DEVELOPMENT AND APPLICATION OF MACHINE LEARNING CLASSIFICATION METHODS

OPTIMAL FEATURE IDENTIFICATION AND PREDICTION OF ANTIMICROBIAL PEPTIDES AND OTHER SOFT MATTER SYSTEMS

BY

Nawisa Jullapech

SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
DEPARTMENT OF MATHEMATICS & STATISTICS
UNIVERSITY OF READING
MARCH 2025

Acknowledgements

First of all, I would like to express my sincere gratitude and appreciation to my supervisor, Dr Fazil Baksh, for his invaluable suggestions, constant support, and a lot of encouragement and motivation throughout the research. Dr Zuowei Wang was my supervisor as well, for his thoughtful suggestions and support of this research. From the beginning, they gave me this PhD study opportunity even though my background was far from Mathematics. During the study, they were patient and kind, providing me with huge knowledge and support, even when I faced challenges or asked seemingly unimportant questions. Also in the future, both of you are a big part of my success.

I would like to thank my Higher Degree Committee members, Professor Sue Todd and Professor Marcus Tindall, for their extremely useful advice and encouragement. I also wish to thank the doctoral examiners, Dr Chieh-Hsi Wu and Dr Patrick Ilg, for all their comments and good suggestions during my memorable viva exam. Special thanks to all my colleagues in the Department of Mathematics and Statistics for sharing the good and the bad times with me. I would also like to extend a great deal of appreciation to all my friends in the United Kingdom and all my friends in Thailand for their constant support and friendship.

I gratefully acknowledge The National Science and Technology Development Agency and the Royal Thai Government for the financial support for 4 years that enabled me to study in the United Kingdom. I am grateful to the Faculty of Science and

Technology, Nakhon Si Thammarat Rajabhat University, Thailand for their support and understanding. I would also like to thank my faculty staff in the Department of Data Analytics and Digital Technology, who allowed me to study and encouraged me.

Finally, I would like to reserve a special thank you to my beloved family, especially my parents, Mr Phongsak Jullapech and Mrs Janpen Jullapech, for their love, support and encouragement. I would also like to thank my younger siblings, Mr Jakkapong Jullapech and Mrs Phatphichcha Jullapech, who always gave me love and encouragement.

Abstract

The research in this thesis integrates the complementary strengths of five machine learning (ML) methods, namely logistic regression (LR), elastic net logistic regression (ENET), support vector machines (SVM), random forests (RF), and neural networks (NN), to simultaneously optimize predictive accuracy and refine variable selection in binary classification tasks. Predictive performance of the ensemble with these five base predictors is evaluated and the impacts of heterogeneity in predictive accuracy and dependency between base predictors are quantified. A novel ensemble prediction and feature selection method based on the majority vote approach, denoted as MV-FS (majority vote-feature selection), is developed to mitigate the limitations of individual predictors and so ensure robust performance across diverse datasets. Theoretical and simulation evaluations of the proposed methodology provide insights into how variations in the model performance and inter-model relationships influence overall effectiveness, which in turn can be used to ensure generalisation and stability, even in challenging classification scenarios.

To validate its practical effectiveness, the ensemble is applied to study two distinct cases in the biological and soft matter areas. In the first case, the MV-FS method is employed to explore the relationship between the physicochemical features of antimicrobial peptides (AMPs) and their antibacterial activities, a task of significant importance in drug discovery due to the growing need for novel antibiotics. In this regard, ML has emerged as a powerful tool for predicting peptide sequences with enhanced antimicrobial activity and selectivity, revolutionizing the way researchers approach

the development of novel antimicrobial agents. By extending existing ML methods via incorporating scientific knowledge of physicochemical and structural characteristics of AMPs with a data-driven ensemble approach, the MV-FS method is able to deepen confidence in identifying the key physicochemical features governing antimicrobial activity and predict regions in the physicochemical descriptor space with high probabilities to find active AMPs.

In the second case, the ensemble is preliminarily tested for predicting the conformational transitions of single charged polymer chains, which can help understand the structural variations of biomacromolecules in different environments and the association behaviours of synthesised polymers for developing novel functional materials. Using molecular dynamics simulation results as training datasets, the conformational regimes predicted by the ensemble method agree well with theoretical expectations, indicating the strong potential of ML methods in predicting the structural properties of macromolecules, especially in regimes where brute force simulations are computationally very costly.

Overall, the research presented in this thesis not only advances the state-of-the-art in ensemble learning for binary classification but also provides a scalable and adaptable framework that can be extended to other domains. By combining interpretability, robustness, and high predictive accuracy, the proposed methodology offers a powerful tool for researchers and practitioners seeking to address classification problems with high precision and reliability.

Table of Contents

List of Tables

List of Figures

Introduction

Antimicrobial peptides (AMPs) are short chains of amino acids with a broad spectrum of antimicrobial activity. Found in various organisms, including humans, animals and plants, AMPs can inhibit growth and replication of bacteria, fungi, viruses, and even cancer cells (Zasloff, 2002). This anti-infective attribute coupled with a relatively low likelihood of developing bacterial resistance make AMPs attractive candidates for addressing the growing problem of antibiotic resistance. Central to the application of AMPs in antimicrobial drug development is knowledge on how their physicochemical properties and structural features combine to influence antimicrobial activity. This information is required to help identify the most effective existing naturally occurring peptides, and is also used in the design of new peptides, to guide structural modifications (such as amino acid substitutions or chemical modifications) so as to tailor AMPs for specific microbial strains or conditions, and to enhance stability and efficacy of active AMPs.

Machine learning (ML) has become a vital tool (Xu et al., 2021; Vishnepolsky et al., 2018) in helping understand the mechanisms and identify key factors governing the antimicrobial activity of AMPs. ML methods have been applied to identify hidden patterns and relationships between AMPs and predictors of activity, to predict

peptide activity based on their sequences, structural information, physicochemical properties and other characteristics (Torrent et al., 2011; Porto et al., 2012; Xiao et al., 2013; Lira et al., 2013; Khamis et al., 2015), and suggest modifications to existing peptides or propose entirely novel sequences with enhanced antimicrobial activity (Wang et al., 2011; Maccari et al., 2013; Bhadra et al., 2018).

Notwithstanding the wealth of research in understanding and predicting the general antimicrobial potency of AMP, in many instances the precise compositions and mechanisms for AMPs being active against particular bacterial strains (Vishnepolsky et al., 2018) or other microbial strains remain unclear. There is also a lack of high-performing ML models in the literature for predicting new amino acid sequences with high therapeutic effects against particular microbial strains (Vishnepolsky et al., 2019). These shortcomings motivate the research presented in this thesis. In particular, ML methods that are able to reliably identify the key features of AMPs driving anti-infective activity against a particular infective agent, and to predict regions of the feature space where such AMPs are highly likely to be found with high degrees of sensitivity and specificity will be developed and evaluated. Ensemble learning is used to realise both the feature identification and prediction objectives set in this work.

Feature selection techniques include filter methods that rank features based on intrinsic properties of the data (such as correlation with the outcome variables) and then select the top-ranked features, wrapper methods (e.g. best subset and stepwise selection) that search for the subset of features that achieves the best performance (Kursa & Rudnicki, 2010), and embedded methods (e.g. lasso and trees) that incorporate feature selection directly into the model building process (Tibshirani, 1996; Breiman,

1995). Moreover, ensemble feature selection aims to improve the robustness and performance of feature selection processes by combining the outputs of multiple individual feature selection algorithms to make more informed and reliable decisions about which features to include in a model. By leveraging the strengths of various algorithms, ensemble feature selection approaches are able to identify features consistently exhibiting high relevance across different methods, thereby increasing the robustness of feature selection. They can also reduce the impact of biases or limitations associated with individual methods (Saeys et al., 2008), leading to enhanced feature ranking and selection accuracy, effective dimensionality reduction and improved model generalization. In this thesis, a novel ensemble feature selection method is developed and evaluated. Although methods have been proposed for improved identification of anti-cancer peptides (Hu et al., 2011; Akbar et al., 2017; Ge et al., 2020), there is no evidence in the literature on the use of ensemble feature selection methods within the context of AMP activity.

ML methods based on different principles, or models based on different training sets, can provide different predictions. An ensemble machine prediction method leverages the strengths of a set of different methods and/or models by combining the predictions of a set of base predictors to produce a final result that is optimal in the sense that the combination reduces both bias and variance and provides better predictive power than that of each constituent base predictor. These base predictors are typically simpler models that may individually have limitations or shortcomings but can contribute collectively to a stronger final prediction when combined strategically. In particular, when the data is nonlinear or involves intricate interactions, combining multiple models allows complex relationships in the data to be captured.

Ensemble methods have been widely and successfully deployed in various application areas (Polikar, 2006). A crucial issue for designing a good ensemble prediction method is the proper selection of base predictors (Polikar, 2006). Base predictors are often designed to capture certain patterns or relationships in the data, but might not be able to capture all the complexities of the underlying problem. To ensure the predictions of the ensemble methods are statistically pairwise independent and correct with high probability, base predictors should be both diverse and accurate. Another crucial issue is the combination rule for integrating the outputs of the base predictors (Polikar, 2006). One example of the combination rule for discrete or categorical outcomes is the majority vote which predicts the outcome as the most frequent class predicted by the base predictors.

The thesis begins with an overview of ML methods and the theory underpinning the techniques used therein. Common ensemble learning and feature selection techniques will also be covered in the first chapter. Chapter 2 begins with a basic description of peptides and their properties followed by a literature review on machine learning methods used for predicting AMP activity. Next, the DB-SCAN method implemented by Vishnepolsky et al. (2018) is programmed and applied to the same peptide sequences, as well as to a larger dataset, in efforts to validate the regions of the physicochemical space where the authors suggested antimicrobial peptides (AMPs) active against the *Escherichia coli* (e-coli) bacteria are highly likely to be found. Chapter 3 explores performances of other machine learning methods, and incorporates scientific knowledge of key physicochemical features and structural characteristics with a data-driven ensemble approach, to identify key factors governing antimicrobial activity of AMPs against bacteria, and predict regions of the physicochemical space

with a high probability of active peptides. The influence of diverse and accurate base predictors and the majority vote combination rule in ensemble learning are evaluated theoretically and via simulations before subsequently being applied to the analysis of antimicrobial peptides in Chapter 4. In Chapter 5, the developed methods are applied to enhance the prediction of conformational properties in charged polymers, specifically through the construction of a conformational phase diagram for diblock polyampholyte chains. Chapter 6 summarizes the key results and findings of the thesis, while also addressing the limitations of the study and outlining potential directions for future research. Notwithstanding the use of ensemble learning in the thesis, it should be noted that the research was conducted with due recognition of the fact that interpretable ML models are crucial for understanding the rationale behind predictions and making informed decisions.

ML methods have gained significant prominence in various fields due to their ability to extract valuable insights from data, automate decision-making processes, and enhance predictive capabilities. For example, in healthcare they are used for disease diagnosis, drug discovery, and patient management. There deep learning models can analyze medical images to detect diseases like cancer (e.g., Koh et al., 2022), and predictive models can forecast patient readmission rates (e.g., Huang et al., 2021). In finance, examples of machine learning applications include the analysis of transaction data to identify suspicious activities (fraud detection) (e.g., Hernandez Aros et al., 2024), the prediction of stock prices based on historical data (e.g., Phuoc et al., 2024), and credit scoring (e.g., Tyagi, 2022). ML methods also enable machines to generate and manipulate human language in applications that include sentiment analysis, chatbots, and language translation (e.g., Jurafsky & Martin, 2025), as well as to interpret and

analyze visual information from images or videos in facial recognition, autonomous vehicles, and object detection applications. Developed and evaluated in this thesis are approaches that will allow researchers to effectively leverage this transformative technology in the design of antimicrobial peptides and related research areas.

Chapter 1

Machine Learning Methods

Machine learning (ML) is a subset of artificial intelligence (AI) that utilises a wide range of mathematical, statistical and computational techniques and algorithms with the objective to enable computers to learn from data, and to make predictions or decisions without explicit programming. Described in this chapter is a methodology for the ML techniques used in this thesis. Consider a collection of functions or models indexed by $\boldsymbol{\Omega}$, $\{f(\boldsymbol{x};\omega);\omega \in \boldsymbol{\Omega}\}$, with input observation(s) $\boldsymbol{x}$ and let $y(\boldsymbol{x};\omega)$ denote the output of $f(\boldsymbol{x};\omega)$. The input observations are also known as predictor variables, independent variables or features; the output is also called the response, outcome, dependent variable or target. These terms will be used interchangeably throughout the thesis. Also, for convenience, arguments of $y(\boldsymbol{x};\omega)$ and $f(\boldsymbol{x};\omega)$ may frequently be dropped. The ML training process involves determining from the collection the model $f(\boldsymbol{x};\omega)$ that optimises an objective defined by the output. The learning process encompasses the training and the trained model's ability to generalize and make predictions on new, unseen data.

The work in this thesis focuses on supervised learning. Here training is achieved using a dataset consisting of N paired observations $(y_i, \boldsymbol{x_i})$, where the observed response y_i and predictor variables $\boldsymbol{x_i}$; $i = 1, \ldots, N$, are typically realisations of a random response variable Y and a random vector variable X, respectively. This type of dataset is known as labelled data in the computer sciences. The primary objective of supervised learning is to be able to use previously unseen observations $\boldsymbol{x}$ to predict the outcome y with a high level of certainty (Hastie et al., 2011). Common supervised learning algorithms include linear regression, logistic regression, neural networks, decision trees, and support vector machines. The theory underlying these and other supervised ML methods are covered in this chapter.

When training is done using data $\boldsymbol{x_i}$; $i = 1, \ldots, N$, that do not include a response or output, otherwise known as unlabelled data, the learning is said to be unsupervised. Here the goal is to discover interesting features about the problem studied using observations of X only. Clustering and dimensionality reduction are common tasks in unsupervised learning, with methods such as k-means clustering (MacQueen, 1967), hierarchical clustering (Lance & Williams, 1967) and principal component analysis (PCA) (Pearson, 1901) used to identify patterns, structures, or groupings in the training data. Interestingly, an unsupervised learning density-based spatial clustering method (DB-SCAN) (Ester et al., 1996) was implemented in the motivating work (Vishnepolsky et al., 2018) for this research. This method will be introduced in this chapter.

Alongside supervised and unsupervised learning, there is reinforcement learning (for example, see Sutton & Barton, 2018) where the training is the trade-off between exploration and exploitation along with reward, which can be positive for all state-action

pairs, to make sequential decisions that maximize a cumulative reward function. This type of learning is prevalent in robotics, game-playing AI, and autonomous systems and is achieved using algorithms such as Monte Carlo, state-action-reward-state-action (SARSA), Q-learning and deep Q-networks (DQN). Reinforcement learning is not covered in this thesis.

Methods used for training can also be classed as either parametric, non-parametric or semi-parametric, depending on how the models in the collection used for training are defined. Parametric methods use a collection of models that are well-defined in a finite-dimensional parameter space, meaning their properties are completely and unambiguously determined by their parameters. A simple example is linear regression where

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p, \tag{1.1}$$

for observed predictor variables $\boldsymbol{x} = (x_1, \ldots, x_p)$ and $\boldsymbol{\beta} = \begin{pmatrix} \beta_0 & \beta_1 & \cdots & \beta_p \end{pmatrix}^T$ an unknown parameter vector in $(p+1)$ dimensional parameter space. Other methods include principal component analysis, discriminant analysis, logistic regression, neural networks, ridge regression, lasso and elastic net. In contrast, non-parametric methods do not assume a rigid parametric form for the fitted model (James et al., 2023) and include methods such as k-means clustering, kernel logistic regression, decision trees and support vector machines. Nonparametric models are not completely lacking in parameters but rather, the number of parameters is flexible. In fact, as will be seen later, the specification of the models in non-parametric methods requires an assumption of an infinite-dimensional parameter space. Semiparametric methods use models that combine parametric and nonparametric models and so have a model

component with finite-dimensional parameter space, and another with an infinite-dimensional parameter space. A well-known example is the Cox model (Cox, 1972) $f(t) = f_0(t)e^{\beta_0+\beta_1 x_1+\ldots+\beta_p x_p}$, where t typically denotes time. The function $f_0(t)$ (a step function) is not pre-specified in terms of any parameters and represents the non-parametric component, whereas $\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$ is the parametric component.

It is often computationally much easier to estimate a parametric model than a non-parametric model and the parametric model is also easier to interpret and understand. However, a parametric model may not match the true form of the relationship between the response Y and predictor variables X. Indeed, as its parameter space is unrestricted, the non-parametric approach has the potential to fit a wider range of possible shapes for this relationship. In other words, it is more flexible than the parametric method. This modelling flexibility however comes at a cost. The non-parametric approach usually requires a larger number of observations and is more likely to overfit the training data, which can reduce the model's generalisability. Also, complicated models may be difficult to interpret and may limit understanding of the relationship between an individual predictor variable and the response. Semi-parametric methods seek to capitalise on the strengths of parametric and non-parametric models, but increasing both model flexibility and interpretability.

As will become evident in the following section, components of the training procedure for different ML methods tend to be different, which leads to learning algorithms with different strengths and weaknesses. This variation in training coupled with the difference between parametric and non-parametric models produces a diverse repertoire of ML methods. This thesis explores conditions under which this diversity is advantageous and investigates how it may be exploited to improve the learning

process.

1.1 Supervised Learning

Consider the random response variable Y and a random vector X and suppose of interest is to learn the relationship f between Y and X using N paired observations $(y_i, \boldsymbol{x}_i);\ i = 1, \ldots, N$. Often very little is known about the form of f and the primary concern of the training process is to find an approximation or estimate to be used in making predictions of the output Y for a given value of X, and to make inferences about the relationship between Y and X. As distinguishing between the various types of estimators is not crucial to understanding the work in this thesis, for simplicity of notation and unless it compromises clarity and comprehension, all estimators will be denoted using the caret '$\widehat{\ }$' notation. A good estimator $\hat{f}$ of f produces accurate estimates of Y with low variability. The data used to estimate f is called training data. If the predictive power of $\hat{f}$ is evaluated using a second independent dataset, this is called the test dataset. The structure of f and its estimation for various supervised learning procedures are covered below. Methods for evaluating the predictive power of f are introduced in Section 1.4.

1.1.1 Linear regression

Linear regression is a fundamental, widely used technique in supervised machine learning and statistics. It is a simple but powerful method for modelling the relationship between a continuous outcome Y and a set of predictor variables $X = (X_1, \ldots, X_p)$. It is also very useful for explaining the driving principles and concepts for all ML

methods. Following this section is an introduction to the ML methods for discrete or categorical outcomes used in the thesis. In the realm of ML terminology, techniques designed for handling continuous outcomes are often denoted as regression methods, whereas those tailored to discrete or categorical outcomes are labelled as classification methods.

The multiple linear regression model is $Y = f(\boldsymbol{x}) + \epsilon$, where $f(\boldsymbol{x})$ is provided by equation (1.1) and ϵ quantifies the deviation between Y and $f(\boldsymbol{x})$. The training process uses training data $(y_i, \boldsymbol{x}_i);\ i = 1, \ldots, N$ to determine the form of $f(\boldsymbol{x})$ that minimises the sum of squared errors loss function or residual sum of squares,

$$RSS = \sum_{i=1}^{N} \left[y_i - f(\boldsymbol{x}_i)\right]^2 = \sum_{i=1}^{N} \left[y_i - (\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})\right]^2, \tag{1.2}$$

over the parameter space $\boldsymbol{\beta}$. The minimising function is called the fitted model and is denoted $\hat{f}(\boldsymbol{x})$. The label "squared errors" for the objective function RSS comes from the fact that $y_i - f(\boldsymbol{x}_i)$ is the error e_i or "loss" incurred as a result of using $f(\boldsymbol{x}_i)$ to estimate y_i. This objective function is also sometimes referred to as the $\mathcal{L}_2$ norm loss function as it is the square of the $\mathcal{L}_2$ norm of the error vector $\begin{pmatrix} e_1 & \cdots & e_N \end{pmatrix}$. While for inferential purposes the error random variable ϵ is assumed to follow a normal distribution with mean $\mu = 0$ and variance σ^2, that is $\epsilon \sim N(0, \sigma^2)$, this assumption is not necessary when determining the optimal model in the training procedure.

From the matrix $\boldsymbol{X}$ such that its i^{th} row is the observed predictor vector $\boldsymbol{x}_i$ but

with a leading 1 appended. Thus $\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N1} & \cdots & x_{Np} \end{pmatrix}$. As the i^{th} element of the product $\boldsymbol{X}\boldsymbol{\beta}$ is simply $f(\boldsymbol{x}_i)$, the linear regression optimisation problem can be written as

$$\min_{\boldsymbol{\beta}} R = \min_{\boldsymbol{\beta}} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \right\}, \tag{1.3}$$

where $\boldsymbol{y} = \begin{pmatrix} y_1 & \cdots & y_N \end{pmatrix}^T$ is the vector of observed outcomes. Using straightforward differential calculus it can be shown that this has solution $\hat{\boldsymbol{\beta}} = \begin{pmatrix} \hat{\beta}_0 & \hat{\beta}_1 & \cdots & \hat{\beta}_p \end{pmatrix}^T$ given by

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^T \boldsymbol{X} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}, \tag{1.4}$$

and the equation of the fitted model is therefore $\hat{f}(\boldsymbol{x}) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p$.

Note that if the model does not include an intercept term β_0, the first column of the matrix $\boldsymbol{X}$ is dropped. Also note from equation (1.4) that multiplying any observed predictor variable $\boldsymbol{x}_j$ by a constant c simply leads to scaling of the estimate $\hat{\beta}_j$ by $\frac{1}{c}$ and therefore the product $x_j \hat{\beta}_j$, and hence the predicted value $\hat{y}$, remains unchanged. In other words, the least squares estimates are scale invariant.

Clearly existence of the estimates $\hat{\boldsymbol{\beta}}$ depends on whether the matrix $\boldsymbol{X}^T \boldsymbol{X}$ is invertible, in other words whether observed values of $X_1, X_2, \ldots, Xp$ are linearly independent. More importantly, although these estimates may exist they can be highly unstable, that is a small change in the training dataset can lead to a big change

in the estimates, if the observed predictor variables are nearly dependent (multicollinear). This is because it can be shown that the variance-covariance matrix $Var\left(\hat{\boldsymbol{\beta}}\right) = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\sigma^2$ and hence variances of the estimates can be large if the determinant of $\boldsymbol{X}^T\boldsymbol{X}$ is close to zero. We can reduce this variance by shrinking or setting some elements of the parameter vector $\boldsymbol{\beta}$ to zero.

In shrinkage methods, all p predictor variables are fitted, but the estimates of the model parameters $\beta_1, \ldots, \beta_p$ are shrunk towards zero relative to the least squares estimates by introducing an additional penalty in the objective function provided in equation (1.3). This shrinkage, also known as regularization, has the effect of reducing the variance of the parameter estimates (Hastie et al., 2011), and can also be used to do variable selection, see Section 1.5. In particular, predictor variables for which the parameter estimates are shrunk to zero may not be relevant to the response. The three regularisation techniques introduced here are ridge regression, lasso regression and elastic net regression.

Ridge regression

Ridge regression (Hoerl & Kennard, 1970) shrinks the regression parameter estimates by restricting the $\mathcal{L}_2$ norm, $\|\boldsymbol{\beta}\|_2 = \sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}} = \sqrt{\sum_{j=1}^{p}\beta_j^2}$, of the parameter vector $\boldsymbol{\beta} = \begin{pmatrix}\beta_1 & \ldots & \beta_p\end{pmatrix}^T$ in the optimisation problem given by equation (1.3), without the intercept term β_0. The ridge regression estimates $\hat{\boldsymbol{\beta}}^{ridge}$ are therefore defined by,

$$\hat{\boldsymbol{\beta}}^{ridge} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\left\{\left(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\right)^T\left(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\right)\right\} \quad \text{subject to } \boldsymbol{\beta}^T\boldsymbol{\beta} \leq s,$$

for some $s > 0$. Reformulated in terms of "Loss + Penalty", or Lagrangian form, the ridge regression estimates are

$$\hat{\boldsymbol{\beta}}^{ridge} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta} \right\}, \tag{1.5}$$

where $\lambda \geq 0$ is a tuning (smoothing) parameter, to be determined separately. As was the case for linear regression, this is a straightforward optimisation problem with a solution,

$$\hat{\boldsymbol{\beta}}^{ridge} = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}, \tag{1.6}$$

where $\boldsymbol{I}$ is the Identity matrix.

Note that, if it is included in the model, the intercept parameter β_0 in ridge regression is usually unpenalised (Hastie et al., 2011) and thus the ridge regression estimates with intercept included are

$$(\hat{\beta}_0, \hat{\boldsymbol{\beta}}^{ridge}) = \underset{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ (\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta} \right\},$$

where $\mathbf{1}$ is a column vector of all ones. If the columns of $\boldsymbol{X}$ are centered, that is each element x_{ij} gets replaced by $x_{ij} - \bar{x}_j$ where $\bar{x}_j = \sum_i x_{ij}/N$, the intercept estimate is simply $\hat{\beta}_0 = \bar{y} = \sum_i y_i/N$. Typically columns of $\boldsymbol{X}$ and the vector $\boldsymbol{y}$ are both centred and so an intercept parameter is not included in the model (Hastie et al., 2011). Also, whereas the least squares estimate in linear regression are scale-invariant, in contrast, because of the penalty $\sum_{j=1}^{p} \beta_j^2$, the ridge regression estimate $\hat{\beta}_j^{ridge}$ can change substantially when multiplying $\boldsymbol{x}_j$ by a constant. To avoid this scale invariance introducing biases into predicting the outcome y, it is best to apply ridge regression after scaling each predictor variable by its standard deviation (James

et al., 2023).

The tuning parameter λ helps to control the relative importance of the data-dependent empirical error and the penalty term (Ogutu et al., 2012). When $\lambda = 0$, the penalty term has no effect and ridge regression estimates are the same as least squares estimates. Whereas $\lambda \to \infty$, the penalty term has increasing impact and the ridge regression estimates will converge to zero (James et al., 2023). In general, ridge regression works better than standard regression if the number of predictor variables p is almost as large as the number of training observations N as in this case, provided the relationship between Y and X is close to linear, the standard regression estimates $\hat{\boldsymbol{\beta}}$ will have low bias but may have high variance. Also, if $p > N$, standard regression estimates do not have a unique solution. As a variable selection method, ridge regression has computational advantages, in terms of the number of models fitted, over other variable selection methods, (see Section 1.5), but one obvious disadvantage is that the ridge estimates are not typically shrunk to zero and hence all p predictor variables are potentially included in the final model.

Lasso regression

Least Absolute Shrinkage and Selection Operator or lasso (Tibshirani, 1996), is a relatively recent alternative to ridge regression that is able to exactly shrink some parameter estimates to zero. The method was motivated by a proposal published in Breiman (1995) for shrinking the standard regression estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$ by minimising

$$\sum_{i=1}^{N} \left[y_i - \hat{\beta}_0 - \left(c_1 \hat{\beta}_1 x_{i1} + \ldots + c_p \hat{\beta}_p x_{ip} \right) \right]^2 \text{ subject to } c_j > 0 \; \forall j, \; \sum_{j=1}^{p} c_j \leq t, \; t > 0.$$

This optimisation procedure was termed a "nonnegative garotte" by the author. Starting with the standard regression estimates, the garotte shrinks each $\hat{\beta}_j$ by the non-negative factor c_j under the constraint $\sum_{j=1}^{p} c_j \leq t$ and $t > 0$. Using simulation studies, Breiman (1995) showed that the garotte compares well with ridge regression except when the true model has many small non-zero parameters. However, the Garotte solution is dependent on the sign and magnitude of the standard regression estimates and can suffer if the model overfits the training data, or the predictor variables are correlated (Tibshirani, 1996).

Tibshirani (1996) proposed modifying the optimisation of equation (1.3) without the intercept term by constraining the $\mathcal{L}_1$ norm, $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$, of $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \cdots & \beta_p \end{pmatrix}^T$. Use of the $\mathcal{L}_1$ norm penalty gives rise to the lasso estimates,

$$\hat{\boldsymbol{\beta}}^{lasso} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \right\} \quad \text{subject to } \|\boldsymbol{\beta}\|_1 \leq t, \ t > 0,$$

or in Lagrangian form,

$$\hat{\boldsymbol{\beta}}^{lasso} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{1.7}$$

where $\lambda \geq 0$ is a tuning parameter.

The lasso optimisation problem is very similar to ridge regression, with the $\mathcal{L}_2$ penalty replaced by the $\mathcal{L}_1$ penalty. As in ridge regression, the data are typically standardised before use in training, and the intercept term is unpenalised if included in the model. Also, the smoothing parameter acts in a similar way as in ridge regression. However, unlike linear and ridge regression, there is no closed form expression for the lasso estimates $\hat{\boldsymbol{\beta}}^{lasso}$ as the ℓ_1 penalty leads to solutions that are nonlinear in the observed

outcomes $\boldsymbol{y}$ (Hastie et al., 2011). Furthermore, unlike ridge regression, the ℓ_1 norm penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter (λ) is sufficiently large (James et al., 2023).

Elastic net regression

Elastic net regression, proposed by Zou and Hastie (2005), is a shrinkage method that is a mixture of lasso and ridge regression. Like lasso regression, it is able to do intrinsic variable selection, while like ridge regression, it is able to shrink together the estimates of correlated predictor variables. Thus, it can be viewed as a generalization of lasso regression that is strong against extreme correlations between the predictor variables (Ogutu et al., 2012).

Assume the observed response vector $\boldsymbol{y}$ is centred and the predictor variables x_{ij} are standardized, as above, so that the training observations satisfy,

$$\sum_{i=1}^{n} y_i = 0, \quad \sum_{i=1}^{n} x_{ij} = 0 \text{ and } \sum_{i=1}^{n} x_{ij}^2 = 1, \text{ for } j = 1, 2, \ldots, p.$$

The naïve elastic net parameter estimates are defined as (Zou & Hastie, 2005),

$$\hat{\boldsymbol{\beta}}^{naive} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda_2 \sum_{j=1}^{p} \beta_j^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| \right\}, \tag{1.8}$$

for $\lambda_1, \lambda_2 > 0$. Denoting $\alpha = \lambda_2/(\lambda_1 + \lambda_2)$ the authors argue that this is equivalent to

$$\hat{\boldsymbol{\beta}}^{naive} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \right\} \text{ subject to } (1-\alpha) \sum_{j=1}^{p} |\beta_j| + \alpha \sum_{j=1}^{p} \beta_j^2 \leq t.$$

This convex combination of lasso and ridge regression penalities, $(1-\alpha)\sum_{j=1}^{p} |\beta_j| +$

$\alpha \sum_{j=1}^{p} \beta_j^2;\ 0 \leq \alpha \leq 1$, is the elastic net penalty. If $\alpha = 1$ the solution $\hat{\beta}^{naive}$ is the ridge estimates whereas if $\alpha = 0$ the lasso estimates are obtained.

Using data from a real study, as well as simulations, Zou and Hastie (2005) show that the naive elastic net procedure tends to overshrink the parameters unless the penalty is very close to either ridge regression or lasso, and propose to address this via the modification,

$$\hat{\boldsymbol{\beta}}^{enet} = (1 + \lambda_2) \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda_2 \sum_{j=1}^{p} \beta_j^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| \right\}, \quad (1.9)$$

where $\hat{\boldsymbol{\beta}}^{enet} = (1 + \lambda_2)\hat{\boldsymbol{\beta}}^{naive}$ are the elastic net estimates of $\boldsymbol{\beta}$. The authors provide theoretical justification for this scaling in Section 3.2 of their manuscript. If the predictor variables are not standardised, the elastic net parameter estimates are obtained by replacing $1 + \lambda_2$ in equation (1.9) by $1 + \lambda_2/N$ (Zou & Zhang, 2009).

The minimising the sum of squared error loss function procedure for parameter estimation introduced in this section is known as the least squares method, and is one of a variety of parameter estimation methods introduced in this chapter. Different estimation methods tend to produce different estimates but this is not always the case. For example, Maximum Likelihood Estimation (MLE), which is introduced below, produces estimates for the linear regression model that are identical to the least squares estimates.

There are several other issues (see James et al., 2023, for example) to consider before the fitted model is used in prediction and inference. A common question asked of the fitted model is whether or not all the predictor variables help to explain Y, or whether only a subset will suffice. This question motivates the feature (variable) selection

component of this research and will be addressed later in the chapter. Another common question which will be addressed later in detail relates to accuracy of the predictions. Issues that are equally important, such as how well the model fits the data, whether some responses in the training data are far from their predicted values (outliers) and whether some observations have a strong influence on estimation of model parameters, are not the focus of this thesis and so are not covered.

1.1.2 Logistic regression

Logistic regression analysis (Cox, 1958) is a well-established statistical technique used to build and interpret a model of the relationship between predictor variables $\boldsymbol{X}$ and a binary or binomial response variable Y. In machine learning, logistic regression may be used to model the probability of a $K = 2$ class output or outcome via the linear function in the predictor variables given by $\beta_0+\beta_1 X_1+\ldots+\beta_p X_p$. For ease of notation this linear function is written here as $\beta_0 + \boldsymbol{X}\boldsymbol{\beta}$, where the vector $\boldsymbol{X} = (X_1, \ldots, X_p)$ and the vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$, as before. Assuming the binary response Y takes values 0 or 1, and denoting the probability that $Y = 1$ conditional on the observations $\boldsymbol{x}$ of $\boldsymbol{X}$ as $\mathcal{P}(\boldsymbol{x}; \beta_0, \boldsymbol{\beta})$, the logistic regression model is given by

$$logit(\mathcal{P}(\boldsymbol{x}; \beta_0, \boldsymbol{\beta})) = \beta_0 + \boldsymbol{x}\boldsymbol{\beta}, \tag{1.10}$$

where $logit(\mathcal{P}) = \log \mathcal{P}/(1 - \mathcal{P})$ is the logit transformation. This probability model can be used to predict class membership for given values of $\boldsymbol{x}$ if values for the parameters β_0 and $\boldsymbol{\beta}$ are available. Note that, as will be seen in the models of AMP activity in this thesis, the linear component $\beta_0 + \boldsymbol{x}\boldsymbol{\beta}$ can be extended to quadratic and higher

order polynomials in $\boldsymbol{x}$ by simply linearly adding the appropriate terms. Another extension of the logistic model to non-linearly separable data classification problems is kernel logistic regression (Zhu & Hastie, 2005), which uses concepts similar to support vector machines. This method is not covered in the thesis.

Estimates of β_0 and $\boldsymbol{\beta}$ are obtained by maximising, over $\beta_0, \boldsymbol{\beta}$, the objective function

$$L(\beta_0, \boldsymbol{\beta}) = \prod_{i=1}^{N} \mathcal{P}\left(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta}\right)^{y_i} \left(1 - \mathcal{P}\left(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta}\right)\right)^{(1-y_i)}, \tag{1.11}$$

where $(y_i, \boldsymbol{x}_i);\ i = 1, \ldots, N$ is the training data. This function is algebraically the joint conditional probability of observing the binary outcomes $y_1, \ldots, y_N$ given the predictor values, but it is taken as a function of the parameters β_0 and $\boldsymbol{\beta}$, and therefore it quantifies how likely the observed data is under various parameter values. In particular, the observations are most probable under the assumed model when $L(\beta_0, \boldsymbol{\beta})$ attains its maximum value. The rationale for estimating β_0 and $\boldsymbol{\beta}$ by maximizing $L(\beta_0, \boldsymbol{\beta})$ is that the best estimators are those which make the observed data most probable. The function $L(\beta_0, \boldsymbol{\beta})$ is known as the likelihood function, or simply the likelihood, and a parameter value $\hat{\boldsymbol{\beta}}$ which $L(\beta_0, \boldsymbol{\beta})$ attains its maximum value is known as a maximum likelihood estimate (MLE).

Equivalently, and for ease of the computational complexity, the log-likelihood

$$\ell\left(\beta_0, \boldsymbol{\beta}\right) = \sum_{i=1}^{N} \left\{ y_i \log \mathcal{P}\left(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta}\right) + (1 - y_i)\, log\ \left(1 - \mathcal{P}\left(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta}\right)\right) \right\}, \tag{1.12}$$

is maximised. Substituting for $\mathcal{P}\left(\boldsymbol{x}_i; \beta_0, \beta\right)$, see equation (1.10), and simplifying gives

$$= \sum_{i=1}^{N} \left\{ y_i(\beta_0 + \boldsymbol{x}_i\boldsymbol{\beta}) - \log\left(1 + e^{\beta_0 + \boldsymbol{x}_i\boldsymbol{\beta}}\right) \right\}. \tag{1.13}$$

Setting the derivatives of $\ell(\beta_0, \boldsymbol{\beta})$ to zero produces the score equations,

$$U = \begin{pmatrix} \frac{\partial \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}_0} \\ \frac{\partial \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{1.14}$$

where

$$\frac{\partial \ell(\beta_0, \boldsymbol{\beta})}{\partial \beta_0} = \sum_{i=1}^{N} (y_i - \mathcal{P}(\boldsymbol{x}_i; \beta_0, \beta)) \quad \text{and} \quad \frac{\partial \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \boldsymbol{x}_i^T (y_i - \mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta})). \tag{1.15}$$

These are $p+1$ nonlinear equations in β_0 and $\boldsymbol{\beta}$ which are solved numerically, as there is no analytic solution in the general case. In particular, the score equations may be solved using the Newton-Raphson algorithm, which requires the second-derivative or Hessian matrix,

$$H = \begin{pmatrix} \frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \beta_0^2} & \frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T \partial \beta_0} \\ \frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \beta_0 \partial \boldsymbol{\beta}} & \frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T \partial \boldsymbol{\beta}}, \end{pmatrix} \tag{1.16}$$

where

$$\frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \beta_0^2} = -\sum_{i=1}^{N} \mathcal{P}(\boldsymbol{x}_i; \beta_0 \boldsymbol{\beta}) (1 - \mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta})), \tag{1.17}$$

$$\frac{\partial^2 \ell(\beta_0; \boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T \partial \boldsymbol{\beta}} = -\sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i \mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta}) (1 - \mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta})), \tag{1.18}$$

$$\frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T \partial \beta_0} = -\sum_{i=1}^{N} \boldsymbol{x}_i \mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta}) (1 - \mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta})), \tag{1.19}$$

and $\frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \beta_0 \partial \boldsymbol{\beta}} = \left(\frac{\partial^2 \ell(\beta_0, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T \partial \beta_0} \right)^T$. Starting with $\begin{pmatrix} \beta_0 & \boldsymbol{\beta} \end{pmatrix}^{old}$, a single Newton update is

$$\begin{pmatrix} \beta_0 & \boldsymbol{\beta} \end{pmatrix}^{new} = \begin{pmatrix} \beta_0 & \boldsymbol{\beta} \end{pmatrix}^{old} - H^{-1} U, \tag{1.20}$$

where the score U and hessian H are evaluated at $\begin{pmatrix} \beta_0 & \boldsymbol{\beta} \end{pmatrix}^{old}$.

Note that the hessian H of second derivatives of the log-likelihood function $\ell(\beta_0, \boldsymbol{\beta})$ is the negative of observed Fisher Information $\hat{I}(\beta_0, \boldsymbol{\beta})$ while taking expectation over the random vector of observations $\boldsymbol{Y}$ produces Fisher Information matrix I. In this case, the above second derivatives do not include $\boldsymbol{Y}$ and thus $E(\hat{I}) = I$, i.e. observed and expected Fisher Information are identical. Replacing H in equation (1.20) with I, called Fisher scoring, can help to improve convergence and is used in numerical algorithms to obtain the maximum likelihood estimates $\hat{\beta}_0$ and $\hat{\boldsymbol{\beta}}$, which are then used to estimate the probability $\hat{\mathcal{P}}(\boldsymbol{x}) = \mathcal{P}(\boldsymbol{x}, \hat{\beta}_0, \hat{\boldsymbol{\beta}})$.

1.1.3 Logistic elastic net regression

Similarly to linear regression with the elastic net penalty, logistic elastic net regression (Zou & Hastie, 2005) fits the logistic regression model but with the combined lasso and ridge regularization. Parameter estimation is via regularized maximum likelihood. In particular, the logistic elastic net estimates are provided by Friedman et al. (2010),

$$(\hat{\beta}_0, \hat{\boldsymbol{\beta}}^{enet}) = \frac{1}{N} \underset{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}^p}{\arg\max} \left\{ \ell(\beta_0, \boldsymbol{\beta}) - \lambda \left(\frac{\alpha}{2} \sum_{j=1}^{p} \beta_j^2 + (1-\alpha) \sum_{j=1}^{p} |\beta_j| \right) \right\}, \quad (1.21)$$

where $\ell(\beta_0, \boldsymbol{\beta})$ is given in equation (1.13). As stated earlier, the $\mathcal{L}_1$ norm penalty generates a sparse model while the $\mathcal{L}_2$ norm penalty encourages grouping effects and stabilizes the $\mathcal{L}_1$ regularization path (Zou & Hastie, 2005). The grouping effect occurs when parameters corresponding to highly associated predictor variables differ only slightly (Zhou, 2013; Yi et al., 2022), while stability is defined as the robustness

to slight changes in data (Kamkar et al., 2016). Choice of the parameter α allows adjustment of the trade-off between sparsity and stability.

Elastic net regression's inherent ability to perform variable selection via regularization makes it a valuable tool in situations where controlling overfitting is essential. Overfitting occurs when a machine learning model learns the training data too well, capturing noise and specific patterns that do not generalize to unseen data, leading to poor performance on new observations. Elastic net can be particularly useful if the dataset has multicollinearity issues (where predictor variables are highly correlated). As shown above, parameter estimation for the traditional logistic regression model is via weighted least squares and, as argued in Section 1.1.1, this method can produce unstable parameter estimates with high variability in the presence of multicollinearity. On the other hand, the logistic elastic net regression estimates are less likely to suffer from these issues. Application of logistic elastic net regression is prevalent in various domains, including healthcare for disease prediction, finance for credit scoring, and marketing for customer churn prediction. Software is readily available in many popular data analysis and machine learning libraries, including glmnet (Friedman et al., 2010) in the R software (R Core Team, 2021), making it easily accessible.

Maximum likelihood estimation, utilised by both logistic and logistic elastic net regression methods, provides the most efficient estimators when the assumptions of the likelihood function are met. In particular, unbiased maximum likelihood estimators achieve Rao-Cramér lower bound (Rao, 1945; Cramér, 1946) while, in general, they are usually asymptotically unbiased and have the smallest possible variances. MLEs can also be robust to outliers if the likelihood function used is less affected by extreme observations. However, a limitation of the maximum likelihood estimation method is

its dependence on specification of a statistical model of the data distribution. This explicit specification of a likelihood function is not required in the other machine learning methods considered in this chapter.

1.1.4 Classification trees

A tree model constructs the relationship between the response Y and its predictor variables $X_1, \ldots, X_p$ by stratifying or segmenting the set of possible values for the predictor variables (the predictor space) into several smaller and more homogeneous regions, and fitting a simple model, (such as the mean response in the case of a continuous response, or the most frequent outcome, for discrete or categorical responses), in each region. The relationship between response and predictor variables is often depicted using a tree diagram, but upside down in the sense that the leaves are at the bottom of the tree. The root is the top node of the tree and represents the entire dataset or population being classified. Each internal node is characterised by a predictor variable, or feature, and a condition or rule that splits the population into two or more subsets based on that predictor variable. Branches emanate from internal nodes and represent the path to follow based on the condition specified at the internal node. Finally, the leaves (or terminal nodes) represent the predicted outcomes for observations from the population. The tree diagram used for depicting the tree model of a continuous response Y is known as a regression tree while that for a discrete or categorical response is called a classification tree.

Figure 1.1 illustrates a classification tree model that involves making a sequence of binary decisions. The prediction model for an observation $(X1, \ldots, X5)$ follows the

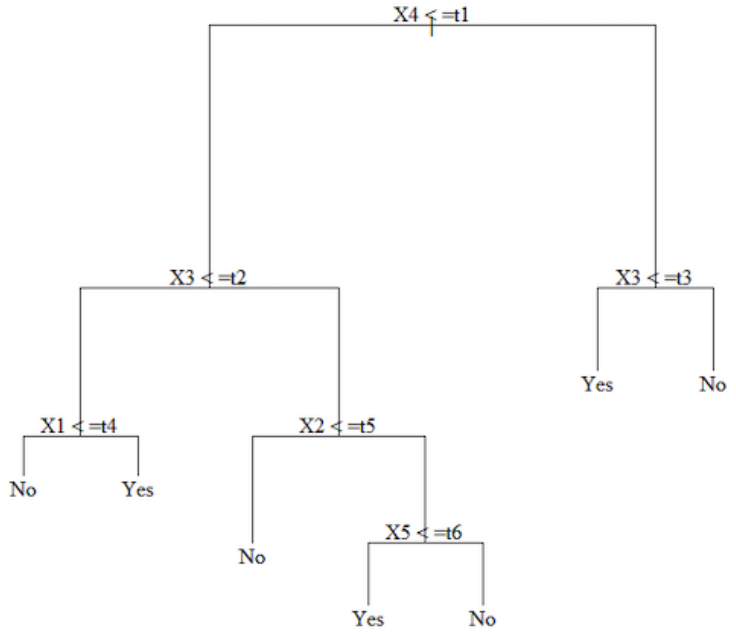


Figure 1.1: The tree diagram. A classification tree depicting the tree model of a binary (Yes/No) outcome and five predictor variables (X1-X5) as a sequence of binary decisions.

path determined by whether or not $X4$ is greater than some known constant $t1$, and continues by following the conditions specified at each node until a leaf is reached. The depth of a tree is the maximal length of a path from the root to a leaf. The example in the figure has a depth equal to four. A deeper tree can capture more complex patterns in the data, but is likely to reduce the model's generalisability to unseen data (overfitting). A tree with a depth equal to one is called a stump.

The classification trees used in this thesis are fitted using recursive binary partitioning (Hastie et al., 2011). This process starts by splitting the predictor space $X_1, \ldots, X_p$ into two regions, modelling the response Y in each region, and choosing the predictor variable X_j and the point s at which this predictor variable is split to achieve the best fit. Next, both of these regions are split into two more regions, and

this process of splitting continues, until some stopping criterion is reached. The nodes of the tree are defined by the regions obtained in the splitting process.

For classification trees, a stopping criterion is that splitting stops if the region (node) is pure, that is all of the training observations within the region are of the same class, but this criterion produces a perfect fit to the training data and potential overfitting. An alternative is to set the tree depth appropriately to avoid the tree being both too deep or too shallow, and underfitting the data. In practice, the optimal tree depth can be set as a hyper-parameter in the fitting algorithm, and is determined using the training data. Methods for doing so will be discussed later. Other options for stopping include not splitting if the number of observations in a region is less than some minimum number, or if some minimum level of gain is not achieved by the split. Measures of gain are discussed below.

Consider the problem of splitting a particular region constructed in the recursive process into two regions $R_1(j,s) = \{\boldsymbol{X}|X_j \leq s\}$ and $R_2(j,s) = \{\boldsymbol{X}|X_j > s\}$ in order to achieve the best split. The fitted value within R_1 is its most commonly occurring outcome, and so, given j and s, a natural measure of fit is the fraction of observations that do not belong to the most common class,

$$E_1 = 1 - \max_k \hat{p}_k, \tag{1.22}$$

where $\hat{p}_k$ denotes the proportion of training observations in R_1 belonging to class k; $k = 1, \ldots, K$. Similarly for R_2. A good fitting model will have a small value for the overall error E, given by a weighted average of E_1 and E_2. So this problem

reduces to solving

$$\min_{j,s} \left\{ \frac{n_1(j,s)}{n} E_1(j,s) + \frac{n_2(j,s)}{n} E_2(j,s) \right\}, \tag{1.23}$$

by scanning all possibilities for j and s, where $n_1(j,s)$ is the number of observations in $R_1(j,s)$, $n_2(j,s)$ is the number of observations in $R_2(j,s)$ and $n = n_1(j,s) + n_2(j,s)$. However, it turns out (James et al., 2023) that E is not sufficiently sensitive for constructing the tree and either Gini Index (Breiman et al., 1984), defined for a region as

$$G = \sum_{k=1}^{K} \hat{p}_k \left(1 - \hat{p}_k\right), \tag{1.24}$$

or cross-entropy (deviance) (Shannon, 1948),

$$D = -\sum_{k=1}^{K} \hat{p}_k \; log \; \hat{p}_k, \tag{1.25}$$

is preferred. Notice that $\log(p) = \log(1 - (1 - p)) \approx -(1 - p)$, by Taylor Theorem, and so G and D are numerically similar.

Gini Index is a measure of variance between the K classes. It is small if all the $\hat{p}_k$ are close to zero or one, and so is referred to as a measure of node purity. As the objective of the tree-building process is construction of homogeneous regions, the chosen split at each stage of the process produces the lowest Gini Index. In other words, as in the above, the splitting criteria for a region are given by the values for j and s that solve

$$\min_{j,s} \left\{ \frac{n_1(j,s)}{n} G_1(j,s) + \frac{n_2(j,s)}{n} G_2(j,s) \right\}. \tag{1.26}$$

Similarly, the chosen split produces the highest reduction in cross-entropy, where the change in cross-entropy due to splitting into the two regions $R_1(j, s)$ and $R_2(j, s)$ is given by the difference between cross-entropy D of the region and the average cross-entropy, $\frac{n_1(j,s)}{n} D_1(j, s) + \frac{n_2(j,s)}{n} D_2(j, s)$.

Note that in cases where the predictor variable X_j is numeric, as assumed in the above description, split points s are usually restricted to mid-points between consecutively ordered and distinct observed values of X_j. In the case of a categorical variable, the split points are levels of the variable. Note also that recursive binary splitting does not always lead to the optimal model as the best split is made at each step, rather than looking ahead and picking a split that leads to a better model in some future step. This approach is often described as greedy. Furthermore, to prevent overfitting, trees may be grown using pruning (see James et al., 2023, for example) which is a process of simplifying the tree by removing branches that do not contribute significantly to the accuracy of the classification.

Wei Yin Loh (2014) credits the regression tree algorithm of Morgan and Sonquist (1963) as the first of its kind to be published in the literature but asserts that the book Classification and Regression Trees (CART) (Breiman et al., 1984) was instrumental in regenerating interest in the subject. One of the advantages of classification trees is their interpretability. They provide a clear, human-readable structure that can be easily understood and visualized, making them a valuable tool for explaining the decision-making process to non-technical stakeholders. Classification trees are widely used in various fields, including finance, healthcare, marketing, and natural language processing. They are the basis for more advanced ensemble methods like random

forests and gradient boosting, which combine multiple decision trees to improve classification accuracy and robustness.

Random forest

Ho (1995, 1998) proposal to address the tree model's lack of generalisation by constructing a collection of trees, or forest, using randomly selected subspaces of the predictor space is the first appearance of this concept found in the literature. With p predictor variables, there are 2^p possible subspaces over which trees can be constructed. For each selected subspace, Ho constructs the tree using the entire training dataset and used examples to illustrate that his approach maintains the highest accuracy on the training data and improves on generalization accuracy. The author advises that independence between trees is critical if the forest is to achieve better accuracy than individual trees.

Subsequently, Breiman (2001) provides a formal definition of a random forest as a collection of trees with the same distribution, and with each tree in the collection dependent on values of a random vector, sampled independently. Breiman (2001) additionally presents a theoretical underpinning for random forests and uses this to argue that this approach solves the overfitting problem. The author also shows that accuracy of a random forest depends on the accuracy of individual trees and the dependence between them, in concurrence with Ho (1995, 1998). This issue will be covered is some detail in Chapter 4 within a more general context.

In practice, a random vector is generated for each tree, independent of previously generated random vectors but with the same distribution, and the tree is grown

using the training dataset and the random vector. For instance, suppose we have a single training dataset with N observations, indexed by 1 to N. The dataset used to grow a particular tree in the collection can be N random observations selected with replacement from this dataset, in which case the random vector associated with this tree is N randomly selected members from $\{1, \ldots, N\}$. In general, this resampling method for model fitting and parameter estimation is known as bootstrapping. To further decouple trees in the forest, Breiman (2001) introduces the concept of using a random selection of the p predictor variables at each node to determine the split and uses empirical evidence to show that accuracy is unaffected by the number of predictor variables selected. After a large number of classification trees are generated, the fitted value for a given observation is the most popular class predicted by the trees. This approach for constructing a forest of trees is utilised in this thesis.

The random forest's potential gain in prediction accuracy over a single tree comes at the cost of interpretability, as it is more difficult to identify from a collection of models the variables that are most important in predicting the outcome. However, summary measures of variable importance can help. One such is Gini Importance, or Mean Decrease in Impurity, which is based on how much a predictor variable decreases the Gini Index (or any other impurity measure like deviance) when it is used for splitting nodes. Every time a variable is used to split the data, the change in Gini Index from before to after the split is measured and the importance score of the variable is calculated by summing up the change in Gini Index across all nodes where the variable was used, averaged over the whole tree. This process is repeated for each tree in the random forest, and the final importance score for each variable is averaged across all the trees. Other measures (for example see Zhou & Hooker,

2020) are based on the mean decrease in accuracy when a variable is dropped, how much each variable contributes to each prediction and how often a variable is used for splitting.

1.1.5 Neural networks

Neural networks are built on the artificial neuron, the concept of which was inspired by the structure and function of biological neurons in the human brain. The foundation for the artificial neuron was laid by the McCulloch-Pitts model (1943) which uses propositional logic to construct a mathematical representation of how neurons could be connected to perform basic logical operations. The representation is based on the premise that a neuron takes input signals from other neurons and combines these into a single "output" which is then sent on to other neurons only if the combined input signals exceed some threshold. The authors claim that this "all-or-none" assumption is sufficient to ensure neuron activity can be represented as a proposition.

Frank Rosenblatt (1957, 1958) introduced the perceptron in 1957, a single-layer neural network capable of binary classification. Intended to mimic the behaviour of a real neuron, a perceptron takes an input vector $\boldsymbol{x} = \begin{pmatrix} x_1 & \cdots & x_p \end{pmatrix}$, linearly combines its elements using weights vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_p)^T$ and produces an output y based on a function ϕ applied to the combined elements, called an activation function. This is illustrated in Figure 1.2. Activation functions enable the modelling of complex relationships between y and $\boldsymbol{x}$. The simplest activation function is the Heaviside, or

unit, step function given by

$$\phi(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x}\boldsymbol{\alpha} > \text{some threshold value;} \\ 0 & \text{otherwise.} \end{cases}$$

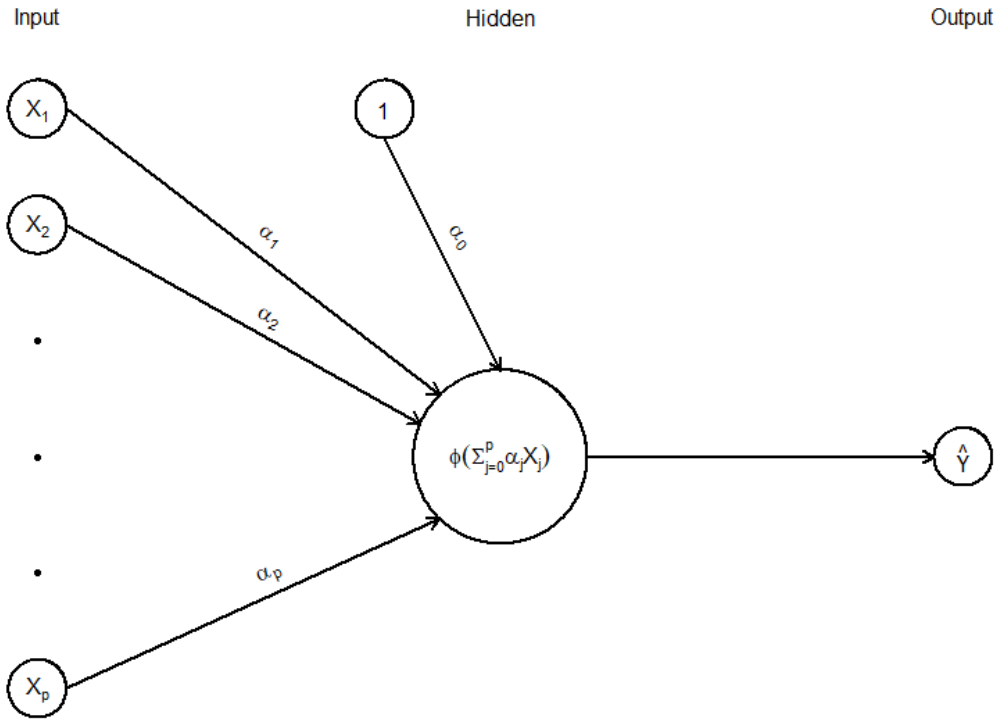


Figure 1.2: Perceptron. The activation function $\phi(\cdot)$ takes inputs $\begin{pmatrix} X_1 & \cdots & X_p \end{pmatrix}$ combined with weights $(\alpha_1, \ldots, \alpha_p)$ and bias α_0 and outputs $\hat{Y}$ if the threshold is exceeded. The intermediate stage between input and output is the hidden layer.

The threshold value can be viewed as a measure of how easy it is to get the perceptron to "fire". Moving the threshold to the left hand side of the above inequality gives

$$\phi(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \alpha_0 + \boldsymbol{x}\boldsymbol{\alpha} > 0; \\ 0, & \text{otherwise,} \end{cases}$$

where $\alpha_0 = -$(threshold value) is known as the bias of the perceptron. Note that the output of the function is more likely to be 1 if the bias α_0 is positive than if it is negative. This fact is used in training a perceptron. Other commonly used activation functions (Nwankpa et al., 2021) are the sigmoid function,

$$\phi(\boldsymbol{x}) = \frac{e^{(\alpha_0 + \boldsymbol{x}\boldsymbol{\alpha})}}{1 + e^{(\alpha_0 + \boldsymbol{x}\boldsymbol{\alpha})}}, \tag{1.27}$$

which is a smoothed version of the Heaviside function; the hyperbolic tangent function,

$$\phi(\boldsymbol{x}) = \frac{e^{(\alpha_0 + \boldsymbol{x}\boldsymbol{\alpha})} - e^{-(\alpha_0 + \boldsymbol{x}\boldsymbol{\alpha})}}{e^{(\alpha_0 + \boldsymbol{x}\boldsymbol{\alpha})} + e^{-(\alpha_0 + \boldsymbol{x}\boldsymbol{\alpha})}};$$

the linear function,

$$\phi(\boldsymbol{x}) = \alpha_0 + \boldsymbol{x}\boldsymbol{\alpha};$$

and the rectified linear unit (ReLU), which is a piecewise linear function given by,

$$\phi(\boldsymbol{x}) = \max\left\{0, \alpha_0 + \boldsymbol{x}\boldsymbol{\alpha}\right\},$$

and so outputs a linear function if this is positive, otherwise it outputs zero.

Training a perceptron involves initially guessing values for the bias and the weights, using the activation function to predict each of the N training responses $y_i, i = 1, \ldots, N$ and adjusting the weights (and bias) based on the error in predicting $y_i, i = 1, \ldots, N$. The adjustment is referred to as back-propagation. This is an iterative procedure, continuing until some stopping criteria is met. A similar approach, described next, is used for training the neural network, which is formed by simply adding together a set of perceptrons and progressively combining outputs at the hidden layers to produce the final output. Figure 1.3 depicts an artificial neural network (ANN)

with a single hidden layer.

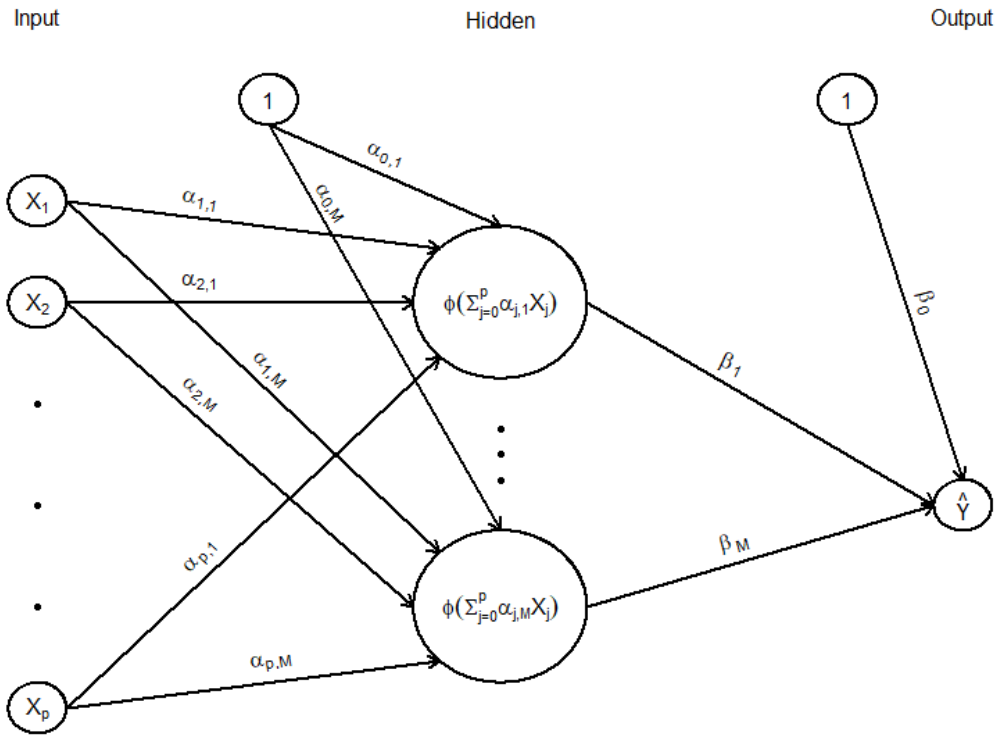


Figure 1.3: A single hidden layer ("vanilla") neural network comprising M derived features, $Z_m = \phi(\alpha_{0,m} + \boldsymbol{X}\boldsymbol{\alpha})$; $m = 1, \ldots, M$, at the hidden layer. The output $\hat{Y}$ is obtained using a function of the linear combination of $Z_1, \ldots, Z_M$ with weights β_0 and $\beta_1, \ldots, \beta_M$. Additional layer(s) can be added between the hidden layer and output; the input for such a layer is the output of the layer that precedes it in the network.

Numerical algorithms for estimating the weights in a single hidden layer neural network with M derived features $Z_1, \ldots, Z_M$, are described below. The estimation process is often referred to as training the network. Consider the classification problem with K classes, so that there are K possible outcomes (target measurements) for the response Y, denoted by $Y_1, \ldots, Y_K$, with each Y_k a Bernoulli $(0-1)$ variable, and

write the derived features as

$$Z_m = \phi\left(\boldsymbol{X}\boldsymbol{\alpha}_m\right);\ m = 1, \ldots, M, \tag{1.28}$$

where $\boldsymbol{\alpha}_m = (\alpha_{0m}, \alpha_{1m}, \alpha_{2m}, \ldots, \alpha_{pm})^T$ is a vector of weights incorporating the bias, and vector $\boldsymbol{X} = (1, X_1, X_2, \ldots, X_p)$ comprise of the p predictor variables augmented with the constant 1 as an additional input feature. Notice that there are M weights $\alpha_{j,1}, \ldots, \alpha_{j,M}$ associated with each predictor variable $X_j;\ j = 1, \ldots, p$, which added to the M biases give a total of $M(p+1)$ weights at the input. The target Y_k is modelled as a function of the linear combination,

$$T_k = \boldsymbol{Z}\boldsymbol{\beta}_k;\ k = 1, 2, \ldots, K, \tag{1.29}$$

where $\boldsymbol{\beta}_k = (\beta_{0k}, \beta_{1k}, \beta_{2k}, \ldots, \beta_{Mk})^T$ is a vector of weights at the output and vector $\boldsymbol{Z} = (1, Z_1, Z_2 \ldots, Z_M)$ comprise the M derived features augmented with the constant 1. There are K weights $\beta_{m1}, \ldots, \beta_{mK}$ associated with each Z_m, which added to the K biases $\beta_{01}, \ldots, \beta_{0K}$, gives $K(M+1)$ weights at the output layer.

