Identifying Phenotypes Based on TCR Repertoire Using Machine Learning Methods

Qin Qianqian

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 27.4.2020

Supervisor

Prof. Harri Lähdesmäki

Advisor

Msc Emmi Jokinen

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Qin Qianqian

Title Identifying Phenotypes Based on TCR Repertoire Using Machine Learning Methods

Degree programme Automation and electrical engineering

Major Control, Robotics and Autonomous Systems **Code of major** ELEC3025

Supervisor Prof. Harri Lähdesmäki

Advisor Msc Emmi Jokinen

Date 27.4.2020 **Number of pages** 53+2 **Language** English

Abstract

The adaptive immune system can prevent human beings being infected by pathogens. T cells, a kind of lymphocytes in the adaptive immunity, recognise antigens by T cell receptors (TCRs) and then generate cell-mediated immune responses. After primary immune responses, the adaptive immunity can generate corresponding immunological memory. TCRs are generated by a process of somatic gene rearrangement and therefore have high diversity. An individual's TCR repertoire can reveal his pathogen exposure history, which can assist in biological studies such as disease diagnosis.

This master thesis targets to make predictions about phenotype statuses based on high-throughput TCR sequencing data using machine learning approaches, to see how accurate the phenotype identification based on TCR repertoire can be. The raw TCR data is preprocessed in three different ways and then proceed the next steps separately. Several feature selection approaches are applied to obtain the most important TCRs. The machine learning algorithms including Beta-binomial model (baseline), Logistic regression, Random forest and a Boosting algorithm LightGBM are trained and evaluated.

Two datasets, Cytomegalovirus (CMV) and rheumatoid arthritis (RA), are explored. For the CMV dataset, Random forest performs best, even though only a little bit better than the baseline model. However, the classification results of the RA dataset are not so good whatever models used, and the best classifier is LightGBM. The results imply that the TCR data needs to be large enough to make powerful predictions. Using a sufficiently large dataset, the prediction ability of the baseline model is great, and there may exist certain algorithms such as Random forest outperform it.

Keywords Machine learning, Immunology, T cell receptor, Phenotype status prediction

Preface

I want to express my gratitude to my supervisor Prof. Harri Lähdesmäki for offering me this thesis topic and providing support and guidance throughout the thesis work. I thank my advisor Msc Emmi Jokinen for her patience, help and feedback during the process, as well as the other members in the group Computational system biology. Finally, I wish to give thanks to my parents for their support and encouragement.

Otaniemi, 27.4.2020

Qin Qianqian

Contents

Abbreviations and Acronyms

TCR	T cell receptor
DNA	deoxyribonucleic acid
CMV	cytomegalovirus
RA	rheumatoid arthritis
V	Variable genes of the T cell receptor
D	Diversity genes of the T cell receptor
J	Joining genes of the T cell receptor
MHC	major histocompatibility complex
CDR	complementarity determining region
CV	cross-validation
LOOCV	Leave-One-Out cross-validation
AUROC	Area Under the Receiver Operating Characteristic curve
MLE	maximum likelihood estimation
MAP	Maximum A Posteriori
RFE	recursive feature elimination
HC	healthy controls
MOM	method of moments
s.d.	standard deviation
LR	logistic regression
RF	random forest
LightGBM	Light Gradient Boosting Machine

1 Introduction

The immune system, which consists of the innate immune system and the adaptive immune system, can defend vertebrates from being infected by many kinds of pathogens like viruses and bacteria. The adaptive immune system is the second defence line, activated to fight against invaders if a pathogen still survives after the innate immune responses. The adaptive responses are pathogen-specific, which is different from the innate ones. In the adaptive immune system, first the antigen is recognised by antigen-specific receptors of lymphocytes, and then lymphocytes are stimulated to generate corresponding immune responses to the targeted pathogen. [1]

T cells are a kind of lymphocytes that produce T-cell-mediated immune responses. They may attack against antigen-presenting cells directly or assist the immune responses of other cells. Each T cell has a unique T cell receptor (TCR), a molecule on the exterior of the T cell, whose function is recognising antigens. The theoretical diversity of TCRs is 10^{15} in mice [2] and 10^{18} in humans [15], while a recent study estimated that there are $<10^8$ unique TCRs in around 10^{12} T cells in total in a person [3].

Even though the diverse population of TCR sequences of a person (TCR repertoire) is very different between individuals, a small proportion of TCRs is still shared by most people [4]. These TCR sequences produce public T cell responses, targeting the same antigens. If a person has never been exposed to such an antigen, the associated public TCRs will only be intermittently detected in his TCR repertoire [5]. On the other hand, the T cells which give antigen-specific responses expand clonally after being stimulated by the antigen and generate corresponding immunological memory, which increases the probability of observing those TCRs in the TCR repertoire. Because of such properties, the TCR repertoire can dynamically characterise pathogen exposure history [6]. Recently, rapid development in high-throughput DNA sequencing has boosted TCR repertoire studies.

Due to the high diversity of TCRs, in a dataset which contains TCR repertoires of multiple individuals, the population of unique TCRs could reach millions while the size of samples may be small. Manually analysing such massive amounts of data is difficult and inefficient, and possibly limited by the existing knowledge about TCRs. Therefore there is a strong need to develop computational methods to automate TCR repertoire analysis. Machine learning methods are powerful to handle big data and to discover underlying relationships of data. They have already been utilised in a few immunology-related studies [5, 7].

Machine learning is a science aiming to improve the performance of a task by experience, involving diverse fields such as mathematics, statistics and computer science. It has been used in solving various kinds of problems and shows a strong power. Now it is also more and more applied in biology-related studies. With the size of available DNA sequences is increasing exponentially[8], it is becoming increasingly

important that developing machine learning algorithms to handle some genomics problems automatically. An important related application is gene prediction, which estimates the protein coding regions in the DNA or the functional parts of genes. Salzberg [9] proposed classification trees to find out the DNA regions coding targeted genes, and Castelo and Guigo [10] developed a advanced Bayesian classifier for predicting splice site. In proteomics, Wang et al. [11] have developed deep convolutional neural networks to identify the secondary structures of proteins. Machine learning has also been widely used in the System biology field„ e.g. probabilistic graphical models have been implemented to model genetic networks [8].

1.1 Problem Statement

The TCR repertoire can help to better understand the adaptive immune system, e.g. it can reflect the previous immune responses. In the past, the studies on TCR repertoire were limited due to the constraint of immunosequencing technology. Now thanks to the advancement of TCR deep sequencing, the researches can go deeper. This thesis aims to explore machine learning approaches for identifying phenotypes through TCR repertoire, hoping to support future related studies.

We mainly explored two datasets: One contains a training set of 666 subjects with over 89 million unique TCRs and a test set with 120 subjects, based on which we aimed to identify the cytomegalovirus (CMV)-serostatus. Another one contains 85 subjects with over 1.4 million unique TCRs, which is studied to analyse the relationships between TCRs and rheumatoid arthritis (RA). Emerson et al. [5] had developed a beta-binomial model for the CMV dataset, which reduces the feature space to only two and achieves excellent prediction performance. Their work did not explore other possible factors affecting phenotype identification like relationships between TCRs and different contribution degrees of TCRs, and it is possible that there exist some other models which could perform better. We reimplemented their model to the two datasets and extended their study, applying new approaches to the datasets we used.

In this thesis, the raw TCR data is preprocessed in three ways, generating three data representations: TCR presence/absence data, TCR count data and TCR frequency data. Various feature selection methods are attempted to select the TCRs most associated with the phenotypes and to decrease the dimension of feature space. Also, feature extraction and feature construction are considered. After the data preparation stage, different machine learning models are built and trained, including the Beta-binomial model, Random forest, Logistic regression, and LightGBM. Grid search is the primary technique to optimise the model parameters. Based on the size and the availability of the dataset, 10-fold cross-validation (10-fold CV) or Leave-One-Out cross-validation (LOOCV) is chosen as the model selection technique, and evaluating on a testing set or LOOCV is selected as the evaluation method. The main evaluation metric used is Area Under the Receiver Operating Characteristic curve (AUROC).

1.2 Structure of the Thesis

In Chapter 2, the background information related to immunology is introduced. We will have a look at the mechanism of the adaptive immune system and the knowledge of TCRs. Chapter 3 elaborates on machine learning methods used in this thesis. We start by going through some general concepts, including what supervised learning is and several common problems, the techniques and the metrics used to evaluate model performance in this thesis and the general machine learning workflow. Then the chapter introduces the methods we used, including the machine learning algorithms, the feature selection methods and the model optimization methods. Chapter 4 presents the implementation of this thesis. We will describe how we preprocessed the raw data and analysed the two datasets we explored, and then the experimentation and the results are presented. Finally, Chapter 5 states the conclusions we made from our work, the contributions and the limitations of this thesis, and the possible improvements.

2 The Adaptive Immune System and TCRs

This chapter presents the background knowledge in the immunology field. First, we will explain the mechanism of the adaptive immune system and the lymphocytes that play important roles in it. Secondly, we will give a detailed introduction about the TCR, including its structure, diversity and the process of generation.

2.1 The Adaptive Immune System

The immune system defends human beings from being infected by pathogens like viruses, preventing diseases or even death. It is made up of two subsystems: the innate immune system and the adaptive immune system, which exhibit their effects in different immune stages and handle potentially harmful agents together. The physical barriers such as the skin, and the chemical barriers such as mucus secretions, are the components of the innate immune system, which may keep pathogens from invading the body. Once pathogens overcame those barriers and invaded the body, the innate immune system will quickly generate innate immune responses in a generic way, which is not antigen-specific. If the pathogens break the defence line of the innate immune system, then the adaptive immune system will come to work, stimulated by innate immune responses and the pathogens, activating adaptive immune responses to eliminate the invaders. [1, 12]

The adaptive immune system has its characteristics that differ from the innate immune system. It produces highly antigen-specific but not immediate responses. It can generate long-term immunological memory which provides long-time protection against those pathogens that once have attacked the body. The first time the body is exposed to a pathogen, a primary immune response is activated: the antigen stimulates naïve cells to proliferate and differentiate, producing effector cells and memory cells targeting the same antigens. Effector cells generate immune responses, while memory cells provide immunological memory. If the pathogen invades the body again later, the memory cells will reproduce effector cells and memory cells, generating a faster, stronger and more efficient secondary immune response. [1, 12]

Two kinds of lymphocytes, B cells and T cells, play important roles in the adaptive immune system. B cells mature in the bone marrow and activate antibody responses. They produce antibodies, also called immunoglobulins. Antibodies recognise specific antigens and bind with them, generating the antigen-antibody complexes. By such bindings, antibodies can prevent infectious agents such as viruses from infecting host cells. The generated complexes can be destroyed or deactivated, and the pathogens will be eliminated by the innate immune system more easily. [1, 12]