Methodology for estimating the $M(p+1)$ and $K(M+1)$ input and output weights are described here using activation function $\phi\left(\boldsymbol{x}\boldsymbol{\alpha}_m\right)$ and the softmax function for transforming the derived variables $t_1, \ldots, t_K$ at the output,

$$g_k\left(t_1, \ldots, t_K\right) = \frac{e^{t_k}}{\sum_{l=1}^{K} e^{t_l}};\ k = 1, 2, \ldots, K. \tag{1.30}$$

For any given observation $\boldsymbol{x} = (x_1, \ldots, x_p)$, each of the K outputs of the softmax function is a number between 0 and 1 and the sum of the outputs is 1. Indeed $g_k\left(t_1, \ldots, t_K\right)$ is an estimate of the probability that $\boldsymbol{x}$ is in the kth class. After

training, the observation $\boldsymbol{x}$ is classified to the class for which $f_k(\boldsymbol{x}) = g_k(t_1, \ldots, t_K)$ is largest.

As is the case for all the methods described in this chapter, the weights in the neural network optimise an objective function that quantifies the fit of the model to the training data. The objective used in the description of the estimation methodology here is the deviance (1.25) used in constructing the classification tree. Noting that y_{ik} is unity if the i^{th} observation in the training dataset belongs to class k otherwise y_{ik} is zero, the deviance for the training data is $D(\boldsymbol{\theta}) = -\sum_{i=1}^{N} D_i(\boldsymbol{\theta})$ where

$$D_i(\boldsymbol{\theta}) = \sum_{k=1}^{K} y_{ik} \log f_k(\boldsymbol{x}_i), \tag{1.31}$$

vector $\boldsymbol{x}_i$ denotes the i^{th} observed set of predictor values and the argument $\boldsymbol{\theta}$ denote the weights. It can be seen that this expression for the deviance is a scalar multiple of equation (1.25), albeit with a different estimator for probability of class membership.

Presented below is a methodology for minimising $D(\boldsymbol{\theta})$ using gradient descent methods, which iteratively search for the optimal weights by moving along the curve $D(\boldsymbol{\theta})$ in a direction based on the gradient. One simple choice for the search direction is the negative gradient of $D(\boldsymbol{\theta})$, leading to the method of steepest descent. An alternative is the conjugate gradient method (Hestenes & Stiefel, 1952). Both methods are described in Appendix A.1. Here, the methodology for training the neural network using the steepest descent method is first described. Next, the method is adapted to the conjugate gradient approach.

For $r = 0, 1, 2, \ldots$, the updates of the weight vectors at the $(r+1)^{th}$ iteration in the

steepest descent method are given by

$$\boldsymbol{\beta}_k^{(r+1)} = \boldsymbol{\beta}_k^{(r)} - \gamma_{(r)} \sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r)}};\ k = 1, \ldots, K, \tag{1.32}$$

and

$$\boldsymbol{\alpha}_m^{(r+1)} = \boldsymbol{\alpha}_m^{(r)} - \gamma_{(r)} \sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r)}};\ m = 1, \ldots, M, \tag{1.33}$$

where $\gamma_{(r)}$ is the step-size or learning rate. An appropriate learning rate is crucial for good performance of the algorithm. In particular, too small a rate will lead to slow convergence while too large a rate can make the algorithm diverge. As a consequence, many techniques for learning rate selection (Allred & Kelly, 1990; Konar, Khandelwal & Tripathi, 2020; Wu & Martin, 2023) have been proposed. An option implements the exact steepest descent method (see Appendix A.1) which requires solving one-dimensional optimization problems and gives,

$$\gamma_{(r)} = \begin{cases} \min\limits_{\gamma} D\left(\boldsymbol{\beta}_k^{(r)} - \gamma \sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r)}}\right), & \text{for } \boldsymbol{\beta}_k; \\ \min\limits_{\gamma} D\left(\boldsymbol{\alpha}_m^{(r)} - \gamma \sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r)}}\right), & \text{for } \boldsymbol{\alpha}_m. \end{cases} \tag{1.34}$$

The above derivatives are obtained using the chain rule.

First consider the partial derivative of D_i with respect to vector $\boldsymbol{\beta}_k$ for some k. Equations (1.29) and (1.30) show that $\boldsymbol{\beta}_k$ occurs only in the function T_k and that T_k is in every $f_l(\boldsymbol{x}_i);\ l = 1, \ldots, K$, hence

$$\frac{\partial D_i}{\partial \boldsymbol{\beta}_k} = \sum_{l=1}^{K} \frac{\partial D_i}{\partial f_l(\boldsymbol{x}_i)} \frac{\partial f_l(\boldsymbol{x}_i)}{\partial t_{ik}} \frac{\partial t_{ik}}{\partial \boldsymbol{\beta}_k};\ k = 1, \ldots, K, \tag{1.35}$$

where t_{ik} is the "current" value of T_k for the i^{th} observation. Differentiating equation (1.31),

$$\frac{\partial D_i}{\partial f_l(\boldsymbol{x}_i)} = y_{il}\frac{1}{f_l(\boldsymbol{x}_i)};\ l = 1, \ldots, K, \tag{1.36}$$

while differentiating equation (1.30) gives

$$\frac{\partial f_l\left(\boldsymbol{x}_i\right)}{\partial t_{ik}} = \begin{cases} f_l\left(\boldsymbol{x}_i\right)\left(1 - f_k\left(\boldsymbol{x}_i\right)\right); & \text{if } l = k, \\ -f_l\left(\boldsymbol{x}_i\right) f_k\left(\boldsymbol{x}_i\right); & \text{if } l \neq k. \end{cases} \tag{1.37}$$

Since

$$\frac{\partial t_{ik}}{\partial \boldsymbol{\beta}_k} = \boldsymbol{z}_i,$$

combining the above terms give

$$-\frac{\partial D_i}{\partial \boldsymbol{\beta}_k} = \delta_i \boldsymbol{z}_i;\ k = 1, \ldots, K, \tag{1.38}$$

where

$$\delta_i = \begin{cases} -\sum_{l=1}^{K} y_{il}\ \left(1 - f_k\left(\boldsymbol{x}_i\right)\right); & \text{if } l = k, \\ \sum_{l=1}^{K} y_{il}\ f_k\left(\boldsymbol{x}_i\right); & \text{if } l \neq k. \end{cases} \tag{1.39}$$

Next, for the partial derivative with respect to weight vector $\boldsymbol{\alpha}_m$ for fixed m, equation (1.29) also shows that Z_m occurs in every T_k and therefore,

$$\frac{\partial D_i}{\partial \boldsymbol{\alpha}_m} = \sum_{k=1}^{K}\sum_{l=1}^{K} \frac{\partial D_i}{\partial f_l(\boldsymbol{x}_i)} \frac{\partial f_l(\boldsymbol{x}_i)}{\partial t_{ik}} \frac{\partial t_{ik}}{\partial z_{im}} \frac{\partial z_{im}}{\partial \boldsymbol{\alpha}_m};\ m = 1, \ldots, M.$$

Differentiating equation (1.29) with respect to z_m and equation (1.28) with respect

to $\boldsymbol{\alpha}_m$ give

$$\frac{\partial t_{ik}}{\partial z_{im}} = \beta_{mk} \quad \text{and} \quad \frac{\partial z_{im}}{\partial \boldsymbol{\alpha}_m} = \phi'(\boldsymbol{x}_i \boldsymbol{\alpha}_m)\boldsymbol{x}_i,$$

where ϕ' is the derivative of activation function ϕ. It follows that

$$-\frac{\partial D_i}{\partial \boldsymbol{\alpha}_m} = s_{im}\boldsymbol{x}_i;\ m = 1, \ldots, M, \tag{1.40}$$

for

$$s_{im} = \sum_{k=1}^{K} \beta_{mk}\phi'(\boldsymbol{x}_i \boldsymbol{\alpha}_m)\delta_i. \tag{1.41}$$

The quantities δ_i and s_{im} can be viewed as "errors" from the current model fit at the output and hidden layer units, respectively. In fitting the neural network, the weights are updated using a two-pass algorithm. In the forward-pass, the current weights are fixed and estimated probabilities $\hat{f}_k(\boldsymbol{x}_i)$ are computed using equation (1.30). In the backward-pass, the errors δ_i (equation (1.39)) for each observation are computed using current values of $\hat{f}_k(\boldsymbol{x}_i)$. These values for δ_i are then "back-propagated" via the "back-propagation equations" (1.41) to give "errors" $s_{im};\ m = 1, \ldots, M;\ i = 1, \ldots, N$ associated with current values of weights $\boldsymbol{\beta}_{mk}$ and $\boldsymbol{\alpha}_m$. Both sets of errors are then used in the gradient descent equations (1.32) and (1.33) to update the weights. The algorithm starts using initial guesses $\boldsymbol{\beta}_k^{(0)}$ and $\boldsymbol{\alpha}_m^{(0)}$ for the weights. This back-propagation algorithm, developed by Rumelhart et al. (1986), has also been called the delta rule.

Back-propagation, a key training technique, is a simple approach and is easy to program on a parallel architecture computer as each hidden unit Z_m passes and receives information only to and from units that share a connection. In other words, the training is efficient as computations can be done separately (parallel processing). But

the steepest gradient descent estimation process can be very slow, (see for example Hastie et al., 2011), and methods such as conjugate gradient are preferred.

Conjugate gradient and steepest descent methods differ in the search direction. While steepest descent uses as search direction the negative gradient of the objective function, conjugate gradient searches for the minimum in a direction conjugate to all previous search directions; see Appendix A.1 for an explanation and a derivation of the method. Thus for $r = 0, 1, 2, \ldots$, the weights at the $(r+1)^{th}$ iteration are updated as in equations (1.32) and (1.33) but with the derivatives replaced by search directions $[d(\boldsymbol{\beta}_k)]^{(r)}$ and $[d(\boldsymbol{\alpha}_m)]^{(r)}$ to now get,

$$\boldsymbol{\beta}_k^{(r+1)} = \boldsymbol{\beta}_k^{(r)} + \gamma_{(r)}[d(\boldsymbol{\beta}_k)]^{(r)};\ k = 1, \ldots, K, \tag{1.42}$$

and

$$\boldsymbol{\alpha}_m^{(r+1)} = \boldsymbol{\alpha}_m^{(r)} + \gamma_{(r)}[d(\boldsymbol{\alpha}_m)]^{(r)};\ m = 1, \ldots, M. \tag{1.43}$$

Normally set as in the steepest gradient method, the first search directions are (Fletcher & Reeves, 1964),

$$[d(\boldsymbol{\beta}_k)]^{(0)} = -\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(0)}} \quad \text{and} \quad [d(\boldsymbol{\alpha}_m)]^{(0)} = -\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(0)}}.$$

For $r \geq 1$, the search directions are recursively defined as (Fletcher & Reeves, 1964),

$$[d(\boldsymbol{\beta}_k)]^{(r)} = -\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r)}} + \delta_{(r)}[d(\boldsymbol{\beta}_k)]^{(r-1)},$$

and

$$[d(\boldsymbol{\alpha}_m)]^{(r)} = -\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r)}} + \delta_{(r)}[d(\boldsymbol{\alpha}_m)]^{(r-1)},$$

where $\delta_{(r)}$ is a scalar. A well-known form for $\delta_{(r)}$ is the ratio of the square of the $\mathcal{L}_2$ norm of successive gradient vectors in the iterative process, provided by the Fletcher-Reeves (FR) formula (1964),

$$\delta_{(r)}^{FR} = \frac{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r)}}\right)^T \left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r)}}\right)}{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r-1)}}\right)^T \left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r-1)}}\right)} \quad \text{and} \quad \delta_{(r)}^{FR} = \frac{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r)}}\right)^T \left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r)}}\right)}{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r-1)}}\right)^T \left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r-1)}}\right)},$$

for updating weights at the output and hidden layers respectively. An alternative form called the Polak-Ribière-Polyak (PRP) formula (Polak & Ribière, 1969; Polyak, 1969) modifies the numerators in the above to get

$$\delta_{(r)}^{PRP} = \frac{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r)}}\right)^T \left(\sum_{i=1}^{N} \left[\frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r)}} - \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r-1)}}\right]\right)}{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r-1)}}\right)^T \left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\beta}_k^{(r-1)}}\right)}$$

and

$$\delta_{(r)}^{PRP} = \frac{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r)}}\right)^T \left(\sum_{i=1}^{N} \left[\frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r)}} - \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r-1)}}\right]\right)}{\left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r-1)}}\right)^T \left(\sum_{i=1}^{N} \frac{\partial D_i}{\partial \boldsymbol{\alpha}_m^{(r-1)}}\right)}.$$

Performances of the FR and PRP formulas have been compared for efficiency and convergence on various occasions (see for example Powell, 1986; Touati-Ahmed &

Storey, 1990; Gilbert & Nocedal, 1992; Babaie-Kafaki, 2012; Babaie-Kafaki & Ghanbari, 2015; Mishra, Chakraborty, Samei & Ram, 2021; Hasibuan, Hendraputra, GS & Saragih, 2022). Which of the two is preferred is unclear as this depends on the particular problem; Powell (1986) concludes that "practical experience and careful consideration of the ingredients of a calculation are vital to the development" of any optimisation algorithm. As in the steepest descent method, various options for the learning rates $\gamma_{(r)}$ exist, including equation (1.34), and exact and inexact line search techniques (Nocedal & Wright, 2006).

Early work in classification used the identity function, $g_k(t_1, \ldots, t_K) = t_k$, at the output layer but this was later abandoned in favour of the softmax function (1.30). Note that for $K = 2$ it is easy to show that the softmax function and the logit transformation (1.10) in logistic regression are equivalent. Additionally, the single hidden layer neural network model with a linear activation function and the logistic model are identical, but the models are fitted in different ways.

There are many acknowledged issues with training the neural network. Typically the global minimiser of the objective function can lead to overfitting and to avoid this, some regularisation is needed (Hastie et al., 2011). Techniques include randomly dropping neurons during training and ℓ_1 or ℓ_2 regularisation, which help prevent overfitting by introducing penalties on large weights. Additional challenges include class imbalance (Zhou et al., 2006), hyperparameter tuning (such as tuning the number of hidden layers) (Diaz et al., 2017), the quality and quantity of the data used to train the neural network, and the availability of sufficient computational resources.

1.1.6 Support vector machine

The approach for classifying an outcome Y described here uses a separating hyperplane in a space defined by the predictor vector $\boldsymbol{X} = (X_1, \ldots, X_p)$. A hyperplane in p-dimensional space is a flat affine subspace of dimension $p - 1$. In particular, a hyperplane separating the two classes of a binary outcome is constructed by maximising the margin between classes, where the margin is defined as the distance between the hyperplane and the nearest training observations from each class. A maximal margin classifier achieves perfect linear separation of the two classes in the p-dimensional space of the predictor variables (feature space). The support vector classifier addresses problems where classes overlap so that linear separation in the p-dimensional feature space is not possible by allowing some training observations to be on the wrong side of their margin. Generalising these two classifiers is the support vector machine (SVM) which enlarges the feature space to allow linear separation.

Writing on the early history of SVMs, Chervonenkis (2013) asserts that the theoretical foundation for maximal margin classifiers and support vector classifiers, which underpins the SVM, was developed in the 1960's by Vladimir N. Vapnik and Alexey Ya. Chervonenkis (Institute of Control Sciences of the Russian Academy of Sciences, Moscow, Russia) during their work on statistical learning theory. The concept of using kernels was introduced by Boser et al. (1992) to handle situations where linear separation of the training data was not possible in p-dimensional feature space. Their methodology is described below.

Consider training the binary classification model using data $(y_i, \boldsymbol{x}_i)$; $i = 1, \ldots, N$, with y_i taking values -1 or 1 and with $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$ a vector in p-dimensional

feature space and write the equation of the separating hyperplane as

$$\boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{\beta} + \beta_0 = 0,$$

where $\boldsymbol{h}(\boldsymbol{x})^T = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_M(\boldsymbol{x}))$ is an M dimensional vector of predefined functions $h_m(\boldsymbol{x})$; $m = 1, \ldots, M$, and $\boldsymbol{\beta}$ is an M-dimensional vector of coefficients. Notice the assumption here that the boundary separating the two classes is linear in its parameters but it is not restricted to linear dependence in $\boldsymbol{X}$. Using the convention that observations "above" the hyperplane are labelled 1 and those "below" are labelled -1, a hyperplane that perfectly separates the training observations according to class labels has the property that

$$(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0) \begin{cases} > 0; & \text{if } y_i = 1, \\ < 0; & \text{if } y_i = -1, \end{cases} \tag{1.44}$$

or equivalently that,

$$y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0\right) > 0; \quad \text{for all } i = 1, \ldots, N.$$

Also, as the distance of a point $\boldsymbol{h}(\boldsymbol{x}_i)$ in the transformed space from the hyperplane is $\frac{1}{\|\boldsymbol{\beta}\|}\left(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0\right)$ if the point is above the hyperplane, and is it $-\frac{1}{\|\boldsymbol{\beta}\|}\left(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0\right)$ if below, the minimum distance of the training data from the hyperplane is given by

$$\Delta = \min_i \left[\frac{1}{\|\boldsymbol{\beta}\|} y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0\right)\right],$$

and the objective is to find values of $\boldsymbol{\beta}$ and β_0 that maximize this margin, subject to the constraints that all observations are on the correct side, and at a distance of

at least the margin from the hyperplane. This provides the constrained optimisation problem,

$$\max_{\boldsymbol{\beta},\beta_0} \Delta \;\text{ subject to } \frac{1}{\|\boldsymbol{\beta}\|} y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T \boldsymbol{\beta} + \beta_0\right) \geq \Delta;\; i = 1, \ldots, N. \tag{1.45}$$

The optimal margin $\hat{\Delta}$ is attained for those training observations satisfying

$$\frac{1}{\|\boldsymbol{\beta}\|} y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T \boldsymbol{\beta} + \beta_0\right) = \hat{\Delta},$$

called support vectors. Also, it is easy to see that if $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ satisfies the constraints in equation (1.45), then so does any scalar multiple $(b\hat{\boldsymbol{\beta}}, b\hat{\beta}_0),\; b > 0$ and therefore there exists an infinite number of possible optimising hyperplanes that differ only in scaling. This problem can be solved by setting $\|\boldsymbol{\beta}\|$ to some fixed value or alternatively, by fixing the product of the margin Δ and $\|\boldsymbol{\beta}\|$ (Boser et al., 1992). In particular, setting

$$\Delta \|\boldsymbol{\beta}\| = 1,$$

gives that maximising Δ is equivalent to minimising $\|\boldsymbol{\beta}\|$ and the problem of finding the maximal margin separating hyperplane $\boldsymbol{h}(\boldsymbol{x})^T \hat{\boldsymbol{\beta}} + \hat{\beta}_0^* = 0$ reduces to solving the convex quadratic problem,

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2} \|\boldsymbol{\beta}\|^2 \;\text{ subject to } y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T \boldsymbol{\beta} + \beta_0\right) \geq 1;\; i = 1, \ldots, N. \tag{1.46}$$

Note that multiplying by $1/2$ does not change the solution and is "included for cosmetic reasons" (Boser et al., 1992). The maximal margin is $\Delta^* = 1/\|\hat{\boldsymbol{\beta}}\|$.

The above can be extended to the problem where the classes overlap and linear

separation is not possible by modifying the constraints to allow some observations to be on the wrong side of their margin, as done in the support vector classifier (Hastie et al., 2011). Two natural ways for modify the constraints in (1.45) are (James et al., 2023),

$$\frac{1}{\|\boldsymbol{\beta}\|} y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T \boldsymbol{\beta} + \beta_0\right) \geq \Delta - \xi_i \quad \text{and} \quad \frac{1}{\|\boldsymbol{\beta}\|} y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T \boldsymbol{\beta} + \beta_0\right) \geq \Delta(1 - \xi_i),$$

where $\xi_i \geq 0;\ i = 1, \ldots, N$, are constrained slack variables. These two ways lead to different solutions. The first modification seems natural as it measures the overlap ξ_i in actual distance from the margin, but leads to a non-convex optimisation problem. The slack ξ_i in the second way measures the proportional amount by which the i^{th} observation is on the wrong side of the margin and so measures overlap in relative distance. This changes the width of the margin but produces a convex optimisation problem and so is preferred (James et al., 2023).

Modifying the constraints using the second way, the optimisation given by equation (1.46) becomes,

$$\min_{\boldsymbol{\beta}, \beta_0} \frac{1}{2} \|\boldsymbol{\beta}\|^2 \ \text{ subject to } \begin{cases} y_i \left(\boldsymbol{h}(\boldsymbol{x}_i)^T \boldsymbol{\beta} + \beta_0\right) \geq 1 - \xi_i, \\ \xi_i \geq 0,\ \sum_{i=1}^{N} \xi_i \leq \textit{constant};\ i = 1, \ldots, N. \end{cases}$$

Bounding $\sum_{i=1}^{N} \xi_i$ places a limit on the total proportional amount by which training observations fall on the wrong side of the margin. Computationally it is more convenient to incorporate this sum into the objective function (Hastie et al., 2011) to

get,

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i \;\text{ subject to } y_i\left(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta}+\beta_0\right) \geq 1-\xi_i,\; \xi_i \geq 0;\; i=1,\ldots,N, \tag{1.47}$$

where C is a "cost" parameter. Using the Method of Lagrange Multipliers generalised to inequality constraints, (see Appendix A.1), to solve this problem gives the Lagrange (primal) function,

$$L_P = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\left[y_i\left(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta}+\beta_0\right)-(1-\xi_i)\right] - \sum_{i=1}^{N}\nu_i\xi_i,$$

where $\lambda_i, \nu_i \geq 0;\; i=1,\ldots,N$, are Lagrange multipliers, or Kühn-Tucker coefficients. Solving equation (1.47) is equivalent to searching a saddle point at which L_P is at a minimum with respect to $\boldsymbol{\beta}$ and a maximum with respect to $\lambda_i,\; i=1,\ldots,N$ (Boser et al., 1992).

Partially differentiating L_P with respect to the vector $\boldsymbol{\beta}$ and setting the derivative to the zero vector gives,

$$\boldsymbol{\beta} = \sum_{i=1}^{N}\lambda_i y_i \boldsymbol{h}(\boldsymbol{x}_i). \tag{1.48}$$

Similarly, differentiating with respect to β_0 and ξ_i gives,

$$0 = \sum_{i=1}^{N}\lambda_i y_i, \tag{1.49}$$

$$\lambda_i = C - \nu_i. \tag{1.50}$$

These equations must be satisfied at the optimum. They are used to produce the Wolfe dual L_D (Hastie et al., 2011) by eliminating $\boldsymbol{\beta}, \beta_0$ and $\xi_i, \nu_i;\; i=1,\ldots,N$, from

L_P.

Substituting for $\boldsymbol{\beta}$ in $\frac{1}{2}\|\boldsymbol{\beta}\|^2 = \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{\beta}$ gives this term as

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\lambda_i\lambda_{i'}y_iy_{i'}\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{h}(\boldsymbol{x}_{i'}),$$

while substituting in $\sum_{i=1}^{N}\lambda_i y_i\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta}$ gives

$$\sum_{i=1}^{N}\sum_{i'=1}^{N}\lambda_i\lambda_{i'}y_iy_{i'}\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{h}(\boldsymbol{x}_{i'}).$$

Further $C - \nu_i - \lambda_i = 0$ leads to $C\xi_i + \lambda_i(1-\xi_i) - \nu_i\xi_i = (C - \nu_i - \lambda_i)\xi_i + \lambda_i$ being simply λ_i, and noting that $\sum_{i=1}^{N}\lambda_i y_i = 0$ gives,

$$L_D = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\lambda_i\lambda_{i'}y_iy_{i'}\langle\boldsymbol{h}(\boldsymbol{x}_i),\boldsymbol{h}(\boldsymbol{x}_{i'})\rangle, \tag{1.51}$$

where $\langle\boldsymbol{h}(\boldsymbol{x}_i),\boldsymbol{h}(\boldsymbol{x}_{i'})\rangle = \boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{h}(\boldsymbol{x}_{i'})$ is the inner product of $\boldsymbol{h}(\boldsymbol{x}_i)$ and $\boldsymbol{h}(\boldsymbol{x}_{i'})$. Incorporating the conditions under which this objective function was constructed gives the optimisation problem,

$$\max_{\lambda_1,\ldots,\lambda_N}\left\{\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\lambda_i\lambda_{i'}y_iy_{i'}\langle\boldsymbol{h}(\boldsymbol{x}_i),\boldsymbol{h}(\boldsymbol{x}_{i'})\rangle\right\};\ 0\le\lambda_i\le C,\ \sum_{i=1}^{N}\lambda_i y_i = 0,$$

which is solved under the additional Karush-Kuhn-Tucker conditions (Hastie et al., 2011)

$$\lambda_i\left[y_i(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0) - (1-\xi_i)\right] = 0, \tag{1.52}$$

$$\nu_i\xi_i = 0, \tag{1.53}$$

$$y_i(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0) - (1-\xi_i) \ge 0. \tag{1.54}$$

Writing the optimisers of the above as $\hat{\lambda}_1, \ldots, \hat{\lambda}_N$, then by equation (1.48) the solution for $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{N} \hat{\lambda}_i y_i \boldsymbol{h}(\boldsymbol{x}_i).$$

By constraint (1.52), $\hat{\lambda}_i$ will be non-zero only if constraint (1.54) is exactly met and therefore the support vectors are those observations $\boldsymbol{x}_i$ for which $y_i(\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0) = 1 - \xi_i$. Support vectors on the margin will have $\hat{\xi}_i = 0$ and so any of these points can be used to solve equation (1.52) for $\hat{\beta}_0$. For numerical stability, an average of all such solutions is used (Hastie et al., 2011). The tuning parameter of this procedure is the cost parameter C. Too large a value allows too few observations to be on the wrong side of their margin, which leads to an overly wiggly boundary and overfitting in the original feature space. Too small a value provides a smoother boundary but the classifier may miss important patterns in the data.

Substituting for the estimates $\hat{\boldsymbol{\beta}}$ and $\hat{\beta}_0$ in $\boldsymbol{h}(\boldsymbol{x}_i)^T\boldsymbol{\beta} + \beta_0$ gives the decision function,

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{N} \hat{\lambda}_i y_i \langle \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}_i) \rangle + \hat{\beta}_0. \tag{1.55}$$

By equation (1.44), the support vector machine classifies an observation $\boldsymbol{x}_0$ as 1 if $\hat{f}(\boldsymbol{x}_0) > 0$ and as -1 if $\hat{f}(\boldsymbol{x}_0) < 0$. Importantly, as both this decision function and equation (1.51) involve the transformation of the predictor space $\boldsymbol{h}(\boldsymbol{x})$ through inner products only, it is sufficient to know only a symmetric, positive-definite kernel function,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}') \rangle \tag{1.56}$$

that computes inner products in the transformed space (Hastie et al., 2011; Boser et al., 1992). Three popularly used kernel functions are the d-degree polynomial

kernel,

$$K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d , \tag{1.57}$$

the radial kernel,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2\right), \tag{1.58}$$

and the sigmoid kernel,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tanh\left(\kappa_1 \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \kappa_2\right). \tag{1.59}$$

A review of properties of kernels is provided in (Cervantes et al., 2020) who also conclude that choice of the kernel used should be based on characteristics of the data and further that the results will also depend on the value(s) of the kernel parameter(s). Grid search and cross-validation methods are proposed (Hastie, Tibshirani & Friedman, 2011) for hyper-parameter tuning; that is to determine the cost C, and the kernel and it parameter values (e.g. $\gamma, \kappa_1, \kappa_2$).

Osuna et al. (1957) present a robust discussion on the underlying theory of support vector machines. Since development, this method has gained popularity across various fields, including finance, biology, and computer vision and the SVM remains a valuable supervised machine learning tool. They are particularly well-adapted to high-dimensional spaces. Over the years various extensions and variations, including multi-class SVMs, semi-supervised SVMs, and online SVMs, have been proposed to address specific challenges.

1.2 Ensemble Methods

An ensemble machine learning method leverages the strengths of a set of different methods and/or models by combining the predictions of a set of base predictors to produce a final result that is optimal in the sense that the combination reduces both bias and variance and provides better predictive power than that of each of its constituent base predictors. In particular, when the data is nonlinear or involves intricate interactions, combining multiple models allows complex relationships in the data to be captured. Covered here are ensemble methods that use a single machine learning method, such as a decision tree, to construct a set of base predictors constructed using variants, or modifications, of the training data. The base predictors here are simpler models that may individually have limitations or shortcomings but can contribute collectively to a stronger final prediction when combined strategically. The base predictors need not be treated equally when combining base predictions; well-performing base predictors can be given more weight than poorly performing ones. Three common methods covered here are Bagging, Boosting and Stacking. Bagging is part of the process of fitting a random forest model while Boosting and Stacking are included in the thesis for completeness.

1.2.1 Bagging

One way to reduce variance and enhance the predictive accuracy of a statistical learning method is to generate multiple training datasets from the population, build a separate prediction model for each, and combine their predictions (James et al., 2023). Introduced by Breiman (1996a), bagging applies this principle in a more practical

setting where only a single training set is available. It is particularly effective for complex and noisy datasets, helping to mitigate overfitting. The bagging process begins with bootstrapping (Efron, 1979), a technique that creates multiple subsets of the original training dataset by randomly sampling data points with replacement. This results in subsets of similar size to the original dataset, though some instances may be repeated while others are omitted. Each subset is then used to train a separate base model. Bagging, short for bootstrap aggregating, combines these models' predictions to improve overall performance.

Consider predicting the response y_0 at input $\boldsymbol{x}_0$. Given a training set,

$$\mathbf{Z} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\},$$

from the population, generate B bootstrapped training sets to produce the bootstrap samples $\mathbf{Z}^b,\ b = 1, 2, ..., B$. Next, use a machine learning method to fit a prediction model on the b^{th} bootstrapped training set and obtain the prediction $\hat{f}^b(\boldsymbol{x}_0),\ b = 1, 2, ..., B$. Thus each base model generates its own prediction for a given input instance. Finally, combine the B predictions to create a final prediction $\hat{f}_{bag}(\boldsymbol{x}_0)$. If the response Y is quantitative the final prediction is often computed as the average, $\hat{f}_{bag}(\boldsymbol{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(\boldsymbol{x}_0)$, or median of the individual base model predictions. In the case of a classification problem, the final prediction is typically determined by a majority vote among the predictions of all the base models. For this thesis, bootstrapping was applied to provide quantification of uncertainty associated with the estimation of the real AMP.

Random Forest, introduced in Section 1.1.4, is a well-known example of a bagging-based ensemble algorithm, where each base model is a decision tree constructed using, not all, but only a pre-determined random subset of the p predictor variables. Thus, Random Forest further enhances bagging by introducing additional randomness during the tree-building process, which helps to diversify the ensemble even more.

By training multiple models on different subsets of data, bagging reduces the variance of predictions. This is particularly beneficial when base models are prone to overfitting. The combined ensemble model thus tends to produce more stable and reliable predictions, but this comes with a loss of interpretability of the model. Bagging also helps improve the generalisation performance of the ensemble by reducing the impact of outliers and noise in the training data. The averaging or voting process across multiple models tends to smooth out individual model errors, leading to a more accurate final prediction. Breiman (1996a) argues that substantial gains in accuracy can be achieved if perturbing the training dataset causes significant changes in the base models. However, if the learning method has high bias, the bagged estimate will also be highly biased. Finally, an important characteristic of bagging is that it can be parallelized, as the training of each base model is independent of the others. This makes bagging well-suited for distributed computing environments.

1.2.2 Boosting

The core idea behind boosting is to sequentially train a series of base models, each focusing on correcting the mistakes of its predecessors. Each base model tends to be a weak learner, such as a decision tree with limited depth or performance, and

so individually may not perform well on the entire dataset, but it possesses enough discriminatory power to classify some observations better than random chance. During each iteration of the boosting algorithm, a new base model is obtained using the same dataset as the original data but with different weights assigned to the training observations. The weights indicate the importance of each observation in terms of its misclassification in previous rounds of the sequential training. By iteratively adjusting the weights, and emphasizing those outcomes that were misclassified in previous rounds, boosting gradually creates a robust and accurate composite model. Popular boosting algorithms include AdaBoost (Adaptive Boosting) (Freund & Schapire, 1997), Gradient Boosting (Friedman, 2001, 2002; Mason et al., 1999) which is similar to AdaBoost but models are trained in a more general gradient descent framework minimizing a loss function, and XGBoost (Extreme Gradient Boosting) (Chen & Guestrin, 2016), among others.

The algorithm AdaBoost.M1 (Freund & Schapire, 1997) is outlined here for binary classification and using the decision tree method to construct the base models. Assume the response is coded as $Y \in \{-1, 1\}$ and a total of B iterations are performed, and let w_i denote the weight applied to training observation $(\boldsymbol{x}_i, y_i);\ i = 1, ..., N$. The objective is to construct base models $\hat{f}_b$ and weights $\alpha_b;\ b = 1, \ldots, B$ using iteratively re-weighted training data, and hence to produce the weighted voting prediction at $\boldsymbol{x}_0$,

$$\hat{f}(\boldsymbol{x}_0) = \text{sign}\left(\sum_{b=1}^{B} \alpha_b \hat{f}_b(\boldsymbol{x}_0)\right). \tag{1.60}$$

Observations $(\boldsymbol{x}_i, y_i)$ that are misclassified by the current ensemble model are assigned higher weights in subsequent iterations. This ensures that the new base model focuses more on those instances that were challenging for the ensemble up to that point. The

aim is to correct the mistakes of previous models and improve overall predictive performance. In addition, the weight α_b assigned to base model $\hat{f}_b$ is determined based on the performance of $\hat{f}_b$ in classifying all training outcomes. Better-performing models are assigned higher weights, indicating their relative importance in the final ensemble.

Setting weights $w_i = 1/N$ initially, a tree with d splits, $(d + 1$ terminal nodes), is fitted to the training data using weights w_i and estimates $\hat{f}_1(\boldsymbol{x}_i),\ i = 1, \ldots, N$, are obtained. Next, the weighted misclassification error rate,

$$\text{err}_1 = \frac{\sum_{i=1}^{N} w_i I\left(y_i \neq \hat{f}_1(\boldsymbol{x}_i)\right)}{\sum_{i=1}^{N} w_i},$$

and the log-odds,

$$\alpha_1 = \log\left(\frac{1 - \text{err}_1}{\text{err}_1}\right),$$

are computed. Notice that only misclassified observations $(y_i \neq \hat{f}_1(\boldsymbol{x}_i))$ contribute to the error err_1 and that α_1 gets larger and more positive as err_1 approaches zero. For the next iteration, the weights w_i are updated by $w_i \leftarrow w_i \cdot \exp\{\alpha_1 I(y_i \neq \hat{f}_1(\boldsymbol{x}_i))\};\ i = 1, \ldots, N$, and thus observations misclassified by $\hat{f}_1$ have their weights scaled by a factor $\exp(\alpha_1)$, increasing their relative influence for the next iteration. The above steps are repeated B times. Since at step b, observations misclassified by the tree at the previous $b - 1$ step have their weights increased while those correctly classified have their weights decreased, as the iterations proceed, each successive tree is forced to concentrate on training observations that were missed by the previous tree.

To prevent overfitting, boosting algorithms often include regularization techniques such as limiting the depth d of decision trees or applying a learning rate that controls

the impact of each new base model on the ensemble. The number d of splits in each tree in a boosted ensemble controls the complexity of the ensemble. Frequently $d = 1$ works well, in which case each tree is a stump, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables. Boosting can also overfit the training data if the number of base models B is too large. To address this, cross-validation can be used to select the number of trees B. Finally, a shrinkage parameter λ which is a small positive number and controls the boosting learning rate is often used. Normally, values are 0.01 or 0.001 but the value depends on the problem. If very small λ can require using a very large value of the number of trees B to achieve good performance. Finally, while the final prediction of the boosting ensemble is often a weighted combination of the predictions from individual base models, see equation (1.60), other aggregation strategies are possible, (see Chen & Guestrin, 2016, for example).

Boosting can significantly enhance the accuracy of models, especially when the underlying base models are only slightly better than random guessing. While boosting can potentially lead to overfitting if the algorithm is not controlled properly, for example by finding the right balance between the number of iterations and the learning rate, it is generally less prone to overfitting compared to training a single, highly complex model. It is also possible to overfit the boosting model if the base models are too complex. Boosting can be sensitive to noisy data and outliers. If the base models focus too much on noisy samples, the final boosted model might suffer from reduced generalisation performance. Depending on the class distribution and how

boosting is implemented, the boosting classifier may be biased towards the majority class, affecting its ability to predict minority classes effectively. Finally, boosting involves creating multiple base models sequentially, which can make it computationally intensive and time-consuming, especially if the dataset is large.

1.2.3 Stacking

Unlike bagging and boosting, which focus on combining multiple instances of a single base model, stacking leverages the diversity of multiple base models to create a more robust and accurate final prediction. While the stacking method, also known as stacked generalisation, was formalised and evaluated by Breiman (1996b) from an idea by Wolpert (1992), the concept of combining predictions is well-known in statistics (Rao & Subrahmaniam, 1971; Efron & Morris, 1973; Rubin & Weisberg, 1975; Berger & Bock, 1976; Green & Strawderman, 1991). Breiman (1996b) presented evaluations of stacking when base models are constructed using regression trees, using linear regression with subset selection, and using ridge regression. For each of these three situations Breiman (1996b) showed that stacking can produce predictions with substantially reduced errors, that stacking never does worse than the single best base model used in the ensemble, and advocated that biggest gains came when dissimilar sets of predictors were stacked.

The stacking process involves multiple stages. In the first stage, a variety of diverse base models are constructed, often using different methods. Thus these models can include decision trees, support vector machines, neural networks, k-nearest neighbours, and more. Each base model is individually trained on the same training dataset but

can employ different feature representations, parameter settings, and learning strategies. To prevent overfitting and ensure generalization, a validation set is often used during this stage. In the second stage, a meta-model, also called the blender or aggregator, is employed to combine the first-level predictions from the base models. This meta-model is typically a simpler model, such as a linear regression, logistic regression, or another machine learning algorithm, which takes the first-level predictions as inputs and generates the stacked prediction.

The following illustration of stacking for binary classification follows the approach outlined in (Ting & Witten, 1999), but with leave-one-out cross-validation. Let $\hat{f}_m^{(-i)}(\boldsymbol{x}_i);\ m = 1, \ldots, M$, denote the predicted probability at $\boldsymbol{x}_i$ using base model $m^{(-i)}$, obtained when method m is applied to the training dataset without the i^{th} observation. The set $\left\{\hat{f}_1^{-i}(\boldsymbol{x}_i), \ldots, \hat{f}_M^{-i}(\boldsymbol{x}_i)\right\}$ are then the M first-level predictions of the i^{th} observation, $i = 1, \ldots, N$, to be used in the second stage. Assuming a linear regression meta-model, the stacked prediction $\hat{f}(\boldsymbol{x}_0)$ for a new observation $\boldsymbol{x}_0$ is the weighted linear combination,

$$\hat{f}(\boldsymbol{x}_0) = \sum_{m=1}^{M} w_m^{st} \hat{f}_m(\boldsymbol{x}_0), \tag{1.61}$$

where estimates of the weights $w^{st} = \{w_1^{st}, \ldots, w_M^{st}\}$ are obtained from the least squares linear regression of y_i on $\hat{f}_m^{-i}(\boldsymbol{x}_i);\ m = 1, ..., M$. Following Breiman (1996b), the weights are constrained to be non-negative, $\hat{w}_m^{st} \geqslant 0$, and to sum to 1, $\sum_m \hat{w}_m^{st} = 1$, and thus the stacking weights are given by

$$\hat{w}^{st} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} w_m^{st} \hat{f}_m^{-i}(\boldsymbol{x}_i) \right]^2 \quad \text{subject to } \hat{w}_m^{st} \geqslant 0; \sum_m \hat{w}_m^{st} = 1. \tag{1.62}$$

The final prediction is $\sum_m \hat{w}_m^{st} \hat{f}_m(\boldsymbol{x}_0)$, where $\hat{f}_m(\boldsymbol{x}_0)$ is obtained by applying method m to the entire training dataset.

Notice that equation (1.61) is simply a weighted linear combination of M predictions of $\boldsymbol{x}_0$ and so, from this perspective, stacking is similar to majority vote (see Chapter 4). The key advantage of stacking is its ability to capture diverse patterns and characteristics from different base models, leading to improved predictive performance. By combining the strengths of various models, stacking can mitigate the weaknesses of individual models and provide more accurate, robust, and stable predictions. It is therefore particularly useful when dealing with complex and challenging prediction tasks, where a single model might struggle to capture all the underlying patterns in the data. But as with bagging, having more than one model hinders interpretability.

1.3 Unsupervised Learning

Covered in this section is a density based spatial clustering method that has been used (Vishnepolsky et al., 2018) for classifying peptides. Spatial clustering methods partition spatial data into subsets, called clusters, such that observations within each cluster are more similar than those in different clusters. Clustering methods may be grouped into three distinct types: combinatorial algorithms, mixture modelling and mode seeking (Hastie et al., 2011). Combinatorial methods use the data directly with no direct reference to a probability model and assign the data to clusters depending on some measure of similarity between pairs of observations. An example is K-means clustering. On the other hand, mixture modelling assumes that each cluster of data

are independent, identically distributed observation from some distribution. The distribution of all observations is then a mixture of the distributions of the clusters. This model is fitted using maximum likelihood or Bayesian approaches. An example is Gaussian mixture model which assumes a certain number of clusters, with observations in each cluster being a sample from some normal (Gaussian) distribution. The third type, mode seeking, attempts to estimate distinct modes of the data distribution non-parametrically. That is, without assuming any form for the distribution. An example is the patient rule induction method (PRIM) which partitions the data space into boxed regions and seeks boxes for which the outcomes have a high average. A thorough coverage of this method is provided by Hastie et al. (2011).

1.3.1 DB-SCAN

Density-Based Spatial Clustering and Application with Noise (DB-SCAN) (Ester et al., 1996) is a combinatorial method that seeks to form dense clusters of data separated by regions of lower density. Here, density is measured by the number of observations close to a given point. The DB-SCAN algorithm is based on the notion of "clusters" and "noise", where noise is defined as the set of data points that are not within any cluster. The key idea is for each point of a cluster, the neighbourhood of a given radius has to contain at least a minimum number of points.

DB-SCAN requires two important parameters, namely ϵ and minimum points ($MinP$). The first parameter (ϵ) defines the radius of the neighbourhood around a point $\boldsymbol{x}$ (called the ϵ-neighbourhood of $\boldsymbol{x}$) while the second ($MinP$) defines the minimum number of points within the ϵ-neighbourhood. The shape of a neighbourhood is

defined by the choice of a distance function between points $\boldsymbol{x}$ and $\boldsymbol{x}'$, denoted by $dist\,(\boldsymbol{x}, \boldsymbol{x}')$. Typically Euclidean distance is used. Core points are those within a cluster, border points are on a cluster border and outlier points (noise) are not in any cluster.

Consider a training dataset $\mathcal{T}$ consisting of points in a space. The ϵ-neighbourhood of a point $\boldsymbol{x}$ in $\mathcal{T}$ is (Birant & Kut, 2007),

$$N_\epsilon\,(\boldsymbol{x}) = \{\boldsymbol{x}' \in \mathcal{T} \mid dist\,(\boldsymbol{x}, \boldsymbol{x}') \leqslant \epsilon\}\,. \tag{1.63}$$

A point $\boldsymbol{x} \in \mathcal{T}$ is a core point if,

$$|N_\epsilon\,(\boldsymbol{x})| \geqslant MinP, \tag{1.64}$$

and a non-core point $\boldsymbol{x}'$ in the neighbourhood of a core point $\boldsymbol{x}$ is a border point if,

$$|N_\epsilon\,(\boldsymbol{x}')| < MinP. \tag{1.65}$$

DB-SCAN works with the concept of density-reachability to find clusters (Ester, Kriegel, Sander & Xu, 1996). Points can be directly density-reachable, density-reachable or density-connected. A point $\boldsymbol{x}'$ is directly density-reachable from a point $\boldsymbol{x}$ if $\boldsymbol{x}' \in N_\epsilon\,(\boldsymbol{x})$ and $\boldsymbol{x}$ is a core point. If $\boldsymbol{x}'$ is also a core point then, as $\boldsymbol{x} \in N_\epsilon\,(\boldsymbol{x}')$, the point $\boldsymbol{x}$ is directly density-reachable from $\boldsymbol{x}'$. In other words, directly density-reachable is symmetric for pairs of core points. A point $\boldsymbol{x}'$ is density-reachable from a point $\boldsymbol{x}$ if there are a set of core points leading from $\boldsymbol{x}$ to $\boldsymbol{x}'$. Density-reachable is a canonical extension of direct density-reachability but, unlike direct density-reachable, is not symmetric for pairs of core points. A point $\boldsymbol{x}'$ is density-connected to a point

$\boldsymbol{x}$ if there are a core point $\boldsymbol{o}$, such that both $\boldsymbol{x}'$ and $\boldsymbol{x}$ are density-reachable from $\boldsymbol{o}$. Density-connectivity is a symmetric relation.

A cluster C is defined as a subset $C \subseteq \mathcal{T}$ satisfying the following "maximality" and "connectivity" requirements. A point $\boldsymbol{x}'$ is in C if $\boldsymbol{x}'$ is density-reachable from $\boldsymbol{x}$ and $\boldsymbol{x}$ is a core point in C. Also, all points $\boldsymbol{x}, \boldsymbol{x}' \in C$ are density-connected. Finally, if $C_1, \ldots, C_K$ are clusters of the dataset $\mathcal{T}$ with respect to parameters ϵ_k and $MinP_k;\ k = 1, \ldots, K$, noise is the set of points in $\mathcal{T}$ not belonging to any C_k. That is,

$$noise = \{\boldsymbol{x} \in \mathcal{T} \mid \forall k : \boldsymbol{x} \notin C_k\}. \tag{1.66}$$

As mentioned earlier, clusters depend on the parameters ϵ and $MinP$. It is recommended (Birant & Kut, 2007) that the value of $MinP$ should be chosen at least 3. Having chosen $MinP$, Ester et al. (1996) suggests the value for ϵ be chosen by using a δ-distance graph that plots the ordered distances of every point in $\mathcal{T}$ from its $\delta = MinP$ nearest neighbour in descending order. The best values of ϵ are where this plot shows a strong bend. Given parameters ϵ and $MinP$, the DB-SCAN algorithm constructs a cluster by first arbitrarily choosing a point $\boldsymbol{x}$ and finding all points that are density-reachable from $\boldsymbol{x}$. This yields a cluster if $\boldsymbol{x}$ is a core point. On the other hand, if $\boldsymbol{x}$ is a border point, then by definition no points are density-reachable from $\boldsymbol{x}$ and the algorithm considers some other point in $\mathcal{T}$.

This section covered the theory for the classification methods that will be used in subsequent chapters. Next, measures for evaluating performance of the classification model will be introduced, following which common feature selection methods for identifying the predictor variables within a set that are most appropriate for predicting

the outcome will be covered.

1.4 Performance Metrics

Let the random variable Y_0 be the true (unknown) value of a categorical response for given predictor variables $\boldsymbol{X} = \boldsymbol{x}_0$ and let $\hat{f}(\boldsymbol{x}_0)$ denote the prediction by a ML model estimated using a training dataset $\mathcal{T}$. A typical choice for measuring the error in predicting $Y_0 = y_0$ using $\hat{f}(\boldsymbol{x}_0)$ is the zero-one loss function,

$$\mathbb{I}(\hat{f}(\boldsymbol{x}_0) \neq y_0) = \begin{cases} 1 \text{ if } \hat{f}(\boldsymbol{x}_0) \neq y_0, \text{ (prediction incorrect);} \\ 0 \text{ if } \hat{f}(\boldsymbol{x}_0) = y_0, \text{ (prediction correct).} \end{cases}$$

Another option is the deviance loss function,

$$D(y_0, \hat{f}(\boldsymbol{x}_0)) = -2 \sum_{k=1}^{K} \mathbb{I}(Y_0 = k) \log \hat{p}_k(\boldsymbol{x}_0),$$

where $\hat{p}_k(\boldsymbol{x}_0)$ is the estimated probability that the observation belongs to class k, $k = 1, \ldots, K$. The zero-one loss function counts the number of incorrect predictions (misclassifications) whereas the deviance is based on the probability of belonging to the predicted class. If the learning method correctly predicts class probabilities with high levels of certainty, the deviance will be close to zero. Conversely, the deviance will be large if these probabilities are small.

The test error for a ML model, also called the generalisation error, is defined as the expectation of the loss function, given $\mathcal{T}$. The test error is therefore specific to the training set and so different datasets will lead to different test errors. Estimation of the test error is best achieved through use of a designated test dataset. This approach is

therefore adopted in this thesis. Alternatively, if a test dataset is unavailable, the test error may be estimated using cross-validation, or using resampling techniques such as bootstrap. Cross-validation methods hold out a subset of the training data from the model fitting process and then apply the fitted model to the held-out observations in order to estimate the error rates. In K-fold cross-validation the training data is randomly divided into K equal-sized sets, with $K-1$ of these sets combined and used for training and the "held-out" set used for estimating the test error. Each of the K sets is held-out once, resulting in K estimates of the test error which are then averaged to produce the final estimate. Note that each observation in the training dataset is used only once as a test dataset for estimating the test error.

A so-called "out-of-bag" estimate of test error can be obtained for ML models such as a random forest that uses bootstrap samples for training (see section 1.1.4). In particular, observations not in the bootstrap sample and thus not used for training the current model are used to estimate the test error. This test error estimation procedure capitalises on the fact that slightly less than two-thirds of the original training observations are expected to appear in a bootstrap sample. To see why this is, consider randomly sampling N observations with replacement from a training dataset of N observations. As the probability that a given observation is not selected at a draw is $1-\frac{1}{N}$ and the bootstrap sample consists N random draws, the probability that the observation is not in the bootstrap sample is $\left(1-\frac{1}{N}\right)^N$ and hence

$$P(\text{the observation } \in \text{ bootstrap sample}) = 1 - \left(1 - \frac{1}{N}\right)^N \approx 1 - e^{-1} = 0.632.$$

Error Rates

The approach adopted in this thesis for finding the best predictive model is to construct models using a selected set of different classification methods and to evaluate each model's performance based on the zero-one loss. A common measure of quality of the classification model for binary outcomes is the test misclassification error rate (ER) defined as

$$ER = \frac{FN + FP}{TP + FN + TN + FP} \tag{1.67}$$

where TP and TN are respectively the numbers of correctly predicted positive and negative outcomes in a test dataset, FP is the number of negative outcomes incorrectly predicted as positive and FN is the number of positive outcomes incorrectly predicted as negative. These counts are often displayed in a "confusion matrix" (Table 1.1). An overall measure of prediction accuracy is $AC = 1 - ER$.

Table 1.1: Confusion matrix. Counts of the observed and predicted outcomes in binary classification.

		Observed	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Positive Predictive Value

There are also a number of other associated measures for specific circumstances. Useful when the cost of false positives is high, precision or accuracy of positive predictions

is measured by the positive predictive value (PPV), and is the ratio of correctly predicted positive observations to the total predicted positives,

$$PPV = \frac{TP}{TP + FP}. \tag{1.68}$$

Sensitivity is defined as the ratio of correctly predicted positive observations to all observations in the actual class,

$$SN = \frac{TP}{TP + FN}. \tag{1.69}$$

Also known as the true positive rate or recall, SN measures the ability of the model to capture all the relevant instances of the positive class. The related false negative rate $1 - SN$ quantifies the positive outcomes that are incorrectly classified. Balancing PPV and SN using their harmonic mean is the $F1$ score,

$$F1 = 2\frac{PPV \times SN}{PPV + SN}.$$

Specificity, or true negative rate, is estimated as the ratio of correctly predicted negative observations to the total observations in the actual negative class,

$$SP = \frac{TN}{TN + FP},$$

and measures the ability of the model to correctly identify negative outcomes, while the false positive rate $1 - SP$ is the ratio of incorrectly predicted positive observations to the total observations in the actual negative class and measures the frequency of false alarms or instances when the model predicts positive when the actual class is

negative. Finally, performance can be measured by the area under a plot of the true positive rate against the false positive rate at various probability thresholds for predicting class membership. Known as Receiver Operating Characteristic (ROC), this plot displays the model's ability to distinguish between positive and negative outcomes as the threshold is varied.

1.5 Variable Selection

Variable (feature) selection involves identifying and prioritizing the most relevant and informative predictor variables in the training data to improve the machine learning model generalisation and to enhance interpretability. The various methods existing in the literature are commonly classified within the machine learning community as either wrappers, filters or embedded methods. Wrapper methods follow an iterative process of evaluating different subsets of features by training and testing a machine learning model. The evaluation is based on performance metrics such as goodness of fit to the training data and measures of prediction accuracy. Embedded methods incorporate feature selection within the process of fitting the ML model to the training data. On the other hand, filter methods assess the relevance of features independently of the machine learning model, using statistical measures such as correlation to evaluate the importance of each feature.

1.5.1 Wrappers

The key steps in a typical wrapper method are subset generation and model evaluation in an iterative process. Wrapper methods generate various subsets of the predictor

variables using different combinations from the training dataset. Each subset is then used to train and test the machine learning model, and performance is assessed using a predefined evaluation metric. The process is repeated for all the generated subsets, and the subset that maximizes the model's performance is selected.

A common example of a wrapper method is all-subsets or best-subset selection in which all possible subsets of the set of predictor variables are evaluated and the best-performing model is selected. Another is forward selection, which starts with an empty set of predictor variables and iteratively adds one feature at a time, selecting the one that improves the model's performance the most, until a stopping criterion is met. Stopping criteria include no further improvement in performance after adding predictor variables, a predefined number of predictor variables have been evaluated, and a computational budget (e.g., time or number of evaluations) is exhausted.

Conversely, backward elimination starts with all predictor variables and iteratively removes the predictor variable that least affects model performance based on a certain criterion (e.g. p-value or some other performance metric), until a stopping criterion is met. In a variant of backward elimination called recursive feature elimination, more than one predictor variable may be removed at each iteration. Combining forward and backward selection is sequential (stepwise) selection which starts either as in forward selection with no predictor variables, or with a randomly selected subset. At each iteration, the predictor variable that most improves model performance is added, if any, and the one that least affects model performance is removed, if any. This approach addresses the potential problem of eliminating predictor variables that may become relevant later on in the iterative process.

The Boruta algorithm (Kursa & Rudnicki, 2010) is a wrapper designed to identify all relevant predictor variables in a dataset, rather than just selecting a minimal set that results in the highest model performance. It is particularly useful for high-dimensional datasets where many predictor variables might have small, but important contributions to the outcome variable. Boruta operates by using a random forest classifier as the learning model and compares the importance of observed values of a predictor variable to the importance of randomly shuffled copies (called shadow features). The importance of each observed predictor variable is compared to the maximum importance of shadow features. A predictor variable is considered relevant if its importance is consistently higher than the importance of shadow features and irrelevant if its importance is consistently lower. This process ensures that only variables with statistically significant importance are selected.

Wrapper methods directly assess the impact of subsets of predictor variables on the model's performance, ensuring that the selected predictor variables contribute to improved model accuracy. They also account for interactions between predictor variables, capturing nuanced relationships that may be missed by other feature selection methods and the choice of evaluation metric can be tailored to the specific goals of the model. However, wrapper methods can be computationally intensive, especially in problems with a large number of predictor variables and complex models.

1.5.2 Embedded methods

Embedded methods select predictor variables during the model's training phase. Examples include regularisation methods such as ridge regression, lasso and elastic net

that shrink the estimates of less relevant predictor variables, effectively eliminating them from the model. In particular, addition of the ℓ_1 penalty term to the objective function in logistic regression encourages variable selection by forcing some parameter estimates to be exactly zero. This results in a sparse model, which means that only a subset of the predictor variables is considered in the final model, effectively selecting the most important. The ℓ_2 penalty discourages large parameter estimates which helps to prevent overfitting, and stabilises the model by spreading the influence of all predictor variables, rather than emphasizing a few. The combination of both ℓ_1 and ℓ_2 regularization allows the logistic elastic net model to benefit from the variable selection capability of lasso while still enjoying the stabilizing effects of ridge regression. Similarly, tree-based algorithms such as random forest prune less relevant predictor variables after evaluating their relevance during the tree-building process, while neural networks can incorporate techniques like penalising large weights, discouraging the model from relying heavily on specific predictor variables.

Embedded methods eliminate the need for separate variable selection algorithms, making the overall model development pipeline more efficient. They also automatically prevent overfitting if the chosen penalty parameter value is large enough, enhancing the model's ability to perform well on unseen data, but they are limited to the variable selection capabilities embedded in the chosen algorithm.

1.5.3 Filters

Unlike wrapper methods, which involve training a model multiple times to evaluate feature subsets, filters rely on statistical measures to assess the relevance of predictor

variables independently of the learning algorithm. Many filter methods rank predictor variables based on a specific scoring criterion. Examples include evaluating the correlation between predictor variables and selecting those with the highest relevance to the outcome, using the standard χ^2 test statistic to assess statistical significance of individual categorical predictor variables and selecting those with the most significant impact on the model, and quantifying the amount of information provided by a predictor variable regarding the outcome and dropping those features with low information. Note that the mutual information between two random variables X and Y is

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} \mathcal{P}(x,y) \log \left(\frac{\mathcal{P}(x,y)}{\mathcal{P}(x)\mathcal{P}(y)} \right),$$

where $\mathcal{P}(x,y)$ is the joint probability distribution of X and Y, and $\mathcal{P}(x)$, $\mathcal{P}(y)$ are the marginal probabilities.

Filter models are computationally efficient for high-dimensional datasets and model-agnostic, making them applicable to various algorithms, but they also tend to ignore interactions between predictor variables, which can lead to suboptimal models. Correlation-based filters assume that predictor variables are independent of each other and so are not multicollinear, which is unlikely to hold in real-world datasets and may lead to the selection of redundant predictor variables, creating problems of multicollinearity. Because filter methods do not consider the model's overall performance in the variable selection process, they may result in lower accuracy compared to wrapper or embedded methods.

1.6 Discussion

Key concerns when developing a machine learning method is its accuracy in predicting previously unseen observations and its interpretability. Deeply rooted in statistical and machine learning theory and practice is that all models are imperfect and there is no "best" learning method for a given problem. One model or method may work well on one dataset, but some other model or method may work better on a different but similar dataset and thus selecting a model and method for training the model is often very challenging. This thesis investigates this issue within the context of predicting peptide activity, a complex problem characterized by limited knowledge about its underlying structure and governing principles.

This chapter provides the underlying theory for five different methods used in classification problems, namely logistic regression and the related elastic net logistic regression, random forests, neural networks and support vector machines. With regards to the learning model, logistic regression requires parametric specification of the mathematical form of the model whereas random forests, neural networks and SVMs do not. In particular, logistic regression requires manually accounting for non-linear relationships between the predictor variables and outcome whereas random forests and neural networks can automatically model such relationships, as can support vector machines using the kernel "trick".

Because the logistic model is constrained, it is more resistant to overfitting, especially with small datasets. It is also more interpretable, meaning it is easier to understand how the model arrived at a decision. On the other hand, complex models like neural networks or SVMs may overfit on small datasets. Thus, the best model is often

dependent on the amount of data available: smaller datasets tend to favour simpler models with lower variance models to prevent overfitting. In contrast, larger datasets can support the training of complex models without overfitting and with sufficient data for complex models to learn the general underlying patterns for minimizing error and capturing complex patterns.

Neural networks require careful tuning of hyperparameters such as the learning rate, the number of layers, the number of neurons per layer, and the choice of activation functions. SVMs involve selecting the appropriate kernel function and optimizing the cost parameter. For random forests, determining the number of trees and their maximum depth is critical. This process of hyperparameter tuning plays a crucial role in optimizing model performance, as these choices significantly impact the model's ability to generalize to new data. However, it is often a complex and non-trivial task, requiring techniques like grid search, random search, or more advanced methods to explore the parameter space effectively.

This chapter describes a set of classification methods based on different perspectives for modelling the data. There is no "best" learning method for every problem because each method has its strengths and weaknesses, which depend on the nature of the data, the task at hand, the need for interpretability, the availability of computational resources, and many other factors. The choice of the best method is context-dependent, and often, multiple models must be tested and evaluated on the specific problem to find the most appropriate solution.

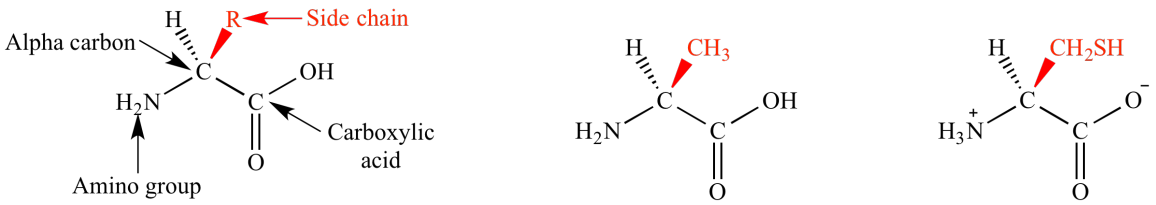
Chapter 2

Machine Learning and Antimicrobial Peptides

This chapter introduces the structure and properties of peptides and reviews the use of machine learning methods in predicting anti-microbial peptide activity. First, we describe the chemical structure of amino acids found in nature and the role they play in determining the physicochemical properties of peptides, which are essential in the immune system for defence against pathogens. Next, the nine physicochemical properties used in the Vishnepolsky et al. (2018) study are introduced in detail. Following this is an overview of the application of contemporary machine learning methods in studying peptide activity. Lastly in this chapter, the adaptation of DBSCAN by Vishnepolsky et al. (2018) to predicting regions of the predictor space with anti-microbial peptides (AMPs) active against the *Escherichia coli* (e-coli) bacteria is described in detail and applied to a larger dataset, to replicate the findings of the authors.

2.1 Peptides

Peptides are fundamental chains of organic molecules called amino acids, serving diverse functions within organisms. There are 20 standard amino acids found in nature and each is made up of a carboxyl group ($COOH$), an amino group (NH_2) and a hydrogen atom bonded to a tetrahedral carbon atom, designated the α-carbon, and a side chain, or R group, which is different for different amino acids. Illustrated in Figure 2.1 is the generic structure of an amino acid and two examples of naturally occurring amino acids with their side chains.



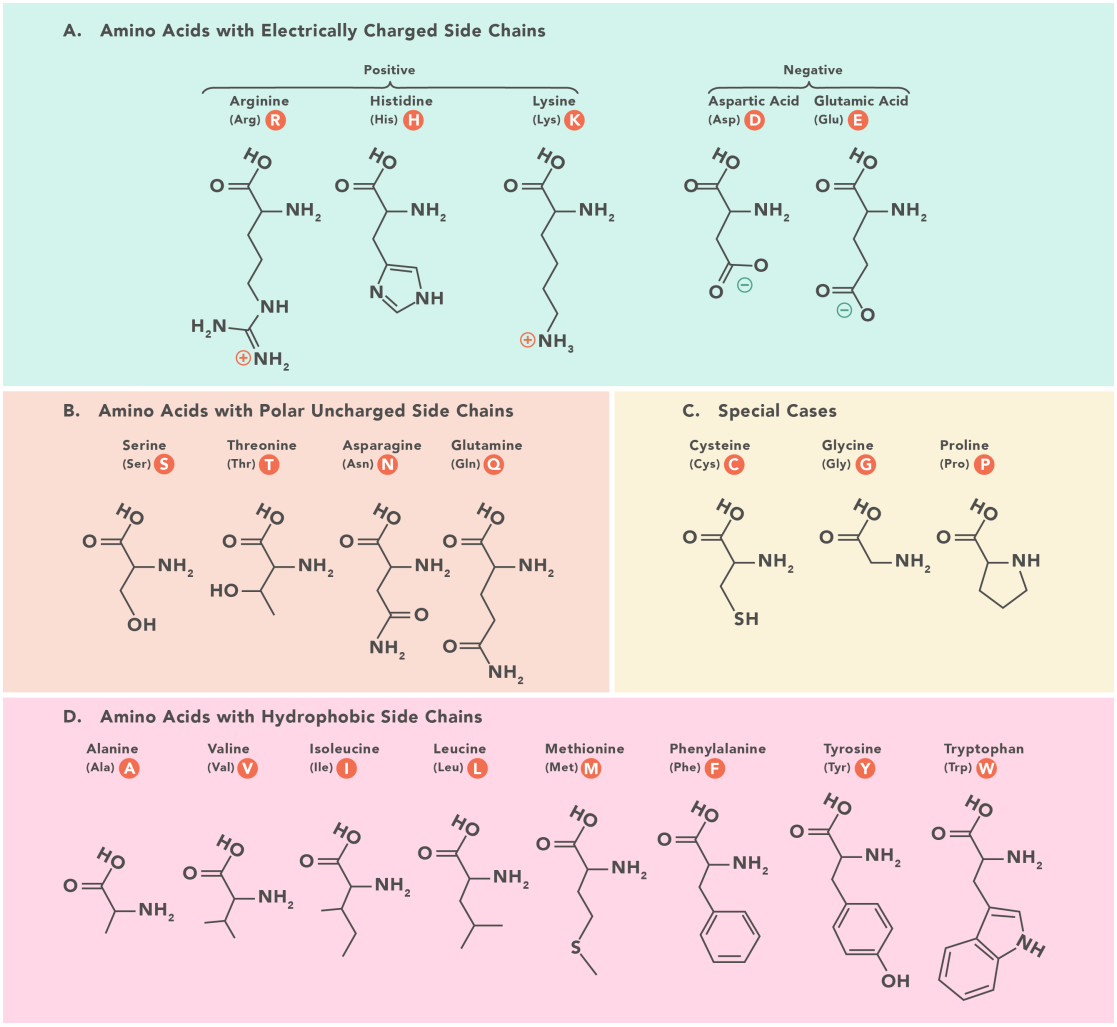
(a) Generic amino acid structure (b) Alanine (c) Cysteine

Figure 2.1: Structure of (a) generic amino acid, (b) alanine (Ala) and (c) cysteine (Cys), two naturally occurring amino acids. The side chain of Ala is CH_3, a methyl group, and for Cys it is CH_2SH. Taken from *https://www.chem.ucla.edu*.

Side chains differ in structure, electrical charge, and polarity (distribution of electric charge) and so are useful in categorising amino acids. For example, the acid/base properties of its side chain are used to classify an amino acid as either acidic, basic, or nonacidic, while the polarity and hydrogen bonding ability of its side chain determines whether or not an amino acid interacts with water. Amino acids that do not interact with water are classed as hydrophobic, whereas those that do are hydrophilic. The hydrophobicity of an amino acid is typically quantified by the energy associated with the amino acid being dissolved in water, or its transfer free energy. Figure 2.2 shows

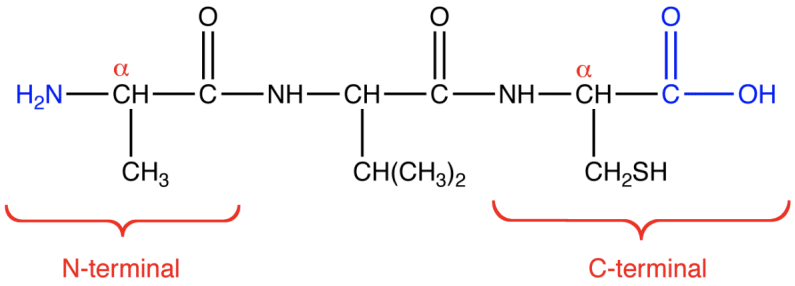
the 20 naturally occurring amino acids grouped by their side-chains.

Figure 2.2: Chemical structure of the 20 amino acids found in nature. Taken from *www.technologynetworks.com/applied-sciences*.



Peptides are formed when amino acids are linked together by peptide bonds, see Figure 2.3, which result from a condensation (dehydration) reaction between the carboxyl group of one amino acid and the amino group of another (Hamley, 2020). As illustrated in the figure, bonding of the amino acids leads to most peptides having a N-terminal or amine group, and a C-terminal or carboxyl group. The remains of

Figure 2.3: Schematic of a peptide, taken from *www.chem.libretexts.org*.



an amino acid after bonding is the residue. Unlike proteins, which are long chains of amino acid residues, the chains of amino acids forming peptides are typically short, ranging from two to fifty residues.

Peptides serve a myriad of functions in biological systems. Peptides act as signalling molecules, facilitating communication between cells and tissues. Examples include neuropeptides, which regulate neuronal function and neurotransmitter release, and peptide hormones like insulin, which control blood sugar levels. Certain peptides function as enzyme inhibitors or activators, modulating the activity of specific enzymes in metabolic pathways. For instance, protease inhibitors regulate the activity of proteolytic enzymes involved in protein degradation and processing. Peptides contribute to the structural integrity of tissues and organs. Collagen, a fibrous protein composed of peptide chains, provides strength and elasticity to connective tissues such as skin, tendons, and cartilage. Of interest in this thesis is the role of peptides in the immune system, in particular the defence against pathogens.

2.2 Physicochemical Properties of Peptides

Peptide versatility is a consequence of their structural and physicochemical diversity. The primary structure of a peptide refers to the linear sequence of amino acids. Its secondary structure, which includes α-helices, β-sheets, random coils or extended linear enriched with specific amino acids (Hamley, 2020), is determined by hydrogen bonding between amino acids in close proximity in the sequence. Tertiary and quaternary structures result from interactions between distant amino acid residues, leading to the formation of stable three-dimensional structures. The structure of a peptide influences its stability, binding affinity, physicochemical properties and biological activity. The models developed in this thesis use the same set of nine physicochemical and activity measures to classify AMPs as in the motivating paper (Vishnepolsky et al., 2018), namely normalized hydrophobicity (H), normalized hydrophobic moment (M), net charge (C), isoelectric point (I), penetration depth (D), tilt angle (O), propensity to disordering (R), linear moment (L) and propensity to aggregation (A). Vishnepolsky and Pirtskhalava (2014) provide detailed definitions of these nine features.

Hydrophobicity of a peptide is defined as the highest value of the hydrophobicities calculated for all fragments of a certain length along the peptide. A fragment's hydrophobicity is based on the sum of the transfer-free energies of its amino acids. Fragment lengths vary within the range of 4-50 amino acid residues. If the peptide length is less than the length of the considered fragment, hydrophobicity is computed for the full peptide. Hydrophobic moment μ_H of a peptide consisting of A amino

acids is defined as

$$\mu_H = \left\{ \left[\sum_{a=1}^{A} H_a sin\left(\delta a\right) \right]^2 + \left[\sum_{a=1}^{A} H_a cos\left(\delta a\right) \right]^2 \right\}^{1/2},$$

where H_a is the numerical hydrophobicity of the ath residue and δ is the angle of rotation of the residue along the helix axis (Hamley, 2020). Hydrophobic moment measures the extent of a peptide having both hydrophobic and hydrophilic regions (amphiphilicity) based on the periodicity of its secondary structure and the transfer-free energies of all residues (Hamley, 2020; Vishnepolsky & Pirtskhalava, 2014). Normalised values of the hydrophobicity (H) and hydrophobic moment (M) are used to construct the classification models.

Linear (hydrophobic) moment (L) is another quantitative characteristic for linear amphiphaticity developed by Vishnepolsky and Pirtskhalava (2014) that measures the degree of segregation of the hydrophobic and hydrophilic residues along the helix axis. In particular,

$$L = D\left(\sum H_a^+ - \sum H_a^-\right),$$

where

$$D = \left| \sum H_a^+ \cdot a \Big/ \sum H_a^+ - \sum H_a^- \cdot a \Big/ \sum H_a^- \right|,$$

is the distance between the centres of the hydrophobic and hydrophilic parts of the considered fragment of length A and $H_a^+,\ H_a^-;\ a = 1, \ldots, A$, are the transfer free energies of the ath residue from water to the hydrophobic environment under the conditions that $H_a^+ > 0$ corresponds to hydrophobic residue, and $H_a^- < 0$ corresponds to hydrophilic residue (Vishnepolsky & Pirtskhalava, 2014). The values of H, M and L used in the current work are determined by the Moon-Fleming hydrophobicity

scale (Moon & Fleming, 2011; Vishnepolsky et al., 2018).

The net charge (C) of a peptide at a given pH is the sum of the charges on all ionizable groups, that is, the side chains and carboxyl and amine groups, in the peptide. The charge on each group depends on whether its dissociation constant pK_a is smaller or larger than the pH value. The isoelectric point (I) is an associated measure corresponding to the pH at which the net charge of the peptide is zero (Hamley, 2020).

Penetration depth (D) and tilt angle (O) represent the location of a peptide relative to the membrane bilayer of a microbe. More specifically D measures the distance of the geometrical centre of the peptide from the membrane surface, and O is the angle between the peptide helical axis and the normal of the membrane surface. The values of D and O are defined using the depth-dependent potentials developed by Senes et al. (2007).

The final two physicochemical properties of peptides introduced here are propensity to disordering (R) and aggregation in solution, or in vitro aggregation, (A). Many short cationic peptides were experimentally shown to have disordered structures in water environments. Their propensity to disordering (R) is estimated using the Uversky's formula

$$R = 2.785 \langle H \rangle - 1.151 - \langle C \rangle ,$$

where $\langle H \rangle$ and $\langle C \rangle$ are the mean hydrophobicity and net charge of the peptide, respectively (Uversky et al., 2000). Disordered peptides have negative R values. Finally, the *in vitro* propensity to aggregation (A) is evaluated using the TANGO software developed by Fernandez-Escamilla et al. (2004) that predicts protein aggregation in

solution by considering only sequence-based structural parameters.

The first five descriptors can be conveniently applied to discriminate and predict the antimicrobial capability of peptides based on the common properties expected for AMPs, i.e., being cationic (C and I), hydrophobic (H) and amphiphilic (M and L), while the other four descriptors (D, O, R and A) can help deduce the mechanisms of action (Vishnepolsky & Pirtskhalava, 2014). It is noted that these descriptors are not fully independent of each other. For example, the net charge of a peptide C at a given pH is determined by the isoelectric point I, and the latter four features listed above could be closely connected to, or even determined by, the net charge, hydrophobicity and amphiphilicity of the peptides. This is relevant when constructing and evaluating the classification models.

2.3 Peptide Activity Studies

This section provides an overview of the application of contemporary machine learning (ML) methods in peptide activity studies. Acknowledging the extensive body of literature employing ML techniques and algorithms to comprehend peptide activity, provided here is a glimpse into the diverse application through a curated selection of publications.

Neural network models

The neural network method has been consistently and successfully used over the past two decades in predicting antimicrobial peptide activity. Cherkasov et al. (2008) used

this method to create antibiotic activity models based on small peptides (nine aa peptides) that are effective against a broad spectrum of antibiotic-resistant superbugs, to predict the activity of $100,000$ virtual peptides and novel antibacterial peptides. The authors claim their approach is effective for designing highly active broad-spectrum peptides smaller than those found in nature and addressed the lack of effective computational approaches for the rational design of antimicrobial peptides on a large scale.

Fjell et al. (2009) used the neural network method to identify antibacterial peptides based on synthetic peptides of nine aa length. The model, built using quantitative structure-activity relationships (QSAR) descriptors to predict and rank the relative activities of antibacterial peptides, was reported to have achieved 94% accuracy in identifying highly active peptides. The authors used the traditional principal component regression (PCR) and partial least squares regression (PLSR) data reduction methods to address the multicollinearity of some QSAR descriptors. From a total of 77 QSAR descriptors, the authors' approach reduced the number of predictor variables to 44 linear combinations of the descriptors.

Torrent et al. (2011) used data obtained from the CAMP (www.camp.bicnirrh.res.in) and Uniprot (www.uniprot.org) databases to build a neural network model to predict antimicrobial peptides and to assess their potency based on isoelectric point, length, α-helix propensity, β-sheet propensity, turn propensity, *in-vitro* and *in-vivo* aggregations, and hydrophobicity. The authors suggest that the predictor variables used are all important for classifying antimicrobial peptides (AMPs) and non-antimicrobial peptides (non-AMPs). In addition, they compared the neural network approach with a support vector machine approach and presented results suggesting that the former

approach is more accurate than the latter.

Mooney et al. (2013) built a neural network model to predict bioactive peptides within protein sequences based on protein having bioactive peptides of length $\leqslant$50 residues from the PeptideDB (www.peptides.be) databases. Their method classifies bioactive peptides into six groups: antifreeze, antimicrobial, cytokines and growth factor, peptide hormones, toxins and venom, and unique/other with an AUC of over 86%. Also, the authors built a model that can predict the bioactivity status (bioactive or non-bioactive) of small (15 aa length) peptides with an accuracy of about 82%.

Support vector machine models

The support vector machine (SVM) method has also been a widely used method for predicting antimicrobial peptide activity over the past two decades. Lata et al. (2010) used the SVM approach to develop the predictive model using antibacterial peptides from the APD database at the N and C terminus residues. The model was constructed based on peptide amino acid composition to predict and classify antibacterial and non-antibacterial peptides. The authors reported to have achieved 92.14% accuracy in identifying antibacterial peptides.

Porto et al. (2012) used the SVM approach with three kernels: linear, polynomial and radial kernels to construct prediction models for predicting antimicrobial activity for cysteine-stabilized peptides. The prediction models were developed based on nine structural/physicochemical properties such as average charge, average hydrophobicity, hydrophobic moment, amphipathicity, α-helix propensity, the flexibility of α-helix, indexes of α-helix, β-sheet and loop formation from the APD database and a subset

of the protein data bank (PDB). The APD database was used to generate the positive data set, while a subset of the PBD was used to generate the negative data set. Both datasets consist of sequences ranging from 16 to 90 amino acid residues. The authors suggest that the five sequence descriptors, indexes of α-helix, loop formation, average charge, average hydrophobicity and flexibility of α-helix are key factors for predicting antimicrobial activity. To simplify the analysis and reduce the dimensionality of these predictor variables, they used principal component analysis (PCA). Also, the authors reported the model from the polynomial and radial kernels achieved 90% accuracy, while the linear model achieved 89.33%.

Ng et al. (2015) used the SVM classification method to predict antimicrobial and non-antimicrobial peptides sequences. A pairwise (or one-vs-one) strategy was implemented to decompose the multi-class problem into multiple binary classification problems, while the LZ complexity algorithm (Lempel & Ziv, 1976), which measures the number of distinct substrings encountered as a sequence is parsed from left to right, was used for sequences that were not identifiable through the sequence alignment method. The authors constructed a predictive model using LIBSVM (a library for SVMs) and based on feature vectors from the CAMP (Collection of Anti-Microbial Peptides) database. Two training sets were used; one had normal sequences while the other comprised sequences with less than 70% similarity. Test datasets were created from the CAMP database and using data from Wang et al. (2011). The authors also evaluated the proposed algorithm's performance using a jackknife test on the two training datasets. The jackknife approach comprises systematically removing one observation at a time from the dataset and recomputing the statistic(sensitivity here). Sensitivity for normal sequences was reported as 95.28% based on jacknife and 87.59%

using the test data, while for sequences with less than 70% similarity, the jackknife and test data sensitivity estimates are 88.74% and 78.70% respectively.

Meher et al. (2017) used the SVM method to predict the propensity of a peptide sequence as antibacterial, antiviral and antifungal peptides using three different sample sizes: small (100 samples), medium (500 samples) and large (983 samples for bacterial, 738 samples for viral, and 1383 samples for fungal), in which the positive and negative sets contain an equal number of samples. The SVM prediction model for identifying antimicrobial peptides was developed using the compositional (consisting of amino acid composition (AAC) with 20 features, pseudo amino acid composition (PAAC) with 20 features and normalized amino acid composition (NAAC) with 20 features), physicochemical properties (consisting of hydrophobicity, net charge and isoelectric point) and structural features of the peptides (consisting of α-helix propensity, β-sheet propensity and turn propensity), derived from benchmark datasets. The authors used the radial kernel with hyperparameter γ equal to the reciprocal of the number of predictor variables, and cost C=1 when fitting the SVM model. They suggest that large sample sizes are more accurate for predicting peptide activity than small sample sizes. The authors reported accuracies of more than 90% for the prediction of AMPs, and that accuracies for predicting antibacterial and antifungal peptides were higher than those for antiviral peptides. They found that physicochemical and structural properties of peptides were more important in the prediction of antibacterial peptides than antiviral and antifungal peptides. Also, physicochemical properties were more important than structural properties in predicting peptide activity in general. Additionally, they discovered that net charge was the most important property in predicting antibacterial and antifungal peptides, followed by isoelectric point, while Cysteine (and

C) composition was the most important in predicting antiviral peptides. However, they reported the amino acids K (Lysine), P (Proline) and I (Isoleucine) were also important for predicting AMPs.

The final paper reviewed in this section, Lata et al. (2007), is one where both neural network and support vector machine methods were used to compare performances in predicting antibacterial peptides. Models were developed focusing on the residues at the N-terminus, C-terminus, and both N- and C-terminus from the antibacterial peptide database (APD). The authors reported accuracies of 83.63%, 87.85% and 84.78% for predicting antibacterial peptides using N-terminal residues for the neural network and SVM models and a method called quantitative matrices (QM) (Lata et al., 2007) which computes matrices of propensities of each residue at a particular position. The accuracy of prediction antibacterial peptides for C-terminal residues using the NN, QM, and SVM methods were 77.34%, 82.03%, and 85.16%, respectively. Finally, the accuracy of prediction antibacterial peptides for N- and C-terminal residues using the NN method is 81.17%, the QM method is 90.37%, and the SVM method is 92.11%. The authors suggest that the N- and C-terminal residues are important features of peptides, and the SVM method performs best in predicting antibacterial peptides, followed by the QM and NN methods.

Random forests models

The random forest (RF) method is also now frequently utilised to predict AMP activity. Maccari et al. (2013) used the random forest approach to predict the peptide α-helix structure and/or antimicrobial activity (a natural and a non-natural amino acid residue/an AMP and a non-AMP). Two predictive models were constructed based on

two different datasets of peptides called Dataset A and Dataset B, both characterized by quantitative structure-activity relationship (QSAR) descriptors. Dataset A contains the function properties for AMP activity consisting of AMPs with 11-40 amino acid residues from the YADAMP (www.yadamp.unisa.it) and CAMP databases (the positive dataset) and from the UniProt databases without antimicrobial annotation (the negative dataset). Dataset B contains the structural characteristics of α-helix AMPs using all-α helical peptide fragments (the positive dataset) and peptide fragments in all-*coli*, all-*sheet* and mixed conformation (the negative dataset) from the CB513 dataset, which is a well-defined non-redundant set of proteins. The authors reported that they have achieved 89.9% accuracy for Dataset A and 87.1% accuracy for Dataset B in predicting the peptide sequence based on the optimal features set. Additionally, their cluster analysis identified five different clusters; clusters 1 and 2 have a significant average net charge with a different distribution of the charged residues, while clusters 3, 4 and 5 have a lower net charge, indicating a higher incidence of negatively charged residues. The authors also designed novel AMP sequences (called *ab-initio* AMP) with non-natural amino acids based on the physicochemical properties of AMPs. Their results showed that using the physicochemical descriptors to analyze and predict non-natural amino acid insertions enhances flexibility in peptide design, and this approach successfully transformed a non-antimicrobial peptide (non-AMP) into a highly active AMP by using non-natural amino acids.

Bhadra et al. (2018) developed a highly accurate random forest classifier to predict AMPs by analyzing distribution patterns of physicochemical properties, which enables the design of novel peptide sequences with potential antimicrobial and therapeutic

effects. The prediction model was constructed based on the physicochemical properties using the positive dataset from the APD3 (https://aps.unmc.edu), CAMPR3 (www.camp3.bicnirrh.res.in), and LAMP databases and the negative dataset from the UniProt databases. The authors discovered that the prediction model can classify with high accuracy (96%) and found that charge was the key factor for antimicrobial activity. They also recommend exploring more modern ensemble learning techniques for training the classifier, as they feel that an advanced and intelligent feature selection strategy can help to improve the model.

Lira et al. (2013) used a tree model with nine nodes, eight predictor variables and ten leaves to represent the level of activity of synthetic peptides, and to predict antimicrobial activity and peptide construction for several therapeutic uses. They developed the model using the physical and chemical properties of antimicrobial peptides from the APD database. The eight predictor variables are net charge, hydrogen, oxygen, isoelectric point, $\log(P)$ of non-ionic species, ASA-P, balaban index and dreiding energy. The jackknife was used to validate the model, which has a classification accuracy of 0.70 for peptide activity. Peptide activity was classified into four categories: none, low, medium, and high.

Youmans et al. (2017) used the long short-term memory (LSTM) recurrent neural networks, random forests (RF) and k-nearest neighbours (KNN) methods to classify antibacterial and non-antibacterial peptides. The authors reported the features used in the LSTM model are simpler and easier to generate compared to the RF model. Moreover, both the LSTM and RF techniques performed better than the KNN classifier.

Logistic regression models

A recent example of use of logistic regression and the Boruta wrapper algorithm for feature selection is Clark et al. (2021), who synthesised peptides comprising all possible sequences up to seven aa length that can be produced from the positively charged amino acid Arginine (R) and the hydrophobic Tryptophan (W), see Figure 2.2, and assessed their inhibitory, microbiocidal and hemolytic activities against the gram-positive bacterium *Staphylococcus aureus*, the gram-negative bacterium *Pseudomonas aeruginosa*, and a yeast *Candida albicans*. Bayesian logistic regression models were fitted using 267 features determined from the peptide sequences.

Other methods

Xiao et al. (2013) used a fuzzy K-Nearest Neighbour (fKNN) method to build a prediction model based on the five physicochemical peptides; hydrophobicity, pK1 (C^{α}-$COOH$), pK2 ($NH3$), PI ($25°C$) and molecular weight, to identify AMPs and non-AMPs. The authors used the Jackknife test to estimate accuracy, reported as 86.32%, of detecting AMPs and non-AMPs. Wang P. et al. (2011) used KNN to predict antimicrobial peptides based on the physicochemical and biochemical properties of amino acids consisting of five features; codon diversity, electrostatic charge, molecular volume, polarity, and secondary structure. The predictive model was developed using an integrated method that combines sequence alignment with feature selection techniques. The authors performed the Jackknife test to determine the accuracy of predicting antimicrobial peptides, achieving an accuracy rate of 80.23%.

They also compared the model with the SVM, RF and discriminant analysis methods, and reported that the KNN method has a higher sensitivity value than each of these methods.

Vishnepolsky et al. (2014) developed a simple approach for discriminating between AMP and non-AMP and to predict the linear cationic antimicrobial peptides (LCAP) based on their membrane interaction properties. Their prediction model was developed using several key characteristics: hydrophobicity, amphiphaticity, the peptide's location relative to the membrane, charge density, propensities to disordered structure and aggregation. A threshold for each characteristic was determined by a point closest to the point (0,1) on the ROC curve of sensitivity against specificity and prediction of the existence of antimicrobial activity of the peptide was done based on these thresholds. The authors compared their approach with more complicated discrimination SVM, RF, NN and discriminant analysis techniques. Their approach uses the LCAP set selection from the APD2 database for the positive dataset, while the negative dataset is obtained from the UniProt database. The authors reported that three descriptors can be used as discriminators of LCAP prediction: hydrophobic moment, charge density, and peptide position along the membranes. They also claim that their approach is comparable to the CAMP prediction methods on the training set while demonstrating superior accuracy on the test set.

Khamis et al. (2015) used k-means clustering with Euclidean distance to classify AMPs into antimicrobial families using features selected by a genetic algorithm. The families and sub-families of peptides that have antimicrobial activity (positive dataset), which has 753 non-redundant natural mature peptides, were obtained from the DAMPD database. In contrast, the negative dataset has 288 peptides obtained

from the UniProt database. The authors claim that they can classify AMPs into 14 families and sub-families.

Jenssen et al. (2008) used principal component analysis (PCA) and the partial least squares (PLS) to analyze the structure-activity relationship of the peptides in the Bac034 and Bac2A libraries and to verify the value of contact energy and inductive and conventional QSAR descriptors, which has amino acid and charge as specific description values. The two models for peptide antibacterial action were reported to have predicted activity of 85% and 71% accuracy.

Machine learning has become a central tool for predicting the antimicrobial activity of peptides, with methods ranging from the traditional logistic regression to the more contemporary neural networks, random forests and support vector machines. Many models rely on features like amino acid composition, hydrophobicity, net charge, and secondary structure propensity, often achieving strong classification performance in distinguishing antimicrobial peptides (AMPs) from non-AMPs. Despite these advances, existing methods also highlight major gaps in our understanding of how peptides act against microbes. Models trained on general AMP datasets often fail to generalize across microbial species, reflecting how context-specific activity is poorly captured. Furthermore, current methods often capture correlations rather than underlying biological mechanisms and highlight a need for better integration of structural and biophysical information and interpretable modelling approaches to bridge prediction with true mechanistic understanding.

2.4 Density-Based Spatial Clustering and Peptide Activity

Presented in this section is a description of the use of DB-SCAN by Vishnepolsky et al. (2018) to cluster linear antimicrobial peptides active against the Gram-negative bacteria *Escherichia coli* ATCC 25922 and hence to predict the physicochemical properties of these peptides. Using the *dbscan( )* function of package *dbscan* (Hahsler et al., 2019) in R (R Core Team, 2021), clusters were constructed based on the nine physicochemical properties introduced in Section 2.1, that is normalized hydrophobic moment (M), normalized hydrophobicity (H), net charge (C), isoelectric point (I), penetration depth (D), tilt angle (O), propensity to disordering (R), linear moment (L) and propensity to aggregation (A). Next, the analysis is reproduced using a larger dataset obtained from the DBAASP (www.dbaasp.org) data repository in an effort to validate the findings of this paper.

2.4.1 Vishnepolsky's method

Using the DBAAP database, training and test data sets were created based on use of the minimum inhibitory concentration (MIC) criterion to decide whether or not a linear AMP is active against *E. coli* ATCC 25922. Active peptides had MIC<25 μg/ml and inactive peptides had MIC>100 μg/ml. In addition, all selected peptides were without intrachain bonds and without non-standard amino acids (or without modification). Two separate sets of clustering models were constructed, one set for peptides of sequence lengths 10-16 aa and the other set for sequences of length 18-27 aa. The training data set for each set of models consisted of a positive set of peptides

and a negative set. The positive set was of peptides active against *E. coli* ATCC 25922 and the negative set were of peptides inactive against the same bacteria. Altogether the full training dataset for the model for peptides of sequence lengths 10-16 aa had a total of 174 active peptides and the same number of inactive peptides. The full training dataset for sequence lengths 18-27 aa had 118 active and the same number of inactive peptides. Each full training set comprised data on the nine physicochemical properties listed above.

Instead of using the full training datasets when building the clustering models, 5 equally-sized subsets of the training data, called validation sets and denoted here by $\text{RTS}_k;\ k = 1, \ldots, 5$, were randomly selected from the training data using sampling without replacement. Thus, for peptide sequence lengths 10-16 aa, models were constructed using 140 active peptides and the same number of inactive peptides while models of peptides of sequence lengths 18-27 aa were constructed from 100 active peptides and the same number of inactive peptides. For peptide sequence lengths 10-16 aa, a test set was constructed using 34 active peptides and the same number of inactive peptides, while for peptide sequence lengths 18-27 aa, the test set had 18 active peptides and the same number of inactive peptides.

Clustering algorithms were run for each of the 511 possible combinations of the nine predictor variables in a training dataset. Also, instead of selecting a single set of values for the DB-SCAN parameters ϵ and for $MinP$, the authors used the range from 0.2 to 2.0 in steps of 0.2 for ϵ and 3 to 15 in steps of 1 for $MinP$, giving a total of 130 combinations of both parameters. Combined with the use of the 5 validation sets, this gave a total of 5×130 clustering models for a given combination of predictor variables and sequence length.

For a given combination of predictor variables and sequence length, the machine learning algorithm for identifying properties of peptides active against ATCC 25922 comprised three steps. First, for the k^{th} validation set RTS_k; $k = 1, \ldots, 5$, and combination of ϵ and for $MinP$, DB-SCAN was run using only active peptides and the training set of active peptides filtered to include only those clusters of size greater than 10 active peptides. Inactive peptides were then taken as belonging to a particular cluster if the values of their features were within an interval determined by the values of the features of the active peptide in the said cluster. A second filtering removed those clusters with less than 75% active peptides. In other words, the retained clusters satisfied a positive predicted value,

$$PPV = \frac{TP}{TP + FP} \geq 0.75,$$

where TP is the number of true positives, or active peptides, and FP is the number of false positives, or inactive peptides. The set of retained clusters over all combinations of ϵ and $MinP$ and using training set RTS_k is denoted here by $\{C_{ik}\}$ where $i = 1, \ldots,$ number of clusters with $PPV \geq 0.75$.

The second step is concerned with finding stable clusters in the five sets $\{C_{ik}\}$; $k = 1, \ldots, 5$. A cluster C_{ik} in set $\{C_{ik}\}$ is considered similar to a cluster $C_{j\ell}$ in set $\{C_{j\ell}\}$, $k \neq \ell$ if the number N_{ij} of peptides that fall in both clusters is large relative to the larger of the two clusters. In particular, cluster C_{ik} is similar to $C_{j\ell}$ $(k \neq \ell)$ if

$$S_{ij} = \frac{N_{ij}}{max(N_i, N_j)} > 0.85,$$

where N_i is the number of peptides in C_{ik} and N_j is the number of peptides in $C_{j\ell}$.

A cluster is stable if similar clusters are found in all five sets. The sets created at this stage are overlapping as they are created using different combinations of ϵ and $MinP$.

In the third and final step of the algorithm, the overlapped sets obtained in the second step are used to construct combinations of non-overlapping stable clusters (CSC). It is unclear from the paper how the combinations were constructed. Clarification was sought via personal communication with the authors but this did not help. In the paper, prediction of activity was done using physicochemical properties of the peptides in the CSC with the highest P_i, defined as

$$P_i = \sum_{j=1}^{\ell_i} (SN_{ij} PPV_{ij}), \tag{2.1}$$

where SN_{ij} and PPV_{ij} are the sensitivity and positive predictive value for the j^{th} cluster of the i^{th} CSC with ℓ_i clusters. A region of the feature space based on the means and standard deviations of the physicochemical characteristics of peptides forming this optimal cluster is constructed and a new peptide is predicted active if the values of its features fall within this region (personal communication with authors).

For peptides with sequence lengths 10-16 aa, Vishnepolsky et al. (2018) reported three clusters for prediction, with two in the space of 7 predictor variables and the third in the space of 3 predictor variables. The largest cluster, with 108 peptides, was found in the 7D space formed by hydrophobic moment (M), hydrophobicity (H), isoelectric point (I), tilt angle (O), linear moment (L), propensity to disordering (R) and propensity to aggregation (A). The second largest, with 21 peptides, was found in the space formed by M, H, I, O, R, A and penetration depth (D). The smallest

cluster, with 18 peptides, was found in the space formed by *I*, *O*, and *D*.

For peptides with sequence lengths 18-27 aa, Vishnepolsky et al. (2018) reported four clusters for prediction. The largest was found in the 6D space *M*, *I*, *D*, *O*, *L* and net charge (*C*). The next was found in 1D space *I*. The third was found in 2D space *I* and *A*. The final cluster was found in 4D space *M*, *I*, *R* and *A*.

Vishnepolsky et al. (2018) evaluated their model for predicting activity status of the shorter peptides using a test dataset of 34 active and 34 inactive peptides. The test set of longer peptides had 18 active and the same amount of inactive peptides. Table 2.1 shows the confusion matrices obtained based on results provided in the paper. The test error rates are calculated as respectively 14/68 or 20.6% for shorter peptides and 8/36 or 22.2% for the longer peptides, while the sensitivities are respectively 25/30 or 83.3% and 14/18 or 77.8%.

Table 2.1: Counts of actual and predicted activity status of peptides against *E coli* ATCC 25922 obtained using a density-based spatial clustering method (Vishnepolsky et al., 2018) and test datasets.

		10-16 aa length		18-27 aa length	
		Actual		Actual	
		Active	Inactive	Active	Inactive
Predicted	Active	25	5	14	4
	Inactive	9	29	4	14

2.4.2 A replication study

In an attempt to establish consistency of the above findings, the method described above is programmed in R, using the *dbscan* function for clustering and implemented using the same set of peptides as in Vishnepolsky et al. (Vishnepolsky et al., 2018). Thus the full datasets, downloaded from the DBAASP repository, consisted of 174 active peptides of 10-16 aa length, 118 active peptides of 18-27 aa length, a set of 173 inactive peptides of 10-16 aa length and 118 inactive peptides of 18-27 aa length. One inactive, 10-16 aa length peptide was not available at the site. As in the paper, a test set for peptide sequence lengths 10-16 aa was constructed from the full dataset using 34 active peptides and 33 inactive peptides, while the test set for 18-27 aa peptides had 18 active peptides and 18 inactive peptides.

Stable clusters were generated as in steps one and two of the Vishnepolsky et al. (Vishnepolsky et al., 2018) algorithm. As the third step of the Vishnepolsky et al. algorithm is vague, implemented here is a reasoned framework for constructing the prediction model. The set of stable clusters of peptides obtained over all combinations of DB-SCAN parameters ϵ and $MinP$ and over all combinations of predictor variables were first partitioned into subsets based on whether or not they shared peptides in common in such a way that every stable cluster in a subset shared peptides (overlap) with at least one other stable cluster in said subset and shared no peptide with clusters outside its subset. The partitioned subsets therefore define disjoint regions of the 9D feature space containing active peptides. Next for each overlapping subset of stable clusters and following Vishnepolsky et al. the cluster with the highest P_i, see equation (2.1), was selected for predicting activity status.

Results for replication study

ML model for 10-16 aa length peptides

Considering the analysis for peptides of 10-16 aa length, a total of 521 stable clusters forming two disjoint subsets of overlapping clusters were found. The first disjoint subset comprises only 2 clusters in the 4D space *MCIL* while the second had 519 distributed across 2D to 9D space. These findings are summarised in Table 2.2. The cluster with maximum P_i within each subset is selected for use in prediction. For the first subset, this cluster is clearly the one in the 4D space *MCIL*. Among all clusters in the second subset, the cluster in 6D space *HCIDOL* and in the 7D space *HCIDOLA* had joint maximum P_i of 1.51, and the cluster in 6D space is therefore selected, using the principle of parsimony. Unlike Vishnepolsky et al. who found a cluster with 108 peptides, clusters found here all have less than 20 peptides. Contrary to Vishnepolsky et al. (2018) who found stable clusters in the 3D space *IDO* and in 7D spaces *MHIORLA* and *MHIDORA*, net charge C is now found to be an important predictor variable for AMP activity status whereas propensity to disordering R is not. This is further explored in Table 2.3.

Table 2.3 shows the mean values and standard deviation of nine physicochemical properties for the optimized clusters for 10-16 aa peptides used for prediction compared with results of Vishnepolsky et al. (2018). As can be seen from the table, peptides from the re-analysis results have similar values as those found by Vishnepolsky et al. The two clusters in the re-analysis differed most strongly from each other by net charge (C), isoelectric point (I) and linear moment (L). While Vishnepolsky et al. reported that clusters differed most strongly from each other by I and tilt angle (O).

Table 2.2: Number (n) of stable clusters of peptides 10-16 aa length and properties of the cluster with maximum P_i within subspaces of the 9D physicochemical space.

All stable clusters		Properties of cluster with maximum P_i		
Dimension	n	P_i	Predictors	Size
			Subset 1	
4D	2	0.12	*MCIL*	14
			Subset 2	
2D	5	0.59	*MO*	17
3D	28	0.72	*HCL*	15
4D	73	1.20	*HCDL*	14
5D	127	1.34	*HCIOL*	15
6D	141	1.51	*HCIDOL*	15
7D	94	1.51	*HCIDOLA*	15
8D	40	1.12	*HCIDORLA*	15
9D	11	1.04	*MHCIDORLA*	15
Total	521			

ML model for 18-27 aa length peptides

Table 2.4 summarises the single set of 243 overlapping stable clusters found for the 18-27 aa length peptides, and the dimension of the spaces where these clusters were formed. The cluster of 88 peptides in 5D space defined by predictor variables *HIDOL* is selected for use in predicting AMP activity. This is in contrast to the four active regions found by Vishnepolsky et al. (2018) in 1D space I, 2D space IA, 4D space $MIRA$ and 6D space $MCIDOL$. The smallest of these clusters comprise 10 peptides while the largest had 41. Notable differences between the two analyses, apart from the number of active regions found, is that hydrophobicity (H) was not identified by Vishnepolsky et al. as an important classifier, whereas hydrophobic moment (M),

Table 2.3: Mean values and standard deviation of nine physicochemical properties for the optimized clusters for peptides of 10-16 aa length using Vishnepolsky's method. Mean values and standard deviation of the key physicochemical properties in the clusters are highlighted in blue.

	mean values ± SD of properties				
	Re-analysis results		Vishnepolsky et al. (2018) results		
	Cluster 1 (*MCIL*)	Cluster 2 (*HCIDOL*)	Cluster 1 (*MHIORLA*)	Cluster 2 (*IDO*)	Cluster 3 (*MHIDORA*)
$M \pm SD$	1.0 ± 0.1	1.4 ± 0.5	1.5 ± 0.6	0.6 ± 0.2	1.4 ± 0.2
$H \pm SD$	-0.7 ± 0.2	0.5 ± 0.3	-0.5 ± 0.7	-0.4 ± 0.5	-0.6 ± 0.4
$C \pm SD$	3.0 ± 0.0	8.4 ± 0.6	5.4 ± 2.1	4.9 ± 1.5	3.2 ± 1.7
$I \pm SD$	11.1 ± 0.2	12.5 ± 0.3	13.8 ± 0.5	12.7 ± 0.8	10.9 ± 0.4
$D \pm SD$	13.9 ± 2.3	16.4 ± 0.8	14.2 ± 4.5	17.6 ± 2.1	13.8 ± 1.2
$O \pm SD$	92.2 ± 18.8	94.7 ± 7.3	90.9 ± 15.6	162.6 ± 12.9	94.0 ± 9.8
$R \pm SD$	0.2 ± 0.2	-0.7 ± 0.2	-0.1 ± 0.5	-0.5 ± 0.3	0.3 ± 0.2
$L \pm SD$	0.3 ± 0.0	0.2 ± 0.0	0.3 ± 0.1	0.3 ± 0.1	0.3 ± 0.1
$A \pm SD$	6.3 ± 12.2	0.0 ± 0.0	2.1 ± 8.8	64.4 ± 157.8	5.7 ± 10.3

net charge (C), propensity to disordering (R) and propensity to aggregation (A) were not identified as important in the re-analysis.

Means and standard deviations of the physicochemical properties for the optimized clusters for 18-27 aa peptides in Table 2.5 show that the cluster found in the re-analysis have similar values to those found by Vishnepolsky et al. apart from propensity to aggregation A which is higher for the former. However, as A is not one of the predictor variables in the ML model from the re-analysis, but is closely linked to C, H and L, it is not obvious at this stage what differentiates the performances of the two sets of models in predicting activity status. This is further examined by evaluating the

Table 2.4: Number (n) of stable clusters of peptides 18-27 aa length and properties of the cluster with maximum P_i within subspaces of the 9D physicochemical space.

All stable clusters		Properties of cluster with maximum P_i		
Dimension	n	P_i	Predictors	Size
2D	15	2.06	*CL*	16
3D	67	8.81	*HIO*	94
4D	72	13.29	*HIDO*	89
5D	62	19.60	*HIDOL*	88
6D	27	6.51	*HIDORL*	89
Total	243			

models on the test datasets.

Evaluation on test sets

The test sets used in the replication analysis here comprise the same peptides as used in Vishnepolsky et al. (2018), but the values of the physicochemical properties can be different as the DBAASP database was updated since publication of their paper (personal communication). The test set data for 10-16 aa length peptides, consisting of 34 active and 33 inactive peptides, are used to estimate the error rates of the re-analysis model using the values coloured blue in Table 2.3 to define active regions. These values are appropriate for defining the active regions, as the use of mean and standard deviation are consistent with the previous analysis by Vishnepolsky et al., to define active regions. Error rates are also obtained using the Vishnepolsky et al. classification model. The results reported here are for active regions defined as three standard deviations from the mean. It is unclear how many standard deviations were used in the original paper. Here, three standard deviations provided a good balance

Table 2.5: Mean values and standard deviation of nine physicochemical properties for the optimized clusters for peptides of 18-27 aa length using Vishnepolsky's method. Mean values and standard deviation of the key physicochemical properties in the clusters are highlighted in blue.

	mean values ± SD of properties				
	Re-analysis results	Vishnepolsky et al. (2018) results			
	Cluster 1 (*HIDOL*)	Cluster 1 (*MCIDOL*)	Cluster 2 (*I*)	Cluster 3 (*IA*)	Cluster 4 (*MIRA*)
$M \pm SD$	1.1 ± 0.4	1.2 ± 0.2	0.9 ± 0.4	1.2 ± 0.5	0.5 ± 0.1
$H \pm SD$	-0.3 ± 0.4	-0.4 ± 0.4	-0.4 ± 0.6	-0.3 ± 0.5	-0.7 ± 0.3
$C \pm SD$	5.3 ± 2.4	4.2 ± 1.2	7.9 ± 1.6	7.2 ± 2.6	6.1 ± 1.4
$I \pm SD$	11.7 ± 0.7	11.4 ± 0.5	12.6 ± 0.2	14.0 ± 0.0	11.7 ± 0.2
$D \pm SD$	14.4 ± 1.9	13.8 ± 0.8	20.4 ± 7.3	15.4 ± 6.1	13.6 ± 4.6
$O \pm SD$	86.5 ± 10.8	88.3 ± 5.8	91.3 ± 30.3	88.5 ± 22.6	75.1 ± 22.7
$R \pm SD$	0.0 ± 0.3	0.1 ± 0.2	-0.4 ± 0.3	-0.1 ± 0.4	0.0 ± 0.1
$L \pm SD$	0.2 ± 0.1	0.2 ± 0.1	0.3 ± 0.1	0.2 ± 0.1	0.4 ± 0.1
$A \pm SD$	51.4 ± 166.4	40.1 ± 161.2	23.5 ± 46.9	10.1 ± 24.4	31.3 ± 26.9

between sensitivity (SN) on the one hand and positive predictive value (PPV) on the other. A similar approach was adopted for 18-27 aa length peptides using a test data set of 18 active and 18 inactive peptides and with active regions defined by the mean and standard deviations coloured blue in Table 2.5.

Results of prediction on the test set of peptides of 10-16 aa length are provided in Table 2.6. However, the sensitivity, positive predictive value and error rates for the Vishnepolsky et al. model in Table 2.6 were calculated using the test sets from the replication study. The two models demonstrate similar error rates but they differ in terms of sensitivity and positive predictive value, with the Vishnepolsky et al. model being more likely to correctly predict active peptides. Performances for both

Table 2.6: Vishnepolsky method performance metrics calculated using replication study and Vishnepolsky et al. models and a test set of peptides of 10-16 aa length.

Clusters	TP	$TP+FN$	FP	$TN+FP$	SN	PPV	ER
Re-analysis model							
Cluster 1 (*MCIL*)	2	34	1	33			
Cluster 2 (*HCIDOL*)	4	34	0	33			
All clusters	6	34	1	33	0.18	0.86	0.43
Vishnepolsky et al. model							
Cluster 1 (*MHIORLA*)	9	34	3	33			
Cluster 2 (*IDO*)	1	34	2	33			
Cluster 3 (*MHIDORA*)	9	34	5	33			
All clusters	19	34	10	33	0.56	0.66	0.37

models here are not as good as in the original paper where the positive predictive value (0.83) was similar, but sensitivity was higher (0.73) and error rate was lower (0.20). Better performances were recorded for the 18-27 aa length peptides, compared with the shorter peptides. As Table 2.7 shows, the two models demonstrate similar sensitivities but they now differ in terms of error rates, with the Vishnepolsky et al. model being more likely to correctly predict peptide activity status. Also, the Vishnepolsky et al. model re-evaluation here is consistent with the metrics obtained in the original paper, where both PPV and SN were 0.78 and the overall error rate was 0.22.

Additional comments on replication study findings

For 10-16 aa peptides, the largest stable clusters found contain 17 peptides and are in 2D and 3D space. Compared with Vishnepolsky et al., who found a cluster with 108 peptides for peptides of 10-16 aa length, this lack of stable clusters with a large number

Table 2.7: Vishnepolsky's method performance metrics calculated using replication study and Vishnepolsky et al. models and a test set of peptides of 18-27 aa length.

Clusters	TP	$TP+FN$	FP	$TN+FP$	SN	PPV	ER
Re-analysis model							
Cluster 1 (*HIDOL*)	16	18	11	18			
All clusters	16	18	11	18	0.89	0.59	0.36
Vishnepolsky et al. model							
Cluster 1 (*MCIDOL*)	9	18	4	18			
Cluster 2 (I)	7	18	1	18			
Cluster 3 (*IA*)	0	18	0	18			
Cluster 4 (*MIRA*)	0	18	0	18			
All clusters	16	18	5	18	0.89	0.76	0.19

of peptides is likely to have negatively impacted the re-analysis model's performance. Conversely, for 18-27 aa peptides, the largest stable clusters found contain 95 peptides and are in 4D space. In fact, 13 stable clusters that have between 87 to 95 peptides were found, two of which are in 3D space, three in 4D space, five in 5D space and three in 6D space. In contrast, Vishnepolsky et al. largest cluster had 41 peptides.

Information obtained in the replication study provides insight into the properties that are relevant for predicting active peptides. Table B.1 in Appendix B lists the predictor variables forming the sub-spaces where stable clusters of active 10-16 aa peptides were found. Comparing physicochemical properties comprising the spaces, it can be seen that net charge (C) is present in the majority of (77) of the 89 subspaces with stable clusters, followed by hydrophobicity (H) which is present in 71. Linear moment (L) found in 63 subspaces, and tilt angle (O) and propensity to disordering (R), found in 52, are the next most common predictor variables. These are followed by penetration

depth (D) and isoelectric point (I), found in 50 and 49 subspaces, respectively. On the other hand, aggregation (A) and hydrophobic moment (M) are present in less than half the subspaces.

Table B.2 lists the 30 subspaces where stable clusters of active 18-27 aa peptides were found. Isoelectric point I is present in the majority (25) of these subspaces, O with 22 occurrences is next, followed by L, H and D present in 17, 16 and 15 subspaces, respectively. Aggregation A, which is in the subspaces containing Clusters 3 and 4 in Table 2.7, is present in only 5 of the subspaces found in the re-analysis.

While the findings of the replication study do not support the consistency of Vishnepolsky's method, evidence is nevertheless provided that this method is potentially promising. To investigate whether the possible lack of consistency can be attributed to the relatively small datasets used to train the ML model, the method is next implemented on larger training datasets. It should also be noted that a substantial number of peptides have been added to the DBAASP repository since the publication of the Vishnepolsky et al. manuscript.

2.4.3 Vishnepolsky method with larger training datasets

Downloaded from DBAASP was a set of 898 active peptides of 10-16 aa length, 508 active peptides of 18-27 aa length, a set of 644 inactive peptides of 10-16 aa length and 406 inactive peptides of 18-27 aa length. These numbers were obtained after removal of duplicates. Next, following the approach in Vishnepolsky et al. (2018) of balancing the active and inactive peptides, the same number of active and inactive peptides were randomly selected for both 10-16 aa length and 18-27 aa length. Consequently,

644 active and 644 inactive 10-16 aa peptides, and 406 active and the same number of inactive 18-27 aa peptides were available to the study. Test sets were constructed using a random selection of 20% of the available data. Thus for 10-16 aa lengths, each of the five validation sets RTS_k; $k = 1, \ldots, 5$ in Vishnepolsky's method had 515 active peptides and the same number of inactive peptides in the training set. For 18-27 aa lengths, the number was 324 peptides each. The test set for peptide sequence lengths 10-16 aa comprised 129 active peptides and the same number of inactive peptides, while for lengths 18-27 aa, the test set had 82 active peptides and the same number of inactive peptides.

Vishnepolsky ML models trained on larger datasets

Three non-overlapping subsets of stable clusters were obtained for peptides of 10-16 aa length after partitioning the set of stable clusters. The cluster with maximum P_i in the first subset comprises 18 peptides in the 5D subspace *MHCDR*. The cluster with maximum P_i in the second had 15 peptides and was in the 5D subspace *MCIDO*, while the third, in the 3D subspace *CIR* had 17 peptides. As obtained in the replication and unlike Vishnepolsky's model, no large, active clusters of 10-16 aa length were found. All stable clusters found for 18-27 aa length peptides were overlapping and hence only one stable cluster is produced by the method for use in prediction. This cluster has 26 peptides and is in the *HCIOLA* subspace.

Table 2.8 shows the mean values and standard deviation of nine physicochemical properties for the optimized clusters for 10-16 and 18-27 aa lengths of peptides. The clusters of 10-16 aa length active peptides are seen here to differ most strongly from each other by C, and less so by I, which uses the minimum distance between clusters

Table 2.8: Mean values and standard deviation of nine physicochemical properties for the optimized clusters of peptides 10-16 and 18-27 aa lengths, found by Vishnepolsky's method and using larger training sets. Mean values and standard deviation of the key physicochemical properties in the clusters are highlighted in blue.

	mean values ± SD of properties			
	Results (10-16 aa length)			Results (18-27 aa length)
	Cluster 1 (*MHCDR*)	Cluster 2 (*MCIDO*)	Cluster 3 (*CIR*)	Cluster 1 (*HCIOLA*)
$M \pm SD$	1.62 ± 0.13	0.51 ± 0.14	0.65 ± 0.52	1.10 ± 0.32
$H \pm SD$	-1.20 ± 0.14	-0.45 ± 0.63	0.23 ± 0.42	-0.34 ± 0.18
$C \pm SD$	2.00 ± 0.00	4.50 ± 0.63	6.00 ± 0.00	3.50 ± 0.71
$I \pm SD$	10.99 ± 0.41	12.46 ± 0.34	12.68 ± 0.13	10.84 ± 0.33
$D \pm SD$	12.05 ± 0.69	19.75 ± 1.65	17.47 ± 4.00	13.69 ± 2.24
$O \pm SD$	84.60 ± 5.82	170.69 ± 4.21	76.53 ± 16.53	88.27 ± 6.91
$R \pm SD$	0.43 ± 0.07	-0.54 ± 0.28	-0.78 ± 0.07	0.16 ± 0.13
$L \pm SD$	0.33 ± 0.04	0.33 ± 0.06	0.22 ± 0.09	0.17 ± 0.02
$A \pm SD$	1.94 ± 2.69	49.15 ± 104.61	0.00 ± 0.00	4.60 ± 9.34

to quantify these differences. A comparison with the values provided in Tables 2.6 and 2.7 show that both sets of analyses identify clusters of active peptides with similar features, but defined in slightly different subspaces. For 10-16 aa peptides, propensity to aggregation A is again not found to be a relevant predictor variable for predicting activity status, but now propensity to disordering R is relevant, while L is not relevant. In the case of the larger peptides, M and D are no longer relevant but now, consistent with the Vishnepolsky et al. model, A is relevant.

Performance metrics for Vishnepolsky method using larger datasets.

Results of prediction on the test sets of peptides of 10-16 aa length and 18-27 aa length

Table 2.9: Results of prediction on the test set of peptides of 10-16 aa length and 18-27 aa length for large datasets.

Clusters	TP	$TP+FN$	FP	$TN+FP$	SN	PPV	ER
10-16 aa length							
Cluster 1 (*MHCDR*)	1	129	1	129			
Cluster 2 (*MCIDO*)	4	129	3	129			
Cluster 3 (*CIR*)	3	129	1	129			
All clusters	8	129	5	129	0.06	0.62	0.49
18-27 aa length							
Cluster 1 (*HCIOLA*)	6	82	2	82			
All clusters	6	82	2	82	0.07	0.75	0.48

are shown in Table 2.9. Sensitivity (SN) values of all clusters were calculated as 0.06 and 0.07, positive predictive values (PPV) were 0.62 and 0.75, and error rates were 0.49 and 0.48 for peptides of 10-16 and 18-27 aa length, respectively. These metrics are consistent with the replication study findings in Tables 2.6 and 2.7. Comparing the quality of the prediction here with Vishnepolsky et al. findings, it is clear that it was not possible to replicate the performances obtained.

Vishnepolsky's method uses an unsupervised clustering algorithm first to identify "regions of active peptides" in the physicochemical space of the predictors, then checks whether inactive peptides also fall into those same regions. So, essentially, the method is borrowing unsupervised structure to inform a supervised discrimination task. Clustering can reveal natural groupings of active peptides, which can reduce the problem from a "black-box classification" to something interpretable. Further, active peptides clustering tightly in certain regions suggest strong feature activity relationships that a classifier can exploit. However, biological data can be noisy, and

the regions found might consequently reflect artefacts of the data rather than true functional activity patterns. Additionally, Vishnepolsky's method does not optimise for separating active and inactive peptides but simply employs a post-hoc check, and so potentially underutilises the full information in the training set. A supervised method (logistic regression, random forest, support vector machines, etc.) would directly exploit all available information to build decision boundaries.

In summary, Vishnepolsky's approach is a sensible exploratory strategy but not the most direct or optimal way to solve a supervised problem. Clustering can highlight potential "active regions" in the physicochemical space, but since activity prediction is inherently a supervised task, the method risks underutilising valuable information in the training data.

2.5 Discussion

The structure of peptides is critical for their biological activity, such as antimicrobial capabilities. Peptides often adopt specific chemical structures such as α-helices or β-sheets, characteristic of naturally occurring amino acid sequences that are essential for effective interaction with microbial membranes. However, the amphipathic nature of peptides is particularly important for disrupting bacterial membranes. Machine learning (ML) approaches are increasingly being used to predict antimicrobial peptide (AMP) activity by learning the behaviour of peptide sequences and their corresponding antimicrobial activities via the large dataset. These ML approaches improve AMP activity identification by analyzing sequence patterns, structural features, and physicochemical properties with higher accuracy and efficiency.

This chapter provided the underlying structure and properties of peptides used in predicting anti-microbial peptide activity by describing the fundamentals of peptides in nature, including the chemical structures of peptides. The structure of a peptide refers to the sequence of amino acids, which influences its stability, binding affinity, physicochemical properties and biological activity. This thesis investigates nine physicochemical properties and biological activity to classify AMPs. These propertiesare normalized hydrophobicity (H), normalized hydrophobic moment (M), net charge (C), isoelectric point (I), penetration depth (D), tilt angle (O), propensity to disordering (R), linear moment (L) and propensity to aggregation (A), as inspired by Vishnepolsky et al. (2018).

This chapter also presented DB-SCAN approaches used by Vishnepolsky et al. (2018), in a replication study using the same set of peptides as in this paper, and a re-analysis with larger datasets. The results of the replication studies with small and large datasets were not consistent. For instance, for peptides of 10-16 aa length, not only were a different number of clusters found, (2 clusters for small datasets and 3 clusters for large datasets), but these were in different regions of the predictor space. While for peptides of 18-27 aa length, one active cluster each was found in the small and large datasets, and these were also in different regions of the predictor space. Also, while the cluster in the small dataset had 88 peptides, no large cluster was found in the large dataset (only 26 peptides).

Comparing the performance between the two datasets, large datasets exhibit a slightly higher error rate of 0.49 compared to 0.43 for small datasets for peptides of 10-16 aa length. The models for peptides of 18-27 aa length exhibited a higher error rate in large datasets (0.48) compared to small datasets (0.36). For large datasets comprising

peptides of 10-16 aa in length, the sensitivity and positive predictive values were 0.06 and 0.62, respectively. These values were lower than those observed in smaller datasets, with a sensitivity of 0.18 and a positive predictive value of 0.86. In contrast, for peptides 18-27 aa in length within large datasets, the sensitivity and positive predictive values were 0.07 and 0.75, respectively. The sensitivity was substantially lower than for smaller datasets (0.89), whereas the positive predictive value was higher than for small datasets (0.59). Overall, there is minimal difference in performances of models constructed using small and large datasets.

The DB-SCAN approach by Vishnepolsky et al. (2018), mentioned in subsection 2.4.1, implements an embedded variable selection method via finding the cluster with the highest P_i to predict peptide activity. For shorter peptides and the small dataset, only three features, net charge (C), isoelectric point (I), and linear moment (L) are common to the spaces of the two clusters. While for the large dataset, only one feature, net charge C is common to the spaces of the three clusters found. Thus there is evidence in favour of C being an important feature for predicting AMP activity in 10-16 aa length peptides, in keeping with current theoretical understanding. For peptides of 18-27 aa length four features, namely hydrophobicity (H), isoelectric point (I), tilt angle (O), and linear moment (L) are common to the spaces of the clusters found in the small and large datasets, but C is not.

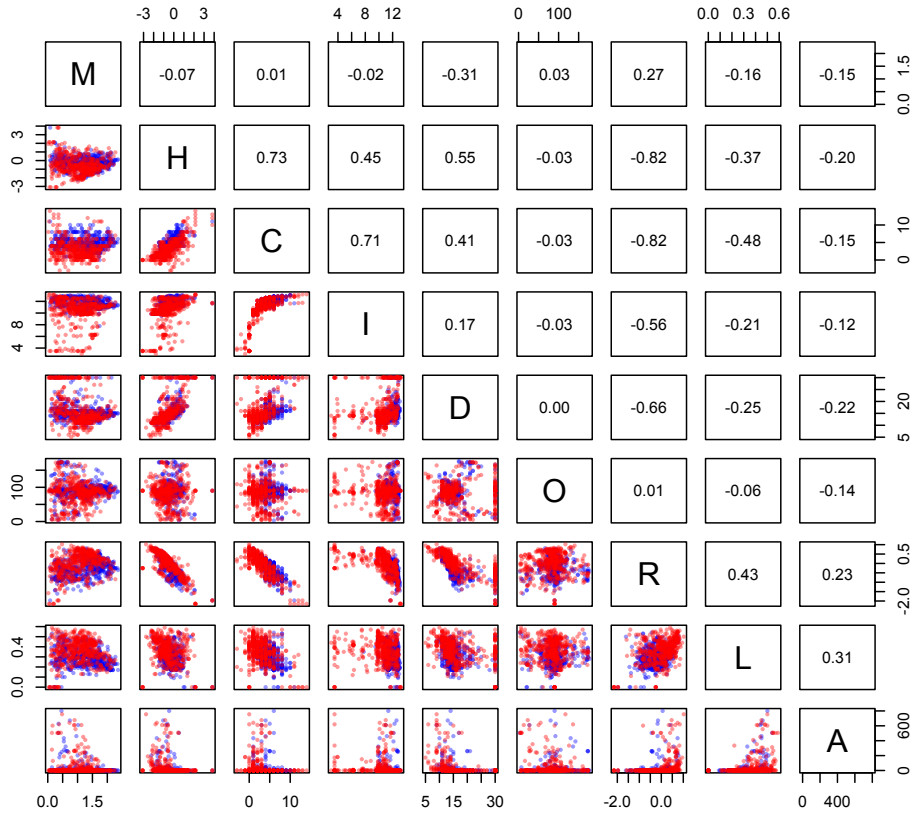


Figure 2.4: Matrix and correlation plots of physicochemical properties for peptides 10-16 aa length.

The findings of the replication study on both small and large datasets do not support the consistency of Vishnepolsky's approach. The misclassification error rates of 49% and 48% obtained for the shorter and longer peptides in larger datasets are much larger than the rates reported by Vishnepolsky et al. (2018), which were 20.6% and 22.2%, respectively. The overall poor performance observed here for the larger datasets is not surprising, given that Vishnepolsky's method found only a handful of small active clusters, thus providing a poor fit to the training data. Note that personal communication with the authors of Vishnepolsky et al. (2018) indicates that the results in the paper are not reproducible because they modified the prediction

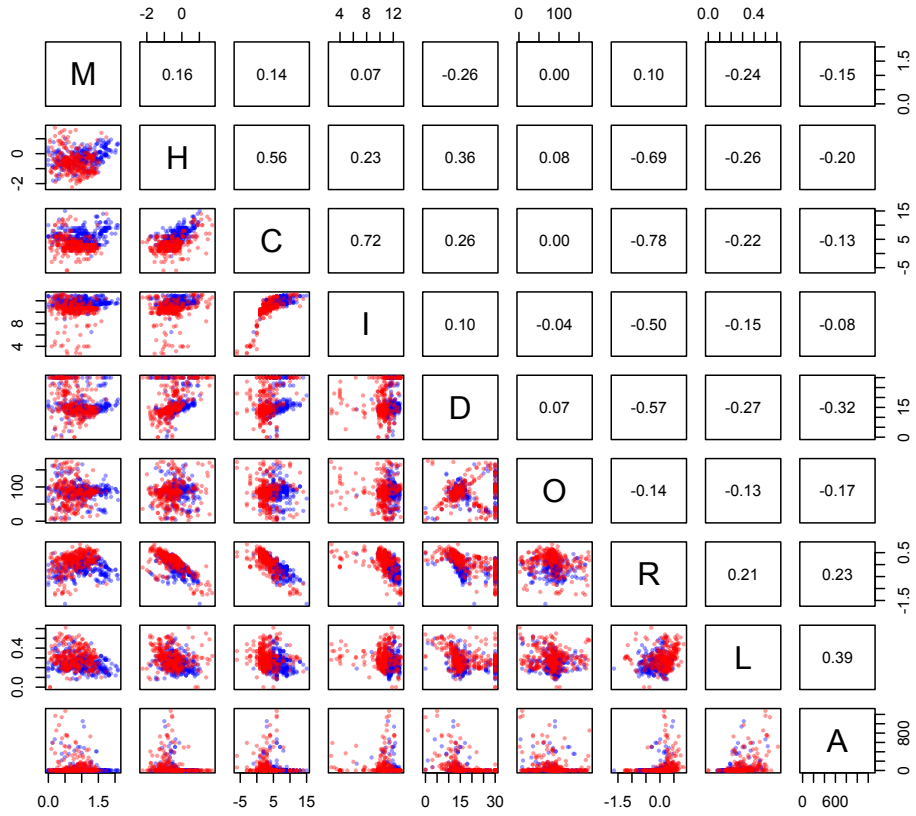


Figure 2.5: Matrix plot and correlation plot between the physicochemical properties for a raw peptide 18-27 aa length dataset.

algorithm in their database, and previously held only a small amount of data. The authors also claim that existing machine learning methods are inadequate for predicting peptide activity.