T cells mature in the thymus. It activate another kind of adaptive immune response called T-cell-mediated immune responses. They can be categorized as three types: cytotoxic T cells, helper T cells and regulatory T cells. On the surface of the cells presenting antigens, T cells recognise antigen peptides which bind with major

histocompatibility complex (MHC) proteins, and then effector cells are generated by proliferating and differentiating the T cells. Effector cytotoxic T cells destroy those infected cells carrying the same pathogens; effector helper T cells assist in stimulating other cells to produce responses; effector regulatory T cells suppress the action of some other cells like self-reactive effector T cells. [1, 12]

MHC proteins are categorized as two kinds: class I MHC proteins are expressed in all cells, while class II proteins only exist in those cells presenting antigens to helper T cells. Two types of co-receptors in the T cells recognise different types of MHC: CD8 expressed in cytotoxic T cells recognises class I MHC proteins, and CD4 expressed on other two classes of T cells recognises class II proteins. It helps T cells take actions to their target cells correctly: cytotoxic T cells recognise antigens bound to class I MHC proteins and then destroy or eliminate any target cells, while helper T cells recognise antigens combined with class II MHC proteins and then handle some specific cells like dendritic cells and B cells. [1, 12]

2.2 T Cell Receptors (TCRs)

T cells recognise antigens via the bindings between their TCRs and the peptide-MHC (pMHC). A TCR is a heterodimer which is composed of two protein chains connecting by a disulfide bond. In human most of the TCRs (90-95%) have one α chain and one β chain, while the minority (5-10%) have one δ chain and one γ chain. Each chain contains one Variable domain (V_α or V_β) and one Constant domain (C_α or C_β), and these two regions are extracellular. [1]. The Constant region anchors the receptor to the T cell membrane by a transmembrane domain (T_m) and a short cytoplasmic tail (CT) [13], while the Variable region interacts with the pMHC to realize antigen recognition. Figure 2.1 shows the structure of $\alpha\beta$ TCR.

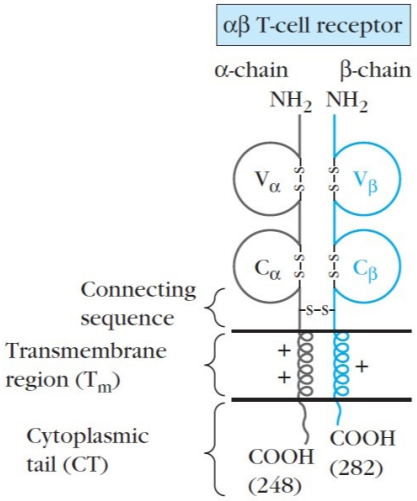


Figure 2.1: Structure of $\alpha\beta$ TCR [12]

The TCR loci consists of variable (V), joining (J), possibly diversity (D), and constant (C) gene fragments through randomly gene rearrangement called V(D)J recombination which generates TCR diversity. The generations of TCRβ chain and TCRδ chain undergo VDJ recombination. First a D gene and a J gene are selected randomly and joined together, forming a D-J rearrangement. Then a V gene is chosen, and it combines with the D-J gene, forming a VDJ recombination. On the other hand, TCRα chain and TCRγ chain only involve V-J rearrangement. [12, 14] The nucleotide insertions and deletions during the process of V(D)J junctions add more diversity of TCRs [15]. So far it is found that there are 42 V genes and 61 J genes in the α locus, and in the β locus there are 47 V, 2 D, and 13 J genes [3]. V(D)J recombination is showed as Figure 2.2.

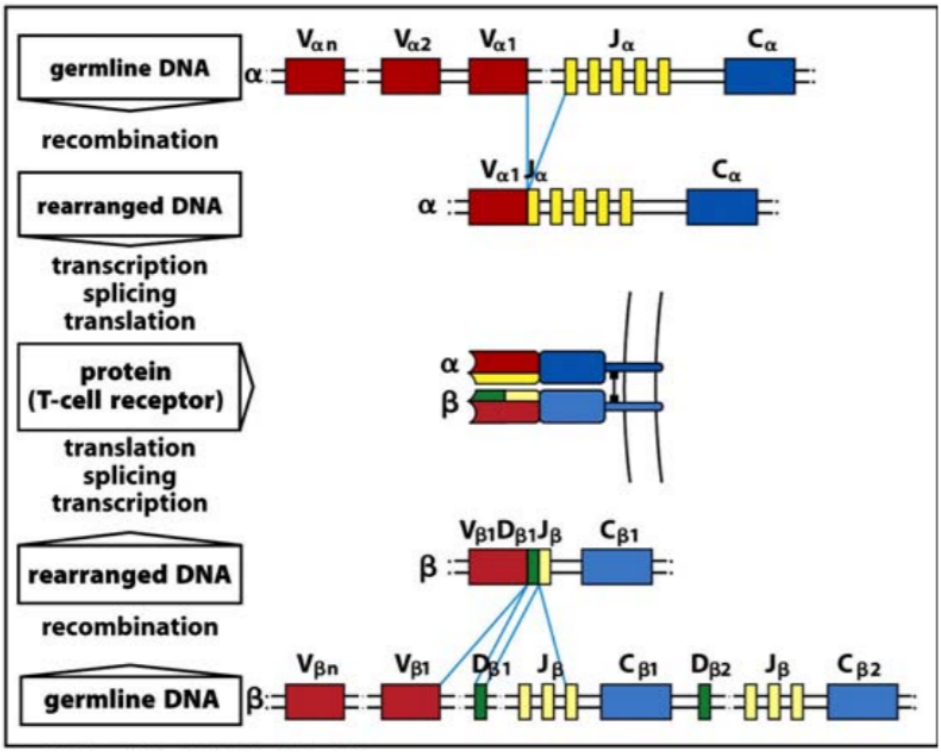


Figure 2.2: V(D)J Recombination [16]

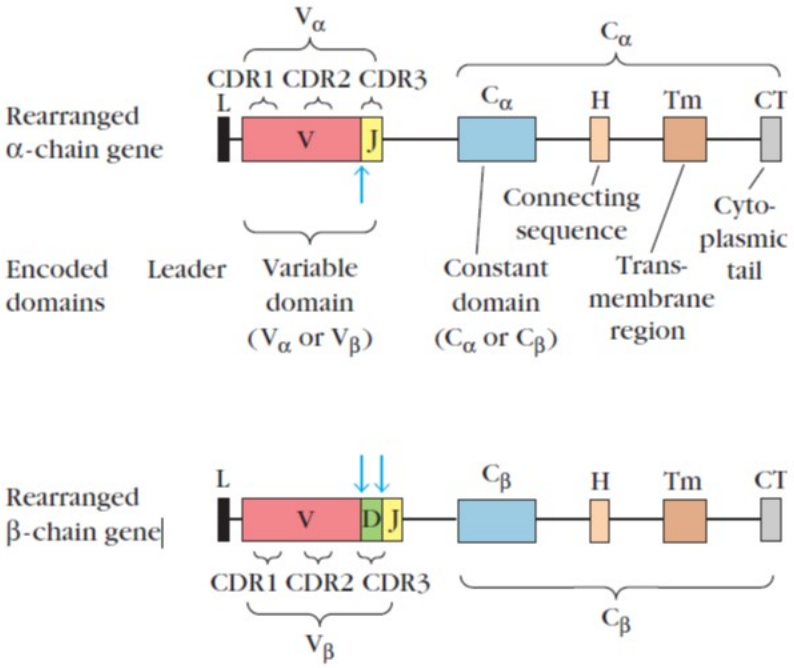


Figure 2.3: Gene decomposition of $\alpha\beta$ TCR [12]

Figure 2.3 presents the gene composition of $\alpha\beta$ TCR. The complementarity determining regions (CDRs) are three hypervariable regions within the Variable region. The CDR1 and the CDR2 are generated by the V gene segment encoding without any nucleotide additions or deletions, and they mainly interact with the peptide binding groove of the MHC. The CDR3 is generated through the junctions between the rearranged V (D) J genes segments along with the addition and deletion of nucleotides, and it interacts with the antigen peptide part. [12, 3, 13] The CDR3 is the most diverse, and it is the key region to determine the specificity of TCR [17].

3 Machine Learning

This chapter talks about the machine learning methods applied in our work. Section 3.1 first introduces a few basic concepts of machine learning and secondly goes deeper into supervised learning. It presents common problems in supervised learning, as well as the techniques and the metrics used to conduct performance evaluation in this thesis. A typical process of solving a machine learning problem is also described in this section. Section 3.2 describes the machine learning models we applied that include the Beta-binomial model, logistic regression, Random forest and Boosting algorithms. Feature selection methods, consisting of filter methods, wrapper methods and embedded methods, are described in Section 3.3, and model tuning approaches are presented in 3.4.

3.1 General Concepts

In this age of "big data", the increasing volume and complexity of data make it almost impossible to process and analyse data manually. For example, the data used in our work contains millions of TCRβ chains, and therefore to find out the most phenotype-related TCRβ chains and to make phenotype predictions by hand is difficult, time-consuming and probably low-quality. Machine learning is powerful in helping people learn from data more automatically. It is the subject that teaches a machine to perform a specific task based on the experience [18]. The tasks include classifications like identifying whether an email is a spam, regressions such as predicting future housing price, explorations of internal properties of an object like applying clustering algorithms to grouping customers in business, and so on. Machine learning is an interdisciplinary field which integrates various subjects mainly including statistics, mathematics and computer science. Some machine learning methods are inspired by other sciences such as physics, neuroscience and biology [19]. From the view of mathematics, machine learning is to find a function that maps a feature set X (input) to a target variable Y (output): $X \to Y$.

In machine learning, each record in a data set is a **sample**. The data set used as the input is the **training set**, and within it, each sample is a **training sample**. A **feature** is an input item that can reflect a kind of property of the object used to solve the problem. Generally, each sample will have multiple features, and the number of features in a sample is called **dimensionality**. The **label** is the target to predict, that is the output Y. The process of fitting a model to data is regarded as **training**. The data set used for testing the trained model is known as the **testing set**. **Generalization** refers to the ability that the model applies to new unseen samples. [19, 20]

Machine learning is generally categorised as supervised learning, unsupervised learning and reinforcement learning. In supervised learning problems, the training samples contain the inputs (features) and the outputs, and the goal is to get a model based on the training feature-label pairs so that the model can predict outputs associated

with new inputs. Different from supervised learning, unsupervised learning deals with unlabeled training samples and aims at finding out the internal structure or the relationships in the data and then analysing new data without the assistance of the labels. Reinforcement learning maps states to actions, and learners need to try different actions by themselves for discovering the best actions which maximise the total numerical reward. [19] We aim to address a supervised learning problem so will focus on the supervised learning part.