Matrix plots of predictor variables illustrating the relationships between physicochemical properties and activity status of peptides with active coded blue and inactive coded red are used to further explore the poor performances observed. From plots of 515 active and 515 inactive peptides for peptides 10-26 amino acid length, shown in Figure 2.4, it can be seen some pairs are related together, for example, H and C, and C and I are highly positively correlated. On the other hand, some pairs are highly

negatively correlated, such as H and R, and C and R. Plots for 324 active and 324 inactive peptides, 18-27 amino acid length are shown in Figure 2.5. Here, again it is clear that some pairs are related together, such as C and I, while some pairs are negatively correlated, for example, H and R, and C and R have correlation values. Based on theoretical reasoning and previous studies, high correlations are often anticipated between variables such as M and L, H and D, or C and I. However, the actual results observed can show quite a divergence from these expectations due to the complexity of biological systems, and multiple variables can influence and obscure these relationships. Here, correlation values between M and L are -0.16 for shorter peptides and -0.24 for longer peptides, and correlation values between H and D are 0.36 for longer peptides.

The matrix plots for each length peptide appear to be similar. More importantly, from these plots, it is clear that distinguishing between regions of active and inactive peptides is not straightforward. As demonstrated in this chapter, several researchers have used machine learning techniques to identify and analyze clusters of peptide activity, such as neural networks, support vector machines, random forests, and logistic regression methods. We next explore how well these methods work in predicting activity status of peptides active against Gram-negative bacteria *Escherichia coli* ATCC 25922.

Chapter 3

An Evaluation of Machine Learning Methods for AMP Activity

Five widely used classification methods-Logistic Regression (LR), Elastic Net (ENET) Logistic Regression, Support Vector Machines (SVM), Random Forest (RF), and Neural Network (NN)-are assessed here for their effectiveness in predicting antimicrobial peptide (AMP) activity. These methods offer diverse approaches to building machine learning (ML) models by employing distinct mathematical and algorithmic frameworks. In LR, the probability of an outcome (e.g., AMP activity status) is modelled as a linear function of predictor variables on the logit scale, with parameters estimated by maximizing the likelihood function of the observed data. ENET extends LR by incorporating regularization constraints ($\mathcal{L}_1$ and $\mathcal{L}_2$ penalties) during parameter optimisation to improve model generalizability and handle multicollinearity among predictor variables. SVM, in contrast, classifies data by constructing a hyperplane in a transformed predictor space, optimized to maximize the margin (distance) between classes (active vs. inactive peptides). RF builds classification models through an

ensemble of decision trees, where each tree partitions the predictor space into binary splits by optimizing measures of homogeneity, such as cross-entropy or the Gini Index, and aggregates their outputs for robust predictions. NN uses a network of interconnected layers to model complex relationships, where the outcome is represented as a weighted combination of the predictor variables, with weights iteratively adjusted to minimize an error metric (e.g., cross-entropy loss). The performance of these methods is evaluated in this chapter using both simulated datasets and comprehensive AMP datasets, derived from the DBAASP data repository and introduced in Chapter 2.

3.1 Simulation Evaluation of Five Common Classification Methods

The performance of the five machine learning (ML) methods is evaluated across four distinct scenarios, each reflecting different data distribution and sampling conditions. These scenarios are designed to illustrate the ML methods' behaviour under varying complexities and challenges inherent in real-world AMP data and to provide insight into how well the methods can capture the complex relationships between predictor variables and AMP activity status. This first scenario examines how the five ML methods perform when the training dataset consists of clearly separated active and inactive regions. This setting serves as a baseline to understand the methods' capabilities under ideal conditions with minimal overlap between classes. In the second scenario, the focus shifts to a more challenging situation where the predictor variables are subject to measurement inaccuracies, and the sampling density varies across the predictor space. Specifically, some inactive regions are sampled more densely than

others, potentially introducing biases and highlighting the methods' robustness to uneven sampling and noisy data. The third scenario builds on the second scenario by introducing variation in sampling density across both active and inactive regions. This setting tests the ability of the ML methods to handle uneven sampling distributions that affect all parts of the predictor space, simulating real-world complexities where both active and inactive regions may be unequally represented. Motivated by observed patterns in the relationship between activity status and key predictor variables hydrophobicity (H) and net charge (C), shown in Figures 2.4 and 2.5, the fourth scenario explores a data distribution reflecting these empirical patterns.

3.1.1 Data generation

The simulation study assumes four predictor variables, X_1, X_2, X_3, X_4, with values distributed within a four-dimensional unit hypercube (unit tesseract). Predictor variables are generated using Beta $B(\alpha, \beta)$ distributions,

$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1};\ 0 \leq x \leq 1,$$

encompassing the uniform distribution as a special case. The activity status Y of an observation is determined by whether it falls within a predefined region of the predictor space. Specifically, these regions are defined by ellipses in the space of X_1, X_2 while X_3, X_4 are unassociated with activity status. This approach provides a nonlinear relationship between Y and predictor variables X_1, X_2, X_3, X_4. The data simulated here are designed to serve as idealised representations of the patterns in the AMP datasets used in this thesis, illustrated in Figures 2.4 and 2.5. However, the

elliptical structure shown in Figure 3.1 is consistent with the empirical AMP datasets, most notably in the patterns observed between H and R and between C and R shown in Figure 2.5

As described above, the simulation evaluation explores four different data patterns, defined by varying the sampling density and distribution of the simulated data used for training the models. In keeping with Vishnepolsky et al. (2018) findings of more than one active region, the predictor space in the first three scenarios described above has two active regions defined by,

$$\frac{(X_1 - 0.10)^2}{0.03^2} + \frac{(X_2 - 0.85)^2}{0.10^2} \leq 1,$$

rotated $4\pi/3$ radians relative to the X_1 axis, and

$$\frac{(X_1 - 0.80)^2}{0.15^2} + \frac{(X_2 - 0.30)^2}{0.05^2} \leq 1, \tag{3.1}$$

rotated $\pi/3$ relative to the X_1 axis. See top right plot in Figure 3.1.

The predictor space in the fourth scenario has one active region defined by

$$\frac{(X_1 - 0.50)^2}{0.35^2} + \frac{(X_2 - 0.55)^2}{0.10^2} \leq 1,$$

rotated $\pi/3.5$, and one inactive region,

$$\frac{(X_1 - 0.40)^2}{0.40^2} + \frac{(X_2 - 0.25)^2}{0.72^2} \leq 1, \tag{3.2}$$

rotated $\pi/4$, both relative to the X_1 axis. See top right plot in Figure 3.2. In this

scenario, there is no distinct separation between active and inactive regions, reflecting a more complex and realistic situation where the mechanism of action for some observations is influenced by unobserved or latent factors. This overlap suggests that the observed features alone are insufficient to fully capture the underlying dynamics governing activity status. Such situations are likely in real-world AMP data, where hidden variables, noise, or interactions between physicochemical properties introduce ambiguity, complicating the classification task.

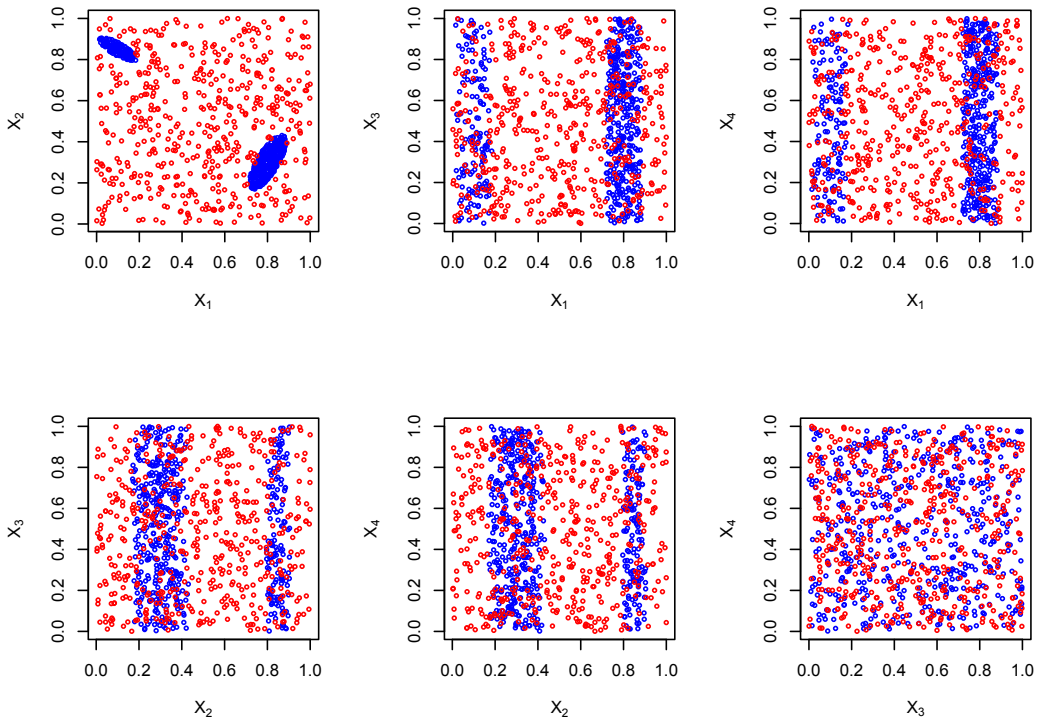


Figure 3.1: **Test dataset for Scenario 1**. Matrix plots of predictor variables X_1, X_2, X_3, X_4 for 500 active (blue) and 500 inactive (red) observations in the simulated test dataset for Scenario 1 described in the text. Variables X_1 and X_2 are associated with activity status whereas X_3 and X_4 are not.

Each training dataset is constructed to have 500 active and 500 inactive observations

by randomly generating 4-tuples X_1, X_2, X_3, X_4 and establishing whether or not they fall within an active region. Sizes of these datasets were guided by the number of peptides in the AMP dataset. Situations where predictor variables are inaccurately measured are evaluated by randomly perturbing observed values of X_1, X_2, X_3, X_4 in the training sets after activity status is assigned.

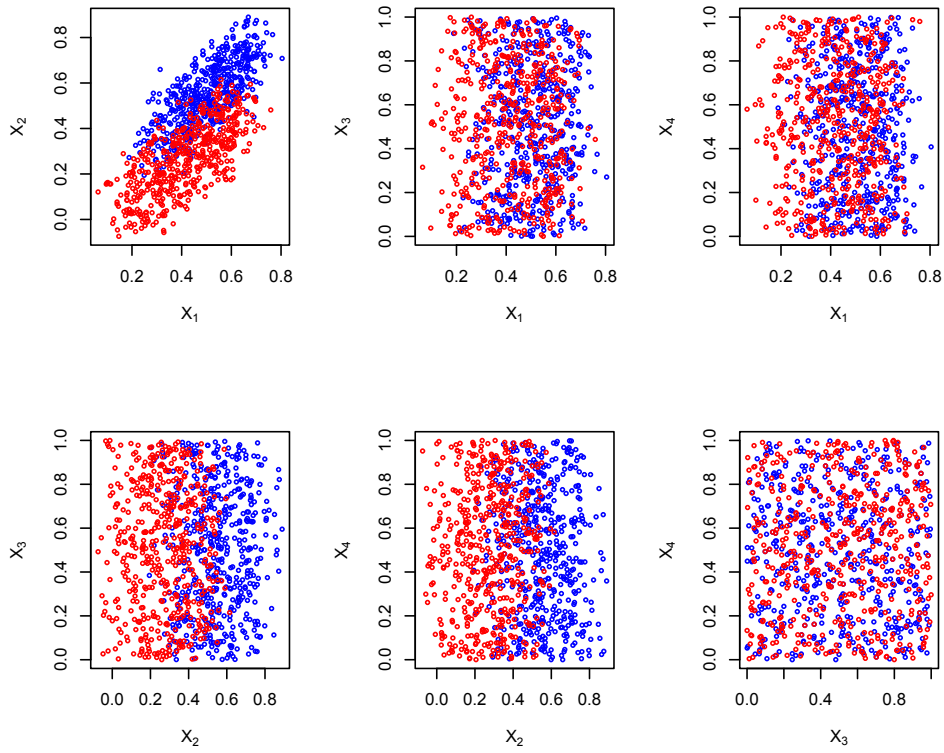


Figure 3.2: **Test dataset for Scenario 4**. Matrix plots of predictor variables X_1, X_2, X_3, X_4 for 500 active (blue) and 500 inactive (red) observations in the simulated test dataset for Scenario 4 described in the text. Variables X_1 and X_2 are associated with activity status whereas X_3 and X_4 are not. The overlap between active and inactive regions shown in the plots reflects a situation where the mechanism of action for some observations is influenced by unmeasured, or latent, factors.

Test datasets are generated to mirror the composition of the training datasets in terms

of X_1 and X_2, maintaining an equal number of active and inactive observations. This balance ensures consistency between the training and testing phases, enabling an unbiased evaluation of model performance.

Unlike the training data, where potential noise or errors may occasionally affect the predictor variables or activity status, the test datasets are constructed under idealized conditions, with no measurement errors in either the predictor variables or the activity labels, and with uniform sampling over the active and inactive regions. This design allows for an assessment of the model's predictive capabilities under optimal conditions, isolating the effects of model architecture and parameter tuning from confounding influences of data quality. A matrix plot of the simulated test dataset used in the first scenario is illustrated in Figure 3.1; the test sets used in the second and third scenarios are similarly distributed. The test set used in the fourth scenario is illustrated in Figure 3.2.

3.1.2 Model training

Polynomial models are used in the LR, ENET and NN methods to address the nonlinearity in the relationship between activity status and predictor variables. In particular, orthogonal polynomials (see Appendix A.2) are used to avoid multicollinearity problems in the model fitting as a result of the correlation between powers of a predictor variable. Preliminary investigations using simulated data for all four datasets indicated that polynomials of degree $d \leq 4$ were sufficient for the evaluation. Higher-degree polynomials, in contrast, cause the fitting algorithms to fail to converge. The SVM and RF methods inherently capture nonlinearity. So too does the

NN method through its layered architecture, but introducing polynomial variables to its input layer can enhance performance, particularly for NN models with one hidden layer (vanilla NN). Best subset variable selection, with misclassification error on a test dataset as evaluation criteria, is used to determine the best model for each ML method.

Logistic Regression

For $i = 1, \ldots, N$ let outcome,

$$Y_i = \begin{cases} 1 & \text{if active;} \\ 0 & \text{if inactive.} \end{cases} \tag{3.3}$$

Also let $\boldsymbol{x}_i = (x_{1i}, \ldots, x_{pi})$ be the observed predictor values for the d degree polynomial model, and denote the probability that $Y_i = 1$ conditional on $\boldsymbol{x}_i$ as $\mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta})$, where $\beta_0, \boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ are unknown model parameters. The logistic regression model

$$logit(\mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta})) = \beta_0 + \boldsymbol{x}_i\boldsymbol{\beta}, \tag{3.4}$$

is fitted to the training data using the $glm()$ function after computing the orthogonal polynomial basis matrix using the $polym()$ function (R Core Team, 2021). Denote the inverse logit of the fitted model as $\mathcal{P}\left(\boldsymbol{x}; \hat{\beta}_0, \hat{\boldsymbol{\beta}}\right)$, where $\hat{\beta}_0, \hat{\boldsymbol{\beta}}$ are estimates of $\beta_0, \boldsymbol{\beta}$. The logistic regression prediction for new observation $\boldsymbol{x}_0$ is

$$\hat{Y}_0 = \begin{cases} \text{active}, & \text{if } \mathcal{P}\left(\boldsymbol{x}_0; \hat{\beta}_0, \hat{\boldsymbol{\beta}}\right) > \frac{1}{2}; \\ \text{inactive}, & \text{otherwise.} \end{cases} \tag{3.5}$$

Elastic Net Regression

As outlined in Section 1.1.3 of Chapter 1, the elastic net regression model combines logistic regression with both lasso and ridge regularization. The model is here expressed as

$$logit(\mathcal{P}(\boldsymbol{x}_i; \beta_0, \boldsymbol{\beta})) = \beta_0 + \boldsymbol{x}_i\boldsymbol{\beta} - \lambda \left(\frac{\alpha}{2} \sum_{j=1}^{p} \beta_j^2 + (1-\alpha) \sum_{j=1}^{p} |\beta_j| \right), \tag{3.6}$$

where $\boldsymbol{x}_i$ is the vector obtained in the orthogonal polynomial transformation described above. The *train*() function from the *caret* (Kuhn, 2008) package is used to set up a grid for the hyperparameters α and λ, and to fit models over this grid using the *glmnet*() function (Tay et al., 2023). The model to be selected for prediction has the lowest misclassification error, estimated via 5-fold cross-validation on the training dataset. The elastic net prediction for new observation $\boldsymbol{x}_0$ follows the above approach for logistic regression.

Support Vector Machine

The separating boundary for the fitted support vector machine here is

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{N} \hat{\lambda}_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + \hat{\beta}_0, \tag{3.7}$$

where, for $i = 1, \ldots, N$,

$$y_i = \begin{cases} 1 & \text{if active;} \\ -1 & \text{if inactive,} \end{cases} \tag{3.8}$$

the parameters λ_i are model coefficients and

$$K(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{x}_i\|^2\right)$$

is the radial kernel, where γ is a positive constant. This kernel fits an implicit infinite dimensional polynomial; details of the support vector machine are provided in Section 1.1.6 of Chapter 1. The $svm()$ function, in conjunction with the $tune.svm()$ function from the $e1071$ package (Meyer et al., 2024), is used to fit models over a 100×100 grid of γ and cost (C) values ranging from 0.1 to 10, and to construct the final predictive model by selecting the hyperparameter combination that yielded the lowest misclassification error. Similar to the approach used for ENET, misclassification error in the model training process is estimated using 5-fold cross-validation on the training dataset. The prediction for new observation $\boldsymbol{x}_0$ is

$$\hat{Y}_0 = \begin{cases} \text{active}, & \text{if } \hat{f}(\boldsymbol{x}_0) > 0; \\ \text{inactive}, & \text{otherwise.} \end{cases} \tag{3.9}$$

Random Forest

The $randomForest()$ function from the $randomForest$ package (Liaw & Wiener, 2002) is used to construct the predictive model of the RF method. The prediction for new observation $\boldsymbol{x}_0$ is the most common prediction among 500 trees, each grown to their maximum possible size. Trees are constructed using the default settings of the $randomForest()$ function. Here, default settings are appropriate because the algorithm is inherently robust, the defaults strike a balance between variance reduction and tree strength, and extensive testing has shown these values to work well across many problems. And also optimizing the overall default parameter is computionary very demanding. Both foundational theory and empirical evidence support the appropriateness of default random forest parameters (for example, Breiman, 2001). Thus trees are constructed using bootstrap samples of the same size as the training data

while at each binary split within a tree, two of the four variables (X_1, X_2, X_3, X_4) are randomly sampled as candidates for splitting.

Neural Network

The *neuralnet*() function of the *neuralnet* package (Fritsch et al., 2019) is used to construct the predictive neural network (NN) models. Sigmoid activation and output functions, with predictor variables orthogonal polynomial transformations of (X_1, X_2, X_3, X_4), are used in constructing the models. Based on preliminary investigations, a single hidden layer with the number of neurons equal to $0.6p$, where p represents the number of predictor variables at the input layer, is selected for the simulation study. This configuration is found to be sufficient for achieving reliable results in the simulated environment, striking a balance between model complexity and computational efficiency. Furthermore, default settings of the *neuralnet*() function are robust, empirically validated starting points that balance stability, speed, and accuracy (for example, Weerts et al., 2020). Note that a more rigorous approach is employed to determine the optimal architecture of the neural network models for the real data analyses in this thesis. Specifically, the number of hidden layers and the number of neurons within each layer are chosen based on a systematic evaluation using 5-fold cross-validation to assess the misclassification error across a predefined grid of possible configurations.

3.1.3 Results

Following the best subset selection approach, model fitting is performed for each of the 15 possible combinations of predictor variables X_1, X_2, X_3, X_4, excluding cases where only a single predictor variable was used (resulting in 11 distinct cases). As mentioned earlier, to capture the nonlinear relationship, orthogonal polynomials of varying degrees are employed in models fitted by the logistic regression (LR), logistic elastic net regression (ENET) and neural network (NN) methods. These polynomials are chosen to provide a flexible representation of the data while preserving orthogonality, and minimizing multicollinearity issues. Initial results reveal that increasing the degree of the polynomial beyond $d = 4$ does not lead to improved error rates. For exsample, using the LR method with $d = 5$ in Scenario 4, the best model in the space defined by X_2 and X_3 shows an error rate of 0.105, and at least 2 models failed to converge, including the best model. In fact, use of higher-degree polynomials frequently leads to convergence issues, potentially due to overfitting. Thus, reported here are the results for quadratic ($d = 2$), cubic ($d = 3$), and quartic ($d = 4$) polynomials.

Scenario 1

In this scenario, the training dataset is generated uniformly across the space defined by X_1, X_2, X_3 and X_4. As described in Section 3.1.1, the active outcomes are associated with specific values of X_1 and X_2, determined by the elliptic functions described by equations (3.1). These functions define regions of activity, while the inactive outcomes are distributed outside these elliptical regions. Importantly, a deliberate buffer zone, an area of free space, is introduced between the active and inactive regions, as

depicted in Figure 3.3. This design choice allows investigation of the learning methods' ability to discern and accurately classify active and inactive peptides within the buffer zone. By constructing training data with clear separation between active and inactive regions, a controlled scenario to evaluate the methods' capacity to generalize and handle transitional or ambiguous regions in the predictor space is created. This setup ensures that the methods' performances are tested not only within well-defined active and inactive zones but also in intermediate spaces where classification boundaries are less explicit.

Models are fitted to the training data displayed in Figure 3.3 using the techniques outlined in Section 3.1.2 and evaluated against the test dataset illustrated in Figure 3.1. The misclassification test error rates and misclassification rate standard error for all models fitted for each of the five methods evaluated are summarized in Tables 3.1 and 3.2, respectively. For each of the eleven combinations of methods and polynomial models assessed, the best-performing models, determined by the lowest misclassification error rates (highlighted in blue in the table), consistently include the variables X_1 and X_2. Notably, six of these models, LR (d=2), ENET (d=2), SVM, RF, NN (d=3) and NN (d=4), exclusively use X_1 and X_2. In contrast, the poorest-performing models fail to include either X_1 or X_2. This is consistent with the data generation.

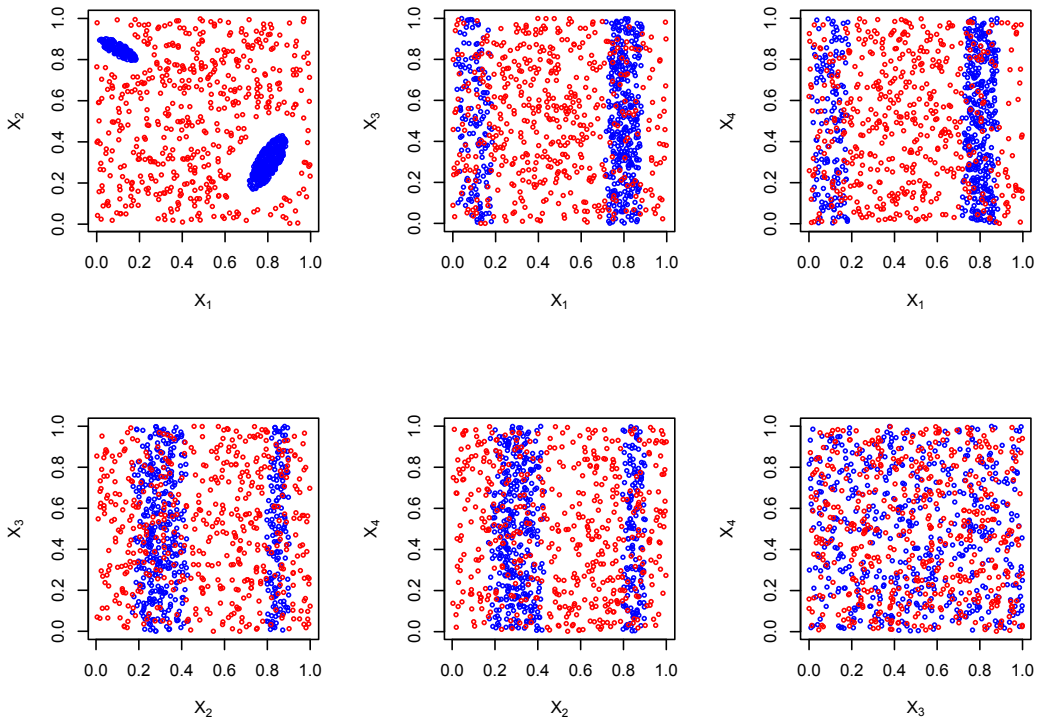


Figure 3.3: **Training dataset for Scenario 1**. Matrix plots of predictor variables X_1, X_2, X_3, X_4 for 500 active (blue) and 500 inactive (red) observations in the simulated training dataset for Scenario 1 described in the text. Variables X_1 and X_2 are associated with activity status whereas X_3 and X_4 are not. Observations are uniformly sampled over the active and inactive regions, but with separation between the two regions.

Table 3.1: Scenario 1 misclassification test error rates for all fitted models. Rates for the top-performing models for each method and polynomial degree, where relevant, are highlighted in blue.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.147	0.150	0.088	0.094	0.158	0.042	0.041	0.022	0.033	0.032	0.033
X_1, X_3	0.310	0.272	0.182	0.296	0.253	0.158	0.187	0.172	0.171	0.255	0.207
X_1, X_4	0.305	0.270	0.175	0.303	0.253	0.170	0.176	0.158	0.163	0.199	0.219
X_2, X_3	0.425	0.338	0.200	0.424	0.281	0.194	0.205	0.208	0.185	0.206	0.226
X_2, X_4	0.436	0.333	0.203	0.424	0.281	0.200	0.204	0.205	0.193	0.198	0.247
X_3, X_4	0.481	0.510	0.507	0.484	0.520	0.506	0.511	0.488	0.483	0.507	0.467
X_1, X_2, X_3	0.151	0.149	0.087	0.100	0.132	0.041	0.043	0.026	0.028	0.035	0.033
X_1, X_2, X_4	0.152	0.148	0.093	0.094	0.156	0.039	0.042	0.025	0.023	0.034	0.033
X_1, X_3, X_4	0.312	0.272	0.184	0.305	0.253	0.158	0.203	0.157	0.181	0.260	0.217
X_2, X_3, X_4	0.435	0.331	0.200	0.424	0.281	0.198	0.221	0.204	0.226	0.275	0.278
X_1, X_2, X_3, X_4	0.148	0.150	0.095	0.122	0.130	0.041	0.051	0.022	0.032	0.035	0.033

Table 3.2: Scenario 1 Misclassification rate standard error for all fitted models, based on 100 replications.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.011	0.011	0.009	0.018	0.073	0.007	0.006	0.005	0.007	0.005	0.007
X_1, X_3	0.020	0.015	0.012	0.020	0.019	0.015	0.012	0.012	0.016	0.020	0.022
X_1, X_4	0.020	0.015	0.011	0.020	0.019	0.014	0.012	0.011	0.015	0.021	0.020
X_2, X_3	0.023	0.018	0.014	0.024	0.018	0.017	0.013	0.012	0.017	0.018	0.024
X_2, X_4	0.023	0.018	0.013	0.022	0.018	0.016	0.012	0.012	0.013	0.016	0.023
X_3, X_4	0.015	0.014	0.016	0.015	0.014	0.017	0.015	0.015	0.016	0.013	0.014
X_1, X_2, X_3	0.011	0.011	0.008	0.015	0.072	0.007	0.007	0.005	0.006	0.006	0.007
X_1, X_2, X_4	0.010	0.011	0.008	0.015	0.068	0.007	0.008	0.005	0.006	0.006	0.007
X_1, X_3, X_4	0.020	0.014	0.013	0.021	0.018	0.014	0.012	0.012	0.015	0.017	0.016
X_2, X_3, X_4	0.023	0.018	0.014	0.023	0.018	0.016	0.013	0.012	0.029	0.020	0.020
X_1, X_2, X_3, X_4	0.011	0.011	0.010	0.014	0.063	0.007	0.007	0.005	0.006	0.007	0.006

With regard to the polynomial model degree, the trends are less uniform. While quartic polynomial models outperform quadratic and cubic models for the LR and ENET methods, quadratic models yield superior results when the NN method is employed. This suggests that the optimal polynomial degree may depend on the specific fitting method used, rather than following a universal trend. For example, the inherently more complex and flexible NN method does not require the high-degree polynomials needed for the LR method. When comparing the methods, it is evident that the ENET regularization of LR enhances performance relative to unregularized LR. However, despite these improvements, ENET's error rates remain higher than those observed for SVM, RF, and NN. These findings emphasize the advantages of more flexible methods like SVM, RF, and NN in scenarios with non-linear classification boundaries.

Next, the top-performing models are further evaluated using the performance metrics detailed in Section 1.4. The parsimony principle is used in cases where there is a tie for best model. The results of this evaluation are summarized in Table 3.3. The findings reveal high sensitivity rates (SN) and a substantial number of true positives (TP) across all methods, with the exception of ENET ($d = 3$), indicating that most models are highly effective in accurately predicting active outcomes. Additionally, low false positive (FP) counts and correspondingly high specificity rates (SP) are observed for the majority of models, although LR and the related ENET models exhibit weaker performance in this regard. Overall, the methods and models demonstrate consistently strong performance across the evaluated metrics.

Table 3.3: **Scenario 1**. Performance measures for the optimal models identified using the best subset selection method, for each of the eleven combinations of methods and polynomial models assessed.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=2)	0	500	147	353	1.00	0.71	0.85	0.77	0.147	X_1X_2
LR (d=3)	3	497	145	355	0.99	0.71	0.85	0.77	0.148	$X_1X_2X_4$
LR (d=4)	0	500	87	413	1.00	0.83	0.91	0.85	0.087	$X_1X_2X_3$
ENET (d=2)	7	493	87	413	0.99	0.83	0.91	0.85	0.094	X_1X_2
ENET (d=3)	92	408	38	462	0.82	0.92	0.87	0.91	0.130	$X_1X_2X_3X_4$
ENET (d=4)	0	500	39	461	1.00	0.92	0.96	0.93	0.039	$X_1X_2X_4$
SVM	0	500	41	459	1.00	0.92	0.96	0.92	0.041	X_1X_2
RF	0	500	22	478	1.00	0.96	0.98	0.96	0.022	X_1X_2
NN (d=2)	0	500	23	477	1.00	0.95	0.98	0.96	0.023	$X_1X_2X_4$
NN (d=3)	0	500	32	468	1.00	0.94	0.97	0.94	0.032	X_1X_2
NN (d=4)	0	500	33	467	1.00	0.93	0.97	0.94	0.033	X_1X_2

Figure 3.4, which presents matrix scatterplots of the predictions from the five classification methods for the variables, X_1 and X_2, provides additional insight into the predictive capabilities of the methods and their sources of error. In these plots, correctly predicted positive (TP) and negative (TN) outcomes are represented by dark blue and red points, respectively. Negative outcomes incorrectly predicted as positive (FP) are shown in blue, while positive outcomes incorrectly predicted as negative (FN) are depicted in green. The plots demonstrate that ENET with a quartic polynomial ($d = 4$) model, SVM, RF and NN construct highly accurate prediction regions,

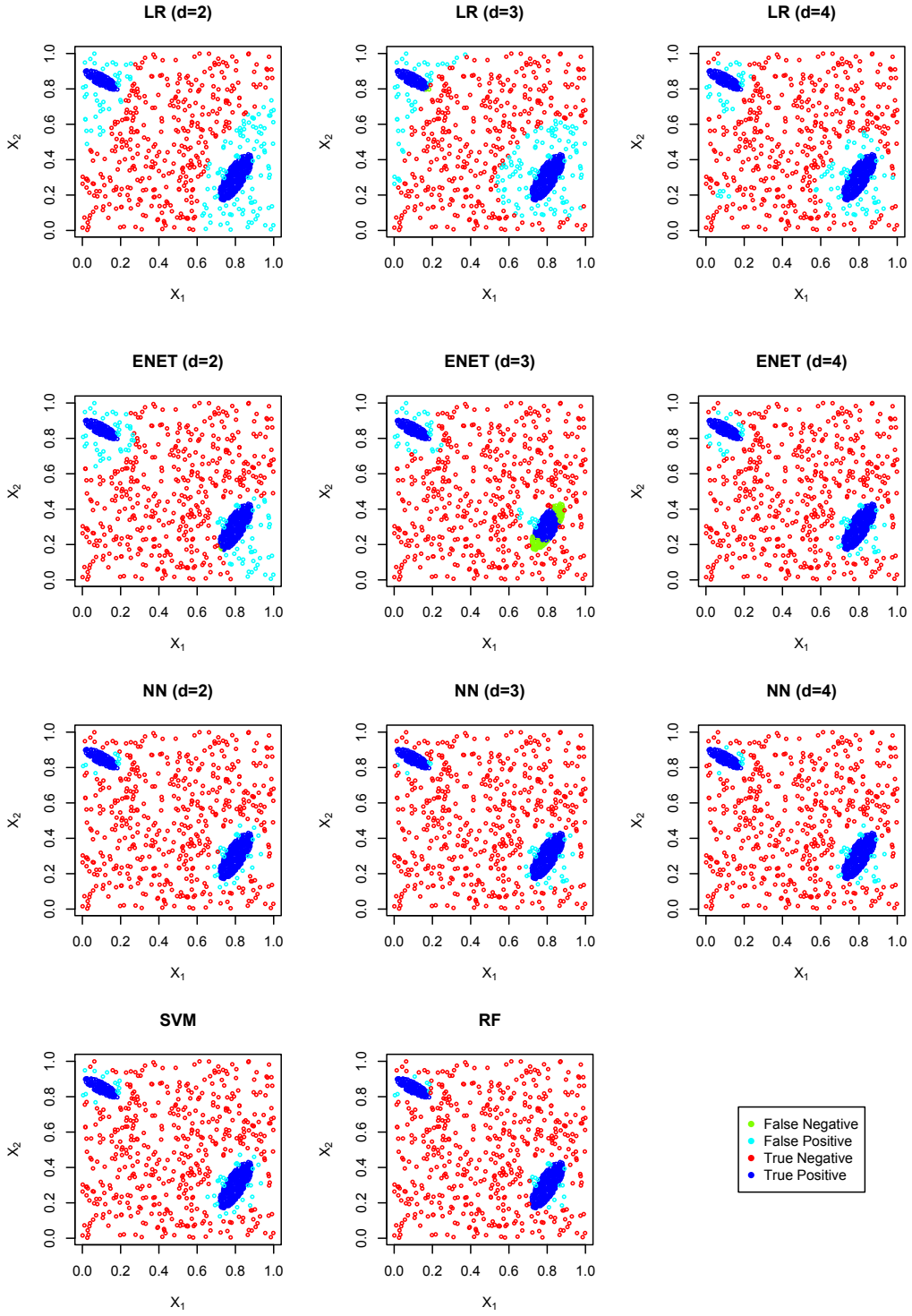


Figure 3.4: **Scenario 1**. Matrix scatterplots of predictions of activity status, by the five classification methods, for outcomes in the simulated test dataset, by variables X_1 and X_2.

effectively capturing both active and inactive outcomes for this scenario. In these cases, the decision boundaries are well-defined, leading to minimal misclassification errors observed and clear separation of active and inactive regions. In contrast, the plots also reveal that the LR method, when applied with quadratic and cubic polynomial models, struggles to construct the curved boundaries of the active regions. This is demonstrated by the number of False Positives shown in Figure 3.4 for the LR method.

Scenario 2

This scenario is designed to evaluate the performance of the methods under conditions where inaccuracies in the measurement of predictor variable values are present. Specifically, the predictor variables are subjected to different noise and distributional characteristics to simulate realistic challenges in data collection and measurement. Values for X_1 and X_2 are initially generated uniformly across the defined space to ensure an even spread. Next, to mimic measurement errors, the simulated values are perturbed by adding random noise sampled from a Normal $N(0, 0.01)$ distribution. This leads to training data with unclear boundaries between active and inactive outcomes. The variance used here (0.01) is of a similar order of magnitude reported in the literature for H, C and I (Audain et al., 2015; Gavva et al., 2023). Values for X_3 and X_4 are generated using Beta distributions, in particular $B(0.5, 0.5)$, which results in uneven sampling across the variable space. This choice reflects scenarios where certain ranges of these predictor variables are more densely represented than others, introducing additional complexity for the methods to handle. The true active regions and observations used for testing the models are simulated as in Scenario 1,

ensuring a consistent framework for comparison.

The process for fitting and evaluating models mirrors the approach used in Scenario 1. The misclassification test error rates and misclassification rate standard error for all fitted models, reported in Tables 3.4 and 3.5, respectively, are larger than the values obtained for Scenario 1, provided in Table 3.1. This consistent elevation in test error rates is not surprising and highlights the greater complexity and challenge of accurately predicting outcomes in the current scenario where there is reduced separability of the active and inactive classes in the training data.

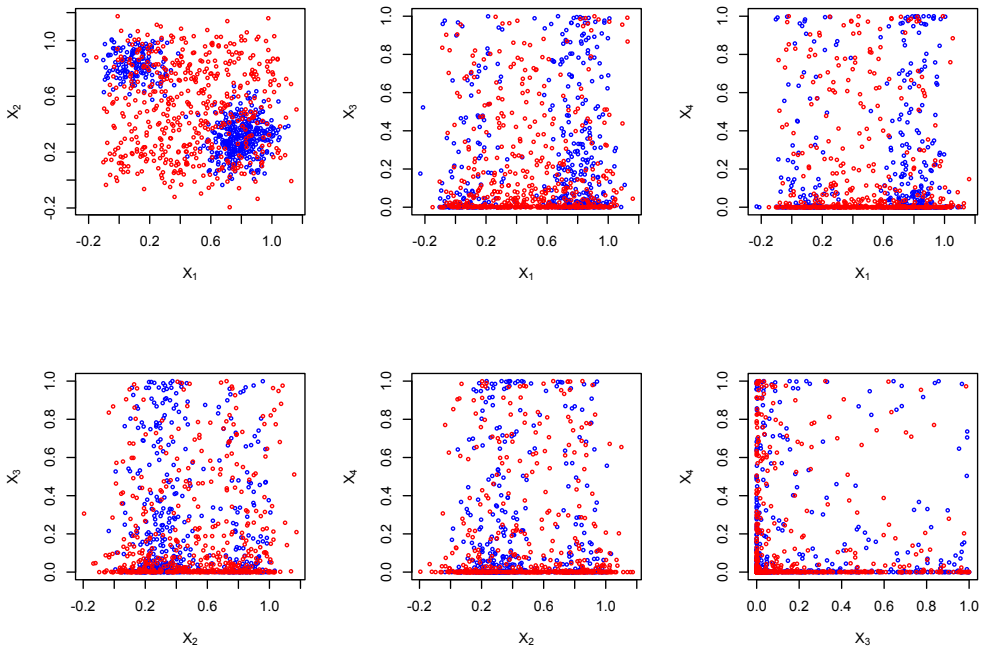


Figure 3.5: **Training dataset for Scenario 2**. Matrix plots of predictor variables X_1, X_2, X_3, X_4 display the relationships between 500 active (blue) and 500 inactive (red) observations from the simulated training dataset for Scenario 2, as described in the text. Variables X_1 and X_2 are associated with activity status whereas X_3 and X_4 are not. The scatterplot of X_1 against X_2 reveals the absence of well-defined boundaries separating active and inactive observations. The uneven sampling is illustrated by the patterns in the scatterplots.

Table 3.4: Scenario 2 misclassification test error rates for all fitted models. Rates for the top-performing models for each method and polynomial degree, where relevant, are highlighted in blue.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.196	0.165	0.111	0.123	0.119	0.079	0.126	0.124	0.098	0.112	0.195
X_1, X_3	0.287	0.268	0.200	0.286	0.269	0.206	0.310	0.318	0.246	0.306	0.372
X_1, X_4	0.288	0.271	0.216	0.279	0.276	0.195	0.303	0.332	0.261	0.386	0.374
X_2, X_3	0.452	0.349	0.243	0.446	0.350	0.259	0.304	0.350	0.244	0.434	0.404
X_2, X_4	0.417	0.326	0.236	0.417	0.324	0.259	0.307	0.379	0.268	0.369	0.403
X_3, X_4	0.490	0.518	0.508	0.490	0.512	0.512	0.498	0.477	0.491	0.499	0.510
X_1, X_2, X_3	0.192	0.164	0.120	0.124	0.120	0.088	0.145	0.121	0.106	0.251	0.265
X_1, X_2, X_4	0.197	0.173	0.117	0.123	0.120	0.086	0.124	0.120	0.218	0.328	0.280
X_1, X_3, X_4	0.287	0.274	0.232	0.279	0.275	0.209	0.322	0.307	0.248	0.396	0.349
X_2, X_3, X_4	0.456	0.351	0.289	0.447	0.346	0.259	0.359	0.369	0.265	0.413	0.475
X_1, X_2, X_3, X_4	0.196	0.176	0.231	0.124	0.120	0.084	0.154	0.118	0.252	0.261	0.209

Table 3.5: Scenario 2 Misclassification rate standard error for all fitted models, based on 100 replications.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.024	0.012	0.012	0.022	0.014	0.013	0.011	0.013	0.023	0.036	0.018
X_1, X_3	0.018	0.019	0.018	0.017	0.021	0.021	0.022	0.026	0.039	0.053	0.088
X_1, X_4	0.018	0.020	0.017	0.017	0.020	0.020	0.024	0.025	0.029	0.047	0.042
X_2, X_3	0.023	0.018	0.025	0.027	0.026	0.026	0.022	0.024	0.042	0.045	0.018
X_2, X_4	0.025	0.019	0.025	0.027	0.028	0.027	0.028	0.032	0.055	0.050	0.082
X_3, X_4	0.017	0.018	0.014	0.016	0.016	0.015	0.016	0.018	0.014	0.015	0.032
X_1, X_2, X_3	0.024	0.013	0.013	0.020	0.014	0.013	0.013	0.012	0.053	0.056	0.033
X_1, X_2, X_4	0.023	0.012	0.014	0.018	0.013	0.012	0.015	0.014	0.050	0.055	0.017
X_1, X_3, X_4	0.019	0.021	0.016	0.018	0.020	0.020	0.026	0.029	0.026	0.034	0.016
X_2, X_3, X_4	0.022	0.022	0.024	0.024	0.029	0.025	0.025	0.033	0.041	0.029	0.017
X_1, X_2, X_3, X_4	0.022	0.013	0.028	0.016	0.013	0.012	0.022	0.015	0.040	0.032	0.054

Considering the best-performing models, highlighted in blue, once again these models consistently incorporate the variables X_1 and X_2 but notably, the best SVM and RF models do not now exclusively include X_1 and X_2. It is also noteworthy that the majority of models utilizing only X_1 and X_2 achieve error rates that are close (within one standard error) to the minimum observed for their respective methods, see Table 3.5 for standard errors. On the other hand, the poorest performing models again fail to include either X_1 or X_2.

Table 3.6: **Scenario 2**. Performance measures for the optimal models identified using the best subset selection method, for each of the eleven combinations of methods and polynomial models assessed.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=2)	0	500	192	308	1.00	0.62	0.81	0.72	0.192	$X_1X_2X_3$
LR (d=3)	4	496	160	340	0.99	0.68	0.84	0.76	0.164	$X_1X_2X_3$
LR (d=4)	0	500	111	389	1.00	0.78	0.89	0.82	0.111	X_1X_2
ENET (d=2)	0	500	123	377	1.00	0.75	0.88	0.80	0.123	X_1X_2
ENET (d=3)	0	500	119	381	1.00	0.76	0.88	0.81	0.119	X_1X_2
ENET (d=4)	0	500	79	421	1.00	0.84	0.92	0.86	0.079	X_1X_2
SVM	0	500	124	376	1.00	0.75	0.88	0.80	0.124	X_1X_2X4
RF	0	500	118	382	1.00	0.76	0.88	0.81	0.118	$X_1X_2X_3X4$
NN (d=2)	0	500	98	402	1.00	0.80	0.90	0.84	0.098	X_1X_2
NN (d=3)	0	500	112	388	1.00	0.78	0.89	0.82	0.112	X_1X_2
NN (d=4)	96	404	99	401	0.81	0.80	0.81	0.80	0.195	X_1X_2

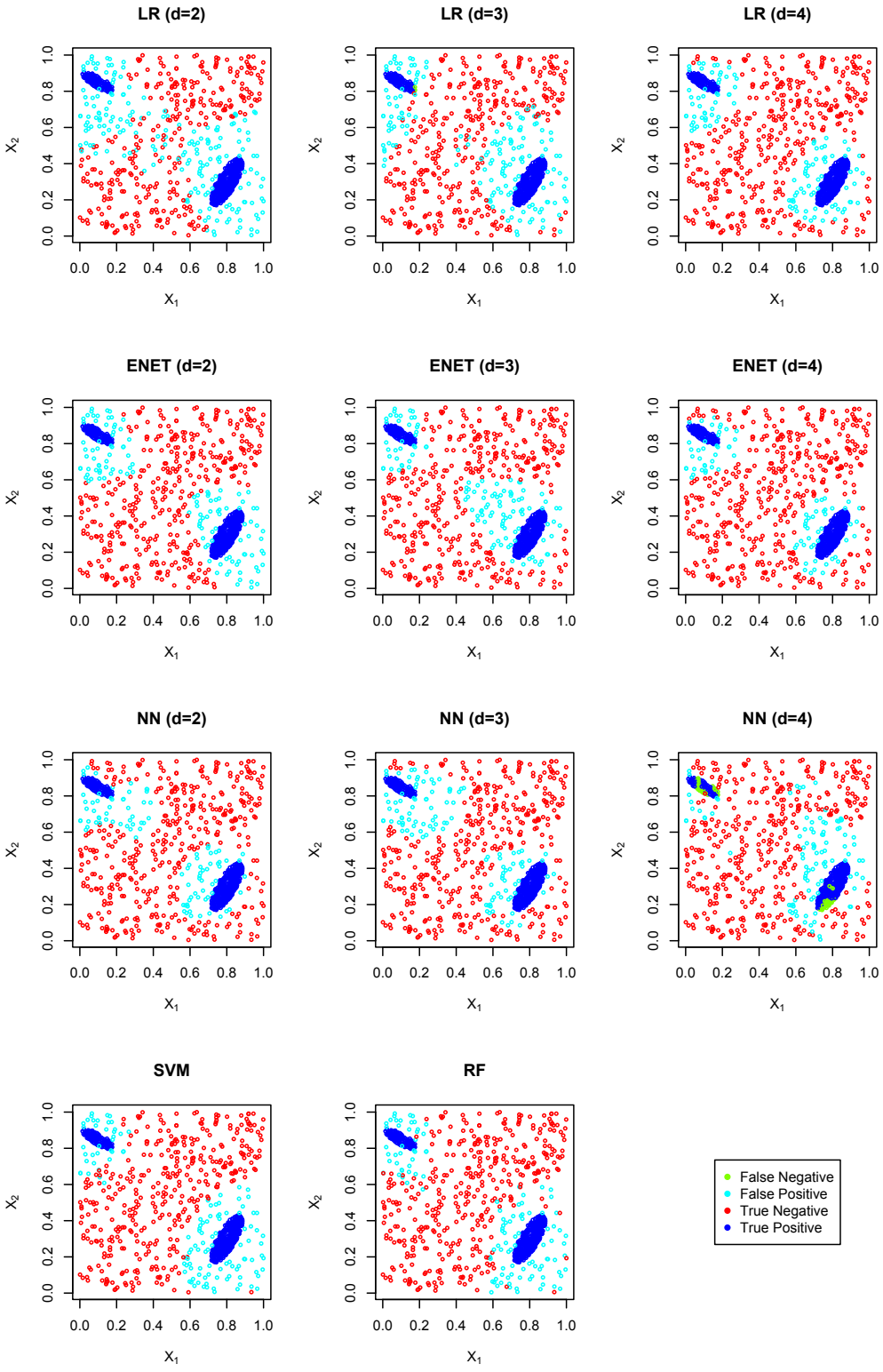


Figure 3.6: **Scenario 2**. Matrix scatterplots of predictions of activity status, by the five classification methods, for outcomes in the simulated test dataset, by variables X_1 and X_2.

Table 3.6 presents the performance metrics of the top-performing models, highlighting that all models, with the exception of NN ($d = 4$), achieve almost perfect sensitivity (SN) values. This indicates that, with the exception of NN ($d = 4$), all models are highly effective at predicting active outcomes. The results also reveal that the higher overall error rates (ER) observed in this scenario, as compared to Scenario 1, can be attributed primarily to an increase in false positive (FP) predictions. This surge in false positives results in a noticeable decline in specificity, relative to Scenario 1. The plots in Figure 3.6 reveal that the overall error rates of the top-performing models for the LR method is largely a consequence of lack of flexibility of the polynomial model, but that the flexibility improves with increasing degree. The plots also reveal that ENET regularisation helps to improve flexibility, more so for the quadratic polynomial model. It is also observed that NN ($d = 4$) produces clusters of false negative (FN) predictions; why this occurs is unclear.

Scenario 3

This scenario introduces an additional layer of complexity to the training data compared to Scenario 2 by simulating X_1, X_2 from Beta $B(0.5, 0.5)$ distributions, X_3 from $B(0.2, 0.8)$ and X_4 from a $B(0.1, 0.5)$ distribution. A notable change is that the training data, plotted in Figure 3.7, is now unevenly sampled across the space of X_1 and X_2, the two predictor variables associated with activity. Creating varying densities of training observations across both active and inactive regions allows examination of the performance of the ML methods when faced with imbalanced and nonuniform training data, a situation commonly encountered in real-world datasets. In particular, the setup in this Scenario permits evaluation of robustness of the ML

methods to learn from regions with high and low data density while still capturing the underlying patterns that distinguish active and inactive outcomes. As in Scenario 2, random noise sampled from a Normal distribution, $N(0, 0.01)$, is added to the simulated values to mimic measurement errors in the predictor variables. The true active regions and testing observations are simulated following the same framework as in the previous scenarios. This consistency ensures that comparisons between scenarios are fair and focused on the effects of the introduced complexities.

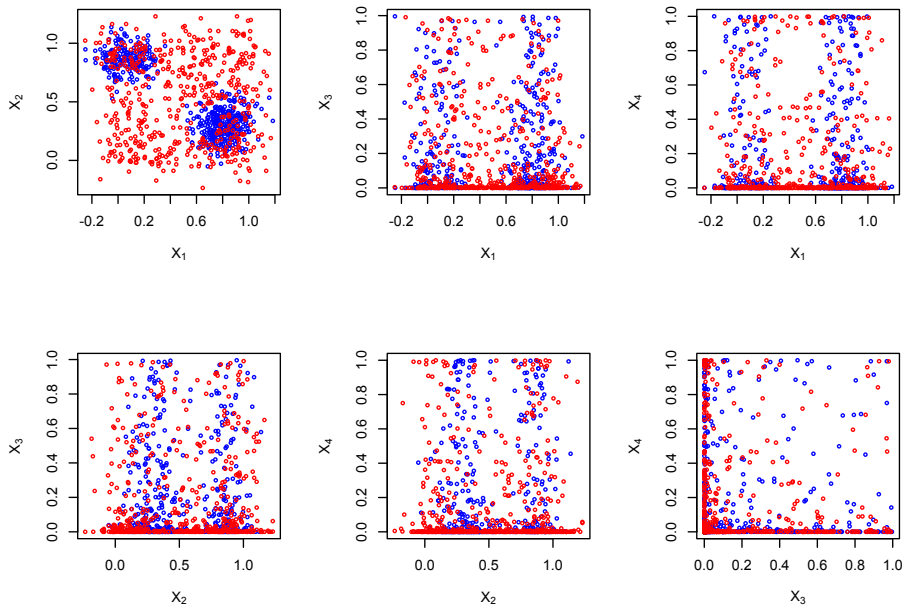


Figure 3.7: **Training dataset for Scenario 3**. Matrix plots of predictor variables X_1, X_2, X_3, X_4 displaying the relationships between 500 active (blue) and 500 inactive (red) observations from the simulated training dataset for Scenario 3, as described in the text. Variables X_1 and X_2 are associated with activity status whereas X_3 and X_4 are not. The scatterplot of X_1 against X_2 reveals the absence of well-defined boundaries separating active and inactive observations. The uneven sampling across all four predictor variables is illustrated by the patterns in the scatterplots.

Table 3.7: Scenario 3 misclassification test error rates for all fitted models. Rates for the top-performing models for each method and polynomial degree, where relevant, are highlighted in blue.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.307	0.262	0.180	0.262	0.230	0.197	0.149	0.133	0.209	0.285	0.143
X_1, X_3	0.332	0.280	0.213	0.316	0.278	0.218	0.285	0.301	0.256	0.349	0.482
X_1, X_4	0.318	0.271	0.212	0.316	0.312	0.215	0.250	0.312	0.265	0.359	0.415
X_2, X_3	0.490	0.430	0.269	0.490	0.452	0.281	0.283	0.358	0.279	0.426	0.368
X_2, X_4	0.477	0.438	0.277	0.491	0.433	0.256	0.341	0.307	0.376	0.374	0.446
X_3, X_4	0.488	0.499	0.516	0.490	0.515	0.511	0.518	0.511	0.468	0.528	0.513
X_1, X_2, X_3	0.304	0.259	0.191	0.243	0.230	0.127	0.158	0.142	0.215	0.521	0.303
X_1, X_2, X_4	0.304	0.263	0.181	0.243	0.230	0.152	0.145	0.124	0.330	0.421	0.304
X_1, X_3, X_4	0.358	0.285	0.244	0.309	0.312	0.218	0.335	0.319	0.359	0.412	0.432
X_2, X_3, X_4	0.482	0.426	0.309	0.491	0.452	0.281	0.371	0.333	0.396	0.433	0.442
X_1, X_2, X_3, X_4	0.303	0.260	0.197	0.243	0.230	0.128	0.145	0.131	0.262	0.331	0.307

Table 3.8: Scenario 3 Misclassification rate standard error for all fitted models, based on 100 replications.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.010	0.024	0.017	0.053	0.051	0.026	0.010	0.012	0.126	0.112	0.081
X_1, X_3	0.085	0.042	0.026	0.090	0.032	0.031	0.023	0.030	0.062	0.047	0.049
X_1, X_4	0.078	0.040	0.028	0.080	0.030	0.030	0.025	0.029	0.054	0.039	0.046
X_2, X_3	0.022	0.029	0.040	0.021	0.031	0.076	0.030	0.026	0.041	0.046	0.036
X_2, X_4	0.021	0.027	0.040	0.021	0.031	0.068	0.036	0.032	0.049	0.050	0.040
X_3, X_4	0.014	0.015	0.014	0.013	0.015	0.015	0.012	0.014	0.015	0.016	0.017
X_1, X_2, X_3	0.011	0.023	0.017	0.051	0.052	0.024	0.014	0.014	0.072	0.070	0.038
X_1, X_2, X_4	0.011	0.024	0.017	0.056	0.052	0.014	0.017	0.014	0.062	0.060	0.044
X_1, X_3, X_4	0.058	0.039	0.026	0.060	0.028	0.035	0.030	0.032	0.039	0.040	0.034
X_2, X_3, X_4	0.021	0.026	0.031	0.021	0.030	0.077	0.029	0.031	0.045	0.031	0.032
X_1, X_2, X_3, X_4	0.012	0.022	0.032	0.054	0.043	0.014	0.025	0.013	0.05 1	0.032	0.030

Results of the test misclassification error rates and misclassification rate standard error for all models fitted using the best subset selection approach, as summarized in Tables 3.7 and 3.8, reveal interesting trends across the different methods. For the neural network (NN) method, the best-performing models consistently identify only X_1 and X_2 as predictors of activity status, aligning with their known association with the true active regions. This pattern is also observed in the best models for logistic regression (LR) with $d = 4$ and elastic net (ENET) with $d = 3$, where the inclusion of X_1 and X_2 alone appears sufficient to optimize predictive performance. In contrast, the support vector machine (SVM) and random forest (RF) methods exhibit different behaviour. While these methods produce models with lower overall error rates compared to the others, the best-performing models for both methods include the spurious variable X_4 as a predictor variable. This suggests that SVM and RF may be more prone to overfitting or leveraging noise in the data, especially when the spurious variable coincidentally correlates with the outcome in certain regions of the predictor space.

The findings here highlight the varying tendencies of different machine learning methods in balancing model complexity and generalization. While NN, LR, and ENET effectively exclude irrelevant predictor variables, SVM and RF appear to benefit from more flexible modelling, albeit at the cost of occasionally incorporating non-informative variables. This distinction underscores the importance of carefully evaluating model selection criteria, and understanding the trade-offs between error minimisation and predictor interpretability.

Table 3.9: **Scenario 3**. Performance measures for the optimal models identified using the best subset selection method, for each of the eleven combinations of methods and polynomial models assessed.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=2)	2	498	301	199	0.99	0.40	0.70	0.62	0.303	$X_1X_2X_3X4$
LR (d=3)	0	500	259	241	1.00	0.48	0.74	0.66	0.259	$X_1X_2X_3$
LR (d=4)	0	500	180	320	1.00	0.64	0.82	0.74	0.180	X_1X_2
ENET (d=2)	35	465	208	292	0.93	0.58	0.76	0.69	0.243	$X_1X_2X_3$
ENET (d=3)	46	454	184	316	0.91	0.63	0.77	0.71	0.230	X_1X_2
ENET (d=4)	0	500	127	373	1.00	0.75	0.87	0.80	0.127	$X_1X_2X_3$
SVM	0	500	145	355	1.00	0.71	0.86	0.78	0.145	X_1X_2X4
RF	7	493	117	383	0.98	0.77	0.88	0.81	0.124	X_1X_2X4
NN (d=2)	113	387	96	404	0.77	0.81	0.79	0.80	0.209	X_1X_2
NN (d=3)	175	325	110	390	0.65	0.78	0.72	0.75	0.285	X_1X_2
NN (d=4)	39	461	104	396	0.92	0.79	0.86	0.82	0.143	X_1X_2

The performances of the methods are further examined using metrics obtained for the optimal best subset selection models and summarized in Table 3.9. As expected, the added complexity in the training data resulted in higher test error rates (ER) compared to Scenario 2. This increase is particularly pronounced for logistic regression (LR), elastic net (ENET), and neural networks (NN), while the error rates for support vector machines (SVM) and random forests (RF) are less affected by the added complexity. These differences reflect the varying levels of robustness among the methods when dealing with uneven sampling and noise in the data. A closer

analysis of the sensitivity and specificity measures provides further insights. Most methods, with the exception of NN, exhibit very high sensitivity values. This indicates that LR, ENET, SVM, and RF are effective at correctly identifying active outcomes under the data conditions evaluated here. However, these methods tend to compromise on specificity, meaning they are more prone to falsely labelling inactive outcomes as active. In contrast, the NN models demonstrate superior performance in specificity, suggesting that they are less likely to generate false positives by labelling inactive outcomes as active. However, the trade-off is lower sensitivity, indicating that NN is likely to adopt a more conservative approach to classification under added complexity in the training data.

The scatterplots in Figure 3.8 provide a visual comparison of the predictive patterns across the different methods, showcasing their ability to capture active and inactive regions under conditions of unevenly sampled training data and noise in measuring predictor values. As observed in previous scenarios, SVM and RF are better able to capture the boundaries between active and inactive regions. LR also performs reasonably well in identifying active outcomes, but consistent with its behaviour in earlier scenarios, its limited flexibility results in imprecise boundaries, especially in regions requiring more complex decision-making. While addition of regularisation via ENET again slightly improves the boundary definitions, this improvement comes at the cost of a marginal increase in false negatives near the boundaries of active regions. The neural network (NN) method, in contrast, produces less well-defined boundaries compared to the other methods, reflecting its potential lack of robustness to uneven sampling and noise in the training data.

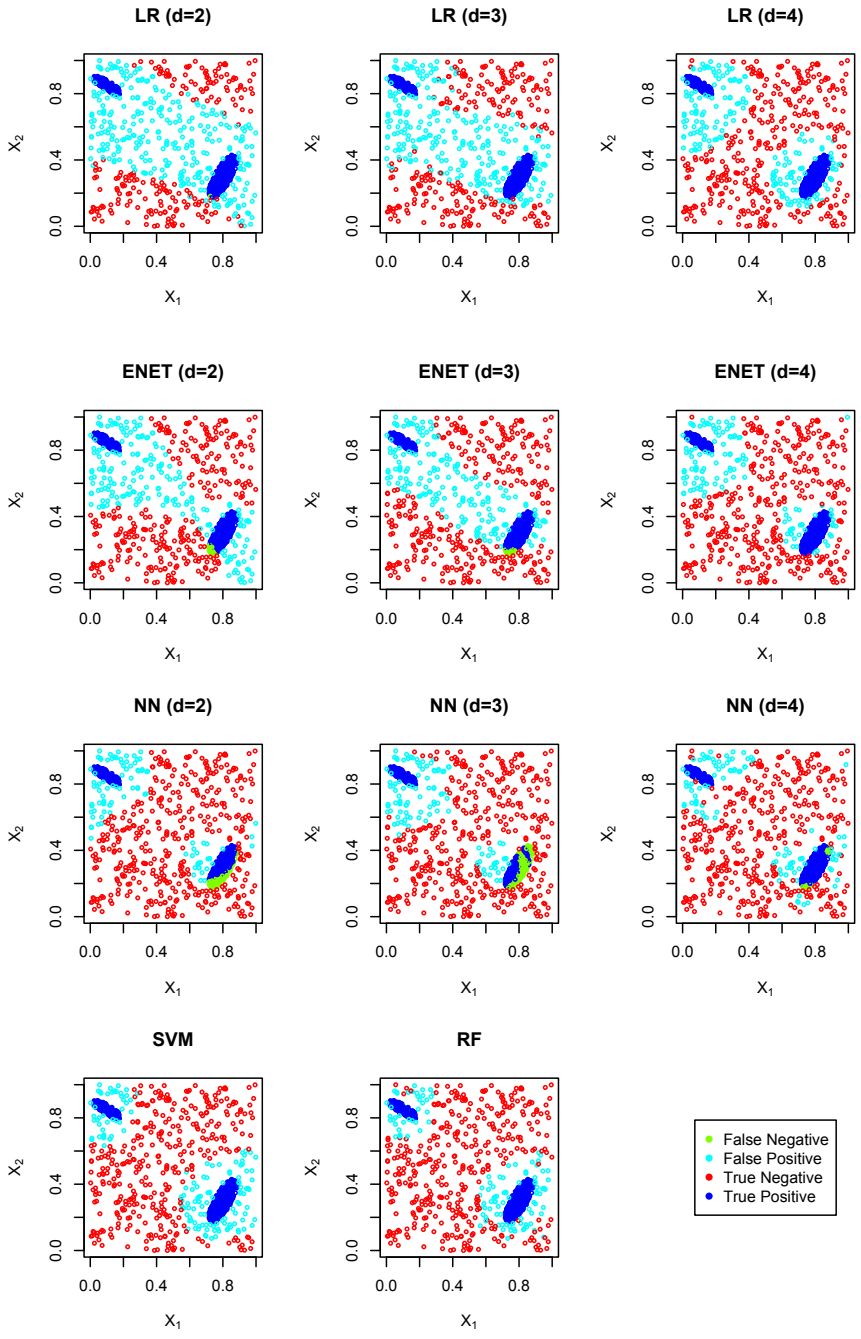


Figure 3.8: **Scenario 3**. Matrix scatterplots of predictions of activity status, by the five classification methods, for outcomes in the simulated test dataset, by variables X_1 and X_2.