3.1.1 Supervised Learning

As mentioned above, the supervised learning problem is to learn from labelled training samples. The data is represented as $(\boldsymbol{x_i}, \boldsymbol{y_i})$, where $\boldsymbol{x_i}$ stands for the feature vector of the i-th sample and $\boldsymbol{y_i}$ is its target. A supervised learning task aims to learn a function approximation h, which closely matches the true function f that maps $\boldsymbol{x}$ to $\boldsymbol{y}$: $\boldsymbol{y} = f(\boldsymbol{x})$ [21]. A function h is a **hypothesis** and all the possible hypotheses compose the **hypothesis space** $\mathcal{H}$. The task is to search for the hypothesis h^* which has the highest possibility given the data D in the hypothesis space: $h^* = \text{argmax}_{h \in \mathcal{H}} P(h \mid D)$. [18, 21]

Supervised learning problems consist of regression and classification. Regression is to predict a continuous value, while classification refers to predicting a category which is a discrete value. People have developed various supervised learning algorithms. Each algorithm has its advantages and drawbacks, and no algorithm can perform best on all the machine learning problems.

In this thesis, we study a binary classification problem using high-dimensional data. In such a case, some issues should be carefully taken into account:

Curse of dimensionality: The increase in the dimensionality of data leads the volume of the feature space to increase and the available training data to become sparse. Since the possible combinations of feature values on the whole space increases exponentially, the quantity of training data needs to increase exponentially in order to get a stable and reliable statistical model. Assuming that we train a model using a training set with a fixed size, the model performance first increases with the number of features increasing until reaching a peaking point, and then it will decrease as the dimension grows [22].

Underfitting and Overfitting: Underfitting is the problem that the insufficiency of model complexity leads to the model can not learn the internal relationships of data. Models that are not complex enough have poor performance on training data and can not generalise to testing data. Underfitting is easy to detect and handle, so it is seldom considered. Conversely, overfitting refers to the over-complex classifier which can fit training data very well but has poor generalization ability on unseen data. Such a classifier learns not only the underlying structure of data but also noise and exceptions in it. Overfitting is a common issue in the implementation of machine

learning. [19, 20]

Bias-Variance Trade-off: Assuming that the true function is $f(x)$ and the fitted hypothesis is $\hat{f}(x)$, the bias is the difference between the expectation of $\hat{f}(x)$ and $f(x)$, written as $bias = E[\hat{f}(x)] - f(x)$, and the variance, denoted as $Var[\hat{f}(x)] = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$, describes how sensitive the model output is to variations of the training data. A high bias means that the predictions far deviate from the true values and the model underfits the data. while if a classifier has a high variance, it is sensitive to fluctuations in the training data and tends to overfitting. With the complexity of a model increasing, the bias will decrease while the variance will increase. We need to find a balance between them without overfitting and underfitting the data.

Some means such as cross-validation can assess how serious the above potential problems are by estimating the generalisation ability of models. Many methods have been developed to overcome or depress those problems, such as feature selection approaches which will be introduced in Section 3.3.

Performance Evaluation

It should be noticed that a classifier performs well on the training set does not indicate that it has good generalisation ability. To find out the best hypothesis, we need some means to assess the performance of supervised learning models.

Hold-out method is the simplest way to estimate model performance. The data set is divided into two subsets. One subset acts as the training set which machine learning models learn from, and another one used as the testing set to make model evaluation. A rule of thumb is that 80% of the samples is used for training while the other 20% is used for testing. The hold-out method only needs to run once, so it has a low computational cost. However, the evaluation is produced from a single one train-test division while different train-test splits are highly possible to produce different evaluation results. Therefore the evaluation performance may have a high variance, especially when the data set is small. [20]

The hold-out method is suitable for the case where the size of data is large. If the dataset size is small, **cross-validation** is a preferred method to evaluate how well a model performs on unseen data. In the beginning, the training set is partitioned into k subsets equally. In each round, one subset is left out and used for model validation. The model is trained on the other k-1 sets, and then the fitted model is evaluated using the validation set. Such a process is iterated k rounds, and finally, the k validation results are averaged to form a prediction of model performance. Generally, the above process is called k-fold cross-validation (k-fold CV). Leave-one-out cross-validation (LOOCV) is a special case where each round just one sample is selected as the validation set and the number of iteration rounds is the same as the size of training set. [20] Compared with the hold-out strategy, cross-validation can provide more reliable evaluation since it uses multiple train-test splits and all the samples

are utilised in training and testing. On the other hand, it is more time-consuming to compute since it needs to run multiple times. Cross-validation is the most widely used technique to assess the severity of the overfitting problem and to help model selection.

There are many metrics used to measure model performance, and the metrics are chosen based on task requirements. When comparing different models, different performance metrics often lead to different evaluation results, which means that the goodness of models is relative and which model to choose depends on not only the algorithm and data but also the task requirement. This thesis targets a classification problem, and therefore we will focus on some metrics on classification.

Confusion matrix is a table presenting the count of correctly predicted samples and incorrectly predicted samples, which can show the quality of a model on a dataset where the true labels are known. A confusion matrix of a binary classifier is shown in Figure 3.1. In a binary outcome case, the samples can be classified as positive or negative. True positives (denoted as a) is the number of correctly classified positive samples, and false negatives (denoted as b) is the number of wrongly classified positive samples. False positives (denoted as c) is the number of wrongly classified negative samples, and true negatives (denoted as d) is the number of correctly classified negative samples. [19, 20]

Actual	Predicted	
	positive	negative
positive	a	b
negative	c	d

Figure 3.1: Confusion matrix of a binary classification

The following metrics can be derived from the confusion matrix of a binary classifier, and they can assess a classifier in different views [19, 20]:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d}, \tag{3.1}$$

$$\text{Sensitivity} = \frac{a}{a+b}, \tag{3.2}$$

$$\text{Specificity} = \frac{d}{c+d}, \tag{3.3}$$

$$\text{Precision} = \frac{a}{a+c}, \tag{3.4}$$

$$\text{F1 score} = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} = \frac{2a}{2a+c+b}. \tag{3.5}$$

Equation 3.1 calculates the ratio of the correctly classified samples. It should be noticed that a high accuracy does not mean good performance of the model when the data set is imbalanced. Equation 3.2 is also known as Recall or True Positive Rate (TPR). TPR presents the percentage of positive samples predicted correctly out of all the positive samples. Equation 3.3, also called as True Negative Rate (TNR), indicates the ratio of correctly predicted negative samples among all the negative samples. Equation 3.4 represents how often a sample is truly positive when it is predicted as positive. Sensitivity and Precision are inversely related to some degree, that is, increasing one will be highly possible to cause decreasing another one. Equation 3.5 is the harmonic mean of these two measures, looking for a balance between them.

Area Under the Receiver Operating Characteristic curve (AUROC) is a common measurement for a binary classification task. ROC curve is a plot of the TPR on the y-axis versus the FPR, defined as $1 - \text{Specificify} = \frac{FP}{TN+FP}$, on the x-axis for various classification thresholds, which gives a visual summary of classification performance and describes relative trade-offs between true positives and false positives. The point (0,1) ((0%,100%)) indicates perfect classification where no incorrect predictions are made, and the diagonal line $y = x$ represents the curve of a random guess. AUROC is the area under ROC, and it can show that when we randomly select a pair of samples from the data, how likely that the pair is correctly ranked (the positive sample is ranked higher than the negative one). A higher AUROC means a better model. [19, 23] The AUROC of an ideal classifier is 1, while that of a random guess is approximately 0.5 [24]. Figure 3.2 shows three different ROC curves.

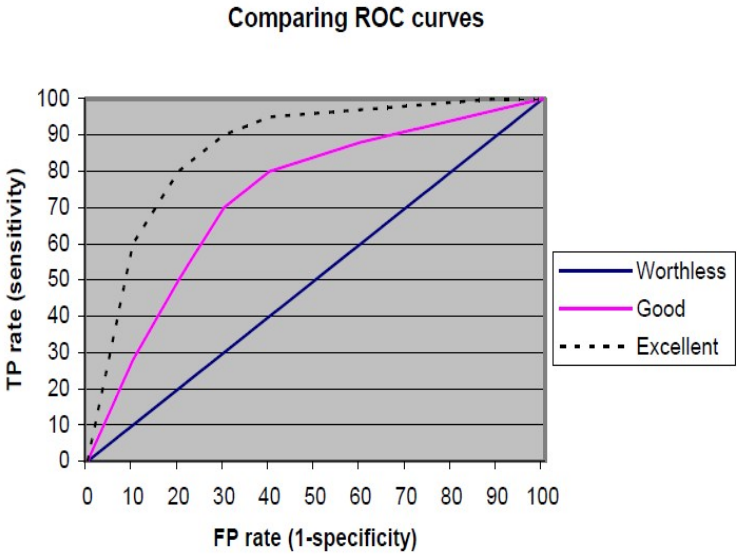


Figure 3.2: ROC curves [25]

3.1.2 Machine Learning Workflow

A common machine learning workflow includes the following stages.

- **Problem Definition and Data Collection**
  First of all, we should define what problem needs to be solved and identify what data should be gathered. The quality and quantity of data determine possible model performances. In some cases there exists pre-collected data or even pre-processed data available, while in the other cases the data needs to be gathered from scratch or assembled from different sources. In this thesis, the raw data can be obtained from a website, so we do not need to collect it from the beginning. [19]

- **Data Preparation**
  The data we collect in the first step is raw and unstructured, and it needs to be preprocessed before being input to models. It is possible that a particular order of data influences prediction results, so the order of data is randomised in such a case. The data cleaning includes error correction, removing duplicate data and irrelevant data, handling missing values, data transformation like normalisation and data type conversion, but not all of these are always needed. [26] Data exploration analysis (DEA) can be conducted at the same time. By DEA, a better understanding of data can be obtained, and the relationships between variables can be learned.

- **Dimension Reduction and Feature Engineering**
  Too many features may cause overfitting and also increase computation cost. Moreover, some features may be unrelated to the target and disturb the prediction. Therefore dimension reduction is an effective step to improve models in many machine learning problems. Dimension reduction includes two parts: feature selection and feature extraction. The purpose of feature selection is to pick the most important features within the complete feature set. Filter methods, Wrapper methods and Embedded methods are three general classes of feature selection methods. Feature extraction involves transforming data into informative features. Principal component analysis (PCA) is one of the most common approaches of extracting features. The details about dimension reduction will be introduced in Section 3.3. Feature engineering, which is the process of constructing new features using domain knowledge, may help to improve the model performance in some cases. [19, 26, 27]

- **Algorithms Choice (Model Training & Evaluation)**
  Many algorithms have been developed over the years. Some of them show good performance on image data; some algorithms are suitable for sequences like text, while others are good at handling numerical data. For different problems and different data set, the suitable algorithms are different. We can train a group of commonly used algorithms, performing cross-validation on the training data or testing on a holdout set, to determine which algorithms to use. One or

several algorithms which have relatively good performance can be selected for the further steps. [19, 26]

- **Parameter Tuning and Model Selection**
  After deciding which algorithm(s) to use, we will further try to improve their prediction ability. Parameter tuning is an approach to realise that. An algorithm often has a set of model parameters can be adjusted, such as a regularisation term for a regularised logistic regression and a learning rate for a neural network, and adjustable parameters may be different for different models. We can try a collection of different parameter values and choose the one which gives out the best evaluation result. Grid search is a common parameter tuning technique and will be presented in Section 3.4. After tuning the parameters of each model, we can choose the best model or combine multiple models using ensemble methods. [19, 26]

- **Deployment and Prediction**
  Finally, the optimised model is deployed to perform prediction on new unseen data. Generally, its performance in real-world applications will be evaluated by a test set. [19, 26]

It should be noticed that a few steps are interactive and several steps may be repeated many times. For example, selecting features using embedded methods is done during the process of training models; the data can be re-preprocessed and then the whole workflow can be repeated many times; the steps from Dimension Reduction and Feature Engineering to Parameter Tuning and Model Selection can be iterated until an acceptable model is worked out.