Scenario 4

The training dataset in this scenario, illustrated in Figure 3.9, was simulated to closely replicate the active and inactive regions observed in the plots of Hydrophobicity (H) and net charge (C) in relation to activity status within the peptide data used in this study. Figures 2.4 and 2.5 presents two-dimensional (2D) projections of this relationship. One active and one inactive region were constructed as elliptic functions of X_1 and X_2 and, as in Scenario 3, the four predictor variables X_1, X_2, X_3 and X_4 in the training dataset were simulated to represent uneven sampling over the regions. Additionally, to reflect the assumption that separation between active and inactive regions cannot be perfectly defined in practice, noise was introduced into the boundary between the active and inactive regions for both the training dataset and test dataset, shown in Figure 3.2. This decision aligns with biological realities, where subtle variations in molecular properties can result in overlapping or transitional zones of activity.

Tables 3.10 and 3.11 present the test misclassification error rates and misclassification rate standard error for all models fitted using the best subset selection approach. The logistic regression (LR) method with quadratic and quartic polynomials achieved minimum error rates of 0.023 and 0.035, respectively, in the space of X_1 and X_2. Similarly, for the elastic net (ENET) method with a quadratic polynomial model, a minimum error rate of 0.023 was achieved in the space of X_1 and X_2, while for the neural network (NN) method, minimum error rates of 0.023, 0.054, and 0.086, respectively were produced in the space of these two predictor variables. The support vector machine (SVM) and random forest (RF) methods yielded minimum error rates

of 0.061 each in spaces incorporating a third predictor variable.

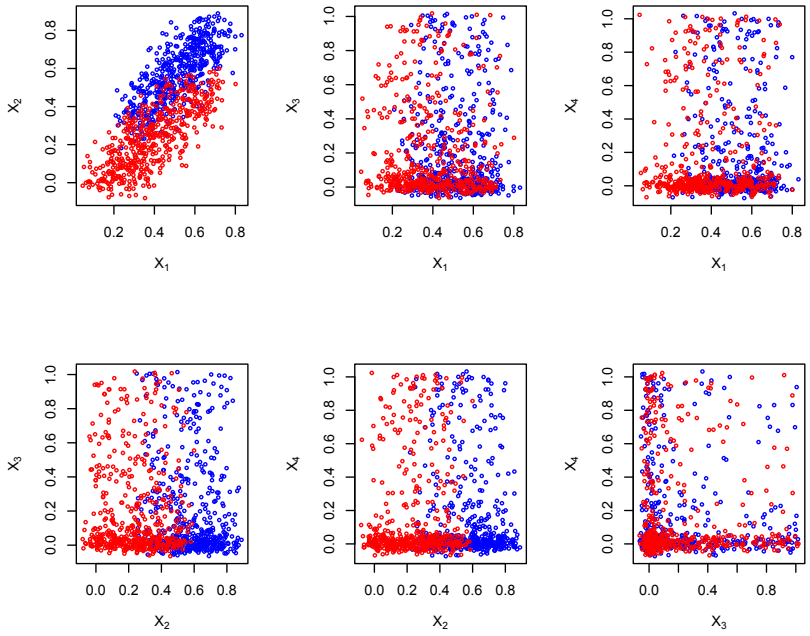


Figure 3.9: **Training dataset for Scenario 4**. Matrix plots of predictor variables X_1, X_2, X_3, X_4 display the relationships between 500 active (blue) and 500 inactive (red) observations from the simulated training dataset for Scenario 4, as described in the text. Variables X_1 and X_2 are associated with activity status whereas X_3 and X_4 are not. The scatterplot of X_1 against X_2 reveals the absence of a well-defined boundary separating active and inactive observations. The uneven sampling across all four predictor variables is illustrated by the patterns in the scatterplots.

The above findings are consistent with previous scenarios in that models incorporating both X_1 and X_2 exhibit lower error rates compared to those that exclude one or both variables. This emerging pattern, that predictor variables strongly associated with the outcome are consistently included among the best-performing models in the best subset selection approach, will be exploited in this thesis to refine the variable selection procedure, and thereby enhance the effectiveness of the model selection process.

Table 3.10: Scenario 4 misclassification test error rates for all fitted models. Rates for the top-performing models for each method and polynomial degree, where relevant, are highlighted in blue.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.023	0.026	0.035	0.023	0.034	0.031	0.062	0.073	0.023	0.054	0.086
X_1, X_3	0.434	0.451	0.449	0.440	0.434	0.436	0.465	0.466	0.439	0.463	0.494
X_1, X_4	0.438	0.451	0.448	0.434	0.435	0.436	0.472	0.464	0.436	0.422	0.460
X_2, X_3	0.072	0.075	0.085	0.069	0.070	0.069	0.079	0.134	0.147	0.095	0.119
X_2, X_4	0.071	0.074	0.071	0.069	0.072	0.072	0.105	0.141	0.069	0.370	0.250
X_3, X_4	0.514	0.524	0.521	0.516	0.507	0.503	0.498	0.495	0.503	0.517	0.505
X_1, X_2, X_3	0.028	0.025	0.048	0.028	0.029	0.036	0.061	0.074	0.059	0.149	0.133
X_1, X_2, X_4	0.030	0.025	0.050	0.029	0.028	0.029	0.063	0.061	0.047	0.117	0.171
X_1, X_3, X_4	0.437	0.446	0.451	0.439	0.434	0.433	0.464	0.454	0.472	0.449	0.493
X_2, X_3, X_4	0.065	0.076	0.089	0.069	0.068	0.070	0.120	0.147	0.178	0.185	0.196
X_1, X_2, X_3, X_4	0.030	0.037	0.116	0.029	0.029	0.027	0.072	0.072	0.147	0.136	0.090

Table 3.11: Scenario 4 Misclassification rate standard error for all fitted models, based on 100 replications.

Models	LR			ENET			SVM	RF	NN		
	d=2	d=3	d=4	d=2	d=3	d=4			d=2	d=3	d=4
X_1, X_2	0.007	0.019	0.026	0.008	0.010	0.012	0.010	0.011	0.027	0.059	0.057
X_1, X_3	0.016	0.015	0.015	0.016	0.018	0.018	0.016	0.017	0.019	0.021	0.022
X_1, X_4	0.015	0.016	0.016	0.016	0.018	0.018	0.017	0.017	0.023	0.024	0.022
X_2, X_3	0.007	0.016	0.015	0.006	0.007	0.008	0.015	0.017	0.035	0.056	0.068
X_2, X_4	0.006	0.016	0.018	0.006	0.007	0.007	0.019	0.021	0.058	0.059	0.069
X_3, X_4	0.015	0.016	0.016	0.016	0.017	0.017	0.015	0.017	0.016	0.016	0.015
X_1, X_2, X_3	0.007	0.021	0.034	0.007	0.010	0.015	0.012	0.014	0.049	0.045	0.025
X_1, X_2, X_4	0.007	0.020	0.048	0.007	0.010	0.012	0.014	0.017	0.053	0.052	0.025
X_1, X_3, X_4	0.016	0.016	0.016	0.017	0.017	0.017	0.017	0.020	0.023	0.020	0.016
X_2, X_3, X_4	0.007	0.016	0.030	0.006	0.008	0.008	0.022	0.023	0.063	0.046	0.027
X_1, X_2, X_3, X_4	0.007	0.022	0.059	0.008	0.011	0.012	0.016	0.015	0.045	0.022	0.018

The performance metrics for all methods evaluated in this scenario are notably high, see Table 3.12. Sensitivity values across all methods exceed 0.85, indicating strong performance in correctly identifying true positives. Specificity values are similarly robust, with most achieving values above 0.90, demonstrating the methods' effectiveness in accurately identifying true negatives. Accuracy and positive predictive value (PPV) metrics further highlight the strong performance of the five methods. Accuracies and PPVs for all exceed 0.90, with one exception: the accuracy of the RF method is 0.89. Despite this slight deviation, the RF method still demonstrates a high level of performance, remaining competitive with the other four classification methods.

In terms of error rates, the minimum values observed for the five classification methods show only slight variation. This consistency suggests that the models are robust and perform reliably across various scenarios, with no single method showing substantial under-performance. In particular, unlike the previous scenarios, here LR demonstrates strong performance. Interestingly, the impact of regularization on the LR error rates is minimal, suggesting that the polynomial models are appropriate for this Scenario, without substantial overfitting or underfitting.

The plots of predictions against X_1 and X_2 for the five classification methods, including polynomial models are presented in Figure 3.10. These plots visually illustrate the prediction regions for active and inactive outcomes, providing insight into the accuracy of the methods and the challenges in the boundary region. In most cases, the predicted areas align well with the true active and inactive regions, demonstrating the methods' effectiveness in correctly classifying the majority of outcomes.

The plots in Figure 3.10 also highlight specific patterns of incorrect predictions. Areas

where negative outcomes were incorrectly predicted as positive are marked in blue, while areas where positive outcomes were incorrectly predicted as negative are marked in green. Not unexpectedly, misclassifications are prominent along the boundary between active and inactive outcomes, reflecting the classification challenges in this transitional region due to the overlapping characteristics of the two classes. Overall, the visualization in Figure 3.10 underscores the effectiveness of the five classification methods while also highlighting the challenges posed by low-density transitional regions.

Table 3.12: **Scenario 4**. Performance measures for the optimal models identified using the best subset selection method, for each of the eleven combinations of methods and polynomial models assessed.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=2)	19	481	4	496	0.96	0.99	0.98	0.99	0.023	X_1X_2
LR (d=3)	6	494	19	481	0.99	0.96	0.97	0.96	0.025	$X_1X_2X_3$
LR (d=4)	2	498	33	467	0.99	0.93	0.96	0.94	0.035	X_1X_2
ENET (d=2)	19	481	4	496	0.96	0.99	0.98	0.99	0.023	X_1X_2
ENET (d=3)	23	477	5	495	0.95	0.99	0.97	0.99	0.028	$X_1X_2X_4$
ENET (d=4)	21	479	6	494	0.96	0.99	0.97	0.99	0.027	$X_1X_2X_3X_4$
SVM	61	439	0	500	0.88	1.00	0.94	1.00	0.061	$X_1X_2X_3$
RF	60	440	1	499	0.88	0.99	0.89	0.99	0.061	$X_1X_2X_4$
NN (d=2)	15	485	8	492	0.97	0.98	0.98	0.98	0.023	X_1X_2
NN (d=3)	28	472	26	474	0.94	0.95	0.95	0.95	0.054	X_1X_2
NN (d=4)	44	456	42	458	0.91	0.92	0.91	0.91	0.086	X_1X_2

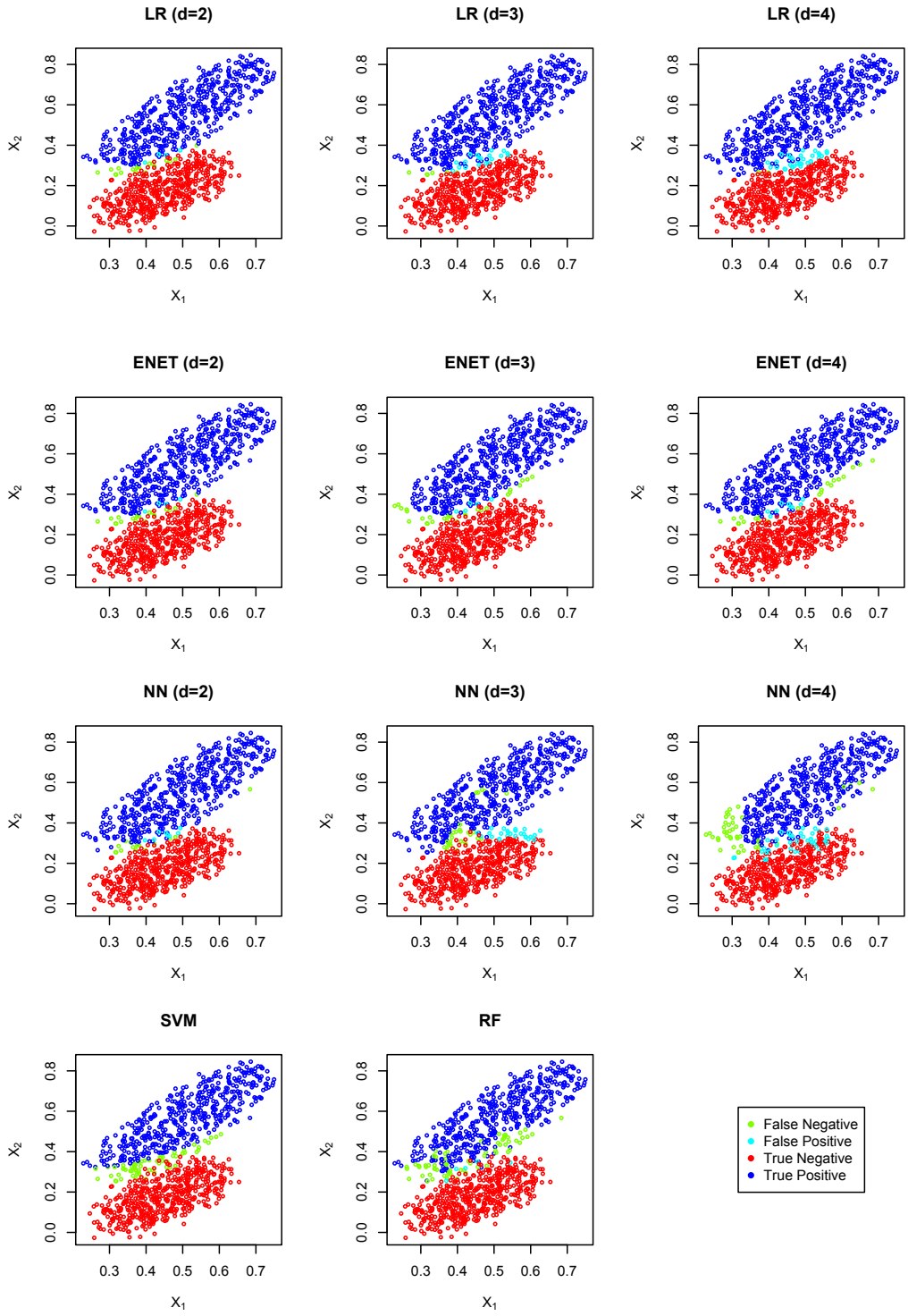


Figure 3.10: **Scenario 4**. Matrix scatterplots of predictions of activity status, by the five classification methods, for outcomes in the simulated test dataset, by variables X_1 and X_2.

3.2 Summary of the Simulation Study Findings

The performance of the five classification methods in predicting outcomes is influenced by the complexity of the data. Logistic regression, while straightforward and interpretable, can struggle with flexibility in capturing the underlying data pattern, resulting in comparatively higher misclassification error rates. Elastic net regularization can offer improvements by enhancing model adaptability, particularly for quadratic polynomial models, but these gains are insufficient to match the performance of more flexible methods in challenging scenarios. Both LR and ENET tend to exhibit high sensitivity and true positive rates, effectively identifying active outcomes. However, they are less effective in maintaining specificity, with elevated false positive rates undermining their overall performance in accurately distinguishing inactive outcomes. The neural network method can demonstrate the contrasting behaviour of prioritizing specificity at the cost of increasing the chance of missing active outcomes. As the data complexity increases, encompassing uneven sampling and measurement errors, the misclassification error rates for LR, ENET, and NN tend to increase as the methods struggle somewhat to adapt. In contrast, by leveraging their intrinsic flexibility, support vector machines and random forests maintain comparatively lower error rates and consistent performance across scenarios.

Regarding variable selection, for LR, ENET and NN methods, the best-performing models tend to include only those predictor variables associated with the outcome, achieving optimal predictive performance while excluding irrelevant predictor variables. In contrast, the SVM and RF methods exhibit a different behaviour. Although these methods achieve lower overall error rates compared to the others, their

best-performing models tend to include spurious variable(s) as predictor(s).

These findings emphasise the varying tendencies of machine learning methods in balancing model complexity and generalization. The simulation results suggest that while NN, LR, and ENET tend to focus on relevant predictor variables, SVM and RF demonstrate a preference for more flexible modelling, occasionally at the expense of interpretability, by incorporating non-informative variables. Nevertheless, for all five methods, models incorporating predictor variables associated with the outcome consistently achieve error rates that are close to the minimum observed among the set of models fitted using the best subset selection approach. Conversely, the poorest-performing models tend to exclude one or more predictor variables that are critical to explaining the outcome.

Overall, the simulation results highlight the trade-offs inherent in each method. Among the five methods, SVM and RF tend to excel in boundary precision but may include spurious variables in their models. LR and ENET provide interpretable solutions with comparable performance, while NN demonstrates potential for non-linear modelling but may struggle with boundary clarity. Based on these findings, the decision was made to evaluate the potential of all five methods for predicting antimicrobial peptides.

3.3 Five Models of AMP Activity

The five classification methods, LR, ENET, SVM, RF and NN are here employed to construct machine learning models for predicting antimicrobial peptide (AMP) activity using the larger datasets obtained from the DBAASP database, as detailed

in Section 2.4.3. As a reminder, after data cleaning and balancing the active and inactive peptide classes, these datasets contained 644 active and 644 inactive peptides of 10-16 amino acid (aa) lengths, and 406 active and 406 inactive peptides of 18-27 aa lengths. As before, training and test sets were generated with an 80 : 20 split giving, for 10-16 aa peptides, a training set consisting of 515 active and 515 inactive peptides, and a test set with 129 active and 129 inactive peptides. Similarly, for the 18-27 aa peptides, the training set comprised 324 active and 324 inactive peptides, while the test set contains 82 active and 82 inactive peptides.

Each ML model is constructed using the training datasets, following the methodology outlined in Section 3.1.2. Thus, best subset variable selection is employed, using misclassification error as evaluation criterion to identify the optimal model for each ML method and orthogonal polynomials were utilized to construct models for the LR, ENET and NN methods. Indeed only quadratic polynomials are fitted as, given the sizes of the training datasets and the total number of non-constant terms in a degree d polynomial with p predictor variables is calculated as

$$\binom{p+d}{d} - 1,$$

fitting models with $d > 2$ is likely to lead to overfitting and was deemed computationally impractical. For illustration, the cubic polynomial model with $p = 9$ predictor variables and fitted using LR has 219 parameters to be estimated, along with the intercept term. Furthermore, the results of the simulation study suggest that the optimal model among the set of possible quadratic polynomial models is likely to contain predictor variables relevant for predicting outcome, and that the quadratic

polynomial models have only slightly higher error rates than cubic and quartic models. The models obtained from each of the five methods are evaluated on the test datasets.

3.3.1 Results for 10-16 aa length peptides

Table 3.13 presents the performance metrics for the optimal models for predicting the activity status of peptides of 10-16 aa length using the five classification methods. For comparison, the DB-SCAN results from Table 2.9 are also included. Among the first five methods, the RF model achieves the lowest misclassification error rate of 0.1240 from a model in six-dimensional space $MHCIRA$ of physicochemical properties, followed by the SVM method, with an error rate of 0.1550 derived from a model in the six-dimensional space $MHCIDL$. The NN method has next lowest error rate of 0.2016, based on a model in the six-dimensional space $MHCIDR$, followed by the ENET method with a slightly higher error rate of 0.2093 and employing a model in a five-dimensional feature space $MIDLA$. Finally, the LR method demonstrates the highest error rate of 0.2171 and the model constructed in the five-dimensional space $MCIDR$. All misclassification error rates are however much less than obtained using DB-SCAN.

Sensitivity values for the first five methods are over 0.80, with the RF method having the highest value of 0.91 while the other four methods have values between 0.81 and 0.85. On the other hand, DB-SCAN has very low sensitivity. Conversely, with a specificity of 0.97, DB-SCAN is highest among all methods. The other five methods have lower specificity, with values ranging between 0.73 and 0.86. Among this lot,

SVM has the highest value of 0.86, followed by RF having a specificity value of 0.84, which is not much different from the SVM method.

Accuracies of the first five methods are between 0.78 and 0.88, with the RF method having the highest value of 0.88, followed by the SVM method having an accuracy value of 0.85. At a value of 0.52, accuracy of the DB-SCAN method is comparatively low. Positive predictive values of the first five methods are between 0.76 and 0.86, with the SVM method having the highest value of 0.86, followed by the RF method having a specificity value of 0.85, which is not much different from the SVM method. Lowest PPV is obtained using the DB-SCAN method.

Table 3.13: Model performance measures using five different methods and the DB-SCAN method to construct models for predicting activity of 10-16 aa length peptides. The bootstrap standard error (SE) of the misclassification rate, calculated from 100 iterations, are in the brackets.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER (SE)	Models
10-16 aa length										
LR	21	108	35	94	0.84	0.73	0.78	0.76	0.2171 (0.01)	*MCIDR*
ENET	20	109	34	95	0.85	0.74	0.79	0.76	0.2093 (0.00)	*MIDLA*
SVM	22	107	18	111	0.83	0.86	0.85	0.86	0.1550 (0.02)	*MHCIDL*
RF	11	118	21	108	0.91	0.84	0.88	0.85	0.1240 (0.02)	*MHCIRA*
NN	25	104	27	102	0.81	0.79	0.80	0.79	0.2016 (0.01)	*MHCIDR*
DB-SCAN	121	8	5	124	0.06	0.97	0.52	0.62	0.4884	*MHCDR*[1]
										MCIDO[2]
										CIR[3]

[1] Cluster 1;
[2] Cluster 2;
[3] Cluster 3.

Comparing physicochemical properties appearing in the optimal models identified

using best subset variable selection, for the first five methods shown in Table 3.13, it can be seen that hydrophobic moment (M) and isoelectric point (I) are identified by all methods. In terms of physicochemical structure (see Chapter 2, Section 2.2), M is related to linear moment (L), which occurs in the ENET and SVM models. Also, I is related to net charge (C), which occurs in all, except the ENET model. On the other hand, penetration depth (D) occurs in all models, except the one constructed using RF. In terms of physicochemical structure, D is related to hydrophobicity (H), which occurs in all models, except ENET. In addition, propensity to disordering (R) occurs in the LR, RF, and NN models, while propensity to aggregation (A) occurs in the ENET and RF methods. In contrast, tilt angle (O) did not show important physicochemical properties for shorter peptides. Compared with the DBSCAN clusters, the other five ML models have two features consistent with cluster 2, that is M and I, one feature, i.e. M, consistent with cluster 1 and one feature, i.e. I, consistent with cluster 3.

Figure 3.11 presents 3D plots illustrating the physicochemical properties M, I, and C alongside the predicted activity status of peptides 10-16 aa length in the test dataset. In these visualizations, correctly predicted active peptides are represented in dark blue, while correctly predicted inactive peptides are shown in red. False positives, where inactive peptides are incorrectly predicted as active, are coloured blue, and false negatives, where active peptides are incorrectly predicted as inactive, are depicted in green. An examination of the plots generated by the five methods suggests clustering of active peptides within specific regions of the 3D space defined by the physicochemical properties. However, these clusters are not clearly distinguishable from the inactive peptides, indicating potential overlap between active and inactive

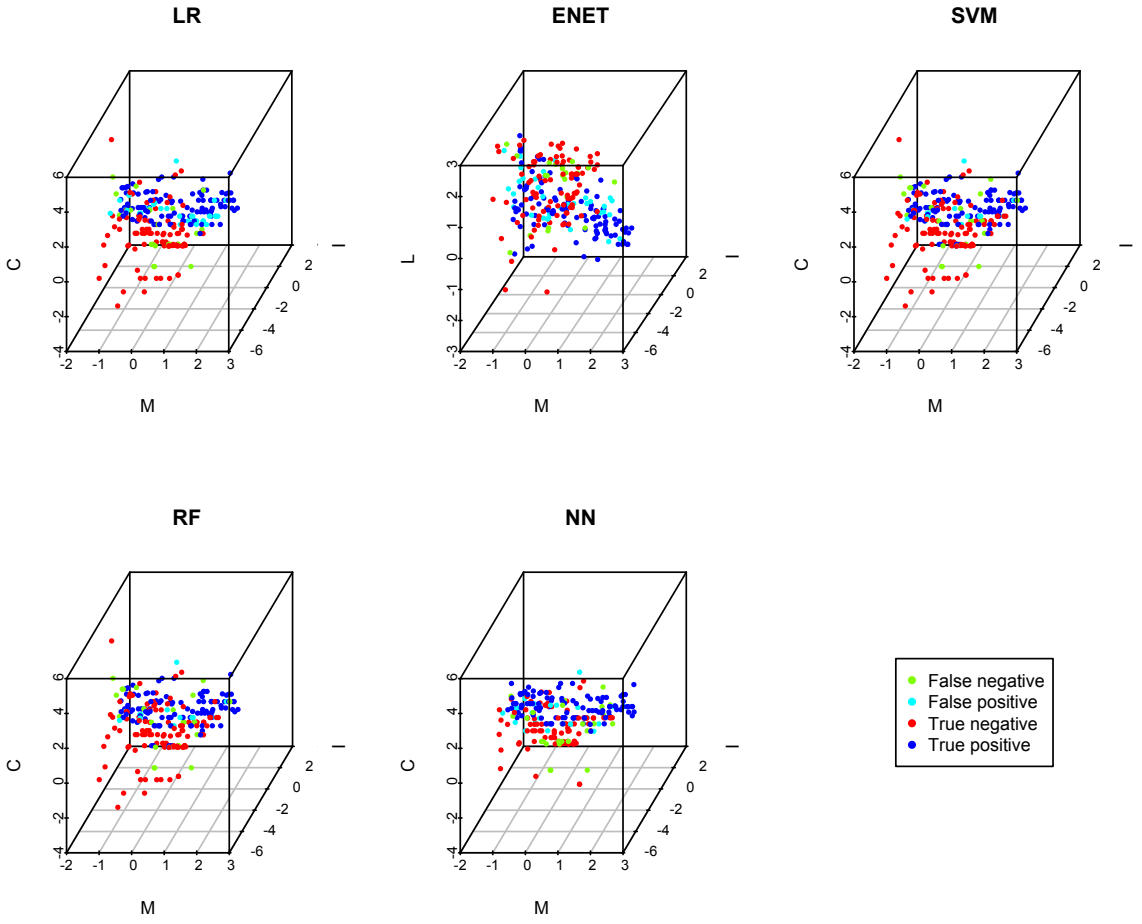


Figure 3.11: 3D scatterplots of predictions of activity status by hydrophobic moment M, isoelectric point I and net charge C for 10-16 aa length peptides.

peptide distributions and regions of the physicochemical space where correct prediction of activity status is more challenging.

3.3.2 Results for 18-27 aa length peptides

Provided in Table 3.14 are performance measures of the five methods for predicting activity of 18-27 aa length peptides. As was the case for the 10-16 aa peptides, the RF method has the lowest misclassification error rate from a model in six-dimensional space $MHCDOL$, followed by the SVM method with model in the five-dimensional space $HCILA$. The LR method with model in five-dimensional feature space $MHCOR$, and ENET and NN methods with models in five-dimensional spaces $MHCDL$ and $HCIDL$ follow. With values between 0.1037 and 0.1341, the misclassification error rates for these five methods are less variable than those for 10-16 aa peptides and substantially less than the error of the DB-SCAN model.

Table 3.14: Model performance measures using five different methods and the DB-SCAN method to construct models for predicting activity of 18-27 aa length peptides. The bootstrap standard error (SE) of the misclassification rate, calculated from 100 iterations, are in the brackets.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER (SE)	Models
18-27 aa length										
LR	9	73	12	70	0.89	0.85	0.87	0.86	0.1280 (0.02)	*MHCOR*
ENET	7	75	15	67	0.91	0.82	0.87	0.83	0.1341 (0.00)	*MHCDL*
SVM	11	71	8	74	0.87	0.90	0.88	0.90	0.1159 (0.02)	*HCILA*
RF	11	71	6	76	0.87	0.93	0.90	0.92	0.1037 (0.02)	*MHCDOL*
NN	12	70	10	72	0.85	0.88	0.87	0.88	0.1341 (0.00)	*HCIDL*
DB-SCAN	76	6	2	80	0.07	0.98	0.52	0.75	0.4756	*HCIOLA*

Exploring the performances further, the sensitivity of all five methods is more than or equal to 0.85, with the ENET method having the highest value of 0.91 while LR, SVM, RF and NN have values between 0.85 and 0.89. The specificity values of all

five methods are over 0.80, with the RF method having the highest value of 0.93, followed by the SVM method having a specificity value of 0.90 and LR, ENET and NN methods having values between 0.82 and 0.88. Accuracies of all five methods are values between 0.87 and 0.90, with the RF method having the highest value of 0.90, which is not much different from other methods. Positive predictive values of all five methods are over 0.80, with the RF method having the highest value of 0.92, followed by the SVM method having a specificity value of 0.90 and other methods having values between 0.83 and 0.88. Conversely, the DB-SCAN model again has the highest sensitivity but its other performance measures are much lower than those of the other methods.

Comparing the predictor variables used to construct the models shown in Table 3.14, we can see that hydrophobicity (H) and net charge (C) occur in all six cases. As outlined Section 2.2 of Chapter 2, H is related to penetration depth (D) and C is related to the isoelectric point (I). The predictor variable D occurs in the ENET, RF and NN methods, and I occurs in the SVM and NN methods. Also, it can be seen that linear moment (L) occurs in the ENET, SVM, RF and NN methods, and is related to the hydrophobic moment (M) that occurs in the LR, ENET and RF methods. In addition, tilt angle (O) occurs in the LR and RF methods, the propensity to disordering (R) occurs in the LR method, and the propensity to aggregation (A) occurs in the SVM method.

The 3D scatterplots of the physicochemical properties M, H, and C alongside the predicted activity status of peptides 18-27 aa length in Figure 3.12 suggest a similar narrative as observed for the shorter peptides in that all five methods suggest the existence of clusters of active peptides but with some overlap between the two classes.

As for the shorter peptides, it is important to recognize that these 3D scatterplots are projections of the 9-dimensional space constituted by the full set of physicochemical properties utilized in the analysis and therefore caution is warranted in interpreting the apparent clustering and overlap.

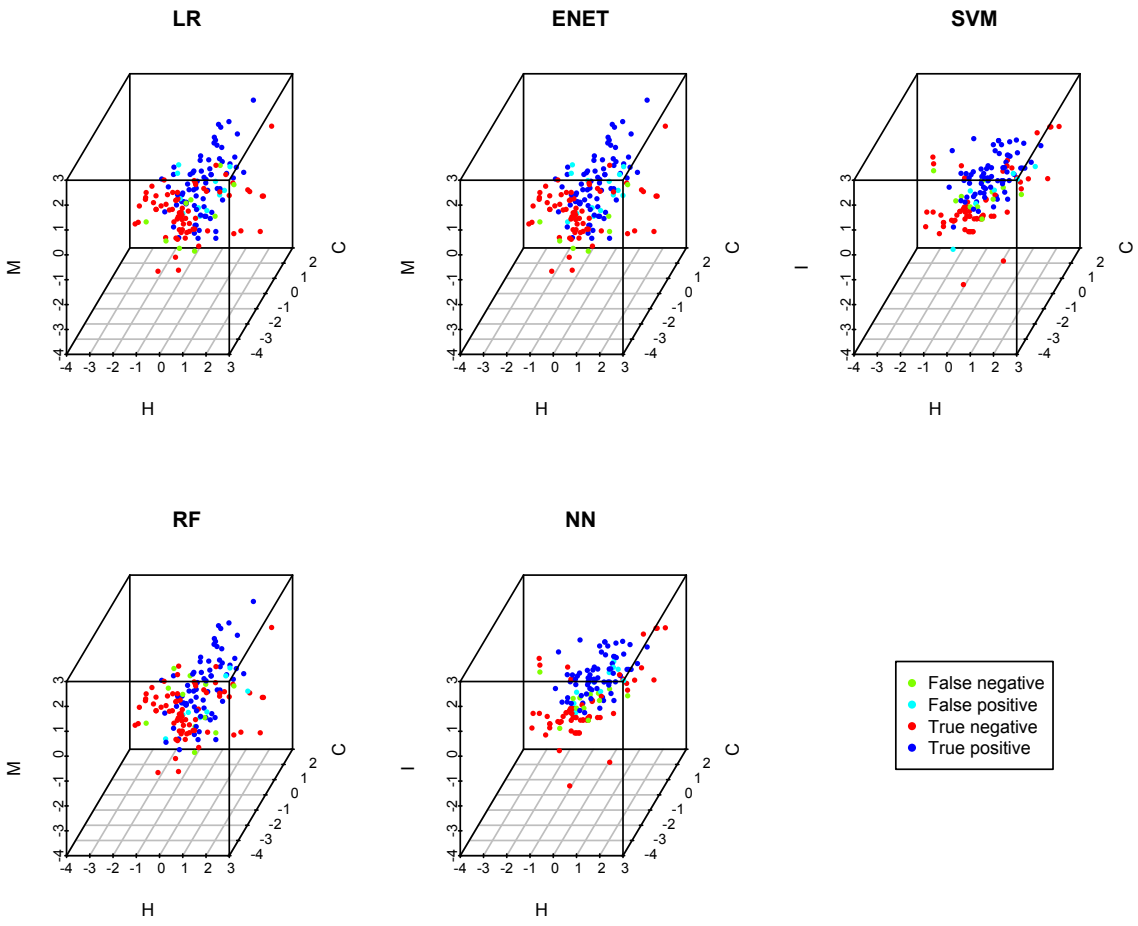


Figure 3.12: 3D scatterplots of predictions of activity status by hydrophobic moment M, hydrophobicity H and net charge C for 18-27 aa length peptides.

3.4 Discussion

Evaluating the performance of DB-SCAN in comparison to the LR, ENET, SVM, RF and NN classification methods it is evident that, while DB-SCAN demonstrates high specificity, indicating a low likelihood of falsely predicting inactive peptides as active, it falls short in replicating the overall performance achieved by the other five methods. Notably, the sensitivity of DB-SCAN is relatively low, leading to a higher probability of missing active peptides. This high specificity coupled with low sensitivity can be attributed to the fact that the DB-SCAN method is based on identifying dense clusters of active peptides which limits its ability to accurately classify new, unseen peptides. In contrast, as the simulations demonstrate, supervised classifiers such as LR, ENET, SVM, RF, and NN are able to learn complex decision boundaries that better separate active from inactive peptides, even in cases of overlapping distributions.

With regards to the performances of LR, ENET, SVM, RF and NN, while it is clear that the flexibility of SVM and RF is advantageous and leads to a reduction in misclassifications in this setting, comparable performances are possible from the other three methods, as demonstrated in the models of activity for 18-27 aa length peptides and in scenarios 1 and 4 in the simulation study. Further, the simulation study also indicates that SVM and RF may inadvertently include spurious predictor variables in the models, which can hamper interpretability, whereas LR, ENET and NN are likely to exclude these predictor variables.

Clearly each method has strengths and limitations, complicating the selection of the most appropriate approach for predicting AMP activity. To address this issue, the next chapter evaluates methodology to mitigating individual limitations while

leveraging the complementary strengths of the five methods. The goal is to develop a more accurate and reliable prediction framework for antimicrobial peptide activity.

Chapter 4

Ensemble Prediction of AMP Activity

This chapter explores the integration of the five classification methods in the previous chapter within an ensemble framework to enhance predictive accuracy and deepen scientific insights into the key determinants of antimicrobial peptide (AMP) activity. By harnessing the complementary strengths of the methods, the ensemble approach aims to mitigate individual model biases, improve generalization, and provide a more robust understanding of the biological and physicochemical factors that influence AMP efficacy. The chapter begins by evaluating the majority vote method as a strategy for combining base predictors via theory and empirical testing through simulations, with a focus on scenarios involving a small number of base predictors. Special attention is given to the role of variable selection in optimizing the ensemble's performance, as well as potential challenges posed by the lack of independence among base predictors and their heterogeneous performance levels. The impact of these factors is quantified, providing a foundation for understanding how they influence the overall

predictive accuracy and reliability of the ensemble. Following this, the proposed ensemble approach is applied to the DBAASP datasets to map the active regions within the physicochemical space that are associated with AMP activity and the results are compared with previous findings.

While many ensemble methods have been widely and successfully deployed in various application areas (Polikar, 2006), a review of the literature suggests that the specific approach proposed in this study has not yet been evaluated within the context of AMP activity. However, ensemble variable selection and prediction methods have been proposed for improved identification of anti-cancer peptides. For instance, Hu et al. (2011) evaluated accuracy of an ensemble of four base predictors, three of which were neural network (NN) models while the fourth utilised a stabilization matrix alignment method SMM-align (Nielsen et al., 2007), in an investigation to improve prediction of the binding affinity of peptides to Human Leukocyte Antigen Class I molecules. Akbar et al. (2017) ensemble classification of anti-cancer peptides consisted of five base predictors, two of which were neural network (NN) models with the remaining three being support vector machine (SVM), random forest (RF) and K-nearest neighbours (KNN) models. More recently, Ge et al. (2020) introduced a two-stage ensemble for anti-cancer peptide identification whereby a gradient boosting decision tree algorithm LightGBM (Ke et al., 2017) is used for variable selection and initial prediction in the first stage then the initial predictions as taken as input for a SVM in the second stage, while Liang et al. (2021) proposed a stacking ensemble consisting of KNN, Naive Bayes, LightGBM and SVM models, and combining the base predictions using logistic regression (LR).

A crucial issue for designing a good ensemble method is selection of the base predictors (Polikar, 2006). Each base predictor is designed to capture certain patterns or relationships in the data, but it might not be able to capture all the complexities of the underlying problem. To ensure base predictions that are independent and correct with high probability, base predictors should be both diverse and accurate. Thus, a base predictor could be a simple decision tree, support vector machine, logistic regression, linear regression, or other regression algorithms. Another crucial issue for designing a good ensemble method is the combination rule for integrating the outputs of the base predictors (Polikar, 2006).

Methods for combining predictions can be conceptually reasonable, uncomplicated and straightforward to implement. For example, consider a set of m models fitted using training data and denote by y_k the predicted value for outcome Y using the k^{th} model; $k = 1, \ldots, m$. For a continuous outcome, a combined prediction may be obtained using the average $y_c = s_m/m$ where $s_m = \sum_{k=1}^{m} y_k$, while the most frequent (mode) of the m predictions, or a majority vote, may be used for a discrete or categorical outcome. Support for this combination method in the case of a continuous outcome is provided by the fact that the variance of the combined prediction tends to be less than the variance of any individual prediction. Support in the discrete case is provided by the Jury Theorem in an article by Marquis de Condorcet (1784) entitled "an essay on the application of analysis to the probability of decisions rendered by plurality of votes". Developed for situations with a binary outcome, this theorem establishes conditions under which a majority vote is optimal.

4.1 Majority Vote

Suppose M methods are individually used to make a decision between two alternatives. Let the random variables $Y_m;\ m = 1, \ldots, M$ denote the set of decisions and let random variable Y_{MV} denote the combined (majority vote) decision. Without loss of generality, the two alternatives can be denoted by 0 and 1 in which case

Definition 4.1.

$$Y_{MV} = \begin{cases} 1, & \text{if } \frac{S_M}{M} > \frac{1}{2}; \\ 0, & \text{if } \frac{S_M}{M} < \frac{1}{2}; \\ 0 \text{ or } 1 \text{ with equal probability,} & \text{if } \frac{S_M}{M} = \frac{1}{2}, \end{cases}$$

where

$$S_M = \sum_{m=1}^{M} Y_m, \tag{4.1}$$

is the sum of the M random variables.

Theorem 4.1 (Condorcet's Jury Theorem)**.** *If the individual decisions $Y_m,\ m = 1, \ldots, M$ are independent of each other, and each method makes the correct decision with common probability $p > \frac{1}{2}$, then as $M \to \infty$, the probability that the majority vote (MV) decision is correct tends to* 1.

Proof. Without loss of generality, assuming the correct decision is 1 the Y_m are iid Bernoulli random variables with parameter p and hence by Khintchine's Weak Law of Large Numbers (Chung,), $\frac{S_M}{M}$ converges in probability to p. Next, by definition of convergence in probability (Chung,), given any $\epsilon, \delta > 0$ there exists an integer M'

such that for every $M > M'$,

$$\mathbb{P}\left(\left|\frac{S_M}{M} - p\right| < \epsilon\right) > 1 - \delta.$$

Choosing $\epsilon = p - \frac{1}{2}$ and using the fact that the set $\left\{\frac{S_M}{M} : \left|\frac{S_M}{M} - p\right| < p - \frac{1}{2}\right\}$ is a subset of $\left\{\frac{S_M}{M} : \frac{S_M}{M} > \frac{1}{2}\right\}$ gives

$$\mathbb{P}\left(Y_{MV} = 1\right) \geq \mathbb{P}\left(\frac{S_M}{M} > \frac{1}{2}\right) \geq \mathbb{P}\left(\left|\frac{S_M}{M} - p\right| < p - \frac{1}{2}\right) > 1 - \delta, \tag{4.2}$$

which completes the proof. □

The above proof underscores several critical aspects of the Jury Theorem, shedding light on its underlying principles and implications. While the theorem's conclusion is an asymptotic result, meaning it is guaranteed to hold as the number of methods (M) approaches infinity, its validity for small values of M is not established. Additionally, the required conditions of pairwise independence of the predictions, homogeneity (across methods) in the probabilities of a correct decision, and that each method is more likely than not to make the correct decision, may not hold in many practical problems. Explored next is the majority vote (MV) method when the Jury Theorem's conditions are relaxed. Of particular interest is identifying conditions under which an ensemble MV predictor provides a lower prediction error rate than any of its constituent methods.

4.1.1 Monotone properties

Consider a binary classification problem, a set of methods that provide independent predictions with a common probability of a correct prediction and, to emphasise dependence on M and p, write the probability that the majority vote decision is correct as $\mathbb{P}(Y_{MV} = 1; M, p)$. Then by Definition 4.1, as the sum S_M in equation (4.1) follows a binomial distribution,

$$\mathbb{P}(Y_{MV} = 1; M, p) = \mathbb{P}\left(S_M \geq \left\lceil \frac{M}{2} \right\rceil\right) + \frac{1}{2}\mathbb{P}\left(S_M = \frac{M}{2}\right), \tag{4.3}$$

where $\left\lceil \frac{M}{2} \right\rceil = \inf_m \{m : m \in \mathbb{Z},\ m > \frac{M}{2}\}$ denotes the smallest integer greater than $M/2$. As shown below, this probability is monotone in both M and p. These two properties are potentially useful when exploring the majority vote method under finite conditions.

Theorem 4.2 (Monotonicity in p)**.** *If $p_1 \leq p_2$ then $\mathbb{P}(Y_{MV} = 1; M, p_1) \leq \mathbb{P}(Y_{MV} = 1; M, p_2)$ with equality iff $p_1 = p_2$.*

Proof. First note that either $M = 2r + 1$ or $M = 2r$ gives $\left\lceil \frac{M}{2} \right\rceil = r + 1;\ r = 1, 2, \ldots$. Considering equation (4.3) when M is odd, $\mathbb{P}\left(S_M = \frac{M}{2}\right) = 0$ and therefore $\mathbb{P}(Y_{MV} = 1; 1, p) = p$ and

$$\mathbb{P}(Y_{MV} = 1; 2r + 1, p) = \sum_{k=r+1}^{2r+1} \binom{2r+1}{k} p^k (1-p)^{2r+1-k} \text{ for } r = 1, 2, \ldots. \tag{4.4}$$

On the other hand, when M is even, equation (4.3) may be written as

$$\begin{aligned}\mathbb{P}(Y_{MV}=1;2r,p) &= \mathbb{P}(S_{2r}\geq r+1)+\frac{1}{2}\left[\mathbb{P}(S_{2r}\geq r)-\mathbb{P}(S_{2r}\geq r+1)\right],\\ &= \frac{1}{2}\left[\mathbb{P}(S_{2r}\geq r+1)+\mathbb{P}(S_{2r}\geq r)\right].\end{aligned}$$

Since S_{2r} follows a binomial distribution this becomes,

$$= \frac{1}{2}\left[\sum_{k=r+1}^{2r}\binom{2r}{k}p^k(1-p)^{2r-k}+\sum_{k=r}^{2r}\binom{2r}{k}p^k(1-p)^{2r-k}\right]. \quad (4.5)$$

For both the odd and even case, use the relationship between the binomial and incomplete beta distributions that

$$\sum_{k=r}^{M}\binom{M}{k}p^k(1-p)^{M-k}=\frac{M!}{(r-1)!(M-r)!}\int_{t=0}^{p}t^{r-1}(1-t)^{M-r}\,dt \quad (4.6)$$

and replace the sums on the right-hand sides with integrals to get,

$$\mathbb{P}(Y_{MV}=1;2r+1,p)=\frac{(2r+1)!}{r!\;r!}\int_{t=0}^{p}t^r(1-t)^r\,dt \quad (4.7)$$

and

$$\mathbb{P}(Y_{MV}=1;2r,p)=\frac{1}{2}\frac{(2r)!}{r!\;(r-1)!}\int_{t=0}^{p}t^{r-1}(1-t)^{r-1}\,dt.;\;\; r=1,2,\ldots. \quad (4.8)$$

Finally noting that for $0<p_1\leq p_2<1$, $\int_{t=p_1}^{p_2}t^{r-1}(1-t)^{r-1}\,dt\geq 0;\;\; r=1,2,\ldots,$ with equality iff $p_1=p_2$ completes the proof.

Equation (4.6) is derived in Appendix A.3. □

Note that the above result that $\mathbb{P}(Y_{MV} = 1; M, p)$ is strictly monotone increasing in p for fixed M may be inferred given that with larger p each method is more likely to produce a correct prediction and, as a consequence, the majority vote prediction is more likely to be correct. Also, while Condorcet's Jury Theorem guarantees that $\mathbb{P}(Y_{MV} = 1; M, p)$ increases in M for fixed $p > \frac{1}{2}$, it does not require the sequence of probabilities of a correct majority vote decision to be monotone. Next, results on the probability of a correct majority vote decision for M finite are stated and proved.

Theorem 4.3 (Monotonicity in M)**.** *If $M_1 < M_2$ then*

$$\mathbb{P}(Y_{MV} = 1; M_1, p) \leq \mathbb{P}(Y_{MV} = 1; M_2, p) \text{ for } p > \frac{1}{2},$$

$$\mathbb{P}(Y_{MV} = 1; M_1, p) \geq \mathbb{P}(Y_{MV} = 1; M_2, p) \text{ for } p < \frac{1}{2},$$

and when $p = \frac{1}{2}$,

$$\mathbb{P}\left(Y_{MV} = 1; M, p = \frac{1}{2}\right) = \frac{1}{2} \text{ for all } M = 1, 2, \ldots.$$

Proof. First, for any $0 < p < 1$,

$$\mathbb{P}(Y_{MV} = 1; M - 1, p) = \mathbb{P}(Y_{MV} = 1; M, p), \quad \text{when } M \text{ is an even number.} \tag{4.9}$$

To see this when $M = 2$ simply note that, by the definition provided in equation (4.3), $\mathbb{P}(Y_{MV} = 1; M = 2, p) = 2p(1-p)/2 + p^2 = p = \mathbb{P}(Y_{MV} = 1; M = 1, p)$. For $M > 2$ consider equation (4.8) with $= 2r + 2;\ r = 1, 2, \ldots,$ to get

$$\mathbb{P}(Y_{MV} = 1; 2r + 2, p) = \frac{1}{2} \frac{(2r+2)!}{(r+1)!\ r!} \int_{t=0}^{p} t^r (1-t)^r \, dt,$$

which by writing $(2r+2)! = (2r+2) \times (2r+1)!$ and $(r+1)! = (r+1) \times r!$ and simplifying leads to

$$\mathbb{P}(Y_{MV} = 1; 2r+2, p) = \frac{(2r+1)!}{r!\ r!} \int_{t=0}^{p} t^r (1-t)^r \, dt = \mathbb{P}(Y_c = 1; 2r+1, p). \tag{4.10}$$

Hence, to complete the proof, it is sufficient to consider the sequence of probabilities when M is odd.

Now when $M = 3,\ \mathbb{P}(Y_{MV} = 1; 3, p) = \sum_{k=2}^{3} \binom{3}{k} p^k (1-p)^{3-k} = 3p^2(1-p) + p^3$ and when $M = 1,\ \mathbb{P}(Y_{MV} = 1; 1, p) = p$, so the difference is,

$$\mathbb{P}(Y_{MV} = 1; 3, p) - \mathbb{P}(Y_{MV} = 1; 1, p) = 3p^2(1-p) + p^3 - p = p(p-1)(1-2p).$$

Thus as $p(p-1) < 0$,

$$\mathbb{P}(Y_{MV} = 1; 3, p) - \mathbb{P}(Y_{MV} = 1; 1, p) \begin{cases} < 0, & \text{for } 1-2p > 0; \\ = 0, & \text{for } 1-2p = 0; \\ > 0, & \text{for } 1-2p < 0. \end{cases} \tag{4.11}$$

Next consider $\mathbb{P}(Y_{MV} = 1; 2r+3, p) = \mathbb{P}(S_{2r+3} \geq r+2),\ \ r = 1, 2, \ldots$. Recalling that S_{2r+3} is the sum of $2r+3$ Bernoulli random variables, the event $\{s : S_{2r+3} \geq r+2\}$ can be written as the union of disjoint sets $\{s :\ S_{2r+3} \geq r+2 \text{ and } S_{2r+1} \geq r+2\}$, $\{s :\ S_{2r+3} \geq r+2 \text{ and } S_{2r+1} = r+1\}$ and $\{s :\ S_{2r+3} \geq r+2 \text{ and } S_{2r+1} = r\}$, where S_{2r+1} is the sum of $2r+1$ Bernoulli random variables. Consider each disjoint set in turn.

First, as $S_{2r+1} \geq r+2$ implies $S_{2r+3} \geq r+2$ and $\mathbb{P}\left(S_{2r+1} \geq r+2\right) = \mathbb{P}\left(S_{2r+1} \geq r+1\right) -$

$\mathbb{P}\left(S_{2r+1} = r+1\right)$, applying the chain rule for conditional probability gives

$$\begin{aligned}
&\mathbb{P}\left(S_{2r+3} \geq r+2,\ S_{2r+1} \geq r+2\right) \\
&= \mathbb{P}\left(S_{2r+3} \geq r+2 \mid S_{2r+1} \geq r+2\right) \mathbb{P}\left(S_{2r+1} \geq r+2\right), \\
&= \mathbb{P}\left(S_{2r+1} \geq r+2\right), \\
&= \mathbb{P}\left(S_{2r+1} \geq r+1\right) - \binom{2r+1}{r+1} p^{r+1}(1-p)^r.
\end{aligned}$$

Second, given that $S_{2r+1} = r+1$, the sum S_{2r+3} will be greater than or equal to $r+2$ if either Y_{2r+2} or Y_{2r+3}, or both, equals one. Thus, by the chain rule for probability,

$$\mathbb{P}\left(S_{2r+3} \geq r+2,\ S_{2r+1} = r+1\right) = \left(2p(1-p) + p^2\right) \binom{2r+1}{r+1} p^{r+1}(1-p)^r.$$

By a similar argument,

$$\mathbb{P}\left(S_{2r+3} \geq r+2,\ S_{2r+1} = r\right) = p^2 \binom{2r+1}{r} p^r(1-p)^{r+1}.$$

Therefore, summing these three probabilities and using that $\binom{2r+1}{r} = \binom{2r+1}{r+1}$ give,

$$\begin{aligned}
&\mathbb{P}(Y_{MV} = 1; 2r+3, p) = \mathbb{P}(Y_{MV} = 1; 2r+1, p) + \\
&\binom{2r+1}{r} \left[-1 + 2p(1-p) + p^2 + p(1-p)\right] p^{r+1}(1-p)^r, \\
&= \mathbb{P}(Y_{MV} = 1; 2r+1, p) + \binom{2r+1}{r} (p-1)(1-2p) p^{r+1}(1-p)^r,
\end{aligned}$$

and hence for $r = 1, 2, \ldots,$

$$\mathbb{P}(Y_{MV} = 1; 2r+3, p) - \mathbb{P}(Y_{MV} = 1; 2r+1, p) \begin{cases} < 0, & \text{for } 1-2p > 0; \\ = 0, & \text{for } 1-2p = 0; \\ > 0, & \text{for } 1-2p < 0, \end{cases} \tag{4.12}$$

which completes the proof. □

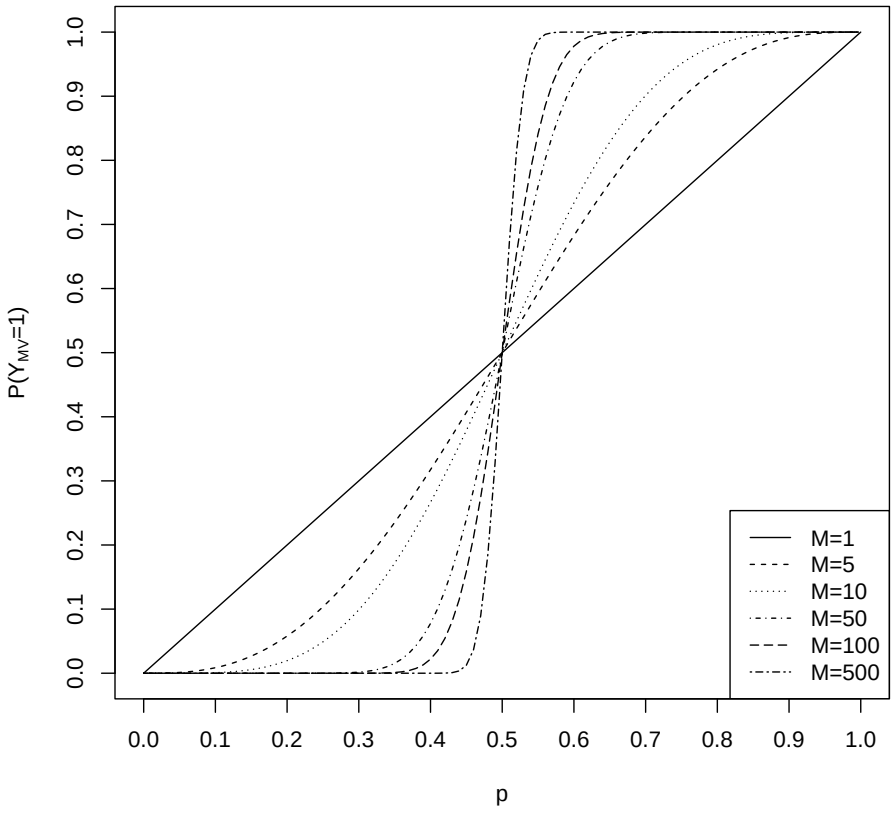


Figure 4.1: Probability of a correct prediction p of a base predictor in a majority vote ensemble against the probability of a correct majority vote prediction $\mathbb{P}(Y_{MV} = 1)$ for various ensemble sizes M.

An immediate consequence of Theorem 4.3 is that majority vote is expected to lead

to a lower error probability than the single method with probability p of a correct prediction, provided that the base predictors in the MV ensemble are pairwise independent and with each having p greater than one-half. Plots of the probability of a correct MV prediction against p in Figure 4.1 illustrate how quickly and by how much the MV probability changes with the number of methods M, generated based on equation (4.3). In particular, for an ensemble of $M = 5$ base predictors with individual probability of a correct prediction $p = 0.80$, the probability of the majority vote (MV) yielding a correct prediction exceeds 94%, and when individual probabilities increase to $p = 0.90$, MV probability approaches near certainty at 99%. Given that the accuracy estimates of the learning methods used to analyse the DBAASP datasets (Section 3.3, Chapter 3) falls within the range of approximately 0.80 to 0.90, it is reasonable to expect that a majority vote ensemble could significantly enhance prediction accuracy. Also, it is worthwhile to note that, as the plots in Figure 4.1 show, with accuracy p between 0.80 and 0.90, there is very little gain from increasing the number of models M to more than five.

4.2 Effect of Base Predictors on Majority Vote Ensemble Properties

The fundamental properties of the majority vote method, established above, serve here as the basis for exploring the small-sample (i.e., small M) behaviour of the MV method in predicting active and inactive peptides under relaxed conditions of the Jury Theorem. Consider M binary classification methods with accuracies $p_1, \ldots, p_M$ and write the accuracy of the m^{th} method p_m as the weighted average of the method's

performance on active and inactive outcomes with weights corresponding to their prevalence in the population. That is write,

$$p_m = p_{A,m}\mathbb{P}(A) + p_{O,m}\mathbb{P}(O), \tag{4.13}$$

where $p_{A,m}$ represents the probability of correctly predicting an active outcome, $p_{O,m}$ represents the probability of correctly predicting an inactive outcome, and $\mathbb{P}(A), \mathbb{P}(O)$ are the respective proportions of active and inactive outcomes in the population. How the MV performance measures are impacted under different scenarios for the base predictors are quantified here. In practice, these probabilities are unknown and have to be estimated. For instance, accuracy AC is an estimate of p, while sensitivity SN and specificity SP provide respective estimates of p_A and p_O.

4.2.1 Independent base predictors

In scenarios where the base predictors are independent, accuracy of the MV method is provided by equation (4.3),

$$p_{MV} = \mathbb{P}\left(S_M \geq \left\lceil \frac{M}{2} \right\rceil\right) + \frac{1}{2}\mathbb{P}\left(S_M = \frac{M}{2}\right), \tag{4.14}$$

where S_M is the number of correct (active or inactive) predictions among the M methods, and follows a Poisson-binomial distribution with probability mass function,

$$\mathbb{P}(S_M = k) = \sum_{F \in \mathcal{F}_k} \prod_{m \in F} p_m \prod_{m' \in F^c} (1 - p_{m'}). \tag{4.15}$$

Here $\mathcal{F}_k$ is the set of all subsets of k integers that can be selected from the set $\{1, \ldots, M\}$, and $F^c = \{1, \ldots, M\} \setminus F$ is the set of integers in $\{1, \ldots, M\}$ that are not

in F. Sensitivity of the MV method $p_{A,MV}$ is similarly defined by equation (4.16) with S_M now the number of correct active predictions and accuracy p_m in equation (4.15) replaced by base predictor sensitivity $p_{A,m}$. Specificity of the MV method $p_{O,MV}$ is similarly defined.

Homogeneous base predictors

In circumstances when all M base predictors in the ensemble are independent with common accuracy p, the Poisson-binomial probability mass function in equation (4.15) reduces to that of a binomial distribution and the results derived in the previous section are directly applicable. Thus for $M > 1$,

$$p_{MV} = \begin{cases} \sum\limits_{k=\frac{M+1}{2}}^{M} \binom{M}{k} p^k (1-p)^{M-k}, & \text{for } M \text{ odd;} \\ \sum\limits_{k=\frac{M}{2}}^{M} \binom{M}{k} p^k (1-p)^{M-k} + \frac{1}{2}\binom{M}{\frac{M}{2}} p^{\frac{M}{2}} (1-p)^{\frac{M}{2}}, & \text{for } M \text{ even.} \end{cases} \tag{4.16}$$

The red curve in Figure 4.2 (on the left side), which represents the probability gain $(p_{MV} - p)$ achieved by the MV ensemble of $M = 5$ base predictors with $p > 0.5$, demonstrates that MV consistently outperforms an individual base predictor. Moreover, for p within the range 0.6 to 0.9, MV improves accuracy by over 10%. Figure 4.2 (on the right side) also includes plots of the probability that an estimate of $\hat{p}_{MV}$ is lower than an estimate of $\hat{p}$,

$$\mathbb{P}(\hat{p}_{MV} < \hat{p}), \tag{4.17}$$

for sample sizes of $N = 80, 125, 160$, and 250. Both plots are generated by applying equation (4.16) for probability p within the range 0.5 to 1 across various data sizes

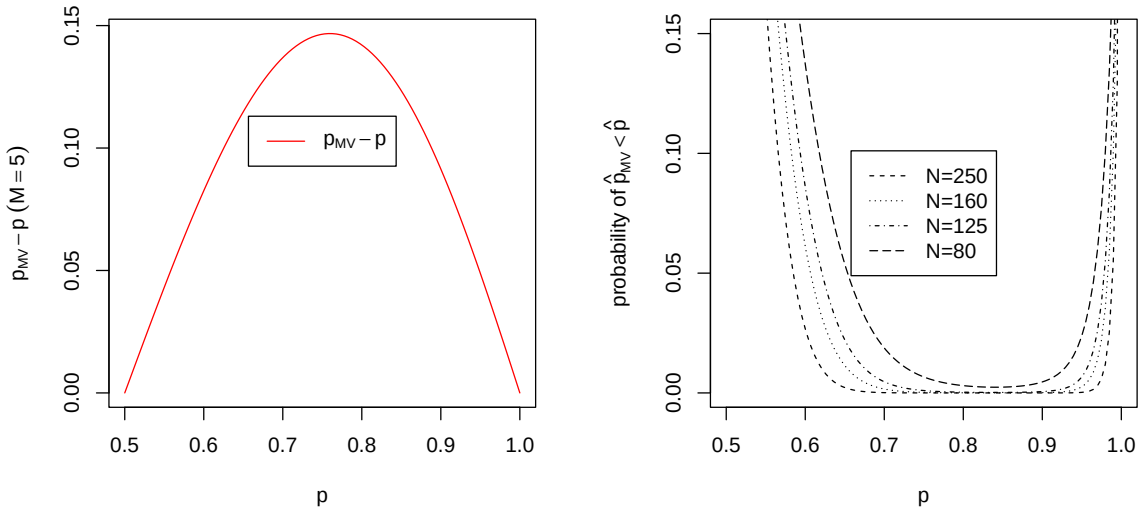


Figure 4.2: Plot of $(p_{MV} - p)$ against values of p (red line on the left side) when $M = 5$. Also shown are probabilities $\mathbb{P}(\hat{p}_{MV} < \hat{p})$ against p for selected test data sizes N (black lines on the right side).

N. These values correspond to the test data sizes used in the AMP dataset analyses. These plots provide insight into the likelihood of an evaluation where MV is observed to under-perform relative to a single base predictor. The probability is computed under the assumption that the estimated proportion $\hat{p}_{MV}$ follows a normal distribution with mean p_{MV} and variance $p_{MV}(1 - p_{MV})/N$. A similar assumption is made for $\hat{p}$.

Heterogeneous base predictors

Accuracy of the MV ensemble of $M = 5$ base predictors with varying levels of performance $p_m;\ m = 1, \ldots, 5$ is evaluated here. The values of p_m (0.5 to 1) used in the evaluation were motivated by the ranges of the estimates of accuracy, sensitivity and specificity (0.52 to 0.98) obtained in the AMP data analyses, provided in Section 3.3

of Chapter 3. Here, the probability of a correct majority vote decision p_{MV} is calculated as for the homogeneous case, but with the binomial probabilities replaced with Poisson-binomial probabilities in equation (4.15). These probabilities are calculated using the *poisbinom* package (Olivella & Shiraito, 2017), for p_m evenly distributed over a range of values.