3.2 Algorithms

In this section, we will explain the classification algorithms chosen for our problem. They are Beta-binomial model (Section 3.2.1), Logistic regression (Section 3.2.2, Random forest (Section 3.2.3) and Boosting algorithms (Section 3.2.4). They have their own characteristics, and the mathematical theories behind them are different.

3.2.1 Beta-Binomial Model

The Beta-binomial model is the baseline model in our work. We will first explain Bayesian inference, the fundamental theory behind the Beta-binomial model. Bayesian inference is a widely used statistical method which uses Bayes' theorem to update a probability distribution on the parameters and estimate a probability point on unseen data conditioned on the observed data [28].

Bayes' theorem can be expressed as:

$$P(B \mid A) = \frac{P(B)P(A \mid B)}{P(A)}, \tag{3.6}$$

where $P(A)$ denotes the probability of the event A, $P(B)$ denotes the probability of the event Y, $P(A \mid B)$ is the conditional probability of A given that B has occurred and $P(B \mid A)$ is the conditional probability of B given that A has occurred. By using Bayes' theorem, we can infer the conditional probability based on the prior knowledge of conditions related to it. Bayesian inference is an application of Bayes' theorem.

When performing Bayesian inference, we first initialize a prior probability distribution of the unknown parameter θ, which is a reasonable guess and noted as $p(\theta)$ [33]. Then the posterior probability can be yielded based on the available data using Bayes' Rule [28]:

Often, the probability of θ is called the prior, and the probability of *data* conditioned on θ is called the likelihood, written as $p(data \mid \theta)$. The joint probability is the product of the prior and the likelihood:

$$p(\theta, data) = p(\theta)p(data \mid \theta). \tag{3.7}$$

Based on the Bayes' Rule, the posterior probability can be obtained:

$$p(\theta \mid data) = \frac{p(\theta)p(data \mid \theta)}{p(data)}, \tag{3.8}$$

where $p(data) = \Sigma_\theta p(\theta)p(data \mid \theta)$. Since $p(data)$ is independent from θ and can be regarded as a constant when *data* is fixed, we can get the unnormalized posterior density:

$$p(\theta \mid data) \propto p(\theta)p(data \mid \theta). \tag{3.9}$$

Finally, the prediction can be done via some estimation methods such as Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP) estimation.

MLE selects the value of the parameter θ which gives the maximum likelihood function [34]:

$$\hat{\theta}_{ML} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(data \mid \theta). \tag{3.10}$$

And in many cases, it is more convenient to maximize the natural logarithm of the likelihood (log-likelihood) [35]. MAP estimation finds the values of θ which maximize the posteriror probability [36]:

$$\begin{aligned} \hat{\theta}_{MAP} &= \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(\theta \mid data) \\ &= \underset{\theta \in \Theta}{\operatorname{argmax}}\, \frac{p(\theta)p(data \mid \theta)}{p(data)} \\ &= \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(\theta)p(data \mid \theta). \end{aligned} \tag{3.11}$$

The difference between MLE and MAP is that MAP incorporates the prior knowledge while MLE does not. MAP can be regarded as regularisation of MLE, weighing

the likelihood based on the prior. When the prior is a constant, the prior could be ignored in the MAP target objective function, and thus the MAP is equal to MLE in this situation.

Let us talk about how Bayes' theorem can be used in classification problems. Given an input data x, we aim to classify x as one of C classes. Supposing that c_i is the class i where $p(c_i)$ is the prior class probability of class i, $i \in \{1, 2, ..., C\}$, and $p(x \mid c_i)$ is the likelihood probability which is the probability of x given that x belongs to class i, the posterior probability $p(c_i \mid x)$, the probability of x belonging to class i given x, can be obtained using Bayes' theorem [37, 38],

$$p(c_i \mid x) = \frac{p(c_i)p(x \mid c_i)}{p(x)}, \tag{3.12}$$

where $p(x) = \sum_{i=1}^{C} P(c_i)P(x \mid c_i)$. A decision rule is to classify x as the class with the highest posterior probability. The classifier using the rule is called Bayes classifier. [37]

The error of the Bayes classifier is known as Bayes error, and it can be given as [37, 38]:

$$p_{Bayes}(error) = 1 - \sum_{i=1}^{C} \int_{R_{c_i}} p(c_i)p(x \mid c_i)\mathrm{d}x, \tag{3.13}$$

where R_{c_i} is the region where class i has the highest posterior probability. The equation shows that the Bayes classifier will minimize Bayes error. However, in most cases, it is impossible to obtain the Bayes error by the equation because of the difficulty in calculating the multi-dimensional integral, and thus the relevant studies focus on its approximation and its bounds estimation [37].

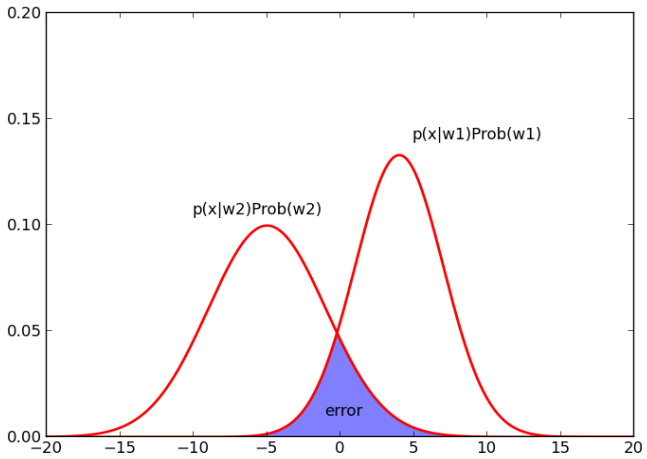


Figure 3.3: An example of two classes' distributions

Figure 3.3 shows an example of a binary classification case where w_1, w_2 denote two classes [39]. Choosing a value x_0 as the decision boundary, the data on the left of x_0 is classified as w_2 and the data on the right of x_0 is classified as w_1. The error occurs in the intersection of the two curves, and it can be given as:

$$p_{Bayes}(error) = \int_{R_{w_2}} p(x \mid w_1) Prob(w_1) \mathrm{d}x + \int_{R_{w_1}} p(x \mid w_2) Prob(w_2) \mathrm{d}x. \quad (3.14)$$

Theoretically the Bayes error can be zero if the classes are completely separable, i.e., for each x if there exists a class $k \in C$ such that $p(c_k \mid x) > 0$ then $p(c_i \mid x) = 0$ for every $i \in C \setminus \{k\}$. However, in practice, it is a very uncommon case. Generally, the Bayes error is non-zero and irreducible, even though we know the true probability distribution that generates data. The class distributions often overlap, and there may exist noises in the distribution, or some invisible variables that are not included the input may affect the output. [40] For example, we know the output of flipping a coin follows a binomial distribution, but we would still make errors when predicting the output of a series of coin flips since the process is inherently stochastic.

The Bayes error provides the minimum prediction error an optimal classifier could achieve for a given machine learning problem [37], which means that the optimal prediction error of any classifiers could be close to but never smaller than the Bayes error. The optimal classifier is typically unknown, and generally, we choose a collection of classifiers denoted as $\mathbb{L}$ and want to find an optimal classifier l^* from $\mathbb{L}$ so that $p_{Bayes}(error) \leq p_{l^*}(error)$. In this thesis, the collection of classifiers we choose include Beta binomial model, Logistic regression, Random forest and LightGBM, and they will be introduced in this chapter later. Given a data set with a finite size, the prediction error rate of each classifier l in $\mathbb{L}$ is evaluated by the given data and denoted as $p_l(error \mid data)$, and then the estimated optimal classifier will be the one with the lowest value of $p_l(error \mid data)$. Therefore, we have $p_{l^*}(error) \leq p_l(error \mid data)$. The values of $p_l(error \mid data)$ are random variables since they depend on the given data coming from an underlying distribution and generated through an inherently stochastic process. Besides, we can not get the exact values of $p_l(error \mid data)$ but compute their estimates $\hat{p}_l(error \mid data)$ by certain error estimation methods, such as cross-validation and bootstraps (which will be introduced in Section 3.2.3). These computed estimates are also random variables existing variance because of the dependence on the given data and the variation during the error estimation process, e.g., the randomness of data splitting on the cross-validation procedure.

Now let we start to discuss Beta binomial distribution, which is a compound distribution of the Beta distribution and the binomial distribution. In many cases, an experiment produces binary outcomes, e.g., coins will land either heads up or tails up after being flipped. Generally two different outcomes can be defined as 'success' with probability p and 'failure' with probability $q = 1 - p$. Each trial of such an experiment is called a Bernoulli trial. [41] A binomial model is a statistical model which has the following properties: 1. it contains a set of repeated Bernoulli trials. 2. The trials are exchangeable, which means that their joint probability does not change

if they are permuted. 3. The trials are independently and identically distributed; that is, they are not affected by each other and follow the same distribution. [28, 42]

The binomial distribution, denoted as $K \sim Bin(n, \pi)$ models the the number of successes K in n trails with a probability of success π in a trial. The probability mass function gives the probability of k successes:

$$Pr(K = k \mid n, \pi) = \binom{n}{k} \pi^k (1-\pi)^{n-k}, \quad k = 0, 1, ..., n, \tag{3.15}$$

where $n \in \mathbb{N}, \pi \in [0, 1]$. For a binomial random variable K with parameters n and π, the expectation is $\mathrm{E}(K) = n\pi$ and the variance is $\mathrm{Var}(K) = n\pi(1-\pi)$.

A beta function is defined as:

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}, \alpha, \beta > 0, \tag{3.16}$$

where $\Gamma(\alpha)$ is the gamma function, $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt$ for $\alpha > 0$.