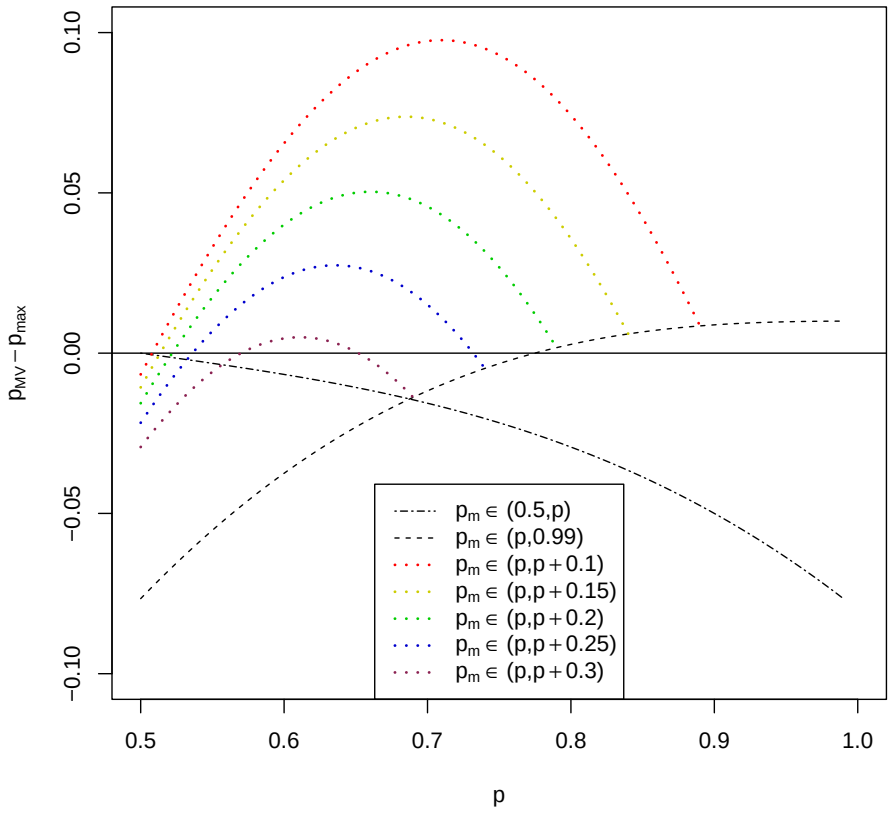


Figure 4.3: Plots of $(p_{MV} - p_{max})$ against values of p, where p_{max} is accuracy of the best performing among $M = 5$ base predictors, for selected ranges of values for base predictor probability p_m.

Figure 4.3, which illustrates the difference between p_{MV} and $p_{max} = \max_m\{p_m\}$, sheds light on conditions under which the majority vote strategy may underperform relative

to the best-performing predictor in the ensemble. For instance, as the long-dashed line in the figure illustrates, MV consistently performs worse than the best-performing base predictor when p_m is approximately evenly distributed over the interval $(0.5, p)$, where $p > 0.5$. Moreover, the performance gap widens as this interval expands. This is confirmed by the violet (interval width 0.3) and blue (interval width 0.25) dotted lines. Thus MV is more likely to underperform relative to the best base predictor when individual performances are spread over a wide range. On the other hand, the performance of MV improves as the range of individual performances narrows, highlighting the importance of homogeneity in predictor performance for the effectiveness of the MV strategy.

4.2.2 Dependent base predictors

First, consider the scenario where all base predictions are identical and thus the set of base predictors are completely dependent. For M homogeneous base predictors, each with individual performances p, it is clear that the performance of the MV ensemble p_{MV} is equal to p when the number of completely dependent base predictors exceeds $M/2$. In such a situation there is no benefit in using the ensemble for prediction.

Next, suppose less than half of the base predictors are completely dependent. Then for $m_2 \leq M/2$ dependent predictors, the number of correct predictions S_M^* equals k in two scenarios: when the dependent set and $k - m_2$ of the independent predictors are correct, or when the dependent set is incorrect and k of the independent predictors are correct. This leads to,

$$\mathbb{P}(S_M^* = k) = p\mathbb{P}(S_{M-m_2} = k - m_2) + (1-p)\mathbb{P}(S_{M-m_2} = k);\ M = 3, 4, 5, \ldots, \quad (4.18)$$

where S_{M-m_2} represents the sum of $M - m_2$ independent base predictors and follows a binomial distribution with parameters $M - m_2$ and p.

Theorem 4.4. *Consider an ensemble with M base predictors with homogeneous performance p and assume a given pair of base predictors is completely dependent but independent of the remaining $M - 2$. Assume further that the remaining $M - 2$ are mutually independent. Then for M odd, the performance of the majority vote ensemble is equivalent to that of the ensemble that excludes one predictor from the dependent pair.*

Proof. Writing $M = 2r + 1$ and using equation (4.18) with $m_2 = 2$ gives performance of the ensemble as

$$\begin{aligned} p_{MV} &= \sum_{k=r+1}^{2r+1} \mathbb{P}(S^*_{2r+1} = k), \\ &= \sum_{k=r+1}^{2r+1} \left[p\mathbb{P}(S_{2r-1} = k-2) + (1-p)\mathbb{P}(S_{2r-1} = k) \right]. \end{aligned}$$

Extracting the first and last terms from the first sum, and noting that the maximum possible value of S_{2r-1} is $2r - 1$, this can be written as,

$$\begin{aligned} =& p\mathbb{P}(S_{2r-1} = r-1) + p\mathbb{P}(S_{2r-1} = 2r-1) \\ &+ \sum_{k=r+2}^{2r} p\mathbb{P}(S_{2r-1} = k-2) + \sum_{k=r+1}^{2r-1} (1-p)\mathbb{P}(S_{2r-1} = k). \end{aligned} \tag{4.19}$$

Using that S_{2r-1} has a binomial distribution the first term in the above sum is

$$\begin{aligned} p\mathbb{P}(S_{2r-1} = r-1) =& p\binom{2r-1}{r-1}p^{r-1}(1-p)^r = \frac{(2r-1)!}{(r-1)!r!}p^r(1-p)^r, \\ &= \frac{(2r)!}{(r)!r!}\frac{r}{2r}p^r(1-p)^r = \frac{1}{2}\mathbb{P}(S_{2r} = r), \end{aligned} \tag{4.20}$$

while the second term,

$$p\mathbb{P}(S_{2r-1} = 2r-1) = p\binom{2r-1}{2r-1}p^{2r-1} = p^{2r} = \mathbb{P}(S_{2r} = 2r). \tag{4.21}$$

Next, shifting the summation index k in the first sum by one unit right, the two sums in equation (4.19) can be merged to get,

$$\begin{aligned} &= \sum_{k=r+1}^{2r-1} \left[p\mathbb{P}(S_{2r-1} = k-1) + (1-p)\mathbb{P}(S_{2r-1} = k)\right], \\ &= \sum_{k=r+1}^{2r-1} \left[p\binom{2r-1}{k-1}p^{k-1}(1-p)^{2r-k} + (1-p)\binom{2r-1}{k}p^k(1-p)^{2r-k-1}\right], \\ &= \sum_{k=r+1}^{2r-1} \left[\frac{(2r-1)!}{(k-1)!(2r-k)!} + \frac{(2r-1)!}{k!(2r-k-1)!}\right]p^k(1-p)^{2r-k}, \\ &= \sum_{k=r+1}^{2r-1} \left[\frac{(2r-1)!}{k!(2r-k)!}(k+2r-k)\right]p^k(1-p)^{2r-k} = \sum_{k=r+1}^{2r-1}\mathbb{P}(S_{2r} = k). \end{aligned} \tag{4.22}$$

Finally, put the expressions obtained in equations (4.20), (4.21) and (4.22) in equation (4.19) to get,

$$p_{MV} = \frac{1}{2}\mathbb{P}(S_{2r} = r) + \sum_{k=r+1}^{2r}\mathbb{P}(S_{2r} = k),$$

which, by equation (4.16), is the majority vote performance with $2r$ independent and homogeneous base predictors. □

A similar result does not hold when M is even, but instead, the ensemble with two

completely dependent base predictors performs slightly worse than the ensemble with one of the dependent pairs removed.

Proof. To see this, first write $M = 2r$ and use equations (4.16) and (4.18) with $m_2 = 2$ to get,

$$\begin{aligned} p_{MV} &= \sum_{k=r+1}^{2r} \mathbb{P}(S^*_{2r} = k) + \frac{1}{2}\mathbb{P}(S^*_{2r} = r), \\ &= \sum_{k=r+1}^{2r} \left[p\mathbb{P}(S_{2r-2} = k-2) + (1-p)\mathbb{P}(S_{2r-2} = k)\right] \\ &\quad + \frac{1}{2}\left[p\mathbb{P}(S_{2r-2} = r-2) + (1-p)\mathbb{P}(S_{2r-2} = r)\right]. \end{aligned}$$

Next, following a similar approach as in the above proof,

$$\sum_{k=r+1}^{2r} \left[p\mathbb{P}(S_{2r-2} = k-2) + (1-p)\mathbb{P}(S_{2r-2} = k)\right] = \mathbb{P}(S_{2r-1} \geq r+1) + p\mathbb{P}(S_{2r-2} = r-1).$$

Now it can also be shown that,

$$p\mathbb{P}(S_{2r-2} = r-1) + (1-p)\mathbb{P}(S_{2r-2} = r) = \mathbb{P}(S_{2r-1} = r),$$

so substituting for $p\mathbb{P}(S_{2r-2} = r-1)$ gives,

$$p_{MV} = \mathbb{P}(S_{2r-1} \geq r) + \frac{1}{2}\left[p\mathbb{P}(S_{2r-2} = r-2) - (1-p)\mathbb{P}(S_{2r-2} = r)\right].$$

The result follows by noting that $\mathbb{P}(S_{2r-1} \geq r)$ is the majority vote performance with $2r-1$ independent and homogeneous base predictors, and that the deviation from

this probability,

$$p\mathbb{P}(S_{2r-2} = r-2) - (1-p)\mathbb{P}(S_{2r-2} = r) = \binom{2r-2}{r-2} p^{r-1}(1-p)^{r-1}(1-2p),$$

is less than zero for $p > 1/2$. □

The results presented above quantify the performance of the MV ensemble when using homogeneous base predictors and confirm that achieving independence among the base predictors is a critical requirement for optimizing ensemble performance. However, the results also strongly suggest that an ensemble composed of homogeneous base predictors will generally not underperform relative to any individual predictor within the ensemble and implies that, even in the absence of perfect independence, the ensemble approach will provide a form of risk mitigation, ensuring that the MV prediction is at least as reliable as that of an individual base predictor.

The emerging pattern for homogeneous base predictors observed here, when combined with the results from the previous section, suggests that in the case of heterogeneous base predictors, the MV ensemble will exhibit a bias towards base predictors that are dependent on one another. This dependency-driven bias can lead to a situation where the ensemble's performance falls short of the best-performing individual base predictor. However, despite this limitation, the ensemble will outperform the worst-performing base predictor.

4.3 Majority Vote with Five Common Classification Methods

This section evaluates the majority vote ensemble method with Logistic Regression, Elastic Net Logistic Regression, Support Vector Machines, Random Forests and Neural Network base predictors, providing further insights into its strengths and limitations. First, the robustness and reliability of the method in a controlled environment is assessed by its performances in the four simulated scenarios introduced in Chapter 3. The base predictors used in this evaluation are the best-performing models identified in Chapter 3, ensuring that the ensemble is built on strong individual components. Next, the MV method is applied to the problem of predicting AMP activity using the DBAASP datasets analysed in Chapter 3 and its performance is compared with those of the five base predictors.

4.3.1 Evaluation of majority vote using simulated scenarios

Recall that the simulation scenarios explored in Chapter 3 involve four predictor variables X_1, X_2, X_3, X_4 and that best subset selection was employed to identify the optimal model for each of the five methods considered. Additionally, quadratic, cubic, and quartic models were fitted for the Logistic Regression (LR), Elastic Net (ENET), and Neural Network (NN) methods to capture non-linear relationships in the data. Here, a comparative analysis is conducted to evaluate the performance of these five individual models against the aggregated performance of the MV ensemble. In particular, this evaluation provides valuable insights into the relationship between the MV ensemble's performance and the dependency and heterogeneity of its base predictors.

The dependence between base predictors is evaluated using Fisher Exact Test (1922), a widely-used statistical significance test to determine if there are nonrandom associations between two categorical variables in a contingency table. This test is particularly useful for small sample sizes or when the data is sparse, as it calculates the exact probability of observing the data which, under the assumption of independence, follows a hypergeometric distribution. In the implementation of this test, it is important to note that not all predicted outcomes are included for the following reason. Two base predictors are considered dependent if they produce identical predictions. However, in the evaluations conducted here, many outcomes are correctly predicted by all five base predictors and these "easy-to-predict" outcomes can bias the hypothesis test results towards indicating dependence, even when it may not exist. To mitigate this bias, such outcomes are excluded from the independence test procedures.

Table 4.1: Cross-tabulated predictions of two base predictors.

		Base predictor 1	
		Correct	Wrong
Base predictor 2	Correct	n_a	n_b
	Wrong	n_c	n_d

Similarity in performance of pairs of base predictors is tested using McNemar test (1947), a widely used hypothesis test procedure to determine if there is a difference in a dichotomous outcome between two related groups. Here this test compares the predictive capabilities of two base predictors by focussing on the number of times the two disagree, shown by n_b and n_c in Table 4.1. The test statistic, with continuity

correction, and which follows a χ^2 distribution with 1 degree of freedom, is given by

$$T = \frac{(|n_b - n_c| - 1)^2}{n_b + n_c}.$$

The null hypothesis of similar predictive performance is rejected for large p-values.

Note that the sensitivity rates of the base predictors are exceptionally high in the evaluation here, reaching perfect or near-perfect levels across all four scenarios. Such performances leave little room for further improvement in sensitivity and consequently, the evaluation in this section primarily focuses on specificity and overall accuracy. As these metrics are equally able to provide meaningful insights into the performance of the ensemble method, this shift in focus does not detract from the primary objective of this section.

Scenario 1

Table 4.2 presents the performance metrics for the best-fitting model from each method based on accuracy alongside the corresponding MV ensemble performance

Table 4.2: **Scenario 1**. Performance measures for the majority vote ensemble and its five base predictors.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=4)	0	500	87	413	1.00	0.83	0.91	0.85	0.087	$X_1X_2X_3$
ENET (d=4)	0	500	39	461	1.00	0.92	0.96	0.93	0.039	$X_1X_2X_4$
SVM	0	500	41	459	1.00	0.92	0.96	0.92	0.041	X_1X_2
RF	0	500	22	478	1.00	0.96	0.98	0.96	0.022	X_1X_2
NN (d=2)	0	500	23	477	1.00	0.95	0.98	0.96	0.023	$X_1X_2X_4$
MV	0	500	31	469	1.00	0.94	0.97	0.94	0.031	

for Scenario 1 dataset. Specifically, the best models are as follows: for LR, a quartic polynomial in X_1, X_2, X_3; for ENET another quartic polynomial but in X_1, X_2, X_4; for NN the quadratic in X_1, X_2, X_4; and for SVM and RF, models using predictor variables X_1, X_2.

Table 4.3: **Scenario 1**: Cross tabulations of correct (C) and incorrect (W) predictions of **inactive outcomes** for pairs of base predictors. The p-values of tests assessing similarity in predictive capability using McNemar's test procedure are given in brackets.

		LR		ENET		SVM		RF	
		C	W	C	W	C	W	C	W
ENET	C	413	48						
	W	0	39						
		(0.0000)							
SVM	C	412	47	451	8				
	W	1	40	10	31				
		(0.0000)		(0.8137)					
RF	C	413	65	461	17	459	19		
	W	0	22	0	22	0	22		
		(0.0000)		(0.0001)		(0.0000)			
NN	C	413	64	461	16	459	18	471	6
	W	0	23	0	23	0	23	7	16
		(0.0000)		(0.0002)		(0.0001)		(1.0000)	

Results in Table 4.2 demonstrate that with a specificity (SP) of 0.94 and accuracy (AC) of 0.97, performance of MV ensemble is comparable to the best-performing base predictors. However, MV does not demonstrate superior performance over these individual predictors. This outcome can be further understood by examining the homogeneity and independence of the base predictors.

Results of the pairwise tests for similar capability p_O in predicting an inactive outcome provided in Table 4.3 show that the LR model has significantly lower specificity than the other four base predictors. Additionally, the ENET and SVM models exhibit similar specificity, though both are significantly lower than that of the RF and NN models. Furthermore, the p-values of the independence tests for Scenario 1 in Table 4.4 indicate that, while there is no evidence of dependence between LR and the other four methods, the prediction models constructed by ENET, SVM, RF and NN are highly significantly dependent. This strong interdependence among four of the five models, and to a lesser extent the heterogeneity in predictive capability, are likely contributors to the observed performance of the MV ensemble in this scenario.

Table 4.4: P-values from Fisher Exact Tests of independence between pairs of base predictors across four scenarios.

Pairs of	P-value			
methods	Scenario 1	Scenario 2	Scenario 3	Scenario 4
(LR, ENET)	1.0000	0.0000	0.0000	0.0000
(LR, SVM)	0.4659	0.0008	0.7146	0.0012
(LR, RF)	1.0000	0.0768	0.0629	0.0495
(LR, NN)	1.0000	0.0002	0.6145	0.0000
(ENET, SVM)	0.0000	0.0005	0.5762	0.0012
(ENET, RF)	0.0000	0.0001	0.0013	0.0495
(ENET, NN)	0.0000	0.0000	0.0012	0.0000
(SVM, RF)	0.0000	0.0000	0.0000	0.0198
(SVM, NN)	0.0000	0.0025	0.0140	0.1746
(RF, NN)	0.0000	0.0013	0.1693	0.3406

Scenario 2

As discussed in Chapter 3, a key difference in the training data here compared to Scenario 1 is that certain regions of the variable space are now densely sampled, while others are sparsely sampled. Additionally, the observed predictor variables are subject to measurement errors, resulting in training data with ambiguous boundaries between active and inactive outcomes. Table 4.5 presents the performance metrics for the MV ensemble and its base predictors in this scenario. First, focusing on the estimated specificity, MV exhibits a four percentage point reduction compared to the best-performing base predictor. The impact on overall accuracy is however minimal as all models achieve perfect sensitivity.

To delve deeper into this performance analysis, Table 4.6 reveals that while there is no evidence for differences in specificity of the LR, SVM and RF models, the ENET model stands out with significantly higher specificity compared to all other base predictors. While this heterogeneity in specificity among the base predictors may have contributed to the observed performance of the MV ensemble, a more plausible

Table 4.5: **Scenario 2**. Performance measures for the majority vote ensemble and its five base predictors.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=4)	0	500	111	389	1.00	0.78	0.89	0.82	0.111	X_1X_2
ENET (d=4)	0	500	79	421	1.00	0.84	0.92	0.86	0.079	X_1X_2
SVM	0	500	124	376	1.00	0.75	0.88	0.80	0.124	X_1X_2X4
RF	0	500	118	382	1.00	0.76	0.88	0.81	0.118	$X_1X_2X_3X4$
NN (d=2)	0	500	98	402	1.00	0.80	0.90	0.84	0.098	X_1X_2
MV	0	500	102	398	1.00	0.80	0.90	0.83	0.102	

Table 4.6: **Scenario 2**: Cross tabulations of correct (C) and incorrect (W) predictions of **inactive outcomes** for pairs of base predictors. The p-values of tests assessing similarity in predictive capability using McNemar's test procedure are given in brackets.

		LR		ENET		SVM		RF	
		C	W	C	W	C	W	C	W
ENET	C	389	32						
	W	0	79						
		(0.0000)							
SVM	C	356	20	365	11				
	W	33	91	56	68				
		(0.0993)		(0.0000)					
RF	C	354	28	370	12	363	19		
	W	35	83	51	67	13	105		
		(0.4497)		(0.0000)		(0.3768)			
NN	C	367	35	387	15	341	61	343	59
	W	22	76	34	64	35	63	39	59
		(0.1120)		(0.0101)		(0.0107)		(0.0549)	

explanation is revealed by an examination of the p-values of the independence tests for Scenario 2 in Table 4.4. The table shows that there are highly significant dependencies between all pairs of models, except for LR and RF, which is likely to have amplified the impact of the collective predictions, thereby overshadowing the individual strength of the ENET model and ultimately shaping the ensemble's performance.

Scenario 3

While the MV ensemble's estimated specificity for this scenario is four percentage points lower than that of the NN model and two points lower than the RF model, as shown in Table 4.7, its overall accuracy remains comparable to the best-performing base predictors. This is primarily driven by the perfect sensitivity achieved by the ensemble, which stems from the flawless performance of the LR, ENET, and SVM models in correctly identifying all active outcomes in the test dataset.

A possible contributing factor to the MV ensemble's performance in the scenario here is the significantly lower specificity of the LR model compared to the other base predictors, see Table 4.8. This weaker performance potentially directly impacts the ensemble's overall specificity, as the majority vote mechanism aggregates predictions from all base predictors, including those with suboptimal performance. Additionally, as the p-values for Scenario 3 in Table 4.4 show, the independence of the LR model from the RF and NN models, the two best-performing base predictors, possibly further exacerbates the issue. Since the LR model's predictions are not correlated with those

Table 4.7: **Scenario 3**. Performance measures for the majority vote ensemble and its five base predictors.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=4)	0	500	180	320	1.00	0.64	0.82	0.74	0.180	X_1X_2
ENET (d=4)	0	500	127	373	1.00	0.75	0.87	0.80	0.127	$X_1X_2X_3$
SVM	0	500	145	355	1.00	0.71	0.86	0.78	0.145	X_1X_2X4
RF	7	493	117	383	0.98	0.77	0.88	0.81	0.124	X_1X_2X4
NN (d=4)	39	461	104	396	0.92	0.79	0.86	0.82	0.143	X_1X_2
MV	0	500	126	374	1.00	0.75	0.87	0.80	0.126	

Table 4.8: **Scenario 3**: Cross-tabulations of correct (C) and incorrect (W) predictions of **inactive outcomes** for pairs of base predictors. The p-values of tests assessing similarity in predictive capability using McNemar's test procedure are given in brackets.

		LR		ENET		SVM		RF	
		C	W	C	W	C	W	C	W
ENET	C	316	57						
	W	4	123						
		(0.0000)							
SVM	C	288	67	307	48				
	W	23	113	66	79				
		(0.0006)		(0.1113)					
RF	C	301	82	330	53	349	34		
	W	19	98	43	74	6	111		
		(0.0000)		(0.3583)		(0.0000)			
NN	C	304	92	346	50	332	64	349	47
	W	16	88	27	77	23	81	34	70
		(0.0000)		(0.0122)		(0.0000)		(0.1824)	

of RF and NN, its errors are not likely to be offset by the stronger performance of these models, but instead, independence possibly allows the LR model's lower specificity to have a more pronounced negative influence on the ensemble's results.

Scenario 4

The performance of the MV ensemble in this scenario, produced in Table 4.9, is primarily driven by the LR, ENET and NN models. These models exhibit a high degree of homogeneity in their predictive capabilities, as evidenced by the results presented

Table 4.9: **Scenario 4**. Performance measures for the majority vote ensemble and its five base predictors.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
LR (d=2)	19	481	4	496	0.96	0.99	0.98	0.99	0.023	X_1X_2
ENET (d=2)	19	481	4	496	0.96	0.99	0.98	0.99	0.023	X_1X_2
SVM	61	439	0	500	0.88	1.00	0.94	1.00	0.061	$X_1X_2X_3$
RF	60	440	1	499	0.88	0.99	0.89	0.99	0.061	$X_1X_2X_4$
NN (d=2)	15	485	8	492	0.97	0.98	0.98	0.98	0.023	X_1X_2
MV	21	479	4	496	0.96	0.99	0.98	0.99	0.025	

in Table 4.10. Furthermore, the p-values for Scenario 4 in Table 4.4 indicate that these models are highly significantly dependent on one another. This interdependence is particularly pronounced between the LR and ENET models, which produce identical predictions on the test dataset (see Table 4.10). Additionally, the results in Table 4.10 reveal no significant difference in the predictive capabilities of the SVM and RF models. These two models also demonstrate a high level of dependence, as supported by the statistical evidence provided in Table 4.4. The high dependence among the two sets of base predictors observed in this scenario, combined with their strong individual performance, highlights the limited potential for further improvement in the MV ensemble's predictive accuracy. However, it also underscores the reliability and stability of the ensemble in leveraging the collective strength of its constituent models.

Table 4.10: **Scenario 4**: Cross tabulations of correct (C) and incorrect (W) predictions of **all outcomes** for pairs of base predictors. The p-values of tests assessing similarity in predictive capability using McNemar's test procedure are given in brackets.

		LR		ENET		SVM		RF	
		C	W	C	W	C	W	C	W
ENET	C	977	0						
	W	0	23						
		( )							
SVM	C	935	4	935	4				
	W	42	19	42	19				
		(0.0000)		(0.0000)					
RF	C	934	5	934	5	919	20		
	W	43	18	43	18	20	41		
		(0.0000)		(0.0000)		(1.0000)			
NN	C	970	7	970	7	930	47	930	47
	W	7	16	7	16	9	14	9	14
		(1.0000)		(1.0000)		(0.0000)		(0.0000)	

4.3.2 Majority vote prediction of AMP activity

The preceding evaluations of the majority vote ensemble indicate that its performance is optimal under conditions of homogeneity and independence and, moreover, it has the ability to align with stronger base predictors and remain competitive even in less-than-ideal scenarios. This motivates the exploration in this section of applying the MV method to predict AMP activity using the DBAASP datasets analysed in Chapter 3. Table 4.11 presents the performance metrics for the MV method with base

predictors the optimal models for predicting activity status of peptides. The performance metrics of these base predictors, provided in Tables 3.13 and 3.14 respectively, are reproduced in Table 4.11 for ease of comparison.

With an estimated sensitivity (SN) of 0.85, specificity (SP) of 0.81, and accuracy (AC) of 0.83, the overall performance of the MV ensemble in predicting the activity of 10-16 aa length peptides is surpassed by RF, the best-performing base predictor in terms of overall error rate (ER), but it outperforms the NN model. Compared to the other three models, its performance is mixed. The SVM and MV models demonstrate comparable SN, but SVM achieves higher SP and, consequently, a higher estimated AC. Additionally, while the MV ensemble's SN is comparable to that of the LR and ENET methods, its improved SP results in a higher overall AC.

While the MV method exhibits reasonable performance in predicting the activity of 10-16 aa length peptides, its performance improves when applied to longer 18-27 aa length peptides. Notably, as Table 4.11 shows, its overall estimated ER of 0.1037 now matches that of the best-performing base predictor, RF. Moreover, the MV method achieves a higher sensitivity (SN) of 0.91, though at the cost of a slightly lower specificity (SP) of 0.88. This improvement in SN appears to result from MV leveraging the collective strengths of the LR and ENET model, thereby enhancing its ability to identify active peptides while mitigating individual model weaknesses.

Further insight into the performance of the MV method here is provided in Table 4.12 and 4.13, which contains the p-values from hypothesis tests assessing the pairwise independence and homogeneity of the base predictors. First, considering the models

Table 4.11: Performance measures for the majority vote ensemble and its five base predictors for predicting activity of 10-16 aa and 18-27 aa length peptides.

Methods	FN	TP	FP	TN	SN	SP	AC	PPV	ER	Models
10-16 aa length										
LR	21	108	35	94	0.84	0.73	0.78	0.76	0.2171	*MCIDR*
ENET	20	109	34	95	0.85	0.74	0.79	0.76	0.2093	*MIDLA*
SVM	22	107	18	111	0.83	0.86	0.85	0.86	0.1550	*MHCIDL*
RF	11	118	21	108	0.91	0.84	0.88	0.85	0.1240	*MHCIRA*
NN	25	104	27	102	0.81	0.79	0.80	0.79	0.2016	*MHCIDR*
MV	19	110	25	104	0.85	0.81	0.83	0.81	0.1705	
18-27 aa length										
LR	9	73	12	70	0.89	0.85	0.87	0.86	0.1280	*MHCOR*
ENET	7	75	15	67	0.91	0.82	0.87	0.83	0.1341	*MHCDL*
SVM	11	71	8	74	0.87	0.90	0.88	0.90	0.1159	*HCILA*
RF	11	71	6	76	0.87	0.93	0.90	0.92	0.1037	*MHCDOL*
NN	12	70	10	72	0.85	0.88	0.87	0.88	0.1341	*HCIDL*
MV	7	75	10	72	0.91	0.88	0.90	0.88	0.1037	

for predicting 10-16 aa length peptide activity, the results in Table 4.12 show no evidence against the pairwise homogeneity of LR, ENET, NN, and SVM in terms of sensitivity, with the first three also demonstrating pairwise homogeneity in specificity and accuracy. Additionally, there is strong statistical evidence for dependence between LR and ENET. Taken together, these findings suggest that the performance of the MV ensemble in predicting the activity of 10-16 aa length peptides is influenced by the dependence between LR and ENET. Extending this, assuming homogeneity and pairwise independence among the remaining base predictors, the result of Theorem 4.4, in conjunction with equation (4.9) in the proof of Theorem 4.3, implies that

the effective performance of the MV ensemble in this case corresponds to that of an ensemble with only $M = 3$ independent base predictors,

Table 4.12: P-values from Fisher Exact Tests of independence, and McNemar's test of similar predictive capability (homogeneity), between pairs of base predictors for activity of peptides 10-16 aa length.

Pairs of	P-value			
methods	Sensitivity	Specificity	Accuracy	Independence
(LR, ENET)	1.0000	1.0000	0.8551	0.0001
(LR, SVM)	1.0000	0.0030	0.0339	0.6869
(LR, RF)	0.0550	0.0140	0.0011	1.0000
(LR, NN)	0.4533	0.1698	0.6434	0.0724
(ENET, SVM)	0.8137	0.0046	0.0553	0.2257
(ENET, RF)	0.0809	0.0311	0.0036	0.6927
(ENET, NN)	0.3320	0.2482	0.8802	0.1628
(SVM, RF)	0.0371	0.6276	0.2684	0.0261
(SVM, NN)	0.6892	0.0953	0.1124	0.8389
(RF, NN)	0.0037	0.3447	0.0061	0.6892

Table 4.13 reveals that, apart from specificity of the ENET and RF models, there is very little evidence against pairwise homogeneity in the predictive capabilities of the base predictors for activity of 18-27 aa length peptides. Consequently, departure in performance from the idealised conditions is likely due to the strongly significant dependence observed between the LR and ENET predictors.

The findings thus far suggest that, by leveraging the diversity of its constituent base predictors, the MV ensemble is able to reduce the risk of systematic errors and enhance overall predictive accuracy. More importantly, the findings demonstrate that the MV

Table 4.13: P-values from Fisher Exact Tests of independence, and McNemar's test of similar predictive capability (homogeneity), between pairs of base predictors for activity of peptides 18-27 aa length.

Pairs of	P-value			
methods	Sensitivity	Specificity	Accuracy	Independence
(LR, ENET)	0.6171	0.3711	1.0000	0.0002
(LR, SVM)	0.7893	0.3865	0.8445	0.3588
(LR, RF)	0.7518	0.0771	0.4795	0.2137
(LR, NN)	0.5465	0.6171	1.0000	0.0358
(ENET, SVM)	0.3428	0.0961	0.6767	1.0000
(ENET, RF)	0.3428	0.0159	0.3827	0.5069
(ENET, NN)	0.1306	0.1824	1.0000	0.0492
(SVM, RF)	1.0000	0.7518	0.8445	0.2268
(SVM, NN)	1.0000	0.7518	0.6276	0.2253
(RF, NN)	1.0000	0.3428	0.4234	0.3726

ensemble often remains competitive with its best-performing base predictors, even in scenarios where the ideal conditions of independence and homogeneity are not fully met. The next section explores how the ensemble of base predictors can be utilised to improve the interpretability of the prediction model without sacrificing performance.

4.4 Ensemble Feature Selection

This section evaluates an ensemble feature (variable) selection method that aims to improve the robustness and accuracy of the best subset selection process by combining the outputs of the five base predictors used above. The approach aims to identify

features consistently exhibiting high relevance across the different methods thereby increasing the robustness of feature selection. By reducing the impact of biases or limitations associated with the individual methods, the approach also aims to achieve enhanced selection accuracy, and improved model interpretability and generalisation.

While the best subset selection guarantees the best possible fit to the training data, it may also include spurious predictor variables, as demonstrated by the simulation findings in Chapter 3, compromising both the interpretability and generalisability of the machine learning model. To mitigate this issue, integration of the outcomes of the five classification methods LR, ENET, SVM, RF and NN using reciprocal rank is proposed and evaluated here. While this proposed use of best subset selection in conjunction with an ensemble of ML methods is conceptually straightforward, there is no documented evidence in the literature of its evaluation or practical application within this context.

In an evaluation of ensemble feature selection methods (Effrosynidis & Arampatzis, 2021), the authors assert that although reciprocal rank has not been previously used for feature selection, it outperformed each of five other methods considered and demonstrated high stability. Equivalent to the harmonic mean, the reciprocal rank is determined by calculating the final rank $r(k)$ of a model using the following equation (Effrosynidis & Arampatzis, 2021);

$$r(k) = \frac{1}{\sum_{m}^{M} \frac{1}{r_m(k)}}, \tag{4.23}$$

where $r_m(k)$ is the rank of the k^{th} model according to the m^{th} of M classification methods; $m = 1, 2, \ldots, M$. The best model has the lowest final ranking score. In

the approach implemented here, the combination of predictor variables used to fit a model serves as a substitute for the model itself and consequently, the goal is to identify the optimal combination of predictor variables across a set of M machine learning methods.

4.4.1 Evaluation of reciprocal ranking using simulated scenarios

Recall from Chapter 3 that best subset selection was employed to identify the optimal combination of the predictor variables X_1, X_2, X_3, X_4 for the five classification methods, and that activity status was defined in the space of X_1, X_2 while X_3, X_4 were unrelated with activity. Results of the best subset selection implementation are provided in Tables 3.1, 3.4, 3.7 and 3.10 for the four respective simulated scenarios. Here, the predictor variable combinations in these tables are first ranked according to the overall error rate (ER) for each method and scenario, before a final ranking score over the five methods is computed (see example of calculating the final ranking score in Appendix C). The final scores for the top three predictor variable combinations are presented in Table 4.14. Note that the optimal degree polynomials (see section 4.3.1) were used for the LR, ENET and NN methods.

The ranking scores in Table 4.14 indicate that X_1 and X_2 are the dominant features in determining activity status in Scenarios 2 and 4. However, in both cases, they are not top-ranked by either the SVM or RF methods, see Tables 3.4 and 3.10. In the remaining two scenarios, X_1 and X_2 rank second, following a three-dimensional space that includes them. For Scenario 1, Table 3.1 shows that models using only

Table 4.14: The final ranking score of the four best combination of predictor variables for four simulation scenarios using reciprocal rank voting.

Order	Ranking score	Predictors
1^{st} Scenario		
1^{st}	0.3158	X_1, X_2, X_4
2^{nd}	0.3333	X_1, X_2
3^{rd}	0.3871	X_1, X_2, X_3
2^{nd} Scenario		
1^{st}	0.2667	X_1, X_2
2^{nd}	0.3750	X_1, X_2, X_4
3^{rd}	0.4855	X_1, X_2, X_3, X_4
3^{rd} Scenario		
1^{st}	0.3158	X_1, X_2, X_4
2^{nd}	0.3429	X_1, X_2
3^{rd}	0.4000	X_1, X_2, X_3, X_4
4^{th} Scenario		
1^{st}	0.2609	X_1, X_2
2^{nd}	0.3871	X_1, X_2, X_3
3^{rd}	0.4000	X_1, X_2, X_4

these two predictor variables rank fourth for the ENET and NN methods while, as can be seen from Table 3.7, in Scenario 3 they rank fourth for ENET and third for both the SVM and RF methods. These findings support the hypothesis that a single machine learning method is likely to be more effective than an ensemble approach in identifying the most relevant predictor variables. They also provide evidence that high-performing predictive methods, such as RF, may exhibit suboptimal performance in feature selection tasks and highlight the advantage of combining multiple methods for more effective feature selection.

4.5 Ensemble Feature Selection and Prediction of AMP Activity

The outcomes of five classification methods LR, ENET, SVM, RF and NN are integrated here into a novel ensemble-based feature selection and prediction framework aimed at enhancing the robustness and accuracy of predictions, and ensuring a more reliable identification of key determinants of AMP activity. Details of the model building approach for the five methods, utilising best subset feature selection to identify the optimal combination of the nine predictor variables of AMP activity, are provided in Section 3.3, with the corresponding performance analyses reproduced in Table 4.11. Here, the 502 predictor variable combinations generated through the best subset selection approach are ranked within each method and these individual rankings are then aggregated using the reciprocal rank vote to determine the optimal combination, as outlined in the previous section. The MV algorithm is then constructed based on this optimal combination of predictor variables.

The results of the ensemble feature selection, presented in Table 4.15, show that the dominant feature combination determining the AMP activity of the shorter peptides is $MHIDLA$, whereas for longer peptides, it is $MHCDOL$. Both combinations define six-dimensional (6D) feature spaces. Notably, as can be determined from Table 4.11, the combination $MHIDLA$ was not ranked highest by any individual method, while $MHCDOL$ was ranked first solely by RF. Furthermore, for shorter peptides, only the fourth-ranked combination, $MHCIDL$, and the fifth-ranked combination, $MHCIDR$, were identified as top-ranked by any of the individual methods. Similarly, for longer peptides, the third-ranked feature combination, $MHCOR$, was

ranked first exclusively by LR.

Table 4.15: The final ranking scores and five best combinations of predictor variables for AMP activity of peptides 10-16 and 18-27 aa length, using reciprocal rank voting. The lowest final ranking scores are highlighted in blue.

	10-16 aa length		18-27 aa length	
Order	Ranking score	Predictors	Ranking score	Predictors
1^{st}	0.6270	*MHIDLA*	0.5288	*MHCDOL*
2^{nd}	0.6640	*MHCIDORL*	0.5567	*MHCOL*
3^{rd}	0.6724	*MCIDRL*	0.5581	*MHCOR*
4^{th}	0.6747	*MHCIDL*	0.6059	*MHCDOR*
5^{th}	0.7103	*MHCIDR*	0.6075	*MHCORL*

An inspection of the optimal feature combinations in Table 4.15 reveals that the four physicochemical features, hydrophobic moment (M), hydrophobicity (H), penetration depth (D) and linear moment (L) are key predictors of antimicrobial peptide (AMP) activity for both the shorter and longer peptides and are therefore likely to play a fundamental role in determining peptide behaviour. Additionally, distinct features emerge as important for the two peptide lengths. For shorter peptides, isoelectric point (I) and propensity to aggregation (A) are selected as relevant predictor variables, suggesting that charge distribution and aggregation tendencies significantly impact their activity. In contrast, for longer peptides, net charge (C) and tilt angle (O) are suggested as key determinants, implying that electrostatic interactions and peptide orientation within membranes are more influential in this category.

Performance measures of the majority vote ensemble, incorporating the five classification methods with the selected predictor variables $MHIDLA$ for peptides of 10-16 and $MHCDOL$ for peptides of 18-27 amino acids, denoted MV-FS, are presented

Table 4.16: Performance measures for ensemble feature selection and prediction of 10-16 aa and 18-27 aa length peptides activity. The majority vote with the feature selection method is labelled MV-FS.

Methods	FN	TP	FP	TN	SN	SP	PPV	ER
10-16 aa length (*MHIDLA*)								
MV-FS	24	105	25	104	0.81	0.81	0.81	0.1899
MV	19	110	25	104	0.85	0.81	0.81	0.1705
DB-SCAN[1]	121	8	5	124	0.06	0.97	0.62	0.4884
18-27 aa length (*MHCDOL*)								
MV-FS	13	69	12	70	0.84	0.85	0.85	0.1524
MV	7	75	10	72	0.91	0.88	0.88	0.1037
DB-SCAN[1]	76	6	2	80	0.07	0.98	0.75	0.4756

[1] Re-Analysis results

in Table 4.16. For comparative analysis, the table also includes performance metrics for the MV method without feature selection and results from the re-analysis of the data using the DB-SCAN clustering approach. Compared to the MV method, the MV-FS approach exhibits no change in the number of false positives (FP) for shorter peptides and a slight increase for longer peptides. In the latter case, this results in a slight decrease in specificity (SP). Additionally, the MV-FS method shows a small reduction in the number of true positives (TP), leading to a slight decrease in sensitivity (SN) for both peptide lengths. The estimated error rates (ER) of the MV-FS method show a slight decrease for shorter peptides, while a modest increase of approximately five percentage points is observed for longer peptides. However, these changes remain minimal, indicating that incorporation of feature selection does not significantly impact the overall performance of the MV ensemble. Notably, both the MV and MV-FS methods consistently outperform the DB-SCAN approach across all

Table 4.17: Minimum and maximum values of the observed predictor variable data and grid values, along with the step size and number of steps, for peptides 10-16 aa length.

Features	Raw data		Grid		Step size	(Steps)
	min	max	min	max		
M	0.02	2.36	0.00	2.40	0.1	(25)
H	-3.09	3.82	-3.10	3.90	0.2	(36)
C	-3.00	14.00	-6.00	15.00	1	(22)
I	3.01	13.21	2.60	13.40	0.4	(28)
D	5.00	30.00	0.00	30.00	1	(31)
O	3.00	177.00	0.00	180.00	5	(37)
R	-2.15	1.02	-2.20	1.10	0.1	(34)
L	0.00	0.59	0.00	0.70	0.02	(36)
A	0.00	799.83	0.00	800.00	20	(41)

performance metrics, except for specificity.

4.5.1 Predicting active regions of the physicochemical space

The machine learning method MV-FS developed in this thesis is next applied to predict regions of active antimicrobial peptides (AMPs) within the physicochemical space of selected predictor variables in order to enhance understanding of the key properties influencing AMP activity, and to provide insights for the identification and rational design of novel AMPs with optimised properties. To enable a systematic exploration of the physicochemical space, predicted activity is obtained for a grid of points designed to encompass the observed data values for each physicochemical property in the DBAASP dataset. Thus, the grid is constructed based on the observed

data ranges, with specified step sizes and the number of steps, as detailed in Table 4.17 for peptides of 10-16 amino acids and Table 4.18 for peptides with 18-27 amino acids. In particular, predictions for the shorter 10-16 aa peptides are obtained over a grid constructed using M (25 steps), H (36 steps), I (28 steps), D (31 step), L (36 steps) and A (41 steps), resulting in a total of $1,153,051,200$ data points. For the longer 18-27 aa peptides, the grid is constructed over M (25 steps), H (36 steps), C (22 steps), D (31 steps), O (37 steps) and L (36 steps), giving a total of $817,581,600$ data points.

Table 4.18: Minimum and maximum values of the observed predictor variable data and grid values, along with the step size and number of steps, for peptides 18-27 aa length.

Features	Raw data		Grid		Step size	(Steps)
	min	max	min	max		
M	0.00	2.09	0.00	2.40	0.1	(25)
H	-2.24	1.74	-3.10	3.90	0.2	(36)
C	-6.00	15.00	-6.00	15.00	1	(22)
I	2.68	13.14	2.60	13.40	0.4	(28)
D	0.00	30.00	0.00	30.00	1	(31)
O	2.00	176.00	0.00	180.00	5	(37)
R	-1.64	0.85	-2.20	1.10	0.1	(34)
L	0.00	0.61	0.00	0.70	0.02	(36)
A	0.00	1274.19	0.00	1320.00	40	(34)

Figures 4.4 and 4.5 present 2D projections of the predicted activity distributions across the physicochemical space for peptides of 10-16 amino acids and 18-27 amino acids, respectively. The contour lines and blue shades in the 2D plots delineate regions of the physicochemical space with the same probability of activity, where the

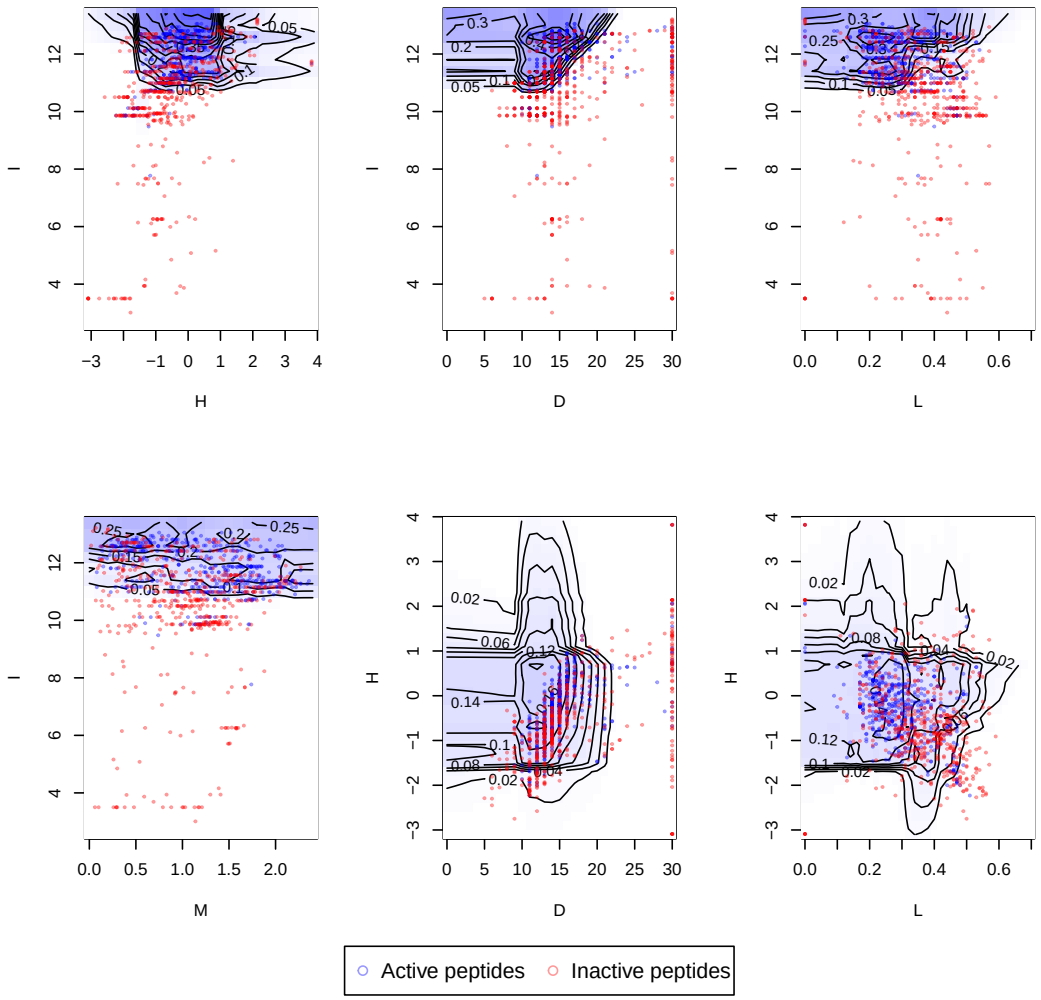


Figure 4.4: Regions of active AMPs predicted by MV-FS for peptides 10-16 aa length, which are visualized as a 2D projection of some pair of physicochemical features, where contour lines and blue regions indicate the highest probability of activity. Matrix plots of predictor variables M, H, I, D and L display the relationships between active (blue) and inactive (red) peptides from the real AMPs datasets.

probability at each point in the 2D space is estimated by the proportion of predicted active outcomes at that point. The selected 2D spaces displayed in the figures were chosen because they reveal regions with the highest predicted activity, which are indicated by the darker blue regions. To provide additional context, scatterplots of

the training data are superimposed on the plots.

The top row of Figure 4.4 suggests that active 10-16 aa peptides are most likely to have isoelectric point (I) around 12 or higher, which is near the boundary of this predictor variable. Further, that there is more than a 30% probability that these peptides will have hydrophobicity (H) in the interval $(-1, 0)$ and linear moment (L) of approximately 0.30. Moreover, the leftmost plot in the bottom row provides mild evidence that hydrophobic moment around 1 corresponds to a region of lower activity.

Regarding the 18-27 aa peptides, a comparison of Figure 4.5 with Figure 4.4 clearly demonstrates that active regions of the physicochemical space are now predicted with greater certainty. This is expected given the performance measures in the previous section. The rightmost plot on the bottom row of Figure 4.5 indicate that there is more than a 90% probability that active 18-27 aa peptides will have net charge (C) in the interval $(7, 10)$ and H in the same interval as the shorter peptides. The tilt angle (O) of these peptides are more likely in the interval $(60 - 100)$ while L tends to be 0.2 or less. Also, notice from the leftmost column of plots that, consistent with observations for the shorter peptides, values of M around 0.5 correspond to a region with lower probability for active peptides.

Finally, it is important to highlight that the ranges of the physicochemical predictor variables identified through MV-FS in this thesis closely align with the values associated with active clusters in the original findings by Vishnepolsky et al. (2018), provided in Tables 2.3 and 2.5 of Chapter 2.

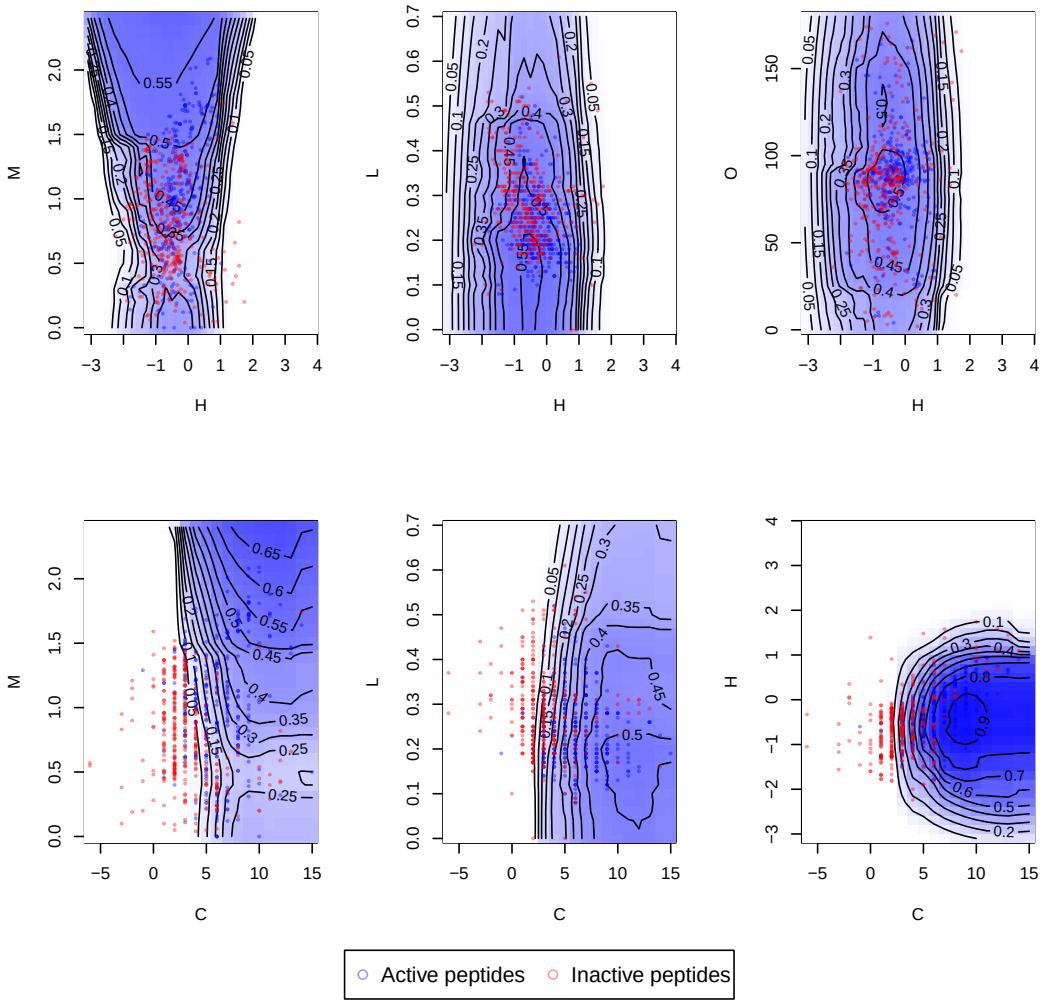


Figure 4.5: Regions of active AMPs predicted by MV-FS for peptides 18-27 aa length, which are visualized as a 2D projection of some pair of physicochemical features, where contour lines and blue regions indicate the highest probability of activity. Matrix plots of predictor variables M, H, C, O and L display the relationships between active (blue) and inactive (red) peptides from the real AMPs datasets.

4.6 Discussion

This chapter used both theory and simulations to delve into the integration of the five classification methods introduced in the previous chapter within an ensemble framework. A systematic evaluation of ensemble performance and the identification of

specific conditions under which the framework may become sub-optimal is presented. By doing so, this chapter offers valuable insights into the strengths and limitations of the ensemble approach and provides a deeper understanding of the conditions when it performs best.

Results proving that the majority vote ensemble will consistently outperform its individual base predictors, provided that the two key conditions of homogeneity and independence are met, are presented. Homogeneity ensures that base predictors are similarly accurate, while independence guarantees that their errors are uncorrelated. When these conditions are satisfied, the ensemble leverages the diversity of its constituent models, reducing the risk of systematic errors and enhancing overall predictive accuracy. Conversely, the behaviour of the MV ensemble can change when these conditions are not met. For example, the majority vote mechanism can amplify the influence of base predictors with similar error patterns, potentially undermining the ensemble's ability to correct for individual weaknesses. Also, if one or a few base predictors significantly outperform the others, the ensemble's performance may closely align with these stronger models, even in the absence of strict independence. Despite these limitations, the theoretical and simulation results presented in this chapter show that the MV ensemble will be competitive with the best-performing base predictors.

The simulation results in this chapter confirm that there is no guarantee that variable selection approaches implemented with individual methods will exclude irrelevant or spurious features. Moreover, existing ensemble feature selection techniques primarily emphasise the individual contributions of features (see Effrosynidis & Arampatzis, 2021, for example), potentially overlooking the importance of feature interactions

and collective relevance. The approach proposed in this chapter addresses these limitations by integrating best subset selection directly into the ensemble framework. This ensures that each base predictor is trained on a subset of features that are both highly relevant and informative, thereby improving model accuracy and robustness. By doing so, this novel methodology addresses a critical challenge in machine learning of reducing the likelihood of models incorporating spurious features and simultaneously enhancing model interpretability. The approach proposed here is expected to be particularly impactful in the domain of peptide design and discovery, where scientific understanding of the key drivers behind biological activity is just as crucial as achieving high predictive accuracy.

The application of the proposed ensemble method presented in this chapter highlights its practical utility in robustly identifying key physicochemical features of antimicrobial peptides that drive antibacterial activity and its ability to capture nuanced relationships that may not be evident through individual methods. Additionally, the method enables robust predictions of regions within the physicochemical space that are linked to functional antimicrobial properties. These predictions provide a valuable foundation for guiding the rational design of peptides with enhanced and optimized antimicrobial properties. By integrating multiple perspectives, the ensemble framework not only improves predictive accuracy but also offers a more comprehensive understanding of the underlying mechanisms, paving the way for more effective peptide-based therapeutic development.

Chapter 5

Predicting Conformational Properties of Charged Polymers

In this chapter, the ML method developed in previous chapters is applied to assist predictions for conformational properties of charged polymers. Charged polymers are macromolecules that carry charges along their backbones and/or side groups, which are abundant in both natural and synthetic forms (Dobrynin et al., 1995, 2004; Dobrynin & Rubinstein, 2001; Shusharina et al., 2005). These polymers can be classified into two categories, i.e., polyelectrolytes (PEs) that carry a single type of charges, such as DNA, and polyampholytes (PAs) that bear both positive and negative charges, such as proteins and peptides. The long-range electrostatic interactions among the charged groups give them rich conformational and dynamic properties in comparison with neutral polymers, but also significantly increase the computational costs of using brute force simulations to study these properties. In this work, the potential of ML methods in facilitating the construction of conformational phase diagrams of single polyelectrolyte chains is explored. The model system selected to study is the

linear diblock polyampholyte chain that consists of one positively charged and one negatively charged block end-linked at the joint point (Wang & Rubinstein, 2006). The training data set is generated using molecular dynamics simulations based on the bead-spring polymer chain model (Kremer & Grest, 1990). The ML results are applied to examine the theoretical model predictions and help identify the key features in controlling the conformational behaviours of charged polymers. The learning outcomes can also contribute to understanding the mechanisms underlying the activities of AMP peptides where the electrostatic and hydrophobic interactions are found to play important roles.

Section 5.1 provides a brief review of the theoretical description of the conformational properties of diblock polyampholyte chains. Section 5.2 introduces the molecular dynamics (MD) simulation method used to simulate these polymers and presents the simulation results on their conformational properties in terms of a set of physiochemical features, which are used as training data for our ML simulations. Section 5.3 details the implementation of the machine learning (ML) approach, presents the resulting predictions, and utilizes these outcomes to examine the theoretical predictions. Conclusions and perspectives are given in Section 5.4.

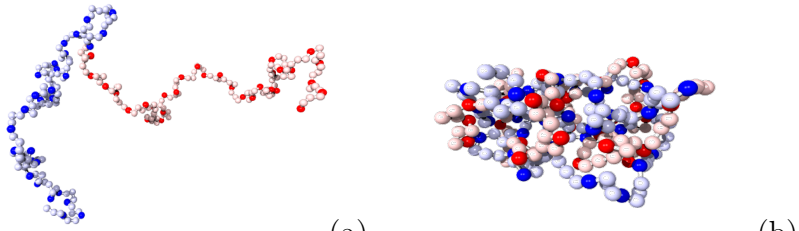


Figure 5.1: Snapshots of a symmetric diblock polyampholyte chain in the swollen (a) and associated (b) states, respectively. The total chain length is $N = 256$, giving the block lengths $N_+ = N_- = 128$. The charge fraction is $f = 1/4$.

5.1 Conformational Transitions of Diblock Polyampholyte Chains Theory

In the coarse-grained (CG) bead-spring model, a linear polymer chain is represented by a sequence of spherical beads or monomers of diameters σ connected by elastic bonds or springs with their near neighbours, see Figure 5.1. The conformations of a neutral polymer chain are determined by the chain connectivity and stiffness, the short-range excluded volume interactions between monomers, and the environmental factors, including the temperature, solvent quality, polymer density, etc.

For a polymer chain composed of N beads located at the Cartesian coordinates $\{\mathbf{r}_i\}, i = 1, \ldots, N$, its overall size can be measured by the end-to-end distance R_{ee} defined as

$$R_{ee} = \left\langle (\mathbf{r}_N - \mathbf{r}_1)^2 \right\rangle^{1/2} \tag{5.1}$$

where $\mathbf{r}_1$ and $\mathbf{r}_N$ are the positions of the two end monomers of the chain and the angle brackets "<>" stand for ensemble average, meaning that this physical quantity is calculated by averaging over a sufficiently large number of polymer chain conformations in the equilibrium state, and the radius of gyration R_g defined as

$$R_g = \left\langle \frac{1}{2N^2} \sum_{i,j=1}^{N} (\mathbf{r}_i - \mathbf{r}_j)^2 \right\rangle^{1/2}. \tag{5.2}$$

In dilute solutions where the polymers are far away from each other and so can be considered as being isolated, the chain size R (i.e., a general representation of R_{ee}

and R_g) follows the scaling law of (Rubinstein & Colby, 2003)

$$R \approx N^{\nu} b \tag{5.3}$$

where $b = \langle (\mathbf{r}_{i+1} - \mathbf{r}_i)^2 \rangle^{1/2}$ is the average bond length and the value of the Flory exponent ν depends on the quality of the solvent.

For an ideal chain where the beads are connected by harmonic springs and do not have excluded volumes (i.e., no short-range interactions), its conformations can be described by the three-dimensional (3D) random walks of N steps with the step size equal to the average bond length b. Its end-to-end distance is then given by (Doi & Edwards, 1988)

$$R_{ee} = N^{1/2} b$$

and the radius of gyration $R_g = R_{ee}/6$. In experiments, polymer chains in the so-called $\Theta-$solvent behave very similarly to the ideal chain and so have size

$$R \approx N^{1/2} b, \tag{5.4}$$

giving the Flory exponent $\nu = 0.5$. In equation (5.4), the approximation sign indicates that the two sides differ by a constant whose value is system-dependent and can be determined in experiments or computer simulations. For polymers in a good solvent, the excluded volume interactions between the monomers lead the chains to take the conformations of swollen coils. The corresponding chain sizes scale as $R \approx N^{0.588} b$, i.e., $\nu = 0.588$ (Rubinstein & Colby, 2003).

The long-range electrostatic or Coulombic interactions between charged groups strongly

alter the conformational properties of charged polymers from their neutral counterparts. The extra controlling factors include the charge distribution along the polymer backbones and the strength of the electrostatic interactions. For example, polyampholytes carry both positive and negative charges which can distribute in random, alternative, or blocked patterns (Imbert et al., 1999; Wang & Rubinstein, 2006; Akinchina & Linse, 2007; Linse, 2007; Ulrich et al., 2007; Narayanan Nair et al., 2014, 2017; Palariya & Singh, 2024). Only considered in this chapter are the linear symmetric diblock polyampholytes composed of one positively and one negatively charged blocks, see Figure 5.1(a) for an example studied by Wang and Rubinstein (2006). The two blocks have an equal number of monomers $N_+ = N_- = N/2$ and equal charge fraction $f \leq 1$. The net charge per chain is thus zero.

The diblock PA studied in Figure 5.1 has $f = 1/4$, meaning that one out of every four monomers is charged either positively (red coloured) or negatively (blue coloured). The electrostatic repulsion between likely charged monomers in each block leads to swelling of the individual block, while the attractions between the oppositely charged monomers in the two different blocks pull them towards each other, leading to chain folding and monomer association (Wang & Rubinstein, 2006).

The electrostatic interaction (or Coulomb) potential between two monomers is

$$U_{Coul}(r_{ij}) = k_B T \frac{l_B q_i q_j}{r_{ij}}, \tag{5.5}$$

where q_i is the charge valence of the ith particle, equal to $+1$ (-1) for the monovalent positive (negative) charges and 0 for the neutral monomers. r_{ij} is the distance between the centers of mass of Particles i and j. The strength of the electrostatic interaction

is represented by the Bjerrum length l_B that is defined as the distance at which the electrostatic interaction energy between two elementary charges is equal to the thermal energy k_BT, i.e.,

$$l_B = e^2/(\epsilon k_B T), \tag{5.6}$$

where e is the elementary charge, ϵ the dielectric constant, k_B the Boltzmann constant and T the temperature. Wang and Rubinstein (2006) found that with the increase of the electrostatic interaction strength, the conformations of a diblock PA can undergo transitions from the swollen coil, to folding, weak association and ion binding states, see Figure 1 in that paper. The authors constructed the diagram of conformational states of symmetric diblock polyampholytes in terms of Bjerrum length l_B and charge fraction f using scaling theory and MD simulations.

Briefly summarized below are the theoretical predictions on the boundaries between different conformational states, which will be used to guide the construction of classification boundaries using our machine learning method. For details please see Wang and Rubinstein (2006) and the references therein.

Consider a flexible symmetric diblock polyampholyte with N monomers and charge fraction f. If the electrostatic interaction strength, as measured by l_B, is very small, the polymer behaves as a neutral chain and takes the conformation of a swollen coil, see Figure 5.1(a). The chain size is given by equation (5.3), i.e.,

$$R_g \approx N^{\nu}\sigma, \tag{5.7}$$

where bond length b is replaced by the monomer size σ for flexible chains, without affecting the scaling law. Recall that the Flory exponent ν is 0.5 for $\Theta-$solvent and

0.588 for good solvent.

The electrostatic interactions start to perturb the chain conformation when the total electrostatic energy of the whole chain is comparable with the thermal energy k_BT, corresponding to the condition on the chain size,

$$R_g \approx \sigma(l_B f^2/\sigma)^{-\nu/(2-\nu)}. \tag{5.8}$$

The electrostatic attractions between the two oppositely charged blocks pull them to fold towards each other. Substituting equation (5.7) into equation (5.8), we get the boundary between the onset of the folding and the unperturbed coil

$$l_B^{fold} = C_f f^{-2} N^{\nu-2} \sigma \tag{5.9}$$

where the value of the numerical coefficient C_f depends on the quality of the solvent. The size of the chain in the folding regime can be calculated numerically, see Appendix in Wang and Rubinstein (2006). The folding mechanism of a diblock polyampholyte can be explained using a simple mean-field model. In this model, the diblock chain is depicted as two equally sized, oppositely charged objects connected by an entropic spring at their centres of mass. The electrostatic attraction between the charged objects and the spring's elastic energy are balanced by the short-range repulsion arising from monomer interactions in the overlapping regions of the blocks. The average distance between the centres of mass of the two objects (block) is denoted as r_{cm}.