The beta distribution denoted as $Beta(\alpha, \beta)$ is a continuous probability distribution with two positive parameters α and β. Its density function is:

$$f(x; \alpha, \beta) = \begin{cases} \dfrac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, & x \in [0, 1]; \\ 0, & \text{otherwise.} \end{cases} \tag{3.17}$$

The mean of a random variable X that follows the distribution $Beta(\alpha, \beta)$ is $\mathrm{E}(X) = \dfrac{\alpha}{\alpha+\beta}$, and its variance is $\mathrm{Var}(X) = \dfrac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$ [33].

The beta binomial distribution is a hierarchical model which has multiple parameters connected in different levels. [28] More specifically, it is the binomial distribution where the probability of success in a trail π is beta-distributed with α, β:

The prior of p follows the Beta distribution:

$$\begin{aligned} &\pi \sim Beta(\alpha, \beta), \\ &p(\pi) \propto \pi^{\alpha-1}(1-\pi)^{\beta-1}. \end{aligned} \tag{3.18}$$

The likelihood is defined as:

$$\begin{aligned} K \sim Bin(n, \pi), \quad & p(k \mid n, \pi) \propto \pi^k (1-\pi)^{n-k}, \\ p(k \mid n, \pi) &= \int p(k, \pi \mid n, \alpha, \beta) \\ &= \int p(k \mid \pi, n) p(\pi \mid \alpha, \beta) d\pi \\ &= \binom{n}{k} \frac{1}{B(\alpha, \beta)} \int_0^1 \pi^{k+\alpha-1}(1-\pi)^{n-k+\beta-1} d\pi \\ &= \binom{n}{k} \frac{B(k+\alpha, n-k+\beta)}{B(\alpha, \beta)}. \end{aligned} \tag{3.19}$$

The posterior for π is

$$\begin{aligned} p(\pi \mid n,k) &\propto \pi^{\alpha-1}(1-\pi)^{\beta-1}\pi^{k}(1-\pi)^{n-k} \\ &= \pi^{\alpha+k-1}(1-\pi)^{\beta+n-k-1} \\ &= Beta(\alpha+k,\beta+n-k). \end{aligned} \tag{3.20}$$

Equations 3.18 and 3.20 show that if the prior is a beta distribution and the likelihood is a binomial distribution, then the generated posterior will be a distribution whose parametric form is the same as that of the prior. The beta distribution is a conjugate prior for the binomial distribution [28]. A conjugate prior is convenient for computation, since the posterior is in a known form and we do not need to compute the integrals.

3.2.2 Logistic Regression

Logistic regression is a well-known algorithm for the purpose of classification. It estimates the relationship between the features and the target via a logistic function. It is very easy to understand and implement, and shows great performance on linearly separable classes, so it is widely used in diverse fields including biostatistical problems. [26] Since our work is a binary classification task, here we just discuss the binary logistic regression, but it can be extended to multiclass problems.

Considering a binary classification task, the input containing d features is denoted as $\boldsymbol{x} = (x_1, x_2, ...x_d)^T$ and the label is $y \in \{0,1\}$. Let $p \in (0,1)$ denote the probability of the positive event, thus $1-p \in (0,1)$ is the probability of the negative event. The log odds, denoted as $\log(p/(1-p))$, is the logarithm of the odds of probabilities. The log odds is modeled by linearly combining features:

$$logit(p) = \log\frac{p}{1-p} = w_0 + w_1x_1 + w_2x_2 + ... + w_dx_d = w_0 + \boldsymbol{w^T x}, \tag{3.21}$$

where $\boldsymbol{w} = (w_1, w_2, ..., w_d)^T$. When $logit(p) > 0$ which means $p > 1-p$, the sample is classified as 1, while it is predicted as 0 if $logit(p) < 0$. So the outcome y of logistic regression is determined as follows:

$$y = \begin{cases} 1, & w_0 + \boldsymbol{w^T x} > 0; \\ 0 \text{ or } 1, & w_0 + \boldsymbol{w^T x} = 0; \\ 0, & w_0 + \boldsymbol{w^T x} < 0. \end{cases} \tag{3.22}$$

When $logit(p) = 0$, the sample can be set to be classified as 0 or 1. [20]

Based on Equation 3.21, we can estimate the conditional probabilities

$$\begin{aligned} P(y=1 \mid \boldsymbol{x}) &= \frac{1}{1+e^{-(w_0+\boldsymbol{w^T x})}}, \\ P(y=0 \mid \boldsymbol{x}) &= 1 - P(y=1 \mid \boldsymbol{x}) = \frac{e^{-(w_0+\boldsymbol{w^T x})}}{1+e^{-(w_0+\boldsymbol{w^T x})}}. \end{aligned} \tag{3.23}$$

Figure 3.4 shows a typical logistic regression curve, where $z = w_0 + \boldsymbol{w}^T\boldsymbol{x}$, the blue curve shows the trend of $P(y = 1 \mid \boldsymbol{x})$ and the red lines denote the decision function of classification. [20]

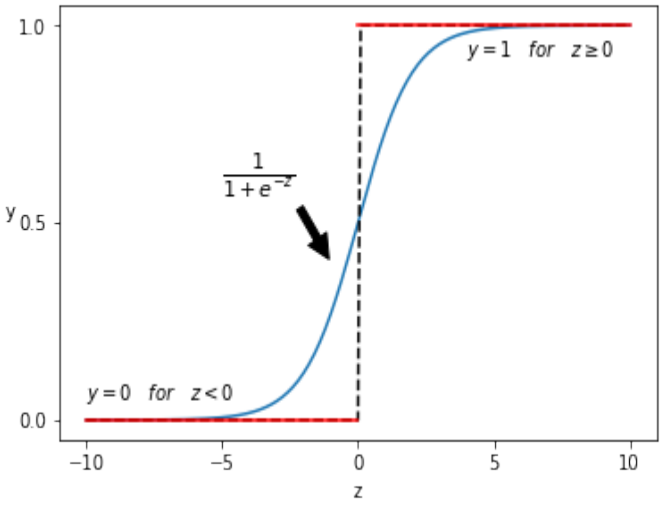


Figure 3.4: a logistic regression curve

MLE is used to estimate the coefficient w_0 and $\boldsymbol{w}$. We denote the estimated $P(y = 1 \mid \boldsymbol{x})$ as $h^{\boldsymbol{w}}(\boldsymbol{x})$, and then we can get the probability of y conditioned on $\boldsymbol{x}$,

$$P(y \mid \boldsymbol{x}) = [h^{\boldsymbol{w}}(\boldsymbol{x})]^y[1 - h^{\boldsymbol{w}}(\boldsymbol{x})]^{1-y}. \tag{3.24}$$

For a dataset $\boldsymbol{X}$ containing N samples, $\boldsymbol{w}$ can be obtained by maximizing the log-likelihood $l(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \log \prod_{i=1}^{N} P(y^{(i)} \mid \boldsymbol{x}^{(i)})$ where $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}..., \boldsymbol{x}^{(N)})^T$ and $\boldsymbol{y} = (y^{(1)}, y^{(2)}..., y^{(N)})^T$. It is equivalent to minimizing its negative form, which is the loss function of logistic regression called log loss and denoted as $J(\boldsymbol{w})$. [20] $J(\boldsymbol{w})$ is written as

$$\begin{aligned} J(\boldsymbol{w}) &= -\log \prod_{i=1}^{N} P(y^{(i)} \mid \boldsymbol{x}^{(i)}) \\ &= \sum_{i=1}^{N} -y^{(i)} \log h^{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - (1 - y^{(i)}) \log(1 - h^{\boldsymbol{w}}(\boldsymbol{x}^{(i)})). \end{aligned} \tag{3.25}$$

So

$$\begin{aligned} \boldsymbol{w}^* &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \{-y^{(i)} \log h^{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - (1 - y^{(i)}) \log(1 - h^{\boldsymbol{w}}(\boldsymbol{x}^{(i)}))\} \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \{-y^{(i)}(w_0 + \boldsymbol{w}^T\boldsymbol{x}^{(i)}) + \log(1 + e^{w_0 + \boldsymbol{w}^T\boldsymbol{x}^{(i)}})\}. \end{aligned} \tag{3.26}$$

Some optimization techniques can be adopted to solve this minimization problem, such as gradient descent method or Newton-Raphson method [20, 29]. When using

the gradient descent algorithm, first we initialize w_j as 0 or a random value for $j = 1, 2, ..., d$. The derivative of $J(\boldsymbol{w})$ with respect to w_j is

$$\frac{\partial J(\boldsymbol{w})}{\partial w_j} = \sum_{i=1}^{N} (h^{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - y^{(i)}) \boldsymbol{x}_j^{(i)}, \tag{3.27}$$

which is easy to compute. Then w_j is updated by

$$w_j = w_j - \eta \frac{\partial J(\boldsymbol{w})}{\partial w_j}, \tag{3.28}$$

where $\eta \in (0, \infty)$ is a learning rate that decides the update size in each step. An overhigh value of η possibly leads to missing minimum points or failing to converge, and an excessively low η may make the computation expensive. Therefore η needs to be carefully chosen. The update is iterated until satisfying a stopping condition, such as reaching the maximum number of iterations or satisfying a tolerance value.

As we mentioned in Section 3.1.1, when $d >> N$, the model is prone to overfitting. For a linear model, regularization is a useful technique to solve this problem. A norm penalty term that shrinks coefficients is added to the loss function:

$$J(\boldsymbol{w}) = -\log \prod_{i=1}^{N} P(y_i \mid \boldsymbol{x}_i) + \lambda \|\boldsymbol{w}\|_r^r \quad \text{for } r = 1 \text{ or } 2, \tag{3.29}$$

where $\|\boldsymbol{w}\|_r$ is the r-norm of $\boldsymbol{w}$, and $\lambda \in [0, \infty)$ is a manually predefined parameter that decides how much the coefficients are penalized. Note that w_0 does not need to be penalized since it does not influence weights of features and thus does not induce overfitting. λ is manually set, and the larger λ is, the more heavily the coefficients are penalized, leading to more coefficients close to zero. When λ is 0, the model is the logistic regression without regularization. Cross-validation is often used to find an optimal value of λ. [20, 21, 29]

When $r = 2$, the penalty term is L_2 norm, and it is referred to as Ridge regression or $L2$ regularisation. Ridge regression prevents the weights from rising too high and drives them to be small. L_2 is differentiable, and therefore it can be solved by gradient descent algorithm. While $r = 1$, the penalty term is L_1 norm and it is called Lasso regression or $L1$ regularization. Lasso regularisation can drive some coefficients of features to zero, and these features will not contribute to y. Therefore it will make the solution more sparse compared with Ridge regression and can be used as a feature selection technique. L_1 is non-differentiable, and thus Lasso has no analytical solution, but it can be solved by certain algorithms such as Proximal Gradient Descent. [20, 21, 29]

We can also apply the logistic regression using a Bayesian approach, and such an algorithm is called Bayesian logistic regression. A prior for $\boldsymbol{w}$ is initialised, and the Gaussian distribution is a common choice. Then we can perform Bayesian inference described in Section 3.2.1. There is no closed form for the predictive posterior,

but some methods, such as Laplace Approximation, can be used to approximate it. [30] Moreover, now there exist automatic software or packages such as Stan or PyMC3 can compute these posteriors using simulation.

3.2.3 Random Forest

Random forest constructs a group of decision trees and ensembles them using Bagging method. Therefore we first have a look at the decision tree and Bagging algorithm and then discuss the random forest. Note that since our work is a classification problem, here only classification models are discussed.

The decision tree is a tree-like model, where the root contains the whole feature set, each internal node is a feature, each branch denotes a rule based on the feature value, and each leaf is a class label. It classifies an instance from the root (top) down to a leaf (bottom) by testing a sequence of features. A path from the root down to a leaf can be explained as a collection of if-then rules pointing to a class label, and thus the decision tree is easy to understand and interpretable. [20] An example of a decision tree is illustrated in Figure 3.5.

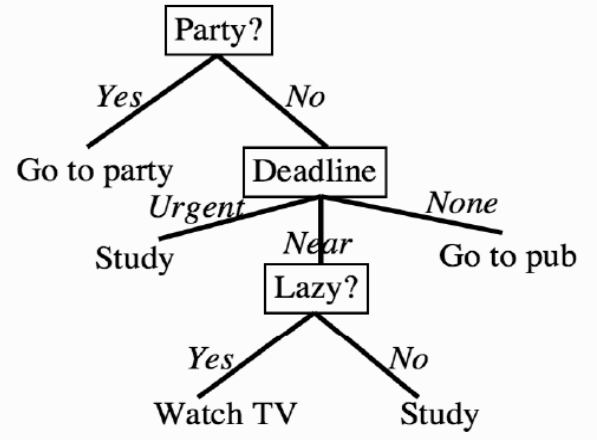


Figure 3.5: A simple example of decision tree [19]

There are different types of decision trees, but the general approach of decision tree learning can be summarised as follows: **1.** At a node, the best feature is selected as the decision feature from the feature set. The best feature is the one that can best separate the data, which is measured by a splitting criterion. **2.** The data is partitioned into smaller subsets and distributed to the new child nodes, based on certain cutoff values of the decision feature. Then the decision feature is removed from the feature set. **3.** Repeating the steps 1-2 for each node until satisfying a stop criterion such as the depth of the node is equal to a preset threshold. [19, 20, 21]

Information gain is a commonly used splitting criterion. Assuming that for a dataset X, the samples with the label c is $X(c)$ and the proportion of $X(c)$ is $p_c = \frac{X(c)}{X}$,

$H(X) = \sum_{c \in C} -p_c \log_2 p_c$ is the information entropy of the dataset X. It measures the purity of data, and a lower information entropy means a higher purity. If all the samples belong to the same class, the information entropy of the data is 0. The information gain of feature A on the dataset X is defined as

$$IG(X, A) = H(X) - \sum_{a \in A} \frac{|X_a|}{|X|} H(X_a), \tag{3.30}$$

where X_a is the subset containing the samples with the feature value a. A feature has larger information gain means that it can improve classification more if it is chosen as the decision feature. The ID3 algorithm uses information gain as its splitting strategy, and the feature that has the highest information gain is selected as the splitting point. [19, 20]

Another popular splitting criterion is the GINI index. The GINI impurity is a metric evaluating the impurity of the data. Using the same notation as the above, $Gini(X) = \sum_{c \in C} \sum_{c' \in C \setminus \{c\}} p_c p_{c'} = 1 - \sum_{c \in C} p_c^2$. The lower GINI value is, the purer the data is. The Gini index of feature A on the data X is

$$GI(X, A) = \sum_{a \in A} \frac{|X_a|}{|X|} Gini(X_a). \tag{3.31}$$

The Gini index indicates how mixed the classes are after the data is split by the feature. If the data is separated perfectly, the Gini index will be 0. Classification And Regression Tree (CART) divides a node based on the feature with the lowest GINI index. [19, 20]

We should notice that the decision tree is prone to suffer from the overfitting problem, especially if there exists noisy data or irrelevant features. To prevent overfitting, we can early stop growing a tree when meeting some certain stopping criteria, e.g., the depth of the tree is limited by a predefined maximum number, or there are some pruning techniques can be used. [21] Another problem is that the decision tree is a high-variance model, and small fluctuations in the data might cause a totally different tree is generated. Besides, since the decision tree learning is greedy, it can not be guaranteed that the generated model is globally optimal.

Bagging, also called Bootstrap aggregating, is a machine learning ensemble method which trains a set of homogeneous classifiers by bootstrap samples and then ensembles them to generate outputs. Bagging first produces a set of bootstrap samples, each of which is produced by randomly sampling with replacement from the training set, and the number of samples in each bootstrap sample is the same as that of the training set. Then a base classifier is trained on each bootstrap sample separately. Finally, combining the outcomes of this collection of classifiers using a certain strategy to generate the prediction for a sample. [19]

For classification problems, a voting strategy is used to make final predictions. Majority voting is a common and simple strategy which has three versions: **1.** the output class is the one that is agreed by all classifiers (unanimous voting) [19]. **2.** the final prediction is the class that receives more than half of the votes (simple majority) [19, 31]. **3.** the class that get the highest total votes is chosen (plurality voting) [31]. In addition to the majority voting, the weighted voting is another often used strategy. The contribution of each classifier to predictions is weighted, and the weights can be decided in a self-defined way [20, 31].

We are often interested in estimating class-probabilities instead of class-labels, or we prefer to classifying samples to the class whose class probability are highest. The class-probabilities are combined by averaging them over all the classifiers, which is called soft voting. For a tree, the class probability is the percentage of samples of the class in the leaf. Soft voting can improve the estimates of class-probabilities and produce bagged classifiers with lower variance. [29]

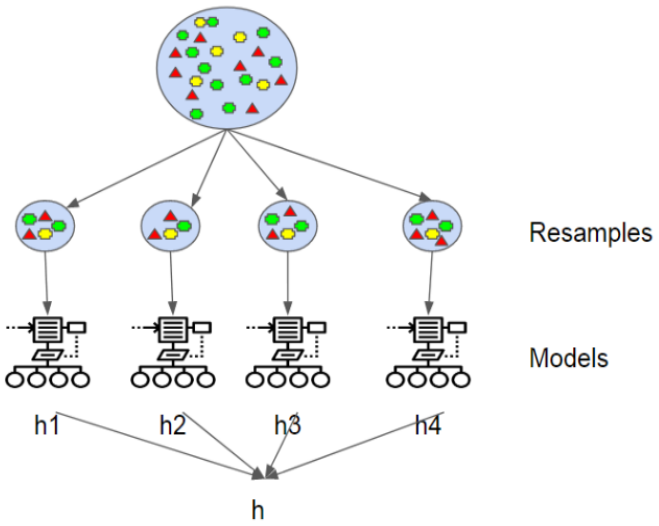


Figure 3.6: An illustration of Bagging framework[45]

Figure 3.6 shows how Bagging works. Bagging is an efficient algorithm since it trains multiple classifiers in parallel and thus has the same level of time complexity as training a single model. For a bootstrap sample with the same sample size as the complete dataset, there are around 63.2% of unique samples while the other 36.8% are the replicates. Those 36.8% samples in the original data that do not appear in the bootstrap sample are out-of-bag (OOB) examples. OOB examples can be used to evaluate how good the generalisation ability of the model is, which is called OOB estimate. [20, 19] Since bootstrap samples are slightly different from each other, a set of different classifiers can be produced. Combining multiple classifiers can help to decrease variance and prevent overfitting, and therefore Bagging is suitable for high-variance models such as decision trees. [19, 29]

Random forest is a variant of Bagging. It uses the unpruned decision tree as the base classifier. The difference happens in the process of decision tree training: at the node to split, not the whole feature set but a subgroup of the features is randomly chosen as the candidates, and the best one is chosen as the decision feature among them. Since there are fewer features to evaluate, the training process is speeded up. Moreover, it adds randomness in the training of each tree, which increases the variance between different trees and then possibly improves the generalisation ability of the classifier. [19] For a classification problem, at a node with d features, typically $\log_2 d$ features are selected as splitting candidates [29].

In a random forest, the feature importance can be evaluated by 'mean decrease in impurity' or 'mean decrease in accuracy' mechanism. The 'mean decrease in impurity' sums the impurity decrease every time the feature is selected as the split point across every tree, and then averages it across all the trees. The more mean decrease in impurity is, the more significant the feature is. Another method 'mean decrease in accuracy' measures the effect of the feature on the model accuracy. For each tree, it randomly permutes the feature values in the OOB samples and measures the decrease in accuracy after permutation. The mean decrease in a forest is obtained by averaging the decreases over all the trees. A lower the mean decrease in accuracy means that the feature has less effect on the model accuracy, and therefore the less important the feature is. [29, 32]

3.2.4 Boosting

Boosting is a collection of ensemble algorithms which combines a collection of weak homogeneous classifiers into one strong classifier. Different from Bagging, a boosting algorithm trains a group of base classifiers sequentially. Each base classifier is trained using a data set which is weighed by the prediction made by the previous classifiers. More specifically, in each round, the misclassified samples are given heavier weights in the newly weighed data set used for the next training. The final prediction is given by combing the classifiers through a weighted majority voting strategy, where models that have lower error rate are given more weights. The Boosting framework is showed in Figure 3.7. [29, 30]

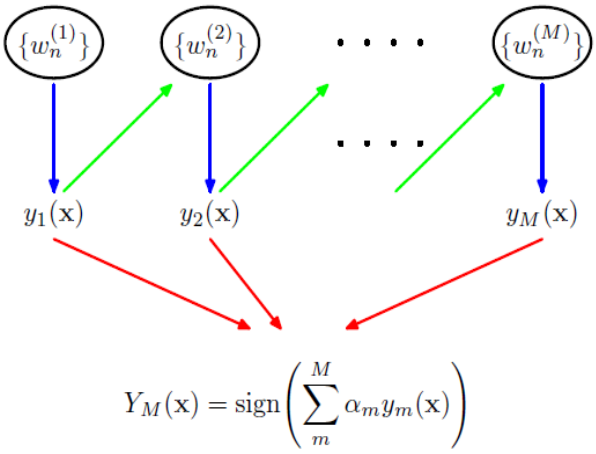


Figure 3.7: An illustration of Boosting framework [30]

Adaptive Boosting (AdaBoost) is a commonly used boosting algorithm. Given a dataset $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n)\}$ and $y_i \in \{-1, 1\}$, we denote the initial weight of the data point $\boldsymbol{x}_i$ as w_i^1 and set it as $1/n$, and fix the number of base classifiers M. For $m = 1, 2, ..., M$, the classifier $h_m(\boldsymbol{x})$ is trained sequentially, aiming to minimizing the weighted loss function

$$J_m = \sum_{i=1}^{n} w_i^m I(h_m(\boldsymbol{x}_i) \neq y_i). \tag{3.32}$$

Then the weighted error rate of $h_m(\boldsymbol{x})$ can be calculated by

$$\epsilon_m = \frac{\sum_{i=1}^{n} w_i^m I(h_m(\boldsymbol{x}_i) \neq y_i)}{\sum_{i=1}^{n} w_i^m}. \tag{3.33}$$

Computing the weight of the classifier using this error rate by the formula $h_m(\boldsymbol{x})$

$$\alpha_m = \frac{1}{2} \ln \frac{1 - \epsilon_m}{\epsilon_m}. \tag{3.34}$$

The weights of the data points are updated by

$$w_i^{m+1} = \frac{w_i^m \exp\left(-\alpha_m y_i h_m(\boldsymbol{x})\right)}{Z_m}, \tag{3.35}$$

where Z_m is a normalization factor that makes sure all the weights sum up to 1.