When the two blocks completely overlap with each other such that the size of the whole chain coincides with that of each block, i.e., $R_g = R_{g,blk}$, the diblock PA enters

the globular or weak association regime. The boundary between the folding and weak association regimes is

$$l_B^{weak} = C_w f^{-2} N^{\nu-2} \sigma. \tag{5.10}$$

In this regime, the diblock PA takes the globular conformation, see Figure 5.1(b), which can be described by a dense packing of the so-called electrostatic blobs (Rubinstein & Colby, 2003). The electrostatic blobs are globules whose sizes ξ_e are determined by the balance between the electrostatic and thermal energies,

$$\xi_e \approx \sigma (l_B f^2/\sigma)^{-\nu/(2-\nu)}. \tag{5.11}$$

It is noticed that the onset of the folding regime is at the point when $R_g = \xi_e$. The picture of densely packed electrostatic blobs can be checked by requiring the average distance between two nearest oppositely charged monomers in the chain

$$\xi_{+-}^{1st} \equiv \left\langle \frac{2}{fN} \sum_{k=1}^{fN/2} (\mathbf{r}_k - \mathbf{r}_k^{near})^2 \right\rangle^{1/2} \approx \xi_e \tag{5.12}$$

where $\mathbf{r}_k$ is the position of charged monomer k and $\mathbf{r}_k^{near}$ is the location of the oppositely charged monomer nearest to monomer k. The resulting size of the diblock PA globule in the weak association regime is

$$R_g \approx \sigma N^{1/3} (l_B f^2/\sigma)^{(1-3\nu)/[3(2-\nu)]}. \tag{5.13}$$

The upper boundary of the weak association regime is reached when the electrostatic blob size reduces to the size of the chain section containing only one charge, giving

$$l_B^{up} \approx f^{-\nu} \sigma. \tag{5.14}$$

The system enters the ion binding or strong association regime when the electrostatic attractions between oppositely charged monomers are strong enough to overcome the entropic penalty and steric repulsion between neutral chain sections and lead them to form stable charge pairs or dipoles. The onset of this regime can be found from the indication that the average distance between nearest oppositely charged monomers is on the order of the monomer size, i.e.,

$$\xi_{+-}^{1st} \approx \sigma.$$

The Bjerrum length l_B^{bind} at this onset point is estimated to be

$$l_B^{bind} \approx -\sigma \ln f. \tag{5.15}$$

With increasing l_B, there is a cascade of multiple formation transitions, from dipoles to quadrupoles, hexapoles and octupoles. These transitions can be identified by calculating the average distances between the charged monomers with their first, second and third nearest likely or oppositely charged neighbours.

5.2 Molecular Dynamics Simulations

From the above section, it can be seen that the conformational transitions of the symmetric diblock polyampholytes in a given solvent are determined by three features or predictor variables: charge fraction f, Bjerrum length l_B and chain length N. The conformational properties in each conformational regime are characterised by the chain (or block) size, R_g (or $R_{g,blk}$), and the average distance between oppositely (or likely) charged monomers, ξ_{+-} (or $\xi_{++(--)}$). The training data sets for investigating

use of the MV-FS method here will then be the outputs $R_{g(,blk)}$ and $\xi_{+-(++,--)}$ in terms of the input parameters f, l_B, and N. These data are generated using molecular dynamics simulations of isolated diblock PA chains (Wang & Rubinstein, 2006).

In the bead-spring chain model used (Kremer & Grest, 1990), all monomers interact via a truncated-shifted Lennard-Jones (LJ) potential,

$$U_{LJ}(r) = \begin{cases} 4\varepsilon_{LJ}\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} - \left(\frac{\sigma}{R_c}\right)^{12} + \left(\frac{\sigma}{R_c}\right)^{6}\right]; & r < R_c; \\ 0; & r \geq R_c, \end{cases} \tag{5.16}$$

where r is the distance between the centers of two monomers, R_c is the cutoff radius, ε_{LJ} is the LJ interaction strength and σ the monomer diameter. The parameter σ is used to set the length scale. Mainly simulated is the good (athermal) solvent condition, for which the cutoff $R_c = 2^{1/6}\sigma$ (at the minimum of the LJ potential), thus presenting a purely repulsive potential U_{LJ}. The interaction parameter of this potential is chosen to be $\varepsilon_{LJ} = 1.0k_BT$. If simulating the Θ solvent condition, a larger cutoff $R_c = 2.5\sigma$ is taken so as to include the attractive tail in the LJ potential and $\varepsilon_{LJ} = 0.34k_BT$.

The monomers are connected by the finite extensible nonlinear elastic (FENE) potential (Kremer & Grest, 1990)

$$U_{FENE}(r) = -\frac{1}{2}k_{FENE}R_0^2 \ln(1 - \frac{r^2}{R_0^2}), \tag{5.17}$$

where the spring constant $k_{FENE} = 7k_BT/\sigma^2$ and the maximum bond length $R_0 = 2\sigma$ at which the elastic energy of the bond becomes infinite. The solvent is modelled as a continuous dielectric medium with dielectric constant ϵ and no added salt. The

charged monomers are thus interacting with each other via the unscreened Coulomb potential given in equation (5.5). For simulating isolated symmetric diblock PAs with zero net charge and no counterions, there is no need to apply the periodic boundary conditions. The Coulomb interaction forces are summed over all charged monomers on the chain directly.

The Langevin equations of motion of the monomers are (Kremer & Grest, 1990),

$$m\frac{d^2\boldsymbol{r}_i}{dt^2} = -\nabla U(\boldsymbol{r}_i) - \zeta\frac{d\boldsymbol{r}_i}{dt} + \Gamma_i(t), \tag{5.18}$$

where $\boldsymbol{r}_i$ is the coordinate of the ith particle, m is the particle mass, and ζ is the friction coefficient equal to $\zeta = 10(mk_BT)^{1/2}/\sigma$. The stochastic force Γ_i is given by a δ-correlated Gaussian noise source. These equations are solved numerically using the velocity Verlet method (Allen & Tildesley, 2017) with a time step $\Delta t = 0.0125\tau$ in most of the simulations here, where $\tau = (m\sigma^2/k_BT)^{1/2}$ is LJ time unit. For simulating systems with very strong electrostatic interactions, the time step is reduced to $\Delta t = 0.003\tau$.

The details about the initialisation, equilibration and data collection processes can be found in Wang and Rubinstein (2006). For each given set of parameters f, l_B, and N several independent simulation runs using different initial configurations were carried out to get reliable results with good statistics. Note that systems subject to very strong ionic binding can be trapped in metastable states, and the simulation results show some dependence on the initial configurations. In this case, a much larger ensemble average is needed.

5.3 Predicting Conformational Transitions of Diblock Polyampholytes

Due to the high computational cost of the MD simulations, especially for the systems with long-range electrostatic interactions, it is impractical to cover the entire parameter phase space and construct a complete conformational phase diagram with well-defined boundaries between different regimes. Machine learning has been increasingly employed as a data-driven approach for simulating, modelling and designing soft matter materials (Jackson et al., 2019; Wang et al., 2020; Noé et al., 2020; Zhang et al., 2024). In an effort to contribute to this exciting field, the potential of the MV classification method developed in Chapter 4 to assist the prediction of conformational properties of diblock PAs is investigated here.

5.3.1 Result of charged polymer simulations

The five classification methods, LR, ENET, SVM, RF and NN and the related MV ensemble without feature selection, are employed to construct machine learning models for building the phase diagram of conformational states of a diblock polyampholyte in a given solvent using MD simulations. Three features determine these conformational states: charge fraction f, Bjerrum length l_B and chain length N. The training dataset consists of 107 chains, each with a length of $N = 256$ monomers and charge fractions of $f = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$ and $\frac{1}{32}$. The value of Bjerrum length l_B runs from $l_B = 0$ for neutral chains, and increases incrementally from $\frac{1}{2048}, \frac{1}{1024}, \ldots,$ up to $l_B = 64$.

Shown in Figure 5.2 are plots of the normalised radius of gyration for a whole chain

R_g/R_{g0} and one block of the chain $R_{g,blk}/R_{g0}$ against $l_B(fN)^2$, where R_{g0} is the radius of gyration for a neutral chain with the same chain length. The point in the graph where R_g/R_{g0} begins to change in value defines the boundary between coil and folding conformational regimes. The point where $R_{g,blk}/R_{g0}$ begins to decrease defines the boundary between folding and weak association regimes for whole chains. As a preliminary investigation, the objective here is to evaluate the effectiveness of the ML ensemble method in reconstructing the boundaries between these regions within the parameter space defined by f and l_B, as well as to assess the reliability of this approach in the given context.

As can be seen from Figure 5.2, the boundary between the coil and folding regimes happens when R_g/R_{g0} deviates from 1. Specifically, the coil regime is defined by $R_g/R_{g0} > 0.95$, whereas the folding regime is defined by $0.65 < R_g/R_{g0} \leqslant 0.95$, with $R_{g,blk}/R_{g0}$ remaining roughly unchanged. In addition, the boundary between the folding and weak association regimes happens when $R_g/R_{g0} \approx R_{g,blk}/R_{g0}$. Therefore, the weak association regime is defined by R_g/R_{g0} and $R_{g,blk}/R_{g0} \leqslant 0.65$, whereas the coil plus folding regime in one block of the chain is defined by $R_{g,blk}/R_{g0} > 0.65$. Here, the boundary between the two regimes is determined differently from that of Wang and Rubinstein (2006) because this study uses only a subset of their data, specifically the case of N=256 monomers. Thus, the boundary values are adjusted to ensure consistency with the training data and to facilitate the construction of a complete conformational phase diagram.

Table 5.1 displays the five-fold cross-validation (CV) estimates of the error rate (ER) for models classifying dichotomised values of R_g/R_{g0} and $R_{g,blk}/R_{g0}$ using f and l_B as predictor variables. A one-versus-all approach was employed to build each

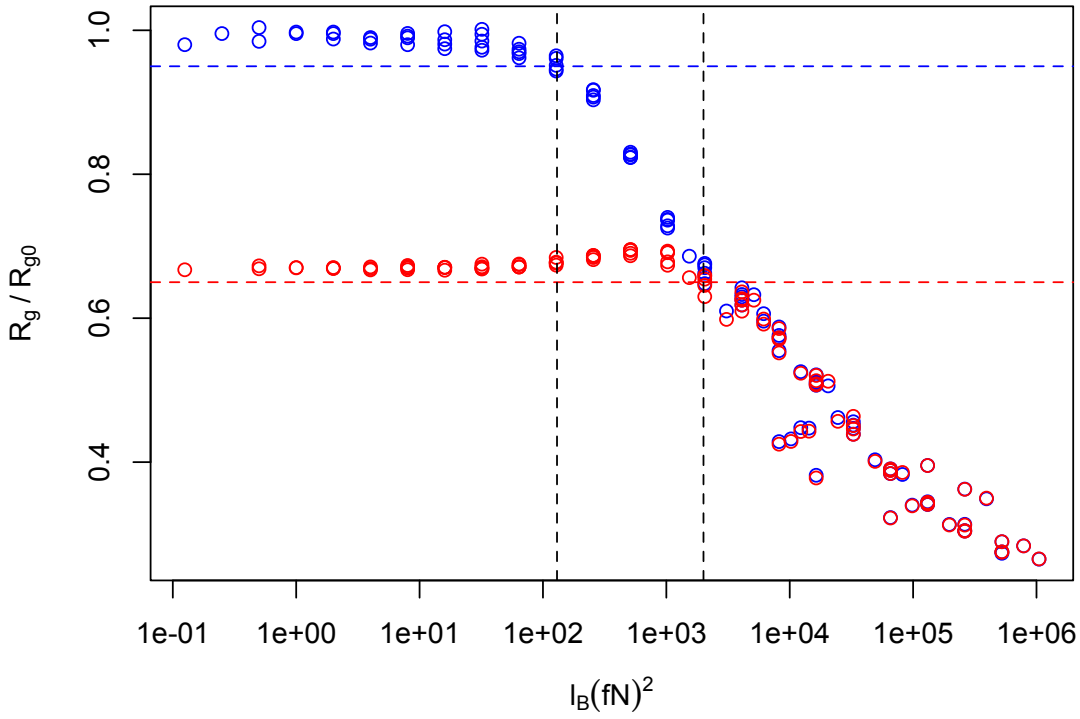


Figure 5.2: Normalized radius of gyration of the whole chain (blue) and one block (red) of diblock polyampholyte chains.

model. Thus, for example, the error rates in the second column are for models distinguishing outcomes classified as "coil" from those classified as either "fold" or "weak association", for a whole chain. The error rates in the last column are for models that classify outcomes as either in a coil plus folding or weak association regime, for one block of the chain.

The results presented in Table 5.1 demonstrate that the five methods employed to delineate the boundary between the coil and folding regimes exhibit comparable performances, each achieving an estimated prediction accuracy of approximately 95%. With prediction accuracies ranging approximately between 90% to 95%, the methods

Table 5.1: Cross-validation (CV) estimates of the error rate (ER) for ML models of dichotomised values of normalized radius of gyration of the whole chain (R_g/R_{g0}), and normalized radius of gyration of one block of the chain ($R_{g,blk}/R_{g0}$) of the diblock polyampholyte chains.

Methods	Error Rate (ER)		
	coil vs (fold+weak)	(coil+fold) vs weak	(coil+fold) vs weak (block)
LR	0.0563	0.1026	0.1117
ENET	0.0558	0.0732	0.0550
SVM	0.0468	0.0636	0.0641
RF	0.0654	0.1013	0.0922
NN	0.0563	0.0550	0.0554

also showed comparable performances in defining the boundary between the folding and weak association regimes for entire chains of the diblock polyampholyte. Similar performances are also observed for the methods when used to define the boundary between the coil plus folding and weak association regimes for a single block of the diblock polyampholyte.

Investigating the performances further, results of the pairwise tests for comparable predictive capability in determining the conformational phase diagram of the whole chain of the diblock polyampholyte chains, as presented in Table 5.2, indicate that the five models exhibit similar predictive performance across all pairs of base predictors, confirming the above suggestion. Moreover, the p-values of the independence tests for the whole chain of the diblock polyampholyte chains in terms of accuracy, as shown in Table 5.4, reveal highly significant dependencies between all pairs of models. The only exception is the pair comprising LR and RF, for which no evidence of dependence is observed.

Table 5.2: **The whole chain of the diblock polyampholyte chains**: Cross tabulations of correct (C) and incorrect (W) predictions of **all outcomes** for pairs of base predictors. The p-values of tests assessing similarity in predictive capability using McNemar's test procedure are given in brackets.

		LR		ENET		SVM		RF	
		C	W	C	W	C	W	C	W
ENET	C	84	9						
	W	6	8						
		(0.6056)							
SVM	C	86	9	93	9				
	W	4	8	0	12				
		(0.2673)		(0.4795)					
RF	C	81	8	82	7	84	5		
	W	9	9	11	7	11	7		
		(1.0000)		(0.4795)		(0.2113)			
NN	C	88	7	91	4	92	3	84	11
	W	2	10	2	10	3	9	5	7
		(0.1824)		(0.6831)		(1.0000)		(0.2113)	

The results presented in Table 5.3 illustrate the pairwise tests for comparable predictive capability in determining the conformational phase diagram of a single block of the diblock polyampholyte chains. These findings reveal that the five models exhibit similar predictive performance across all pairs of base predictors, consistent with the results observed for the whole chain. Furthermore, the p-values of the independence tests for a single block of the diblock polyampholyte chains, evaluated in terms of accuracy and presented in Table 5.4, demonstrate that there is no evidence of dependence between ENET and the other two methods, SVM and NN. However, the

Table 5.3: **A single block of the diblock polyampholyte chains**: Cross tabulations of correct (C) and incorrect (W) predictions of **all outcomes** for pairs of base predictors. The p-values of tests assessing similarity in predictive capability using McNemar's test procedure are given in brackets.

		LR		ENET		SVM		RF	
		C	W	C	W	C	W	C	W
ENET	C	91	10						
	W	4	2						
		(0.1814)							
SVM	C	91	9	100	0				
	W	4	3	1	6				
		(0.2673)		(1.0000)					
RF	C	91	6	95	2	94	3		
	W	4	6	6	4	6	4		
		(0.7518)		(0.2888)		(0.5050)			
NN	C	93	8	99	2	98	3	96	5
	W	2	4	2	4	2	4	1	5
		(0.1138)		(1.0000)		(1.0000)		(0.2207)	

prediction models derived from all other pairs exhibit highly significant dependencies.

Next, a comparison of the LR, ENET, SVM, RF, and NN methods in predicting the conformational phase diagram is conducted by generating predictions over a grid constructed with the range of f from $\log 0.001$ to $\log 1$ in increments of 0.01, and the range of l_B from $\log 0.01$ to $\log 10$ in increments of 0.01, resulting in a total of $90,601$ data points. Figures 5.3 and 5.4 present the predicted conformational state diagrams for the entire chain and a single block of the diblock polyampholyte, respectively, in a good solvent, as obtained from these methods. From Figure 5.3, it is evident that

Table 5.4: P-values from Fisher Exact Tests of independence, between pairs of base predictors for two cases of the diblock polyampholyte chains.

Pairs of	P-value of Accuracy	
methods	The whole chain	A single block of chain
(LR, ENET)	0.0012	0.1716
(LR, SVM)	0.0001	0.5327
(LR, RF)	0.2571	1.0000
(LR, NN)	0.0000	1.0000
(ENET, SVM)	0.0000	0.0049
(ENET, RF)	0.0111	0.5221
(ENET, NN)	0.0000	0.0429
(SVM, RF)	0.0046	0.2778
(SVM, NN)	0.0000	0.2451
(RF, NN)	0.0468	0.3260

the LR, ENET, SVM, and NN methods construct a linear relationship in the log-log plot between the charge fraction f and the Bjerrum length l_B for both the boundary separating the coil and folding regimes and the boundary between the folding and weak association regimes. Similarly, Figure 5.4 shows that the LR, ENET, and SVM methods construct a linear boundary between the coil plus folding and weak association regimes, while the NN method displays a slight curvature, closely approximating a linear trend. Although the weak association regime boundary constructed by the LR method differs from the other three methods in both cases, likely due to the issue of high multicollinearity which can lead to unstable coefficient estimates, it still captures the expected linear behaviour. In contrast, the RF method demonstrates a non-linear relationship in both cases, reflecting the inherent nature of the random

forest modelling process.

The findings in the above paragraph, combined with the similar accuracies observed for the five methods, suggest that, in the absence of prior knowledge, which is frequently the case in such analyses, it is not immediately evident which of the five classification methods is most appropriate for this problem. Integrating the predictions using the majority vote (MV) ensemble method not only resolves this issue but also mitigates the risk associated with selecting a suboptimal method. Indeed, the consistency in accuracy observed across the five methods suggests that the MV ensemble will likely enhance overall prediction accuracy.

The boundary lines in Figures 5.3 and 5.4 were obtained using linear regression on the data points in the log-log plot. Standard error estimates were obtained by fitting a linear regression model to observed data points defining the predicted boundary of the regime. Line I in Figure 5.3 is the boundary between the coil and folding regimes based on the MV prediction of the conformational state diagram for whole chains in a good solvent and with $N = 256$. Theory predicts a power law dependence of l_B^{fold} on f at this boundary, for example, $l_B^{fold} \sim f^{-2}$, see equation (5.9). Line I in Figure 5.3 illustrates the relationship $\log l_B^{fold} = -1.992 \log f - 2.693$, or equivalently $l_B^{fold} \approx 10^{-2.693} f^{-1.992}$, where the exponent -1.992 has a standard error of 0.001. The coefficient C_f can be estimated by dividing the coefficient $10^{-2.693}$ by $N^{\nu-2}$, with $N = 256$ and $\nu = 0.588$, yielding $C_f \approx 5.10$. This value is notably close to $C_f \approx 4.70$ as reported by Wang and Rubinstein (2006), based on fitting the theoretical prediction to the simulation data. The MV ensemble method thus demonstrates a satisfactory prediction power in describing the boundary between the coil and folding regimes. Line II in Figure 5.3 represents the MV prediction for the boundary between the

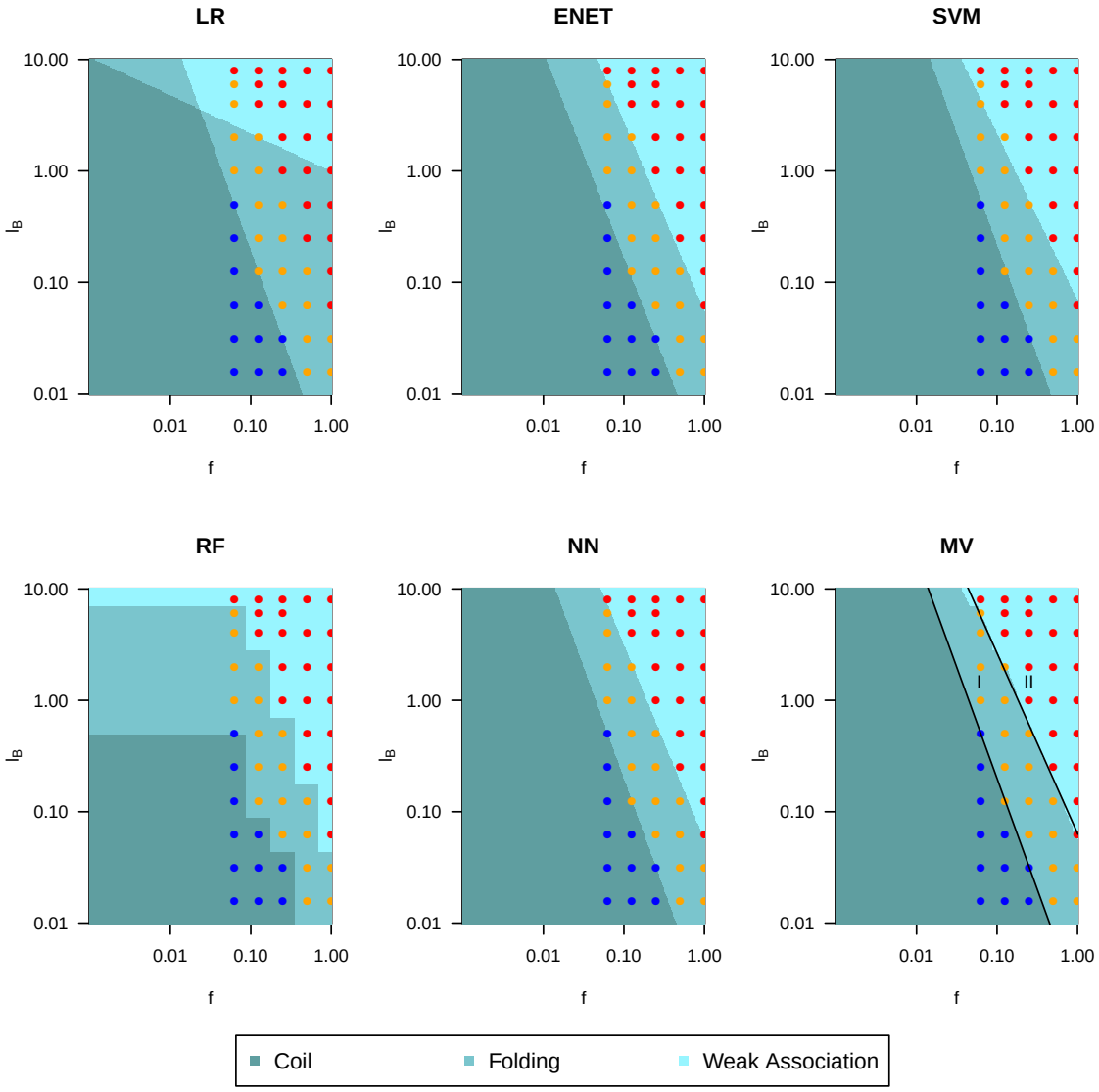


Figure 5.3: Predictions of conformational states of a diblock polyampholyte in a good solvent based on dichotomisation of the normalized radius of gyration of the whole chain R_g/R_{g0}. Line I in the MV plot is the boundary between the coil and folding regimes while line II is the boundary between the folding and weak association regimes. Matrix plots of the predictor variables f and l_B illustrate the conformational regions identified from MD simulations, categorized into distinct regimes: coil (blue points), folding (orange points), and weak association (red points).

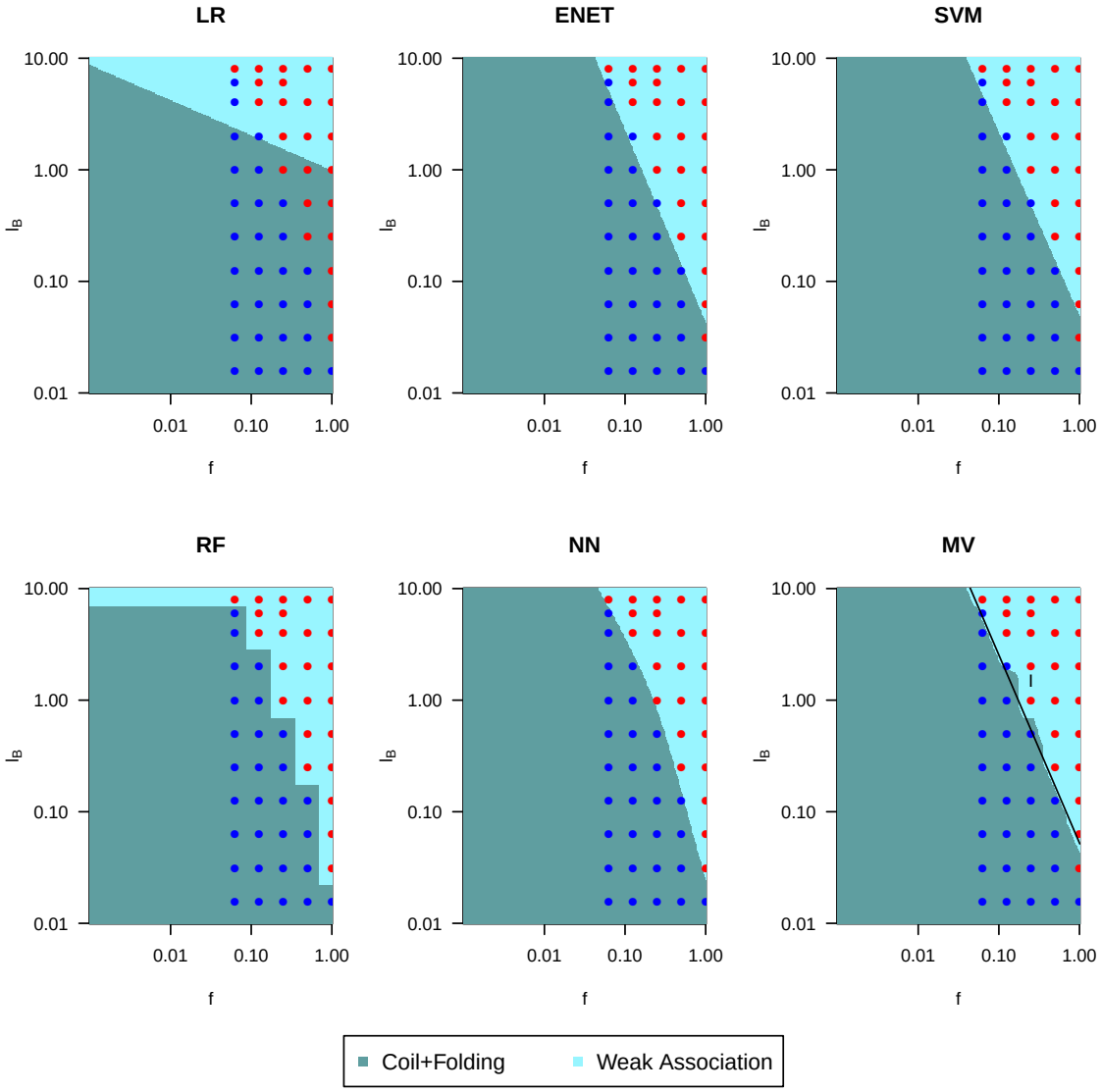


Figure 5.4: Predictions of conformational states of a diblock polyampholyte in a good solvent based on dichotomisation of the normalized radius of gyration of one block of the chain $R_{g,blk}/R_{g0}$. The solid line in the MV plot is the boundary between the coil+folding and weak association regimes. Matrix plots of the predictor variables f and l_B illustrate the conformational regions identified from MD simulations, categorized into distinct regimes: coil+folding (blue points) and weak association (red points).

folding and weak association regimes, that is, $\log l_B^{weak} = -1.618 \log f - 1.194$, or equivalently $l_B^{weak} \approx 10^{-1.194} f^{-1.618}$. The exponent -1.618 has a standard error of 0.006. It shows a relatively large discrepancy from the theoretically predicted value of -2, see equation (5.10). Consequently, the extracted coefficient, $C_w \approx 160.84$, differs significantly from $C_w \approx 60.00$ reported by Wang and Rubinstein (2006) whose data fitting was performed using the -2 power law. Such discrepancy can be related to the limited number of simulation data points in the training data set and the assumptions made in the theoretical predictions. The MV prediction for the boundary between the combined coil plus folding regime and the weak association regime is shown as Line I in Figure 5.4. It represents $\log l_B^{weak} = -1.685 \log f - 1.280$, or equivalently $l_B^{weak} = 10^{-1.280} f^{-1.685}$, where the exponent -1.685 has a standard error 0.009. The corresponding coefficient is $C_w \approx 131.95$. It can be seen that the inclusion of the coil regime data points has led to slightly different power law exponents and C_w values from those obtained from Line II in Figure 5.3 for the boundary between the folding and weak association regimes. It is noted that the individual blocks of the PA chains undergo a small swelling behaviour close to the end of the folding regime, leading to an increase in the $R_{g,blk}$ values. This swelling effect near the boundary between the folding and weakly association regime was not considered in the theoretical model, which may cause some discrepancy between the theoretical and ML predictions.

5.4 Discussion

In this chapter, the application of the majority vote (MV) ensemble method, introduced in Chapter 4, to predict the conformational regimes of single polyelectrolyte

chains has shown that while some of the base predictors within the ensemble produced boundaries deviating from theoretical expectations, integrating their prediction values using the MV ensemble effectively resolved these discrepancies. Similar to the AMP analysis, this application highlights that combining the outputs of multiple methods not only circumvents the challenge of selecting an optimal method but also mitigates the individual limitations of each approach, potentially leading to more accurate and theoretically consistent predictions. The findings here underscore the utility of the proposed ensemble method in addressing complex problems in polymer physics, demonstrating its ability to enhance predictive reliability and robustness in the face of methodological uncertainties.

Note that the criteria used in our ML models for determining the dichotomised values of the diblock polyampholyte chain sizes in different regimes are based on prior knowledge of the conformational behaviours of such chains in the descriptor space. Further study in this direction will involve predicting the true values of the chain sizes over the interested descriptor space and using them to identify different conformational regimes without relying on the prior knowledge of the system. This will help reveal conformational behaviours not captured by analysing a limited amount of simulation data points, examining the prediction power of existing theories and if needed guiding the development of new theoretical models.

This study did not attempt to predict the transition boundary between weak and strong association regimes due to insufficient training data from MD simulations, hindered by the high energy barriers in the strong association regime. This problem can potentially be resolved by combining efforts from two aspects: generating more reliable MD simulation data with the assistance of statistical sampling algorithms for

exploring rare events, and experimenting with MV and ML methods that can work with small training datasets.

Chapter 6

Conclusions

The research presented in this thesis advances the field of ensemble learning for binary classification by introducing a scalable and adaptable framework with broad applicability across diverse domains. By seamlessly integrating interpretability, robustness, and high predictive accuracy, the proposed methodology has proven to be a powerful and reliable tool for addressing complex classification challenges with precision and confidence. This chapter provides a summary of the work undertaken. Additionally, this chapter critically examines the limitations of the research, acknowledging areas where further refinement or exploration is needed. Finally, it outlines promising directions for future work, providing a roadmap for researchers to extend, refine and enhance the proposed methodology.

6.1 Summary

Presented and evaluated in this thesis is a novel method that strategically combines the strengths of diverse machine learning (ML) methods while addressing their individual limitations, to enhance predictive accuracy and strengthen variable selection in binary classification tasks. The proposed method was shown in this thesis to offer a robust framework for improved performance and interpretability of ML models. Inspired by the work of Vishnepolsky et al. (2018), this ensemble learning method is applied to explore the relation between physiochemical features and activities of antimicrobial peptides (AMPs), aiming to facilitate the design of effective AMPs. To the best of our knowledge, such a method has not been reported in the literature within this specific context.

The study of Vishnepolsky et al. (2018) distinguishes itself in the field of peptide research by explicitly incorporating variations in the mechanisms of action of AMPs, via factors such as charge, hydrophobicity, tilt angle and aggregation, and implementing a novel use of a density-based clustering method (DB-SCAN) to develop ML models for AMPs of 10-16 and 18-27 amino acid (aa) lengths active against specific strains of Gram-negative bacteria (*E.coli* ATCC 25922). While this approach for predicting AMP activity acknowledges the diversity in how AMPs interact with bacterial targets and provides a more nuanced representation of their antibacterial effects, a replication study using the same data and ML methodology as the authors used failed to reproduce the results reported in their paper.

The application of DB-SCAN by Vishnepolsky et al. (2018) within a supervised learning framework represents an innovative approach, the reported performance metrics

were promising, and the method was able to clearly define clusters of active peptides in the space of its predictor variables (feature space). However, when the DB-SCAN approach was applied to larger datasets, consisting of 644 active and 644 inactive 10-16 aa peptides, and 406 active and the same number of inactive 18-27 aa peptides obtained from the DBAASP data repository, the resulting ML models demonstrated a high likelihood of correctly identifying inactive peptides (high specificity) but performed poorly in accurately detecting active peptides (low sensitivity). This inability to reliably distinguish active AMPs, which is essential for practical applications, underscores a critical limitation of the clustering method.

A review of the literature showed that typical ML methods for predicting antimicrobial peptide (AMP) potency and aiding in peptide design include neural networks (NN) (Cherkasov et al., 2008; Fjell et al., 2009; Torrent et al., 2011; Mooney et al., 2013), support vector machines (SVMs) (Lata et al., 2007, 2010; Porto et al., 2012; Ng et al., 2015; Meher et al., 2017), random forests (RF) (Lira et al., 2013; Maccari et al., 2013; Youmans et al., 2017; Bhadra et al., 2018), and logistic regression (Clark et al., 2021), with the latter being particularly valued for its interpretability. These four methods, along with the logistic elastic net regression (ENET) method, which incorporates regularisation via a combination of both $\mathcal{L}_1$ and $\mathcal{L}_2$ penalties on its parameter estimates, were evaluated for their ability to predict clusters of active and inactive regions within the feature space under controlled conditions, and to deepen understanding about their comparative performances. Findings revealed that performance is varied and heavily influenced by both the complexity of the data and the model.

Evaluations using simulated data scenarios suggest LR and ENET are effective at identifying clusters of active outcomes but struggle with specificity, leading to higher

false positive rates and misclassification errors, especially in complex scenarios. While ENET improves adaptability of LR, particularly for more complex models, its performance still falls short of more flexible methods. The findings also suggest that, for clusters of active outcomes, NN prioritizes specificity but risks missing active outcomes, and its performance degrades with increasing data complexity. In contrast, due to their intrinsic flexibility, SVM and RF maintain lower error rates and consistent performance across diverse scenarios.

Regarding variable selection, the simulations revealed that LR, ENET, and NN tend to include only relevant predictor variables, and therefore optimize interpretability and scientific understanding of the problem. Conversely, SVM and RF, despite achieving lower error rates, occasionally include spurious variables, sacrificing some model interpretability for flexibility. All methods, however, demonstrated good performance when predictor variables associated with the outcome were included in the model, while excluding critical predictor variables led to poor performance. Overall, the simulation evaluations did not provide a clear understanding about which of these methods was preferable for modelling and interpreting data with clustered active outcomes, as each demonstrated distinct strengths and limitations, depending on the data distribution.

An evaluation of the five classification methods for predicting peptide activity using the larger DBAASP datasets yielded sensitivity and accuracy estimates ranging approximately between $80-90\%$, with slightly lower specificities in the range of $75-85\%$ for peptides of $10-16$ amino acids. For the longer $18-27$ aa peptides, sensitivities, specificities, and accuracies were consistently higher, falling within the $85-90\%$ range. These results demonstrate that all five methods substantially outperform the

DB-SCAN approach overall and have performances that are competitive with those reported in similar studies on peptide behaviour. Furthermore, considered alongside insights from the simulation studies, there is minimal performance differences among the methods themselves, which prompted an investigation into the potential value of integrating these methods into an ensemble framework.

Commonly used in classification tasks, the majority vote (MV) is a simple, yet powerful ensemble method used to combine the predictions of multiple base predictors in order to improve overall performance. Although its general effectiveness is widely acknowledged, there remains a lack of detailed understanding regarding its performance and nuanced behavior in practical scenarios. Addressing this gap, this thesis presents a comprehensive theoretical evaluation of the MV method under finite conditions. The analysis not only confirmed that optimal performance is achieved when base predictors are independent and exhibit similar accuracy levels, but more importantly provides valuable insights for ensemble building and a framework for understanding and predicting ensemble behaviour in scenarios where ideal assumptions may not hold.

A theoretical foundation for optimizing base predictor selection in practical applications is provided by the results established in this thesis that removing two entirely dependent predictors from an ensemble with an odd number of base predictors does not compromise performance. Additionally, the finding that the MV ensemble, while potentially biased towards dependent base predictors and occasionally falling short of the best individual base predictor, consistently outperforms the weakest predictor in the ensemble, underscores its reliability and resilience, and promotes confidence in this approach, even in less-than-ideal conditions.

A simulation-based evaluation of the MV ensemble, with LR, ENET, SVM, RF and NN base predictors, revealed that the majority vote mechanism can amplify the influence of base predictors with similar error patterns, which may compromise the ensemble's ability to mitigate individual model weaknesses. Also, if one or a few base predictors significantly underperform, the ensemble's performance may align more closely with these weaker predictors, particularly if the underperforming predictors are independent of the stronger ones. In such cases the errors of the weaker models are less likely to be offset by the stronger predictors, as their independence means their mistakes do not correlate with those of the better-performing predictors. This highlights the importance of ensuring that all base predictors, even if not equally accurate, maintain a reasonable level of performance and diversity to prevent the ensemble from being disproportionately affected by the weaker ones. Despite these limitations, the results demonstrated that the MV ensemble remains competitive with the best-performing base predictors, and reinforced the value of the majority vote as a reliable and versatile tool in ensemble learning.

The implementation of the best subset selection method within the ensemble framework in this thesis introduces a novel and innovative approach to enhancing optimal feature selection, addressing a significant challenge in machine learning. The proposed method, termed MV-FS to differentiate it from the MV ensemble, was developed in response to the observation that while individual base predictors in the best subset selection method may not consistently identify the most relevant set of features (predictor variables), this set frequently emerges as one of the top-performing candidates. Additionally, conventional ensemble methods tend to identify relevant features one at a time, overlooking the potential synergistic effects of feature combinations,

which can be critical for optimizing model performance. By integrating best subset selection based on models with interactions within an ensemble framework, the proposed MV-FS method addresses these issues by leveraging the individual strengths of the base predictors while simultaneously capturing the collective influence of feature combinations.

The proposed ensemble feature selection method offers other key advantages. By selecting the feature subset that is collectively most informative, the method reduces the likelihood of individual base predictors relying on spurious associations or irrelevant features, thereby providing a mechanism that mitigates the risk of overfitting. This improvement in feature selection accuracy through the use of an ensemble represents an important first step toward developing interpretable machine learning methods as it not only strengthens confidence in the chosen set of predictor variables, but also provides more reliable insights into the underlying structure of the data. Interpretability is essential for uncovering the physical mechanisms that govern the structural and dynamical behaviours of soft matter systems, as it transforms complex data into insights that are understandable to humans. This enables scientists to verify, trust, and build upon model findings, making the understanding of physical mechanisms as important as achieving high predictive accuracy.

Application of the MV-FS ensemble with base predictors LR, ENET, SVM, RF, and NN for predicting peptide activity using the larger DBAASP datasets resulted in only marginal changes to the overall performance of the MV ensemble that relied on the optimal model selected individually by each base predictor. Thus, the MV-FS method demonstrated its ability to maintain competitive accuracy while introducing

a more robust approach to feature selection. Furthermore, the MV-FS method consistently outperformed the DB-SCAN approach by Vishnepolsky et al. (2018) across all performance metrics, with the exception of specificity. Also, the ranges of the physicochemical predictor variables identified through MV-FS closely align with the values associated with active clusters identified by Vishnepolsky et al. (2018), validating robustness of the method.

To further explore the utility of the proposed MV approach, the method was applied to predicting the conformational regions of single diblock polyampholyte chains. The findings revealed that while some individual base predictors produced boundaries inconsistent with theoretical expectations, integrating their predictions through the MV ensemble effectively resolved these inconsistencies. Similar to its application in AMP analysis, this approach demonstrated that combining outputs from multiple methods not only eliminates the challenge of selecting the best single method but also mitigates the limitations inherent in individual predictors, yielding more accurate and theoretically aligned results. These outcomes underscore the value of the proposed ensemble method in addressing complex challenges and highlights its capacity to improve predictive accuracy and its resilience, despite methodological uncertainties.

6.2 Limitations and Further Work

In the majority vote (or "hard voting") method implemented in this thesis, each base predictor in the ensemble casts a single vote for a class, and the class with the majority of votes is selected as the final prediction. While this approach is straightforward and computationally efficient, it may not fully leverage the predictive capabilities of

individual base predictors, particularly when some base predictors are more confident in their predictions than others. An alternative approach, known as "soft voting", allows each base predictor to provide a probability distribution over all possible classes rather than a single class label. The final prediction is then determined by selecting the class with the highest average probability across all models. This method has the potential to improve ensemble performance, as it incorporates the confidence levels of individual base predictors into the decision-making process. For instance, if one base predictor is highly confident in its prediction while others are uncertain, the soft voting approach can give more weight to the confident predictor, leading to more accurate and reliable predictions.

Soft voting is particularly beneficial when the base predictors are well-calibrated, meaning their predicted probabilities accurately reflect the true likelihood of each class. In such cases, averaging probabilities can lead to a more nuanced and robust aggregation of predictions, especially in scenarios where the decision boundaries between classes are ambiguous or overlapping. However, if the base predictors are poorly calibrated, soft voting may introduce bias or noise, potentially degrading performance compared to hard voting. One potential avenue for further exploration is a theoretical evaluation of how soft voting impacts the MV-FS ensemble performance. Such an analysis would provide valuable insights into the advantages and limitations of soft voting compared to hard voting, as well as a deeper understanding of the conditions under which soft voting may be preferred.

Assigning probability distributions to base predictors' performances opens up new avenues for evaluating the ensemble performance beyond the complete dependence case examined in this thesis. Additionally, the use of probability distributions enables

the exploration of more sophisticated ensemble techniques, such as weighted voting or stacking. In weighted voting, the contributions of base predictors can be adjusted based on their performance or confidence levels. Stacking, on the other hand, involves training a meta-model to combine the predictions of the base predictors, potentially accounting for the complex interactions and dependencies among them.

Another promising approach to enhance the performance of the MV-FS method is to expand the ensemble size by incorporating additional classification methods, such as K-Nearest Neighbours (KNN) and Naive Bayes (NB), that bring unique strengths to the ensemble. However, it is important to ensure that base predictors have independent errors and comparable performance, as combining methods with highly dependent errors may degrade performance, while including predictors with vastly different performance levels dilutes the ensemble's effectiveness. To address this, it is proposed to conduct extensive simulation experiments across a wide range of controlled scenarios and datasets with varying characteristics, such as different data sizes, feature dimensions, and class distributions. This would help identify the optimal combination of base predictors for specific data structures and problem domains, and to develop guidelines for selecting the most effective set of base predictors tailored to specific data types and application contexts.

Being able to reliably identify relevant features and to characterise their role in the decision of a machine learning model, in other words to produce a more explainable model, is key to the development of a machine-learning system that is capable of explaining the rationale for its decision, that can critically evaluate its strengths and weaknesses, and that can convey an understanding of how/why decisions in the future will be made. While the findings in this thesis demonstrate that use of an ensemble

can aid in this regard, the current best subset selection approach is constrained to a reasonable number of predictors. Extending this approach to accommodate a larger number of predictors presents a promising direction for future research, but it also introduces the challenge of ensuring compatibility across base predictors in terms of the features used to build their individual models when combining them in an ensemble. One obvious approach to addressing this challenge is to develop a unified feature representation framework that ensures all base predictors operate on a consistent set of features, even as the number of predictors scales. This could involve techniques such as feature embedding or dimensionality reduction, using methods like Principal Component Analysis (PCA) to transform the original features into a shared space that preserves their predictive power while maintaining compatibility across models, but this approach complicates interpretability. Another approach, which is to be developed and evaluated, integrates a method such as forward selection into the ensemble framework to dynamically identify the most relevant set of features across all base predictors simultaneously.

The proposed MV-FS method in this thesis demonstrated strong and competitive predictive power and instilled confidence in the physicochemical properties identified as being involved in the antimicrobial activities of AMPs, as well as pinpointing regions of the feature space where active peptides are likely to be found. Further work involves integrating it with molecular simulation methods to reveal microscopic mechanisms of action underlying antimicrobial behaviours, and to provide guidance for designing novel antimicrobial materials with optimised compositions to fulfil the desired functions. Additionally, it is also planned to apply the methodology on other subsets of peptides in the DBAASP database as well as on a series of block peptide like

molecules with blocks of arginine sequences attached to non-ionic residues (Edwards-Gayle et al., 2019, 2020). In the latter case, it will be challenging to see if the MV-FS method can deal with these very artificial sequences which do not resemble those typically present in databases.

While the method is able to predict a partitioning of the feature space into active and inactive regions, extending this to include indeterminate clusters of peptides and to construct confidence regions for these clusters will be investigated. To achieve this, resampling methods such as bootstrap sampling and the Jackknife-after-bootstrap (Efron, 1979) technique will be explored. Bootstrap sampling involves repeatedly resampling the dataset with replacement to generate multiple subsets of the data, from which clusters of peptides can be constructed. In the basic Jackknife method, an estimator is obtained by deleting one observation from the data and recalculating the estimate using this reduced dataset. Repeating this process N times results in N Jackknife estimates from which an empirical variance can be obtained. Its extension, the Jackknife-after-bootstrap, uses bootstrap samples to generate initial estimates and then applies the Jackknife method to assess the stability and variability of these estimates. Although it is not immediately clear how these methods can be directly applied to construct confidence regions for peptide clusters in this specific problem, their key advantage lies in their non-parametric nature, as they do not require any assumptions about the underlying data distribution.

The preliminary test of the MV method in predicting the conformational transitions of the diblock polyampholyte chains also elucidated the potential of ensemble learning in predicting the structural and even dynamical properties of soft matter systems,

including biological and synthesised polymers, and colloidal systems, over a large parameter (descriptor) space. This is essentially important when the descriptor space is of high-dimensional and has wide ranges of parameter values, because the high computational costs prevent the brute force simulations from generating sufficient data points for clearly defining the boundaries between different structural or dynamical regimes. To achieve such goals, it requires the incorporation of ML methods in the ensemble that can produce model predictions with continuous output values, rather than the dichotomised values as used in the current work, and also base predictors that can work well with small datasets.

In conclusion, the MV ensemble was shown in this thesis to be a reliable and computationally efficient method for combining predictions, particularly in settings where the base predictors exhibit a balance of accuracy and diversity. While it may not always achieve optimal performance, its simplicity and competitive results make it a valuable tool in ensemble learning. The integration of best subset selection within the ensemble framework here represents a novel advancement in feature selection and ensemble learning that combines the strengths of both approaches to improve predictive performance and enhance interpretability. The resulting MV-FS ensemble framework creates not only a more comprehensive and reliable approach to enable deeper understanding of the underlying drivers of peptide activity, but can also be applied to tackle challenges in feature selection and predictions of physical properties of other soft matter systems, such as bio-mimicking charged polymers, and self-healing and self-oscillating gel systems.

Appendix A

Mathematics Results

A.1 Optimization

A.1.1 Gradient descent methods

Some common optimization methods for optimizing a differentiable, scalar function $f(\boldsymbol{x})$ of the vector $\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ are described here. Optimization by gradient descent methods, which choose a search direction and a step size at each iteration. The search direction has two simple choices: the negative gradient and the conjugate gradient algorithms.

The negative gradient algorithm

The negative gradient approach is optimised by moving in the direction of the negative gradient at the current position, which points towards the steepest decrease in the

function value. Suppose we wish to minimize the function $f(\boldsymbol{x})$. Write $\boldsymbol{x} = \boldsymbol{x}_0 + t\boldsymbol{v}$, where $\boldsymbol{x}_0 = \begin{pmatrix} x_{1,0} \\ \vdots \\ x_{n,0} \end{pmatrix}$ is some fixed vector, t is a scalar and $\boldsymbol{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$ is a vector with unit norm, and consider the function

$$g(t) = f(\boldsymbol{x}).$$

Using the chain rule to differentiate $g(t)$ with respect to t gives

$$\frac{dg(t)}{dt} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \ldots + \frac{\partial f}{\partial x_n}\frac{\partial x_n}{\partial t}.$$

Now as

$$x_1 = x_{1,0} + tv_1, \quad \text{we get} \quad \frac{\partial x_1}{\partial t} = v_1.$$

Similarly,

$$\frac{\partial x_2}{\partial t} = v_2, \ldots, \frac{\partial x_n}{\partial t} = v_n,$$

and hence we can write

$$\frac{dg(t)}{dt} = \frac{\partial f}{\partial x_1}v_1 + \ldots + \frac{\partial f}{\partial x_n}v_n = \nabla f(\boldsymbol{x}) \cdot \boldsymbol{v},$$

where $\nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$. As we selected $\boldsymbol{v}$ satisfying $\|v\| = 1$, the inner product $\nabla f(\boldsymbol{x}) \cdot \boldsymbol{v} = \|\nabla f(\boldsymbol{x})\| \cos(\theta)$ and hence we get

$$\frac{dg(t)}{dt} = \|\nabla f(\boldsymbol{x})\| \cos(\theta),$$

where θ is the angle between $\nabla f(\boldsymbol{x})$ and $\boldsymbol{v}$. By definition, the norm is non-negative and hence $\frac{dg(t)}{dt}$ will be a minimum when $\cos(\theta) = -1$ (its minimum value). In other words, $\frac{dg(t)}{dt}$ is a minimum when the angle between $\nabla f(\boldsymbol{x})$ and $\boldsymbol{v}$ is π or $180°$. This occurs when

$$\boldsymbol{v} = -\frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|}.$$

If t is small then $\boldsymbol{x} \approx \boldsymbol{x}_0$ and

$$\boldsymbol{v} \approx -\frac{\nabla f(\boldsymbol{x}_0)}{\|\nabla f(\boldsymbol{x}_0)\|},$$

and we can reduce the problem of minimizing the function $f(\boldsymbol{x})$ over several variables to minimizing $g(t)$ over the single variable t, for this choice of $\boldsymbol{v}$. In particular, for some starting value $\boldsymbol{x}_0$, we find the value t_0 that minimizes

$$g_0(t) = f(\boldsymbol{x}_0 - t\nabla f(\boldsymbol{x}_0)).$$

Next, we set

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - t_0\nabla f(\boldsymbol{x}_0),$$

find the value t_1 that minimises,

$$g_1(t) = f(\boldsymbol{x}_1 - t\nabla f(\boldsymbol{x}_1)),$$

and set

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 - t_1\nabla f(\boldsymbol{x}_1).$$

Continuing in this way we generate a sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$ that converges to the minimum of $f(\boldsymbol{x})$. This is the steepest descent method.

The conjugate gradient algorithm

The conjugate gradient in principle is similar to the method of orthogonal vectors for solving linear systems proposed by Fox et al. (1948). Optimization using the conjugate gradient algorithm is as follows.

First note that two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are said to be *conjugate* with respect to some positive semi-definite matrix $\boldsymbol{A}$ if $\boldsymbol{x}^T\boldsymbol{A}y = \boldsymbol{0}$, which a positive semi-definite matrix $\boldsymbol{A}$ satisfies $\boldsymbol{x}^T\boldsymbol{A}x \geq \boldsymbol{0}$ for all vectors $\boldsymbol{x}$.

Now, we just saw that the k^{th} iteration of the steepest gradient method minimizes

$$f(\boldsymbol{x}_k - t_k \nabla f(\boldsymbol{x}_k)).$$

Thus at the k^{th} iteration this method searches for the minimum of $f(\boldsymbol{x})$ in the direction $-\nabla f(\boldsymbol{x}_k)$. The conjugate gradient method is a modification of the steepest descent method to improve the rate of convergence. In particular, at the k^{th} iteration, the conjugate gradient method searches for the minimum in a direction that is conjugate to the previous search direction. Let the initial search direction by

$$\boldsymbol{d}_0 = -\nabla f(\boldsymbol{x}_0).$$

In the next step, the conjugate gradient method minimizes

$$f(\boldsymbol{x}_1 - t_1 \boldsymbol{d}_1),$$

where $\boldsymbol{d}_1$ is conjugate to $\boldsymbol{d}_0$. In the second iterate, the search direction $\boldsymbol{d}_2$ is chosen to be conjugate to both of the previous directions $\boldsymbol{d}_0$ and $\boldsymbol{d}_1$. Continuing in this way, at

the k^{th} iterate, the search direction $\boldsymbol{d}_k$ is chosen to be conjugate to previous directions $\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{k-1}$.

The conjugate gradient method can be regarded as being between the method of steepest descent, which is based on a first-order approximation of $f(\boldsymbol{x})$, and Newton's method which is based on a second-order approximation.

A.1.2 Lagrange multipliers method

The Lagrange multiplier method is a mathematical methodology for determining the local maxima and minima of a function subject to constraints. To convert a constrained optimisation problem to an unconstrained one by introducing extra variables called Lagrange multipliers. Given an objective function $f(x_1, x_2, \ldots, x_n)$, we want to find the local maximum and minimum values. There is a constraint $g(x_1, x_2, \ldots, x_n) = 0$, for which there may be multiple such constraints. Optimization of the Lagrange multiplier is as follows.

First, we define the Lagrangian function $\mathcal{L}$, which combines the objective function and the constraint by introducing a new variable λ, known as the Lagrange multiplier. The Lagrangian is defined as,

$$\mathcal{L}(x_1, x_2, \ldots, x_n, \lambda) = f(x_1, x_2, \ldots, x_n) + \lambda g(x_1, x_2, \ldots, x_n),$$

where λ is the Lagrange multiplier, which shows how the constraint influences the optimization of f. Next, find the critical points that maximize and minimize the function f subject to the constraint g, we need to solve the system of equations

formed by setting the gradient of $\mathcal{L}$ equal to zero,

$$\nabla\mathcal{L}(x_1, x_2, \ldots, x_n, \lambda) = 0.$$

Thus, the system of equations is

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial x_1} = 0 \\ \frac{\partial\mathcal{L}}{\partial x_2} = 0 \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\lambda} = 0. \end{cases}$$

Then, solve the system of equations, which has a system of $n+1$ equations with $n+1$ unknown variables $x_1, x_2, \ldots, x_n$ and λ. To find the solutions of the system, we solve the system to identify the potential points where the function may be maximized or minimized, subject to the constraint. Finally, interpret the results by representing the critical points consisting of maximum, minimum or saddle points. We also need to further analyze or test points to determine the nature of the solutions.

A.2 Construction of Orthogonal Polynomial

Orthogonal polynomials have a crucial role in classification methods by constructing more efficient models, mitigating overfitting, and improving interpretability by capturing essential data features. A common approach to generating a set of orthogonal polynomials is the Gram-Schmidt process, an orthogonalization technique that converts a set of linearly independent vectors into an orthogonal set.

Consider a dataset $\boldsymbol{X}$ consisting n paired observations $(y_i, \boldsymbol{x}_i)$, where $y_i \in \{1, -1\}$

represents the binary class label for the i-th observation, and $\boldsymbol{x}_i \in \mathbb{R}^d$ denotes the corresponding feature vector with d features. Next, we define a set of polynomial basis functions $\{p_0(\boldsymbol{x}), p_1(\boldsymbol{x}), p_2(\boldsymbol{x}), \ldots, p_m(\boldsymbol{x})\}$, which will be orthogonalized using the Gram-Schmidt process. A common choice for the basis functions is to use the monomials of $\boldsymbol{x}$, such as:

$$p_0(\boldsymbol{x}) = 1,\ p_1(\boldsymbol{x}) = \boldsymbol{x},\ p_2(\boldsymbol{x}) = \boldsymbol{x}^2,\ p_3(\boldsymbol{x}) = \boldsymbol{x}^3.\ \ldots$$

The original basis functions $\{q_0(\boldsymbol{x}), q_1(\boldsymbol{x}), q_2(\boldsymbol{x}), \ldots, q_m(\boldsymbol{x})\}$, are defined as polynomials or other features derived from the dataset.

The Gram-Schmit process is used to orthogonalize these functions. The procedure is as follows: start with the first polynomial $p_0(\boldsymbol{x})$. This is the first orthogonal polynomial,

$$\phi_0(\boldsymbol{x}) = p_0(\boldsymbol{x}).$$

The second step, orthogonalize the second polynomial $p_1(\boldsymbol{x})$. This is the second orthogonal polynomial,

$$\phi_1(\boldsymbol{x}) = p_1(\boldsymbol{x}) - \frac{\langle p_1(\boldsymbol{x}), \phi_0(\boldsymbol{x})\rangle}{\langle \phi_0(\boldsymbol{x}), \phi_0(\boldsymbol{x})\rangle} \cdot \phi_0(\boldsymbol{x})$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. The inner product can be defined as,

$$\langle f(\boldsymbol{x}), g(\boldsymbol{x})\rangle = \sum_{i=1}^{n} f(\boldsymbol{x}_i) g(\boldsymbol{x}_i),$$

which measures the similarity between the two polynomials evaluated at each sample point. The third step, orthogonalize subsequent polynomials $p_2(\boldsymbol{x}), p_3(\boldsymbol{x}), \ldots$. For

each k-th polynomial, subtract the projection of $p_k(\boldsymbol{x})$ onto all previously orthogonalized polynomials,

$$\phi_k(\boldsymbol{x}) = p_k(\boldsymbol{x}) - \sum_{j=1}^{k-1} \frac{\langle p_k(\boldsymbol{x}), \phi_j(\boldsymbol{x}) \rangle}{\langle \phi_j(\boldsymbol{x}), \phi_j(\boldsymbol{x}) \rangle} \cdot \phi_j(\boldsymbol{x}).$$

This guarantees that each new polynomial $\phi_k(\boldsymbol{x})$ is orthogonal to all previously constructed polynomials. The final step is to normalize the polynomials to ensure they are of unit length,

$$\psi_k(\boldsymbol{x}) = \frac{\phi_k(\boldsymbol{x})}{\|\phi_k(\boldsymbol{x})\|}.$$

This step may be necessary if you wish to maintain the orthonormality of the basis functions.

A set of orthogonal polynomials $\{\psi_0(\boldsymbol{x}), \psi_1(\boldsymbol{x}), \psi_2(\boldsymbol{x}), \ldots, \psi_m(\boldsymbol{x})\}$, is prepared and ready for use in classification. Transforming the original dataset $\boldsymbol{X}$ into a new dataset $\boldsymbol{Z}$ by evaluating each orthogonal polynomial on the input features $\boldsymbol{x}_i$,

$$\boldsymbol{Z} = \begin{pmatrix} \psi_0(\boldsymbol{x}_1) & \psi_1(\boldsymbol{x}_1) & \psi_2(\boldsymbol{x}_1) & \cdots & \psi_m(\boldsymbol{x}_1) \\ \psi_0(\boldsymbol{x}_2) & \psi_1(\boldsymbol{x}_2) & \psi_2(\boldsymbol{x}_2) & \cdots & \psi_m(\boldsymbol{x}_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \psi_0(\boldsymbol{x}_n) & \psi_1(\boldsymbol{x}_n) & \psi_2(\boldsymbol{x}_n) & \cdots & \psi_m(\boldsymbol{x}_n) \end{pmatrix}$$

This transformed dataset $\boldsymbol{Z}$ has features that are orthogonal to each other.

A.3 Proof of the Relationship between Binomial and Beta Distributions

Result A.3.1. *For positive integers $\alpha < \beta$ and $0 < p < 1$,*

$$\int_{t=0}^{p} t^{\alpha-1}(1-t)^{\beta-\alpha}dt = \frac{\Gamma(\alpha)\Gamma(\beta-\alpha+1)}{\Gamma(\beta+1)} \sum_{k=\alpha}^{\beta} \binom{\beta}{k} p^k (1-p)^{\beta-k}.$$

The above integral is known as the incomplete Beta function and is related to the Beta function, or Euler's integral of the first kind, which is defined as $\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,dt$ for $\alpha, \beta > 0$.

Proof. Use integration by parts with $u = (1-t)^{\beta-\alpha}$ and $dv = t^{\alpha-1}$ to get

$$\int_{t=0}^{p} t^{\alpha-1}(1-t)^{\beta-\alpha}dt = \frac{1}{\alpha}p^{\alpha}(1-p)^{\beta-\alpha} + \frac{\beta-\alpha}{\alpha}\int_{t=0}^{p} t^{\alpha}(1-t)^{\beta-(\alpha+1)}dt,$$

which can be rewritten as

$$= \frac{(\alpha-1)!(\beta-\alpha)!}{\beta!}\left[\binom{\beta}{\alpha}p^{\alpha}(1-p)^{\beta-\alpha} + \frac{\beta!}{\alpha!\,(\beta-(\alpha+1))!}\int_{t=0}^{p} t^{\alpha}(1-t)^{\beta-(\alpha+1)}dt\right]. \tag{A.1}$$

Using $\Gamma(n+1) = n!$ for n a non-negative integer, and noting that if $\beta = \alpha + 1$ the integral on the right-hand side evaluates to $\frac{1}{\alpha+1}p^{\alpha+1}$ gives,

$$\int_{t=0}^{p} t^{\alpha-1}(1-t)^{\beta-\alpha}dt = \frac{\Gamma(\alpha)\Gamma(\beta-\alpha+1)}{\Gamma(\beta+1)}\left[\binom{\beta}{\alpha}p^{\alpha}(1-p)^{\beta-\alpha} + \binom{\beta}{\alpha+1}p^{\alpha+1}(1-p)^{\beta-(\alpha+1)}\right],$$

and the proof is complete. Otherwise, another application of integration by parts to

the second term of equation (A.1) gives,

$$\int_{t=0}^{p} t^{\alpha}(1-t)^{\beta-(\alpha+1)}dt = \frac{1}{\alpha+1}p^{\alpha+1}(1-p)^{\beta-(\alpha+1)} + \frac{\beta-(\alpha+1)}{\alpha+1}\int_{t=0}^{p} t^{\alpha+1}(1-t)^{\beta-(\alpha+2)}dt,$$

and hence

$$\begin{aligned}\int_{t=0}^{p} t^{\alpha-1}(1-t)^{\beta-\alpha}dt =& \frac{\Gamma(\alpha)\Gamma(\beta-\alpha+1)}{\Gamma(\beta+1)}\left[\binom{\beta}{\alpha}p^{\alpha}(1-p)^{\beta-\alpha} + \binom{\beta}{\alpha+1}p^{\alpha+1}(1-p)^{\beta-(\alpha+1)}\right.\\ &\left. + \frac{\beta!}{(\alpha+1)!(\beta-(\alpha+2))!}\int_{t=0}^{p} t^{\alpha+1}(1-t)^{\beta-(\alpha+2)}dt\right].\end{aligned}$$

If $\beta = \alpha + 1$ the above integral evaluates to $\frac{1}{\alpha+2}p^{\alpha+2}$ and, as above, the proof is complete. Otherwise, continue integrating by parts to get, after r steps,

$$\begin{aligned}\int_{t=0}^{p} t^{\alpha-1}(1-t)^{\beta-\alpha}dt =& \frac{\Gamma(\alpha)\Gamma(\beta-\alpha+1)}{\Gamma(\beta+1)}\left[\sum_{k=\alpha}^{\alpha+r-1}\binom{\beta}{k}p^{k}(1-p)^{\beta-k}\right.\\ &\left. + \frac{\beta!}{(\alpha+r-1)!(\beta-(\alpha+r))!}\int_{t=0}^{p} t^{\alpha+r-1}(1-t)^{\beta-(\alpha+r)}dt\right],\end{aligned}$$

and the proof is complete when the power of $1-t$ in the integral reduces to zero. □

Result A.3.2. *For integers $r = 0, 1, 2, \ldots$,*

$$\sum_{k=r+1}^{2r+1}\binom{2r+1}{k} = 2^{2r}.$$

Proof. Writing

$$\sum_{k=0}^{2r+1}\binom{2r+1}{k} = \sum_{k=0}^{r}\binom{2r+1}{k} + \sum_{k=r+1}^{2r+1}\binom{2r+1}{k},$$

and then increasing the index of the first sum on the right hand side by $r+1$ gives

$$\sum_{k=0}^{2r+1}\binom{2r+1}{k} = \sum_{k=r+1}^{2r+1}\binom{2r+1}{2r+1-k} + \sum_{k=r+1}^{2r+1}\binom{2r+1}{k},$$

which, as $\binom{2r+1}{k} = \binom{2r+1}{2r+1-k}$, becomes

$$= 2\sum_{k=r+1}^{2r+1}\binom{2r+1}{k}.$$

Hence, using the binomial theorem $(a+b)^{2r+1} = \sum_{k=0}^{2r+1}\binom{2r+1}{k}a^k b^{2r+1-k}$ with $a=b=1$,

$$\sum_{k=r+1}^{2r+1}\binom{2r+1}{k} = \frac{1}{2}\sum_{k=0}^{2r+1}\binom{2r+1}{k} = \frac{1}{2}2^{2r+1} = 2^{2r}.$$

□

Result A.3.3. *For integers $r = 1, 2, \ldots$,*

$$\sum_{k=r+1}^{2r}\binom{2r}{k} + \sum_{k=r}^{2r}\binom{2r}{k} = 2^{2r}.$$

Proof. Again, splitting the sum $\sum_{k=0}^{2r}\binom{2r}{k} = \sum_{k=0}^{r-1}\binom{2r}{k} + \binom{2r}{r} + \sum_{k=r+1}^{2r}\binom{2r}{k}$ and using $\binom{2r}{k} = \binom{2r}{2r-k}$ gives,

$$\sum_{k=0}^{2r}\binom{2r}{k} = 2\sum_{k=r+1}^{2r}\binom{2r}{k} + \binom{2r}{r},$$

which, by including $\binom{2r}{r}$ under the summation, can also be written as

$$\sum_{k=0}^{2r}\binom{2r}{k} = 2\sum_{k=r}^{2r}\binom{2r}{k} - \binom{2r}{r}.$$

The result is obtained by adding these two equations and using the binomial theorem.

□

Appendix B

Supplementary Tables

B.1 Re-analysis of Small Datasets

Table B.1: Stable clusters of peptides 10-16 aa length by physicochemical properties forming the space and cluster size.

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
2D	C1	∗					∗				17
3D	C2	∗					∗			∗	17
	C3		∗	∗	∗						16
	C4		∗	∗		∗					15
	C5		∗	∗			∗				14
	C6		∗	∗					∗		15
	C7			∗				∗	∗		14

Continued on next page

Table B.1: continued from previous page

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
4D	C8	∗		∗	∗		∗				15
	C9	∗		∗	∗				∗		14
	C10		∗	∗	∗		∗				14
	C11		∗	∗	∗				∗		15
	C12		∗	∗	∗					∗	16
	C13		∗	∗		∗	∗				11
	C14		∗	∗		∗			∗		14
	C15		∗	∗		∗				∗	15
	C16		∗	∗			∗	∗			14
	C17		∗	∗			∗		∗		14
	C18		∗	∗			∗			∗	14
	C19		∗	∗				∗	∗		15
	C20		∗	∗					∗	∗	15
	C21		∗			∗		∗	∗		16
	C22			∗		∗		∗	∗		14
	C23			∗				∗	∗	∗	14
5D	C24	∗		∗	∗	∗	∗				14
	C25	∗		∗	∗		∗			∗	15
	C26	∗			∗	∗		∗	∗		14
	C27	∗		∗	∗	∗	∗				15

Continued on next page

Table B.1: continued from previous page

Space	Stable Cluster	Physical-chemical properties									Cluster size
		M	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	
	C28		*	*	*	*			*		14
	C29		*	*	*		*	*			14
	C30		*	*	*		*		*		15
	C31		*	*	*		*			*	14
	C32		*	*	*			*	*		16
	C33		*	*	*				*	*	15
	C34		*	*		*	*	*			14
	C35		*	*		*	*		*		14
	C36		*	*		*	*			*	11
	C37		*	*		*		*	*		14
	C38		*	*		*			*	*	14
	C39		*	*			*	*	*		14
	C40		*	*			*	*		*	14
	C41		*	*			*		*	*	14
	C42		*	*				*	*	*	15
	C43		*		*	*	*		*		11
	C44		*		*	*		*	*		15
	C45		*			*		*	*	*	16
	C46			*	*	*		*	*		14
	C47			*	*		*	*	*		14

Continued on next page

Table B.1: continued from previous page

Space	Stable Cluster	Physical-chemical properties									Cluster size
		M	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	
	C48			*		*		*	*	*	14
6D	C49	*	*	*		*		*	*		15
	C50	*	*	*			*	*	*		15
	C51	*		*	*	*	*			*	14
	C52	*			*	*		*	*	*	14
	C53		*	*	*	*	*	*			14
	C54		*	*	*	*	*		*		15
	C55		*	*	*	*	*			*	15
	C56		*	*	*	*		*	*		15
	C57		*	*	*	*			*	*	14
	C58		*	*	*		*	*	*		15
	C59		*	*	*		*	*		*	14
	C60		*	*	*		*		*	*	15
	C61		*	*	*			*	*	*	16
	C62		*	*		*	*	*	*		15
	C63		*	*		*	*	*		*	14
	C64		*	*		*	*		*	*	14
	C65		*	*		*		*	*	*	14
	C66		*	*			*	*	*	*	14
	C67		*		*	*	*	*	*		15

Continued on next page

Table B.1: continued from previous page

Space	Stable Cluster	Physical-chemical properties									Cluster size
		M	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	
	C68		*		*	*	*		*	*	11
	C69		*		*	*		*	*	*	15
	C70			*	*	*		*	*	*	14
	C71			*	*		*	*	*	*	14
7D	C72	*	*	*	*	*		*	*		15
	C73	*	*	*	*		*	*	*		15
	C74	*	*	*		*	*	*	*		14
	C75	*	*	*		*		*	*	*	15
	C76	*	*	*			*	*	*	*	15
	C77		*	*	*	*	*	*	*		15
	C78		*	*	*	*	*	*		*	14
	C79		*	*	*	*	*		*	*	15
	C80		*	*	*	*		*	*	*	14
	C81		*	*	*		*	*	*	*	15
	C82		*	*		*	*	*	*	*	15
	C83		*		*	*	*	*	*	*	15
8D	C84	*	*	*	*	*	*	*	*		15
	C85	*	*	*	*	*		*	*	*	15
	C86	*	*	*	*		*	*	*	*	15
	C87	*	*	*		*	*	*	*	*	15

Continued on next page

Table B.1: continued from previous page

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
	C88		*	*	*	*	*	*	*	*	15
9D	C89	*	*	*	*	*	*	*	*	*	15

Table B.2: Stable clusters of peptides 18-27 aa length by physicochemical properties forming the space and cluster size.

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
2D	C1			*					*		16
	C2				*		*				48
3D	C3		*		*		*				94
	C4			*	*				*		16
	C5			*		*			*		14
	C6			*			*		*		14
	C7			*					*	*	14
	C8				*	*	*				89
	C9				*		*		*		62
4D	C10	*	*		*		*				94
	C11	*			*		*	*			95
	C12		*		*	*	*				89
	C13			*	*	*	*				14

Continued on next page

Table B.2: continued from previous page

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
	C14			*	*		*		*		14
	C15			*	*				*	*	14
	C16			*				*	*	*	23
5D	C17	*	*		*	*	*				88
	C18	*	*		*		*		*		14
	C19		*	*	*	*	*				87
	C20		*	*	*	*			*		89
	C21		*	*	*			*	*		14
	C22		*		*	*	*	*			89
	C23		*		*	*	*		*		88
	C24				*	*	*	*	*		72
6D	C25	*	*	*	*	*	*				21
	C26	*	*		*	*	*	*			88
	C27	*	*		*		*		*	*	14
	C28		*	*	*	*	*	*			87
	C29		*		*	*	*	*	*		89
	C30		*		*	*	*		*	*	14

Evaluation on training sets for small datasets.

The training set used by Vishnepolsky et al. (2018) for peptides of 10-16 aa length

consists of 140 active peptides and 140 inactive peptides. However, the training sets used in the re-analysis were similar to Vishnepolsky et al. (2018), each consisting of 140 active and 140 inactive for 10-16 aa length of peptides. Results of P_i and properties of a stable cluster of peptides of 10-16 aa length for cluster 1 and cluster 2 are reported in Tables B.3 and B.4, respectively. We see that cluster 1 has only one stable cluster, the highest value of P_i is approximately 0.12. So, we decided to select a set of physical-chemical properties in the stable cluster C9 in 4D space for AMP prediction, consisting of hydrophobic moment (M), net charge (C), isoelectric point (I) and linear moment (L). Cluster 2 has 88 stable clusters, with stable clusters C54 in 6D space and C79 in 7D space having the highest P_i values of roughly 1.51, the physical-chemical properties in stable cluster C54 are a subset of stable cluster C79. So, we decided to select a set of physical-chemical properties in stable cluster C54 in 6D space for AMP prediction, consisting of hydrophobicity (H), net charge (C), isoelectric point (I), penetration depth (D), tilt angle (O) and linear moment (L).

Table B.3: Results of P_i and properties of a stable cluster of peptides of 10-16 aa length: cluster 1. The highest of P_i is highlighted in blue.

Space	Stable Cluster	P_i	Properties
4D	C9	0.12	*MCIL*

Table B.4: Results of P_i and properties of a stable cluster of peptides of 10-16 aa length: cluster 2. The highest of P_i is highlighted in blue.

Space	Stable Cluster	P_i	Properties
2D	C1	0.59	*MO*

Continued on next page

Table B.4: continued from previous page

Space	Stable Cluster	P_i	Properties
3D	C2	0.59	*MOA*
	C3	0.09	*HCI*
	C4	0.60	*HCD*
	C5	0.56	*HCO*
	C6	0.72	*HCL*
	C7	0.20	*CRL*
4D	C8	0.21	*MCIO*
	C10	0.20	*HCIO*
	C11	0.29	*HCIL*
	C12	0.09	*HCIA*
	C13	0.47	*HCDO*
	C14	1.20	*HCDL*
	C15	0.60	*HCDA*
	C16	0.60	*HCOR*
	C17	1.10	*HCOL*
	C18	0.56	*HCOA*
	C19	0.28	*HCRL*
	C20	0.72	*HCLA*
	C21	0.11	*HDRL*
	C22	0.40	*CDRL*
	C23	0.20	*CRLA*

Continued on next page

Table B.4: continued from previous page

Space	Stable Cluster	P_i	Properties
5D	C24	0.30	*MCIDO*
	C25	0.21	*MCIOA*
	C26	0.10	*MIDRL*
	C27	0.28	*MCIDO*
	C28	1.10	*HCIDL*
	C29	0.60	*HCIOR*
	C30	1.34	*HCIOL*
	C31	0.20	*HCIOA*
	C32	0.49	*HCIRL*
	C33	0.29	*HCILA*
	C34	0.60	*HCDOR*
	C35	0.70	*HCDOL*
	C36	0.47	*HCDOA*
	C37	0.50	*HCDRL*
	C38	1.20	*HCDLA*
	C39	0.40	*HCORL*
	C40	0.60	*HCORA*
	C41	1.10	*HCOLA*
	C42	0.28	*HCRLA*
	C43	0.31	*HIDOL*
	C44	0.42	*HIDRL*

Continued on next page

Table B.4: continued from previous page

Space	Stable Cluster	P_i	Properties
	C45	0.11	*HDRLA*
	C46	0.40	*CIDRL*
	C47	0.10	*CIORL*
	C48	0.40	*CDRLA*
6D	C49	0.64	*MHCDRL*
	C50	0.43	*MHCORL*
	C51	0.30	*MCIDOA*
	C52	0.10	*MIDRLA*
	C53	0.60	*HCIDOR*
	C54	1.51	*HCIDOL*
	C55	0.28	*HCIDOA*
	C56	0.61	*HCIDRL*
	C57	1.10	*HCIDLA*
	C58	1.12	*HCIORL*
	C59	0.60	*HCIORA*
	C60	1.34	*HCIOLA*
	C61	0.49	*HCIRLA*
	C62	0.94	*HCDORL*
	C63	0.60	*HCDORA*
	C64	0.70	*HCDOLA*
	C65	0.50	*HCDRLA*

Continued on next page

Table B.4: continued from previous page

Space	Stable Cluster	P_i	Properties
	C66	0.40	*HCORLA*
	C67	0.76	*HIDORL*
	C68	0.31	*HIDOLA*
	C69	0.42	*HIDRLA*
	C70	0.40	*CIDRLA*
	C71	0.10	*CIORLA*
7D	C72	0.71	*MHCIDRL*
	C73	0.62	*MHCIORL*
	C74	0.64	*MHCDORL*
	C75	0.65	*MHCDRLA*
	C76	0.43	*MHCORLA*
	C77	1.12	*HCIDORL*
	C78	0.60	*HCIDORA*
	C79	1.51	*HCIDOLA*
	C80	0.61	*HCIDRLA*
	C81	1.12	*HCIORLA*
	C82	0.94	*HCDORLA*
	C83	0.76	*HIDORLA*
8D	C84	1.04	*MHCIDORL*
	C85	0.71	*MHCIDRLA*
	C86	0.62	*MHCIORLA*

Continued on next page

Table B.4: continued from previous page

Space	Stable Cluster	P_i	Properties
	C87	0.64	*MHCDORLA*
	C88	1.12	*HCIDORLA*
9D	C89	1.04	*MHCIDORLA*

For peptides of 18-27 aa length, the training set used by Vishnepolsky et al. (2018) consists of 100 active and the same amount of inactive peptides. While the training sets used in the re-analysis were similar to Vishnepolsky et al. (2018), each consisting of 100 active and 100 inactive for 18-27 aa length of peptides. Results of P_i and properties of a stable cluster of peptides of 18-27 aa length only one cluster is reported in Table B.5. We see that cluster 1 has 30 stable clusters, with stable cluster C23 in 5D space having the highest P_i values of roughly 19.60. So, we decided to select a set of physical-chemical properties in the stable cluster C23 in 5D space for AMP prediction, consisting of hydrophobicity (H), isoelectric point (I), penetration depth (D), tilt angle (O) and linear moment (L).

Table B.5: Results of P_i and properties of a stable cluster of peptides of 18-27 aa length: cluster 1. The highest of P_i is highlighted in blue.

Space	Stable Cluster	P_i	Properties
2D	C1	2.06	*CL*
	C2	0.91	*IO*
3D	C3	8.81	*HIO*
	C4	4.27	*CIL*

Continued on next page

Table B.5: continued from previous page

Space	Stable Cluster	P_i	Properties
	C5	1.82	*CDL*
	C6	0.14	*COL*
	C7	1.26	*CLA*
	C8	2.44	*IDO*
	C9	1.80	*IOL*
4D	C10	12.61	*MHIO*
	C11	4.36	*MIOR*
	C12	13.29	*HIDO*
	C13	0.14	*CIDO*
	C14	0.28	*CIOL*
	C15	3.83	*CILA*
	C16	1.34	*CRLA*
5D	C17	0.84	*MHIDO*
	C18	0.28	*MHIOL*
	C19	8.50	*HCIDO*
	C20	4.12	*HCIDL*
	C21	0.28	*HCIRL*
	C22	11.60	*HIDOR*
	C23	19.60	*HIDOL*
	C24	2.80	*IDORL*
6D	C25	0.57	*MHCIDO*

Continued on next page

Table B.5: continued from previous page

Space	Stable Cluster	P_i	Properties
	C26	0.84	*MHIDOR*
	C27	0.28	*MHIOLA*
	C28	3.40	*HCIDOR*
	C29	6.51	*HIDORL*
	C30	1.26	*HIDOLA*

Results of the prediction on the training set of peptides of 10-16 aa length and 18-27 aa length is shown in Table B.6. Sensitivities values of all clusters of the prediction were equal to 0.19 and 0.85, and PPV were equal to 0.90 and 0.97 for peptides of 10-16 and 18-27 aa length, respectively. While the article has sensitivities values of all clusters of the prediction were equal to 0.84 and 0.79, and PPV were equal to 0.80 and 0.81 for peptides of 10-16 and 18-27 aa length, respectively. If we compare the quality of the prediction with the article, we can see that the sensitivities of all prediction clusters are lower for peptides with 10-16 aa lengths than the article but slightly higher for peptides with 18-27 aa lengths. However, the prediction's PPV was higher than the article for peptides with lengths of 10-16 and 18-27 aa.

Table B.6: Results of prediction on the training set of peptides of 10-16 aa length and 18-27 aa length.

Clusters	TP	$TP+FN$	FP	$TN+FP$	SN	PPV	ER
10-16 aa length							
Cluster 1	11	140	3	140			

Continued on next page

Table B.6: continued from previous page

Clusters	TP	$TP+FN$	FP	$TN+FP$	SN	PPV	ER
Cluster 2	15	140	0	140			
All clusters	26	140	3	140	0.19	0.90	0.42
18-27 aa length							
Cluster 1	85	100	3	100			
All clusters	85	100	3	100	0.85	0.97	0.09

B.2 Analysis on Large Datasets

Table B.7: Stable clusters of peptides 10-16 aa length by physicochemical properties forming the space and cluster size for large datasets.

		Physical-chemical properties									
Space	Stable Cluster	M	H	C	I	D	O	R	L	A	Cluster size
2D	C1	*	*								22
3D	C2	*	*				*				12
	C3	*		*		*					20
	C4	*				*	*				14
	C5		*	*		*					21
	C6		*	*			*				22
	C7		*	*				*			27
	C8		*	*						*	22

Continued on next page

Table B.7: continued from previous page

Space	Stable Cluster	Physical-chemical properties									Cluster size
		M	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	
	C9			*	*	*					15
	C10			*	*			*			17
	C11			*	*			*			16
	C12			*		*	*				21
4D	C13	*	*	*		*					18
	C14	*	*	*			*				17
	C15	*	*	*			*				17
	C16	*	*	*				*			18
	C17	*	*	*					*		21
	C18	*	*			*			*		15
	C19	*	*						*	*	15
	C20	*		*		*	*				23
	C21	*		*		*	*				15
	C22	*		*		*		*			16
	C23	*		*		*				*	20
	C24	*		*			*	*			19
	C25	*		*			*			*	15
	C26		*	*	*		*				15
	C27		*	*	*			*			16
	C28		*	*	*					*	22

Continued on next page

Table B.7: continued from previous page

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
	C29		*	*		*	*				21
	C30		*	*			*	*			23
	C31		*	*			*		*		15
	C32		*	*			*			*	18
	C33		*	*				*		*	14
	C34		*	*					*	*	21
	C35			*	*	*	*				15
	C36			*	*	*	*				21
	C37			*	*	*				*	11
	C38			*	*			*		*	17
	C39			*		*	*			*	23
	C40			*			*	*		*	21
5D	C41	*	*	*	*		*				17
	C42	*	*	*		*	*				15
	C43	*	*	*		*	*				16
	C44	*	*	*		*		*			18
	C45	*	*	*		*				*	18
	C46	*	*	*			*		*		15
	C47	*	*	*			*			*	17
	C48	*	*	*			*			*	14

Continued on next page

Table B.7: continued from previous page

Space	Stable Cluster	Physical-chemical properties									Cluster size
		M	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	
	C49	*	*	*				*		*	18
	C50	*	*			*	*			*	14
	C51	*	*			*			*	*	17
	C52	*	*				*		*	*	15
	C53	*		*	*	*	*				15
	C54	*		*		*	*	*			22
	C55	*		*		*	*	*			15
	C56	*		*		*	*		*		14
	C57	*		*		*	*			*	22
	C58	*		*		*		*		*	21
	C59	*			*	*	*		*		15
	C60		*	*	*	*				*	14
	C61		*	*	*		*			*	15
	C62		*	*		*	*	*			16
	C63		*	*		*	*			*	18
	C64		*	*		*			*	*	18
	C65		*	*			*	*		*	21
	C66		*	*			*		*	*	18
	C67		*			*	*	*	*		11
	C68			*	*	*	*	*			15

Continued on next page

Table B.7: continued from previous page

Space	Stable Cluster	Physical-chemical properties									Cluster size
		M	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	
	C69			*	*	*	*		*		15
	C70			*	*	*		*		*	11
6D	C71	*	*	*	*	*	*				15
	C72	*	*	*	*	*		*			14
	C73	*	*	*	*	*			*		19
	C74	*	*	*	*		*		*		14
	C75	*	*	*	*		*			*	14
	C76	*	*	*		*	*		*		18
	C77	*	*	*		*	*		*		21
	C78	*	*	*		*	*			*	14
	C79	*	*	*		*	*			*	16
	C80	*	*	*		*		*	*		27
	C81	*	*	*		*		*		*	18
	C82	*	*	*			*		*	*	14
	C83	*	*		*	*	*		*		18
	C84	*	*			*	*	*	*		17
	C85	*		*	*	*	*	*			15
	C86	*		*	*	*	*		*		14
	C87	*		*		*	*	*	*		14
	C88	*		*		*	*	*		*	21

Continued on next page

Table B.7: continued from previous page

Space	Stable Cluster	Physical-chemical properties									Cluster size
		M	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	
	C89	*			*	*	*	*	*		15
	C90		*	*		*	*	*		*	14
	C91		*			*	*	*	*	*	11
	C92			*	*	*	*	*	*		14
	C93			*		*	*	*	*	*	12
7D	C94	*	*	*	*	*	*	*			14
	C95	*	*	*	*	*	*		*		14
	C96	*	*	*	*	*		*	*		18
	C97	*	*	*	*	*		*		*	14
	C98	*	*	*	*	*			*	*	19
	C99	*	*	*	*		*		*	*	14
	C100	*	*	*		*	*	*	*		15
	C101	*	*	*		*	*		*	*	15
	C102	*	*	*		*	*		*	*	21
	C103	*	*	*		*		*	*	*	27
	C104	*	*	*			*	*	*	*	14
	C105	*	*		*	*	*	*	*		17
	C106	*	*		*	*	*		*	*	15
	C107	*	*			*	*	*	*	*	17
	C108	*		*	*	*	*	*	*		14

Continued on next page

Table B.7: continued from previous page

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
8D	C109	*	*	*	*	*	*	*	*		16
	C110	*	*	*	*	*	*	*		*	14
	C111	*	*	*	*	*		*	*	*	18
	C112	*	*	*		*	*	*	*	*	15

Table B.8: Stable clusters of peptides 18-27 aa length by physicochemical properties forming the space and cluster size for large datasets.

		Physical-chemical properties									
Space	Stable Cluster	*M*	*H*	*C*	*I*	*D*	*O*	*R*	*L*	*A*	Cluster size
2D	C1	*	*								14
	C2	*						*			15
3D	C3	*						*		*	15
6D	C4		*	*	*		*		*	*	26
	C5		*	*			*	*	*	*	22
7D	C6		*	*	*	*	*	*	*		12
	C7		*	*	*		*	*	*	*	14
8D	C8		*	*	*	*	*	*	*	*	12

While Vishnepolsky et al. (2018) found that clusters differed most strongly from each other by isoelectric point (I) and tilt angle (O). One set of stable clusters was found in longer peptides, the cluster is created in 6D space and includes the hydrophobicity

(H), net charge (C), isoelectric point (I), tilt angle (O), linear moment (L) and aggregation (A). Vishnepolsky et al. (2018) found that clusters differed most strongly from each other by only isoelectric point (I) for this case. This result is not replicated in the analysis of both peptides for the large datasets.

Table B.7 lists the 112 subspaces where stable clusters were found for the 10-16 aa peptides. Comparing physical-chemical properties comprising the spaces, we see that net charge (C) occurs in the majority of cases which is found in the space of 95 clusters, followed by hydrophobicity (H) is found in the space of 76 clusters. Hydrophobic moment (M), penetration depth (D) and tilt angle (O) are found in the space of 74 clusters. On the other hand, aggregation (A), propensity to disordering (R), linear moment (L) and isoelectric point (I) occur in less than half of the cases. The largest clusters contain 27 peptides and are in 3D, 6D and 7D space. Unlike Vishnepolsky et al. no big clusters were found.

For 18-27 aa length of peptides, we also got only one set of stable clusters in 6D space from eight overlapping stable clusters found within any set, see Table B.8, compared with four stable clusters found by Vishnepolsky et al. (2018) in 2D, 3D, 6D, 7D and 8D space. Also, the optimal CSC for a set was simply the stable cluster with maximal P_i within that set. If we compare physical-chemical properties in all clusters, we see that hydrophobicity (H) and propensity to disordering (R) occurs in the majority of cases which is found in the space of six clusters, followed by net charge (C), tilt angle (O), linear moment (L) and aggregation (A) are found in the space of five clusters. On the other hand, hydrophobic moment (M) and penetration depth (D) are found in less than half the cases. The largest clusters contain 26 peptides and are in 6D space. Like Vishnepolsky et al. (2018), no big clusters were found.

Evaluation on training sets for large datasets.

The training set was used by randomly selecting peptides 80% from the full datasets for peptides of 10-16 aa length consisting of 515 active peptides and 515 inactive peptides. Three stable clusters were found for shorter peptides and hence the optimal CSC for each set was the stable cluster with maximal P_i for AMP prediction. Results of P_i and properties of a stable cluster of peptides of 10-16 aa length for cluster 1, cluster 2 and cluster 3 are reported in Tables B.9, B.10, and B.11, respectively.

Table B.9, we see 78 stable clusters occur in cluster 1 with stable clusters C44 in 5D space and C81 in 6D space having the highest P_i values of roughly 0.61. The physical-chemical properties in stable cluster C44 are a subset of stable cluster C81, so we decided to select a set of physical-chemical properties in stable cluster C44 in 5D space for AMP prediction includes the hydrophobic moment (M), hydrophobicity (H), net charge (C), penetration depth (D) and propensity to disordering (R).

Table B.9: Results of P_i and properties of a stable cluster of peptides of 10-16 aa length for large datasets: cluster 1. The highest of P_i is highlighted in blue.

Space	Stable Cluster	P_i	Properties
2D	C1	0.38	*MH*
3D	C2	0.05	*MHO*
	C3	0.03	*MCD*
	C5	0.11	*HCD*
	C6	0.60	*HCO*
	C7	0.09	*HCR*

Continued on next page

Table B.9: continued from previous page

Space	Stable Cluster	P_i	Properties
	C8	0.58	*HCA*
	C9	0.15	*CID*
	C11	0.29	*CIR*
	C12	0.13	*CDO*
4D	C13	0.25	*MHCD*
	C14	0.31	*MHCO*
	C16	0.23	*MHCR*
	C17	0.22	*MHCL*
	C18	0.06	*MHDL*
	C19	0.12	*MHLA*
	C20	0.22	*MCDO*
	C22	0.08	*MCDR*
	C23	0.03	*MCDA*
	C24	0.03	*MCOR*
	C26	0.11	*HCIO*
	C27	0.36	*HCIR*
	C28	0.25	*HCIA*
	C29	0.48	*HCDO*
	C30	0.19	*HCOR*
	C31	0.12	*HCOL*
	C32	0.59	*HCOA*

Continued on next page

Table B.9: continued from previous page

Space	Stable Cluster	P_i	Properties
	C33	0.08	*HCRA*
	C34	0.33	*HCLA*
	C36	0.30	*CIDO*
	C37	0.13	*CIDA*
	C39	0.09	*CDOA*
	C40	0.08	*CORA*
5D	C43	0.11	*MHCDO*
	C44	0.61	*MHCDR*
	C45	0.25	*MHCDA*
	C46	0.12	*MHCOL*
	C47	0.31	*MHCOA*
	C49	0.23	*MHCRA*
	C51	0.34	*MHDLA*
	C52	0.15	*MHOLA*
	C54	0.19	*MCDOR*
	C57	0.18	*MCDOA*
	C58	0.38	*MCDRA*
	C60	0.12	*HCIDA*
	C61	0.11	*HCIOA*
	C62	0.41	*HCDOR*
	C63	0.49	*HCDOA*

Continued on next page

Table B.9: continued from previous page

Space	Stable Cluster	P_i	Properties
	C64	0.24	*HCDLA*
	C65	0.32	*HCORA*
	C66	0.24	*HCOLA*
	C67	0.06	*HDORL*
	C70	0.13	*CIDRA*
6D	C72	0.12	*MHCIDR*
	C73	0.25	*MHCIDL*
	C74	0.11	*MHCIOL*
	C77	0.19	*MHCDOL*
	C79	0.11	*MHCDOA*
	C80	0.38	*MHCDRL*
	C81	0.61	*MHCDRA*
	C84	0.07	*MHDORL*
	C88	0.15	*MCDORA*
	C90	0.16	*HCDORA*
	C91	0.06	*HDORLA*
	C93	0.27	*CDORLA*
	C94	0.08	*MHCIDOR*
	C96	0.19	*MHCIDRL*
	C97	0.12	*MHCIDRA*
	C98	0.25	*MHCIDLA*

Continued on next page

Table B.9: continued from previous page

Space	Stable Cluster	P_i	Properties
	C99	0.11	*MHCIOLA*
	C100	0.17	*MHCDORL*
	C102	0.19	*MHCDOLA*
	C103	0.38	*MHCDRLA*
	C104	0.03	*MHCORLA*
	C107	0.07	*MHDORLA*
	C110	0.08	*MHCIDORA*
	C111	0.19	*MHCIDRLA*
	C112	0.17	*MHCDORLA*

Table B.10, we see 31 stable clusters occur in cluster 2 with stable clusters C53 in 5D space having the highest P_i values of roughly 1.57. So we decided to select a set of physical-chemical properties in stable cluster C53 in 5D space for AMP prediction including the hydrophobic moment (M), net charge (C), isoelectric point (I), penetration depth (D) and tilt angle (O).

Table B.10: Results of P_i and properties of a stable cluster of peptides of 10-16 aa length for large datasets: cluster 2. The highest of P_i is highlighted in blue.

Space	Stable Cluster	P_i	Properties
3D	C4	0.03	*MDO*
4D	C15	0.10	*MHCO*

Continued on next page

Table B.10: continued from previous page

Space	Stable Cluster	P_i	Properties
	C21	0.99	*MCDO*
	C25	0.74	*MCOA*
	C35	0.99	*CIDO*
5D	C41	0.63	*MHCIO*
	C42	0.16	*MHCDO*
	C48	0.08	*MHCOA*
	C53	1.57	*MCIDO*
	C55	0.52	*MCDOR*
	C56	0.33	*MCDOL*
	C59	1.41	*MIDOL*
	C68	0.77	*CIDOR*
	C69	0.33	*CIDOL*
6D	C71	0.20	*MHCIDO*
	C75	0.11	*MHCIOA*
	C76	0.62	*MHCDOL*
	C78	0.08	*MHCDOA*
	C82	0.03	*MHCOLA*
	C83	0.10	*MHIDOL*
	C85	1.15	*MCIDOR*
	C86	1.01	*MCIDOL*
	C87	0.38	*MCDORL*

Continued on next page

Table B.10: continued from previous page

Space	Stable Cluster	P_i	Properties
	C89	0.83	*MIDORL*
	C92	0.27	*CIDORL*
7D	C95	0.16	*MHCIDOL*
	C101	0.48	*MHCDOLA*
	C105	0.16	*MHIDORL*
	C106	0.09	*MHIDOLA*
	C108	0.54	*MCIDORL*
8D	C109	0.03	*MHCIDORL*

Table B.11, we see three stable clusters occur in cluster 3 which stable clusters C10 in 3D space and C38 in 4D space having the highest P_i values of roughly 1.17. The physical-chemical properties in stable cluster C10 are a subset of stable cluster C38, so we decided to select a set of physical-chemical properties in stable cluster C10 in 3D space for AMP prediction including the net charge (C), isoelectric point (I) and propensity to disordering (R).

Table B.11: Results of P_i and properties of a stable cluster of peptides of 10-16 aa length for large datasets: cluster 3. The highest of P_i is highlighted in blue.

Space	Stable Cluster	P_i	Properties
3D	C10	1.17	*CIR*
4D	C38	1.17	*CIRA*

Continued on next page

Table B.11: continued from previous page

Space	Stable Cluster	P_i	Properties
5D	C50	0.05	*MHDOA*

For peptides of 18-27 aa length, we used a training set of 324 active and the same amount of inactive peptides by randomly selecting peptides 80% from the full datasets. Results of P_i and properties of a stable cluster of peptides of 18-27 aa length only one cluster is reported in Table B.12 and the optimal CSC for a set was the cluster with maximal P_i for AMP prediction. We see that cluster 1 has eight stable clusters, with stable clusters C4 in 6D space and C7 in 7D space having similar P_i values of roughly 0.24 and 0.25, respectively. The physical-chemical properties in stable cluster C4 are a subset of stable cluster C7. Also, the stable cluster C4 has the most peptides (26 peptides) and the peptides overlap with all other stable clusters, while the stable cluster C7 has a small number of peptides (14 peptides) and the peptides do not overlap with some stable clusters. So, we decided to select a set of physical-chemical properties in the stable cluster C4 in 6D space for AMP prediction, consisting of hydrophobicity (H), net charge (C), isoelectric point (I), tilt angle (O), linear moment (L) and aggregation (A).

Table B.12: Results of P_i and properties of a stable cluster of peptides of 18-27 aa length for large datasets: cluster 1. The highest of P_i is highlighted in blue.

Space	Stable Cluster	P_i	Properties
2D	C1	0.08	*MH*
	C2	0.14	*MR*

Continued on next page

Table B.12: continued from previous page

Space	Stable Cluster	P_i	Properties
3D	C3	0.14	*MRA*
6D	C4	0.24	*HCIOLA*
	C5	0.07	*HCORLA*
7D	C6	0.15	*HCIDORL*
	C7	0.25	*HCIORLA*
8D	C8	0.15	*HCIDORLA*

Results of the prediction on the training set of peptides of 10-16 aa length and 18-27 aa length for a large dataset is shown in Table B.13. Sensitivities values of all clusters of the prediction were equal to 0.10 and 0.08, and PPV were equal to 0.98 and 1.00 for peptides of 10-16 and 18-27 aa length, respectively. While the article has sensitivities values of all clusters of the prediction were equal to 0.84 and 0.79, and PPV were equal to 0.80 and 0.81 for peptides of 10-16 and 18-27 aa length, respectively. If we compare the quality of the prediction with the article, we can see that the sensitivities of all prediction clusters are lower than the article and PPV of the prediction higher than the article for both peptides.

Table B.13: Results of prediction on the training set of peptides of 10-16 aa length and 18-27 aa length for large datasets.

Clusters	TP	$TP+FN$	FP	$TN+FP$	SN	PPV	ER
10-16 aa length							
Cluster 1	19	515	1	515			
Cluster 2	16	515	0	515			
Cluster 3	17	515	0	515			
All clusters	52	515	1	515	0.10	0.98	0.45
18-27 aa length							
Cluster 1	26	324	0	324			
All clusters	26	324	0	324	0.08	1.00	0.46

Appendix C

Example of Calculating the Final Rank Score

C.1 Calculating the final ranking score

The misclassification test error rates for models fitted for each of the five methods evaluated for Scenario 1 are summarized in Table C.1. Here, the reciprocal rank is determined by calculating the final rank $r(k)$ of a model using the following equation 4.23. Firstly, ranked according to the error rate (ER) for each method shown in Table C.2. Then, calculating the sum of the inverse ranking score of all methods and computing the inverse value of the sum of the inverse ranking score of all methods to get the final ranking score shown in Table C.2 in the last two columns. Finally, the final scores for predictor variable combinations are determined. Apply the same algorithm to the remaining scenarios.

Table C.1: Scenario 1 misclassification test error rates for all fitted models.

Models	LR	ENET	SVM	RF	NN
X_1, X_2	0.088	0.042	0.041	0.022	0.033
X_1, X_3	0.182	0.158	0.187	0.172	0.171
X_1, X_4	0.175	0.170	0.176	0.158	0.163
X_2, X_3	0.200	0.194	0.205	0.208	0.185
X_2, X_4	0.203	0.200	0.204	0.205	0.193
X_3, X_4	0.507	0.506	0.511	0.488	0.483
X_1, X_2, X_3	0.087	0.041	0.043	0.026	0.028
X_1, X_2, X_4	0.093	0.039	0.042	0.025	0.023
X_1, X_3, X_4	0.184	0.158	0.203	0.157	0.181
X_2, X_3, X_4	0.200	0.198	0.221	0.204	0.226
X_1, X_2, X_3, X_4	0.095	0.041	0.051	0.022	0.032

Table C.2: Scenario 1 ranking score of the error rate (ER) for each method. The final score for each method is its inverse ranking score divided by the sum of all inverse ranking scores. The final scores for the top three predictor variable combinations are highlighted in blue.

Models	LR	ENET	SVM	RF	NN	$\sum_{m=1}^{5} \frac{1}{r_m(k)}$	$1/\sum_{m=1}^{5} \frac{1}{r_m(k)}$
X_1, X_2	2	4	1	1	4	3.0000	0.3333
X_1, X_3	6	5	6	7	6	0.8429	1.1864
X_1, X_4	5	7	5	6	5	0.9095	1.0995
X_2, X_3	8	8	9	10	8	0.5861	1.7062
X_2, X_4	10	10	8	9	9	0.5472	1.8274
X_3, X_4	11	11	11	11	11	0.4545	2.2000
X_1, X_2, X_3	1	2	3	4	2	2.5833	0.3871
X_1, X_2, X_4	3	1	2	3	1	3.1667	0.3158
X_1, X_3, X_4	7	5	7	5	7	0.8286	1.2069
X_2, X_3, X_4	8	9	10	8	10	0.5611	1.7822
X_1, X_2, X_3, X_4	4	2	4	1	3	2.3333	0.4286

Bibliography

Akbar, S., Hayat, M., Iqbal, M. & Jan, M. A. (2017). iACP-GAEnsC: evolutionary genetic algorithm based ensemble classification of anticancer peptides by utilizing hybrid feature space. *Artificial Intelligence in Medicine*, *79*, 62–70.

Akinchina, A. & Linse, P. (2007). Diblock polyampholytes grafted onto spherical particles: Effect of stiffness, charge density, and grafting density. *Langmuir*, *23(3)*, 1465–1472.

Allen, M. P. & Tildesley, D. J. (2017). *Computer Simulation of Liquids* (Second Ed.). Oxford: Clarendon Pr, UK.

Allred, L. & Kelly, G. (1990). Supervised learning techniques for backpropagation networks. In *1990 IJCNN International Joint Conference on Neural Networks* (pp. 721–728 vol.1).

Audain, E., Ramos, Y., Hermjakob, H., Flower, D. R. & Perez-Riverol, Y. (2015). Accurate estimation of isoelectric point of protein and peptide sequences: benchmarking and implications. *Bioinformatics*, *32(6)*, 821–827.

Babaie-Kafaki, S. (2012). A quadratic hybridization of polak-ribière-polyak and

fletcher-reeves conjugate gradient methods. *Journal of Optimization Theory and Applications*, *154*, 916–932.

Babaie-Kafaki, S. & Ghanbari, R. (2015). A hybridization of polak-ribière-polyak and fletcher-reeves conjugate gradient methods. *Numerical Algorithms*, *68*, 481–495.

Berger, J. O. & Bock, M. E. (1976). Combining independent normal mean estimation problems with unknown variances. *The Annals of Statistics*, *4*, 642–648.

Bhadra, P., Yan, J., Li, J., Fong, S. & Siu, S. W. I. (2018). AmPEP; sequence-based prediction of antimicrobial peptides using distribution patterns of amino acid properties and random forest. *Scientific Reports*, *8*, 1697.

Birant, D. & Kut, A. (2007). St-dbscan: An algorithm for clustering spatial-temporal data. *Data & Knowledge Engineering*, *60*, 208–221.

Boser, B. E., Guyon, I. M. & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on computational learning theory* (pp. 144–152).

Breiman, L. (1995). Better subset selection using the non-negative garotte. *Technometrics*, *37*, 373–384.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (1996b). Stacked regressions. *Machine Learning*, *24*, 46–94.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R. (1984). *Classification and Regression Trees*. Wadsworth Publishing Company, Belmont.

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L. & Lopez, A. (2020). A comprehensive survey on support vector machine classification: applications, challenges and trends. *Neurocomputing*, *408*, 189–215.

Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY, USA: ACM.

Cherkasov, A., Hilpert, K., Jenssen, H., Fjell, C. D., Waldbrook, M., Mullaly, S. C., Volkmer, R. & Hancock, R. E. (2008). Use of artificial intelligence in the design of small peptide antibiotics effective against a broad spectrum of highly antibiotic-resistant superbugs. *ACS Chemical Biology*, *4(1)*, 65–74.

Chervonenkis, A. Y. (2013). *Early History of Support Vector Machines* (pp. 13–20). Berlin, Heidelberg: Springer Berlin Heidelberg.

Chung, K. L. *A course in probability theory* (Second Ed.). Academic Press, Inc, Dan Diego.

Clark, S., Jowitt, T. A., Harris, L. K., Knight, C. G. & Dobson, C. B. (2021). The lexicon of antimicrobial peptides: a complete set of arginine and tryptophan sequences. *Communications Biology*, *4*, Article no. 605.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, *20(2)*, 215–232.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, *34*, 187–220.

Cramér, H. (1946). *Mathematical Methods of Statistics*. Princeton, NJ: Princeton University Press.

Diaz, G. I., Fokoue-Nkoutche, Nannicini, G. & Samulowitz, H. (2017). An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development*, *61(4/5)*, 9:1–9:11.

Dobrynin, A. V., Colby, R. H. & Rubinstein, M. (1995). Scaling theory of polyelectrolyte solutions. *Macromolecules*, *28(6)*, 1859–1871.

Dobrynin, A. V., Colby, R. H. & Rubinstein, M. (2004). Polyampholytes. *Journal of Polymer Science Part B: Polymer Physics*, *42(19)*, 3513–3538.

Dobrynin, A. V. & Rubinstein, M. (2001). Counterion condensation and phase separation in solutions of hydrophobic polyelectrolytes. *Macromolecules*, *34(6)*, 1964–1972.

Doi, M. & Edwards, S. F. (1988). *The theory of Polymer dynamics* (1 Ed.). Oxford: Claredon Press.

Edwards-Gayle, C. J. C., Barrett, G., Roy, S., Castelletto, V., Seitsonen, J., Ruokolainen, J. & Hamley, I. W. (2020). Selective antibacterial activity and lipid membrane interactions of arginine-rich amphiphilic peptides. *Applied Bio Materials*, *3*, 1165–1175.

Edwards-Gayle, C. J. C., Castelletto, V., W.Hamley, I., Barrett, G., Greco, F., Hermida-Merino, D., Rambo, R. P., Seitsonen, J. & Ruokolainen, J. (2019). Self-assembly, antimicrobial activity, and membrane interactions of arginine-capped peptide bola-amphiphiles. *Applied Bio Materials*, *2*, 2208–2218.

Effrosynidis, D. & Arampatzis, A. (2021). An evaluation of feature selection methods for environmental data. *Ecological Informatics*, *61*, 101224.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*, 1–26.

Efron, B. & Morris, C. (1973). Combining possibly related estimation problems. *Journal of the Royal Statistical Society Series B*, *35*, 379–421.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J. & Fayyad, U. M. (Eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (pp. 226–231). AAAI Press.

Fernandez-Escamilla, A.-M., Rousseau, F., Schymkowitz, J. & Serrano, L. (2004). Prediction of sequence-dependent and mutational effects on the aggregation of peptides and proteins. *Nat Biotechnol.*, *22*, 1302–1306.

Fisher, R. A. (1922). On the interpretation of χ^2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, *85(1)*, 87–94.

Fjell, C. D., Jenssen, H., Hilpert, K., Cheung, W. A., Pante, N., Hancock, R. E. & Cherkasov, A. (2009). Identification of novel antibacterial peptides by chemoinformatics and machine learning. *Journal of Medicinal Chemistry*, *52(7)*, 2006–2015.

Fletcher, R. & Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, *7*, 149–154.

Fox, L., Huskey, H. D. & Wilkinson, J. H. (1948). Notes on the solution of algebraic linear simultaneous equations. *The Quarterly Journal of Mechanics and Applied Mathematics*, *1*, 149–173.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, *38*, 367–378.

Friedman, J. H., Hastie, T. & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*, Issue 1.

Fritsch, S., Guenther, F. & Wright, M. N. (2019). *neuralnet: Training of Neural Networks*. R package version 1.44.2.

Gavva, V., Al Musaimi, O., Bent, C. & Williams, D. R. (2023). Determining the hydrophobicity index of protected amino acids and common protecting groups. *Separations*, *10(8)*.

Ge, R., Feng, G., Jing, X., Zhang, R., Wang, P. & Wu, Q. (2020). EnACP: An ensemble learning model for identification of anticancer peptides. *Frontiers in Genetics*, *11*, Article no. 760.

Gilbert, J. C. & Nocedal, J. (1992). Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization*, *2(1)*, 21–42.

Green, E. J. & Strawderman, W. E. (1991). A James-Stein type estimator for combining unbiased and possibly biased estimators. *Journal of the American Statistical Association*, *86(416)*, 1001–1006.

Hahsler, M., Piekenbrock, M. & Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, *91(1)*, 1–30.

Hamley, I. W. (2020). *Introduction to Peptide Science*. West Sussex: John Wiley & Sons.

Hasibuan, E. H., Hendraputra, S., GS, A. D. & Saragih, L. (2022). Comparison fletcher-reeves and polak-ribiere ann algorithm for forecasting analysis. *Journal of Physics: Conference Series* (p. 2394).

Hastie, T., Tibshirani, R. & Friedman, J. (2011). *The elements of statistical learning*. New York: Springer.

Hernandez Aros, L., Bustamante Molano, L. X., Gutierrez-Portela, F., Moreno Hernandez, J. J. & RodrÃguez Barrero, M. S. (2024). Financial fraud detection through the application of machine learning techniques: a literature review. *Humanities and Social Sciences Communications*, *11(1)*, 1130.

Hestenes, M. R. & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, *49*, 409–436.

Ho, T. K. (1995). Random decision forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition* (pp. 278–282). IEEE.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 832–844.

Hoerl, A. E. & Kennard, R. W. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, *12*, 55–67.

Hu, X., Mamitsuka, H. & Zhu, S. (2011). Ensemble approaches for improving hla class i-peptide binding prediction. *Journal of Immunological Methods*, *374*, 47–52.

Huang, Y., Talwar, A., Chatterjee, S. & Aparasu, R. R. (2021). Application of machine learning in predicting hospital readmissions: a scoping review of the literature. *BMC Medical Research Methodology*, *21(1)*, 96.

Imbert, J., Victor, J., Tsunekawa, N. & Hiwatari, Y. (1999). Conformational transitions of a diblock polyampholyte in 2 and 3 dimensions. *Physics Letters A*, *258(2)*, 92–98.

Jackson, N. E., Webb, M. A. & de Pablo, J. J. (2019). Recent advances in machine learning towards multiscale soft materials design. *Current Opinion in Chemical Engineering*, *23*, 106–114. Frontiers of Chemical Engineering: Molecular Modeling.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2023). *An Introduction to Statistical Learning* (Second Ed.). New York: Springer.

Jenssen, H., Fjell, C. D., Chrekasov, A. & Hancock, R. E. W. (2008). Qsar modelling and computer-aided design of antimicrobial peptides. *Journal of Peptide Science*, *14*, 110–114.

Jurafsky, D. & Martin, J. H. (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd Ed.). Online manuscript released January 12, 2025.

Kamkar, I., Gupta, S. K., Phung, D. & Venkatesh, S. (2016). Stabilizing ℓ_1-norm prediction models by supervised feature grouping. *Journal of Biomedical Informatics*, *59*, 149–168.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)* (pp. 3149–3157).

Khamis, A. M., Essack, M., Gao, X. & Bajic, V. B. (2015). Distinct profiling of antimicrobial peptide families. *Bioinformatics*, *31(6)*, 849–856.

Koh, D.-M., Papanikolaou, N., Bick, U., Illing, R., Kahn, C. E., Kalpathi-Cramer, J., Matos, C., Martí-Bonmatí, L., Miles, A., Mun, S. K., Napel, S., Rockall, A., Sala, E., Strickland, N. & Prior, F. (2022). Artificial intelligence and machine learning in cancer imaging. *Communications Medicine*, *2(1)*, 133.

Konar, J., Khandelwal, P. & Tripathi, R. (2020). Comparison of various learning rate scheduling techniques on convolutional neural network. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)* (pp. 1–5).

Kremer, K. & Grest, G. S. (1990). Dynamics of entangled linear polymer melts: A molecular-dynamics simulation. *Journal of Chemical Physics*, *92(8)*, 5057–5086.

Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software*, *28(5)*, 1–26.

Kursa, M. B. & Rudnicki, W. R. (2010). Feature selection with the boruta package. *Journal of Statistical Software*, *36(11)*, 1–13.

Lance, G. N. & Williams, W. T. (1967). A general theory of classificatory sorting strategies I. hierarchical systems, journal = The computer Journal, volume = 9, issue = 4, pages = 373-380, doi = 10.1093/comjnl/9.4.373.

Lata, S., Mishra, N. K. & Raghava, G. P. (2010). AntiBP2: improved version of antibacterial peptide prediction. *BMC Bioinformatics*, *11*, (Suppl 1):S19.

Lata, S., Sharma, B. & Raghava, G. P. (2007). Analysis and prediction of antibacterial peptides. *BMC Bioinformatics*, *8*, 263.

Lempel, A. & Ziv, J. (1976). On the complexity of finite sequences. *IEEE Transactions on Information Theory*, *22(1)*, 75–81.

Liang, X., Li, F., Chen, J., Li, J., Wu, H., Li, S., Song, J. & Liu, Q. (2021). Large-scale comparative review and assessment of computational methods for anti-cancer peptide identification. *Briefings in Bioinformatics*, *22*, Issue 4.

Liaw, A. & Wiener, M. (2002). Classification and regression by randomforest. *R News*, *2(3)*, 18–22.

Linse, P. (2007). Interaction between colloids with grafted diblock polyampholytes. *The Journal of Chemical Physics*, *126(11)*, 114903.

Lira, F., Perez, P. S., Baranauskas, J. A. & Nozawa, S. R. (2013). Prediction of antimicrobial activity of synthetic peptides by a decision tree model. *Journals Applied and Environmental Microbiology*, *79(10)*, 3156–3159.

Loh, W. (2014). Fifty years of classification and regression trees. *International Statistical Review*, *82*, 329–348.

Maccari, G., Luca, M. D., Nifosí, R., Cardarelli, F., Signore, G., Boccardi, C. & Bifone, A. (2013). Antimicrobial peptides design by evolutionary multiobjective optimization. *PLOS Computational Biology*, *9*, e1003212.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Le Cam, L. M. & Neyman, J. (Eds.), *Berkeley Symposium on Mathematical Statistics and Probability. Vol. 5.1* (pp. 281–297). University of California Press.

Marquis de Condorcet, J. (1784). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Royal Printing, Paris.

Mason, L., Baxter, J., Bartlett, P. & Frean, M. (1999). Boosting algorithms as gradient descent. In Solla, S. A., Leen, T. K. & M′uller, K. (Eds.), *Advances in Neural Information Processing Systems 12* (pp. 512–518). MIT Press.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133.

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, *12(2)*, 153–157.

Meher, P. K., Sahu, T. K., Saini, V. & Rao, A. R. (2017). Predicting antimicrobial peptides with improved accuracy by incorporating the compositional, physico-chemical and structural features into chou's general pseaac. *Scientific Reports*, *7*, 42362.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. & Leisch, F. (2024). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-16.

Mishra, S. K., Chakraborty, S. K., Samei, M. E. & Ram, B. (2021). A q-polak-ribière-polyak conjugate gradient algorithm for unconstrained optimization problems. *Journal of Inequalities and Applications*, *25*.

Moon, C. P. & Fleming, K. G. (2011). Side-chain hydrophobicity scale derived from transmembrane protein folding into lipid bilayers. *Proceedings of the National Academy of Sciences*, *108(25)*, 10174–10177.

Mooney, C., Haslam, N. J., Holton, T. A., Pollastri, G. & Shields, D. C. (2013). Peptidelocator: prediction of bioactive peptides in protein sequences. *Bioinformativs*, *29(9)*, 1120–1126.

Morgan, J. N. & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, *58*, 415–434.

Narayanan Nair, A. K., Jimenez, A. M. & Sun, S. (2017). Complexation behavior

of polyelectrolytes and polyampholytes. *The Journal of Physical Chemistry B*, *121(33)*, 7987–7998.

Narayanan Nair, A. K., Uyaver, S. & Sun, S. (2014). Conformational transitions of a weak polyampholyte. *The Journal of Chemical Physics*, *141(13)*.

Ng, X. Y., Rosdi, B. A. & Shahrudin, S. (2015). Prediction of antimicrobial peptides based on sequence alignment and support vector machine-pairwise algorithm utilizing lz-complexity. *BioMed Research International*, *2015*.

Nielsen, M., Lundegaard, C. & Lund, O. (2007). Prediction of MHC class ii binding affinity using SMM-align, a novel stabilization matrix alignment method. *BMC Bioinformatics*, *8*, 238.

Nocedal, J. & Wright, S. J. (2006). *Numerical Optimization* (Second Ed.). New York: Springer.

Noé, F., Tkatchenko, A., Müller, K.-R. & Clementi, C. (2020). Machine learning for molecular simulation. *Annual Review of Physical Chemistry*, *71*, 361–390.

Nwankpa, C. E., l. Ijomah, W., Gachagan, A. & Marshall, S. (2021). Activation functions: comparison of trends in practice and research for deep learning. In *2nd International Conference on Computational Sciences and Technology, Jamshoro, Pakistan* (pp. 124–133).

Ogutu, J. O., Schulz-Streeck, T. & Piepho, H.-P. (2012). Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. *BMC Proceedings*, *6*, 1–6.

Olivella, S. & Shiraito, Y. (2017). *poisbinom: A faster implementation of the Poisson-binomial distribution*. R package version 1.0.1.

Osuna, E. E., Freund, R. & Girosi, F. (1957). Support vector machines: training and applications. Technical Report A.I. Memo No. 1602; C.B.C.L. Paper No. 144, MIT Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Cambridge, Massachusetts.

Palariya, R. & Singh, S. P. (2024). Structural transitions of a semi-flexible polyampholyte. *The Journal of Chemical Physics*, *161(10)*, 104903.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2(11)*, 559–572.

Phuoc, T., Anh, P. T. K., Tam, P. H. & Nguyen, C. V. (2024). Applying machine learning algorithms to predict the stock price trend in the stock market - the case of vietnam. *Humanities and Social Sciences Communications*, *11(1)*, 393.

Polak, E. & Ribière, G. (1969). Note sur la convergence de méthodes de directions conjuguées. *Revue Française d'Automatique, Informatique, Recherche Opérationnelle*, *3(1)*, 35–43.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6(3)*, 21–45.

Polyak, B. T. (1969). The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, *9(4)*, 94–112.

Porto, W. F., Állan S. Pires & Franco, O. L. (2012). Cs-amppred: An updated svm model for antimicrobial activity prediction in cysteine-stabilized peptides. *PLOS ONE*, *7(12)*, e51444.

Powell, M. J. D. (1986). Convergence properties of algorithms for nonlinear optimization. *SIAM Review*, *28(4)*, 487–500.

R Core Team (2021). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.

Rao, C. R. (1945). Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, *37*, 81–89.

Rao, J. N. K. & Subrahmaniam, K. (1971). Combining independent estimators and estimation in linear regression with unequal variances. *Biometrics*, *27*, 971–990.

Rosenblatt, F. (1957). The perceptron: A perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Inc, Buffalo, New York.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65(6)*, 386–408.

Rubin, D. J. & Weisberg, S. (1975). The variance of a linear combination of independent estimators using estimated weights. *Biometrika*, *62(3)*, 708–709.

Rubinstein, M. & Colby, R. H. (2003). *Polymer Physics*. Oxford University Press Inc., New York.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Saeys, Y., Abeel, T. & Van de Peer, Y. (2008). Robust feature selection using ensemble feature selection techniques. In Daelemans, W., Goethals, B. & Morik, K. (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 313–325). Berlin, Heidelberg: Springer Berlin Heidelberg.

Senes, A., Chadi, D. C., Law, P. B., Walters, R. F. S., Nanda, V. & DeGrado, W. F. (2007). E_z, a depth-dependent potential for assessing the energies of insertion of amino acid side-chains into membranes: Derivation and applications to determining the orientation of transmembrane and interfacial helices. *Journal of Molecular Biology*, *366*, 436–448.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*, 379–423, 623–656.

Shusharina, N. P., Zhulina, E. B., Dobrynin, A. V. & Rubinstein, M. (2005). Scaling theory of diblock polyampholyte solutions. *Macromolecules*, *38(21)*, 8870–8881.

Sutton, R. S. & Barton, A. G. (2018). *Reinforcement Learning: An Introduction* (Second Ed.). Cambridge, MA: MIT Press.

Tay, J., Narasimhan, B. & Hastie, T. (2023). Elastic net regularization paths for all generalized linear models. *Journal of Statistical Software*, *106(1)*, 1–31.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B (Methodological)*, *58*, 267–288.

Ting, K. M. & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research*, *10*, 271–289.

Torrent, M., Andreu, D., Nogues, V. M. & Boix, E. (2011). Connecting peptide physicochemical and antimicrobial properties by a rational prediction model. *PLOS ONE*, *6*, e16968.

Touati-Ahmed, D. & Storey, C. (1990). Efficient hybrid conjugate gradient techniques. *Journal of Optimization Theory and Applications*, *64*, 379–397.

Tyagi, S. (2022). Analyzing machine learning models for credit scoring with explainable ai and optimizing investment decisions. *American International Journal of Business Management*, *5*, 5–19.

Ulrich, S., Seijo, M. & Stoll, S. (2007). A monte carlo study of weak polyampholytes: Stiffness and primary structure influences on titration curves and chain conformations. *The Journal of Physical Chemistry B*, *111(29)*, 8459–8467. PMID: 17411088.

Uversky, V. N., Gillespie, J. R. & Fink, A. L. (2000). Why are "natively unfolded" proteins unstructured under physiologic conditions? *Proteins: Structure, Function, and Genetics*, *41*, 415–427.

Vishnepolsky, B., Gabrielian, A., Rosenthal, A., Hurt, D. E., Tartakovsky, M., Managadze, G., Grigolava, M., Makhatadze, G. I. & Pirtskhalava, M. (2018). Predictive model of linear antimicrobial peptides active against gram-negative bacteria. *Journal of Chemical Information and Modelling*, *58*, 1141–1151.

Vishnepolsky, B. & Pirtskhalava, M. (2014). Prediction of linear cationic antimocrobial peptides based on characteristics responsible for their interaction with the membranes. *Journal of Chemical Information and Modelling*, *54*, 1512–1523.

Vishnepolsky, B., Zaalishvili, G., Karapetian, M., Nasrashvili, T., Kuljanishvili, N., Gabrielian, A., Rosenthal, A., Hurt, D. E., Tartakovsky, M., Grigolava, M. & Pirtskhalava, M. (2019). De novo design and in vitro testing of antimicrobial peptides against gram-negative bacteria. *Pharmaceuticals*, *12(82)*.

Wang, P., Hu, L., Liu, G., Jiang, N., Chen, X., Xu, J., Zheng, W., Li, L., Tan, M., Chen, Z., Song, H., Cai, Y.-D. & Chou, K.-C. (2011). Prediction of antimicrobial peptides based on sequence alignment and feature selection methods. *PLOS ONE*, *6*, e18476.

Wang, T., Zhang, C., Snoussi, H. & Zhang, G. (2020). Machine learning approaches for thermoelectric materials research. *Advanced Functional Materials*, *30(5)*, 1906041.

Wang, Z. & Rubinstein, M. (2006). Regimes of conformational transitions of a diblock polyampholyte. *Macromolecules*, *39*, 5897–5912.

Weerts, H. J. P., Mueller, A. C. & Vanschoren, J. (2020). Importance of tuning hyperparameters of machine learning algorithms.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*, 241–359.

Wu, P. & Martin, R. (2023). A comparison of learning rate selection methods in generalized bayesian inference. *Bayesian Analysis*, *18(1)*, 105–132.

Xiao, X., Wang, P., Lin, W.-Z., Jia, J.-H. & Chou, K.-C. (2013). iAMP-2L: A two-level multi-label classifier for identifying antimicrobial peptides and their functional types. *Analytical Biochemistry*, *436*, 168–177.

Xu, J., Li, F., Leier, A., Xiang, D., Shen, H.-H., Lago, T. T. M., Li, J., Yu, D.-J. & Song, J. (2021). Comprehensive assessment of machine learning-based methods for predicting antimicrobial peptides. *Briefings in Bioinformatics*, *22*, 1–22.

Yi, S., Wan-Ling, G. & Rui-Fang, H. (2022). On the grouping effect of the ℓ_{1-2} models. *Applied Mathematics-A Journal of Chinese Universities*, *37(3)*, 422–434.

Youmans, M., Spainhour, C. & Qiu, P. (2017). Long short-term memory recurrent neural networks for antibacterial peptide identification. *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (pp. 498–502).

Zasloff, M. (2002). Antimicrobial peptides of multicellular organisms. *Nature*, *415(6870)*, 389–395.

Zhang, K., Gong, X. & Jiang, Y. (2024). Machine learning in soft matter: from simulations to experiments. *Advanced Functional Materials*, *34*, 2315177.

Zhou, D.-X. (2013). On grouping effect of elastic net. *Statistics and Probability Letters*, *83*, 2108–2112.

Zhou, Z. & Hooker, G. (2020). Unbiased measurement of feature importance in tree-based methods. *ACM Transactions on Knowledge Discovery from Data*, *15(2)*, Article 26.

Zhou, Z.-H., Member, S., *IEEE* & Liu, X.-Y. (2006). Training cost-sensitive neural

networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, *18(1)*, 63–77.

Zhu, J. & Hastie, T. (2005). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, *14(1)*, 185–205.

Zou, H. & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B (Methodological)*, *67(2)*, 301–320.

Zou, H. & Zhang, H. H. (2009). On the adaptive elastic-net with a diverging number of parameters. *The Annals of Statistics*, *37(4)*, 1733–1751.