After M classifiers being trained, the prediction is made by

$$H(\boldsymbol{x}) = sign(\sum_{m=1}^{M} \alpha_m h_m(\boldsymbol{x})). \tag{3.36}$$

Equation 3.34 shows that a base classifier that has a lower weighted error rate is assigned a greater weight when performing predictions. From Equation 3.35 we can see that those samples wrongly predicted are given more weights more while the other samples correctly predicted are given less weights, and then subsequent classifiers will more focus on samples that have been misclassified before. [19, 29, 30]

Another boosting method is Gradient boosting, which is popular in recent years and shows a strong power. It is widely used in the learning to rank field [43], e.g., Frery et al.(2017) adopted gradient boosting algorithms to solve a anomaly detection problem in a learning to rank approach [44]. Given a dataset $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n)\}$, a loss function $L(y, H(\boldsymbol{x}))$ that is differentiable, and the number of iterations M, we aim at finding a function $H^*(\boldsymbol{x})$ which makes the loss function minimized:

$$H^*(\boldsymbol{x}) = \underset{H}{\operatorname{argmin}} E_{\boldsymbol{x},y} L(y, H(\boldsymbol{x})). \tag{3.37}$$

To optimize the cost function, repeatedly choosing a weak hypothesis that points to its negative gradient direction. Firstly an initial model $H_0(\boldsymbol{x})$ is generated. For $m = 1, 2, ..., M$ and $i = 1, 2, ..., n$, the residual of each sample is computed using the formula

$$r_{im} = -\Big[\frac{\partial L(y_i, H(\boldsymbol{x}_i))}{\partial H(\boldsymbol{x}_i)}\Big]_{H(\boldsymbol{x})=H_{m-1}(\boldsymbol{x})}. \tag{3.38}$$

Then a base classifier $h_m(\boldsymbol{x})$ is trained using the dataset $\{(\boldsymbol{x}, r_i)\}_{i=1}^n$. The multiplier γ_m can be obtained by

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, H_{m-1}(\boldsymbol{x}_i) + \gamma h_m(\boldsymbol{x})). \tag{3.39}$$

The model is updated by

$$H_m(\boldsymbol{x}) = H_{m-1}(\boldsymbol{x}) + \gamma_m h_m(\boldsymbol{x}). \tag{3.40}$$

After M iterations, the $H_M(\boldsymbol{x})$ is the finally classifier. [29, 46]

In this thesis, we applied a boosting algorithm called LightGBM. LightGBM is a further improved algorithm. It optimises computation speed and memory usage by histogram-based algorithms, and the Gradient-based One-Side Sampling and Exclusive Feature Bundling techniques can help handle big data with faster speed and less memory consumption. The trees of LightGBM are grown leaf-wise while the other algorithms grow trees level-wise, which makes LightGBM more accurate but also prone to overfitting when the data is small. LightGBM also supports parallel learning. [48]

3.3 Feature Selection

Feature selection aims to choose features that are most relevant for predicting the target from the complete feature set. By removing irrelevant or redundant features,

feature selection helps simplify constructed models, avoid the curse of dimensionality, improve the performance of classifiers and reduce computation space and time [49]. It also assists in understanding data, e.g. learning about the critical factors that affect the relationship between the object and the target. Feature selection is an significant step in implementing machine learning, especially in the case that data is high-dimensional. In our case, the features are a vast number of TCRβ sequences and plenty of them are not related to the target phenotype, so it is necessary to select an appropriate feature subset before training machine learning models. Feature selection approaches mainly consists of filter methods, wrapper methods and embedded methods.

3.3.1 Filter Methods

Filter methods score and rank each feature in the original feature set by a selected statistical test which assesses the feature relevance with the label and then remove those features whose scores are below a manually set threshold. They are independent of machine learning algorithms, and since they do not take the relationships between features into account and may therefore select redundant features, they are often the first action in the stage of feature selection. [49] There are various statistical criteria could be used to assess features. In this thesis we used the Fisher exact test to filter features firstly before trying other feature selection methods.

Fisher exact test is a statistical measure for assessing the association between two categorical variables by analyzing their contingency table. It is suitable for analyzing small data set [50]. The contingency table is a table displaying the frequency distribution across two nominal variables, and the 2×2 contingency table is the most applied case for inspecting the relationship between a feature and the target variable:

		Feature Presence		
		Present	Absent	Total
Label	Positive	a	b	$a+b$
	Negative	c	d	$c+d$
	Total	$a+c$	$b+d$	N

Table 3.1: An example of a 2×2 contingency table

The null hypothesis H_0 of Fisher exact test is that the two variables are independent. The p-value is the total probability over all the tables which have the probabilities equal to or smaller than that of the observed condition under the null hypothesis. It is computed using the formula [51]

$$p = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{a!b!c!d!N!}. \tag{3.41}$$

A lower p-value indicates a higher significance that the null hypothesis could not explain the observation. If the p-value is lower than a chosen threshold α known

as significance level, the null hypothesis can not be accepted, concluding that there exists a correlation between the variables. This p-value gives a one-tailed probability associated with the top left element of the contingency table. E.g., for Table 3.1, if the p-value is less than α, then it can be concluded that the feature is present more frequently in positive samples.

3.3.2 Wrapper Methods

A wrapper method uses a kind of model to select features by evaluating the prediction performance of candidate subsets. The quantity of all the possible feature subsets is 2^N for a dataset with N features [49], and thus exhaustive searching of subsets will be computational-expensive for high-dimensional data. Therefore, some simplified methods which search for suboptimal solutions are widely used. Even though their solutions are not globally optimal, they have acceptable computational cost and also output good prediction performance. In this thesis, we tried two greedy sequential search methods, Sequential forward selection (SFS) and Recursive Feature Elimination (RFE).

Recursive Feature Elimination (RFE)

Recursive feature elimination (RFE) selects features in the opposite direction, recursively removing features from the feature set. Given a feature set A, the steps of performing RFE are: **1.** Fitting the chosen model on A and ranking the features by a criterion like the feature importance. **2.** a small number of features which rank last is eliminated from A. **3.** Repeating step 1 - 2, until a termination criterion is met. [52] Generally, we could not know how many features should be kept in advance. We iterate steps 1-2 until all the features are eliminated and use cross-validation to evaluate different feature subsets. The subset which shows the best cross-validation result will be selected.

RFE often performs well, but it may remove those features which improve the model performance when combined with some other features.

3.3.3 Embedded Methods

Embedded methods automatically select features during the time of model training. Compared with wrapper methods, they are less computationally intensive but also can yield good results. [49] Tree-based methods and regularisation-based methods are the most popular embedded methods.

Tree-based methods use tree-structure models such as Random forest to select features by feature importance. The feature importance can be computed by 'mean decrease in impurity' or 'mean decrease in accuracy', which are introduced in Section 3.2.3. When training such a model, the importances of all the features are assessed meanwhile. After finishing training, a set of features considered most important will be kept. Regularisation-based methods select features by using linear models with

L1 regularisation such as Lasso regression. Such models shrink the coefficients of the features which are least important as zeros, and therefore those features are removed. [20, 29]

3.4 Hyperparameter Tuning

A model hyperparameter is a configuration variable which is preset before the model is trained, e.g., the regularisation term is a hyperparameter of the regularised logistic regression model. A machine learning algorithm generally has multiple hyperparameters, and different algorithms have different hyperparameters. Hyperparameters govern the process of model training and significantly influence the model performance. Hyperparameter tuning, also called hyperparameter optimisation, is to find out the best hyperparameters which produces the optimal model. [53]

Grid search is a simple and widely used hyperparameter optimisation technique. A range of values is predefined for each hyperparameter, generating a grid of hyperparameters, and a metric used to evaluate model performance is chosen. Then grid search exhaustively evaluates each possible combination in the grid by cross-validation on the training set, evaluation on a held-out set or maximising the marginal log-likelihood with Gaussian processes [54]. The combination of hyperparameters which yield the best evaluation score will be the optimal hyperparameter setting. Grid search is computationally intensive when exploring a big hyperparameter space, because the number of possible combinations increases exponentially with the number of hyperparameters increasing. [55]

Random search is another common technique to tune hyperparameters. Each hyperparameter is defined as following a specific distribution over a range of values, and an evaluation metric and the number of iterations are set. In an iteration, the hyperparameter values are randomly sampled from the defined distributions and then evaluated. After reaching the number of iterations, the set of hyperparameters with the optimal score is selected. When many hyperparameters need to be tuned but only a small amount of hyperparameters affect the model performance significantly, Random search is faster and more efficient, since it can focus on finding optimal values of the important hyperparameters. [55]

The above methods can search a hyperparameter space including both discrete and continuous hyperparameters. In our case, both of these approaches were applied. We assessed the hyperparameters by cross-validation on the training set, and the evaluation metric is AUROC. Since certain hyperparameters are real-valued or unbounded in the hyperparameter space, we should try to set appropriate exploration ranges of hyperparameter values which are reasonable and contain the optimal ones.

4 Experiments and Results

This chapter gives details about the process of our experiments and the experiment results. We give a description about the data we studied and how we preprocessed it in Section 4.1. In Section 4.2, first the process of implementing baseline model is introduced, and then we describe how we applied other algorithms. In Section 4.3 we do some exploration on the data , and finally the results of the experiments is presented in Section 4.4.

4.1 Data Collection and Preprocessing

Two datasets were studied in this thesis, and they can be downloaded and analysed from the Adaptive Biotechnologies immuneACCESS site. The data has been obtained by bulk sequencing method, which is one of high-throughput sequencing methods and commonly chosen to study TCR repertoires in a large cohort (a cohort is a group of subjects with a shared characteristic) [56]

The first dataset is used for the cytomegalovirus (CMV)-related study available at https://clients.adaptivebiotech.com/pub/Emerson-2017-NatGen. Cytomegalovirus (CMV) is a kind of common herpes virus, and 30–90% of adults are infected. Once a person is infected, he will retain it for life. CMV is used to model the immune system for public T cell responses in many studies, and because of its relatively high infection rate, it provides good statistical power. The cohort 1 used as the training set is a group of healthy bone marrow donors, and the cohort 2 used as the testing set is a group of healthy volunteers whose pathogen exposure history was examined. For each cohort, the TCRβ sequences were immunosequenced from peripheral blood samples of the subjects in it. [5]

The second one is used for the study related to rheumatoid arthritis (RA) available at https://clients.adaptivebiotech.com/pub/e19cf008-6363-4bd7-8f75-05bc29e17600. Rheumatoid arthritis (RA) is a kind of chronic autoimmune disease that mainly damages joints and also possibly other organs. It is caused by the immune system wrongly attacking the body's own tissues, and the T cells produce autoantibodies for years before the clinical diagnosis. The TCR repertoires we used were obtained from the synovial fluid of 65 RA patients and 20 healthy controls (HC) via TCRβ sequencing. [57] Since the size of dataset is small, we performed LOOCV to evaluate the model performance instead of evaluating on a testing set divided from the whole dataset.

Since the majority of T cells express $\alpha\beta$ TCRs, $\alpha\beta$ T cells are mainly studied. When using bulk sequencing, we can not get information about paired TCRs ($\alpha\beta, \gamma\delta$), but only single TCR chains. [56] Most of TCRβ chains are generated before the rearrangement of TCRα chains [58, 59], and β chains have a higher diversity compared with α chains. Besides, an $\alpha\beta$ T cell has a unique β chain but possibly two α chains [56], so studying TCRα chains increases the level of complexity. Therefore the TCRβ

chain is the main study target. The CDR3 region directly interacts with antigen peptides and is the most variable region. It is the studied region in many TCR repertoire studies. [56] CDR1 and CDR2 are not the key region that interacts with antigens, but important in recognising MHC modules and beneficial in understanding the TCR and its binding properties [56]. As mentioned in Section 2.2, they are determined by V genes, and thus as we utilize V genes we also utilize CDR1 and CDR2 to some extent. We conducted our research using TCRβ chains, each of which consists of a CDR3 amino acid sequence, a V gene and a J gene. For a subject, we combined a group of elements to form TCRs, and then only kept those productive TCRs. The productive TCRs are those TCRs with in-frame CDR3 [60]. The table 4.1 shows the list of the elements making up of TCRs in our case and an example of a productive TCRβ representation.

Element	Example
CDR3 amino acid	CASSGQGAYEQYF
V Family Name	TCRBV09
V Gene Name	TCRBV09-01
V Gene Allele	null
J Family Name	TCRBJ02
J Gene Name	TCRBJ02-07
J Gene Allele	01
sequenceStatus	In

Table 4.1: The TCR-related columns and an example of a TCRβ

The schematic representation of phenotype and TCR presence/absence data is shown in Figure 4.1. The phenotype is the target to predict which is indicated as positive (1) or negative (0). Each TCR is a feature indicated its presence (1 denotes being present and 0 denotes being absent), and M denotes the total count of unique TCRs.

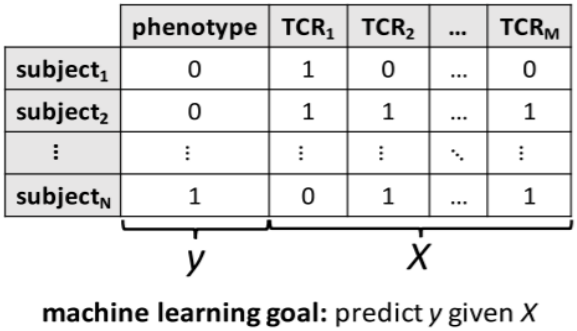


Figure 4.1: Representation of phenotype and TCR presence/absence data [5]

We have a hypothesis that it is possible that the more phenotype-associated TCRs or the higher proportions of phenotype-associated TCRs a subject has, the more likely it is phenotype-positive. Therefore we also studied the TCR count data and the

TCR frequency data, to see whether the prediction performance could be improved if TCR quantity or TCR proportion are taken into account. The TCR count data records the quantity of each unique TCR in each subject, while in the TCR frequency data, an element represents the proportion of a TCR in a subject. Figure 4.2 shows the data representations.

	phenotype	TCR_1	...	TCR_M
$sample_1$	0	5	...	0
⋮	⋮	⋮	⋮	⋮
$sample_N$	1	0	...	8

(a) TCR Count data

	phenotype	TCR_1	...	TCR_M
$sample_1$	0	2.5×10^{-5}	...	0
⋮	⋮	⋮	⋮	⋮
$sample_N$	1	0	...	5×10^{-5}

(b) TCR Frequency data

Figure 4.2: Representations of the TCR count data and the TCR frequency data

The values in the TCR frequency data are extremely small ($10^{-6} - 10^{-3}$), which is not applicable when training linear models. Therefore before training linear models on the frequency data, we scaled the features to fall within certain appropriate ranges. We standardized the values of each feature to using the formula $x^{'} = \frac{x-\bar{x}}{\sigma}$, where $\bar{x}$ is the mean of x and σ is its standard deviation. A standardized feature follows a normal distribution with a 0 mean and a 1 variance.

4.2 Machine Learning Implementation

4.2.1 Baseline model

The Beta-binomial model is our baseline model, developed by Emerson et al. [5] Firstly, we need to determine a list of phenotype-associated TCRβ chains. To measure the association between the TCR incidence and the phenotypes, a one-tailed Fisher's exact test is performed for each of the TCRβ chains. The null hypothesis is that the TCRβ is not associated with the phenotypes. In theory, only the alternative hypothesis that the TCRβ is enriched in phenotype-positive subjects makes sense, but we also had a try of conducting the one-tailed Fisher's test in the opposite direction, where the alternative hypothesis is that the TCRβ significantly appears among phenotype-negative subjects. The purpose is to verify whether it is reasonable to predict phenotypes of subjects using the TCRs that are enriched in phenotype-positive subjects. If using TCRs enriched in positive subjects could give acceptable

classification results while using TCRs enriched in negative subjects could not, the reasonableness is verified in a way. Figure 4.3 presents an example of the contingency table. It presents the incidence of a TCRβ among positive subjects and negative subjects separately, and the top left cell determines that the alternative hypothesis of Fisher exact test is the TCRβ has increased occurrence among positive samples. The p-value of Fisher exact test is 7.96×10^{-13}.

		V: TCRBV09-01 **J**: TCRBJ02-07*01 **CDR3**: CASSGQGAYEQYF	
		present	absent
Phenotype: CMV serostatus	positive	61	228
	negative	12	340

P = 7.96×10^{-13}

Figure 4.3: An example of a contingency table of TCR presence versus CMV serostatus

After computing the p-values of Fisher exact test for all the TCRβs, we can obtain a group of phenotype-associated TCRβs by setting a p-value threshold. The phenotype-associated TCRβs are those TCRβs with p-values lower than the threshold. Given a dataset with N subjects, we counted the amount of phenotype-associated TCRβs k_i among the sum of unique TCRβs n_i for each sample i $(i = 1, 2, .., N)$.

The phenotype burden is defined as the amount of TCRβs, which are associated with the phenotype status, out of all the unique TCRβs in a sample. The probability that a TCRβ in the sample i has association with the phenotype is denoted as p_i. The phenotype burden was modelled using the Beta-binomial model, and it can be interpreted as k_i successes in n_i trials with p_i probability of success in a trial. We trained the model on the two groups of subjects with different phenotype statuses separately. We denote $c_i \in \{0, 1\}$ as the class label, and let p_i independently and identically follow a beta distribution within each class,

$$p_i \sim Beta(\alpha_{c_i}, \beta_{c_i}), c_i = 0, 1. \tag{4.1}$$

Then as discussed in Section 3.2.1, for subject i, the likelihood of k_i conditioned on class assignment c_i is

$$p(k_i \mid n_i, c_i) = \binom{n_i}{k_i} \frac{B(k_i + \alpha_{c_i}, n_i - k_i + \beta_{c_i})}{B(\alpha_{c_i}, \beta_{c_i})}. \tag{4.2}$$

The parameters of the beta priors can be determined by maximizing the joint likelihood on the whole dataset

$$p(k \mid n, c) = \prod_{i=0}^{N} p(k_i \mid n_i, c_i) = \left(\prod_{i:c_i=0} p(k_i \mid n_i, c_i) \right) \left(\prod_{i:c_i=1} p(k_i \mid n_i, c_i) \right). \tag{4.3}$$

Directly dealing with Equation 4.3 is complicated. We formed the log joint likelihood, removed the terms that only rely on the data, and utilized parameter separability, producing Equation 4.4,

$$\ell_a(\alpha,\beta) = -N_a \log B(\alpha,\beta) + \sum_{i:c_i=a} \log B(k_i+\alpha, n_i-k_i+\beta), a = 0,1. \tag{4.4}$$

Then we could get the parameters by maximizing Equation 4.4 using the standard numerical gradient ascent method [61],

$$\{\alpha_a, \beta_a\} = \underset{\alpha,\beta\in\mathbb{R}^+}{\operatorname{argmax}}\, \ell_a(\alpha,\beta), a = 0,1. \tag{4.5}$$

The Equation 4.5 is based on MLE method which is mentioned in Section 3.2.1. An alternative way to estimate model parameters is based on the method of moments (MOM). Assuming that we have N i.i.d. random variables $\boldsymbol{X} = (X_1, X_2, ..., X_N)$ distributed according to a distribution $f_X(x;\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$, we aim to estimate the d unknown parameters $\boldsymbol{\theta} = (\theta_1, ..., \theta_d)$ of $f_X(x;\boldsymbol{\theta})$. The k-th population moment of X is the expectation of its k-th power, and it can be expressed as the function of $\boldsymbol{\theta}$, $\mu^k(\boldsymbol{\theta}) = E[X^k]$ where $k \in \mathbb{N}^+$. The k-th sample moment is $m^{(k)}(\boldsymbol{X}) = \frac{1}{N}\sum_{i=1}^{N} X_i^k$. The MOM estimator estimates the d distribution parameters by solving the equations

$$\begin{cases} m^{(1)}(\boldsymbol{X}) = \mu^1(\boldsymbol{\theta}); \\ \vdots \\ m^{(d)}(\boldsymbol{X}) = \mu^d(\boldsymbol{\theta}), \end{cases} \tag{4.6}$$

leading to $\hat{\boldsymbol{\theta}} = (\hat{\theta}_1, ..., \hat{\theta}_d)$. [62] The expectation μ and the dispersion parameter ϕ of Beta distribution have the properties $\mu = \frac{\alpha}{\alpha+\beta}$ and $\phi = \frac{1}{\alpha+\beta+1}$. With MOM, we initialized $\tilde{p}_i = \frac{k_i}{n_i}$ and estimated $\hat{\mu}$ and $\hat{\phi}$ in an iterative way, until the sum of squared differences between the old and new weights was less than a predefined tolerance [63, 64, 65]. For a relatively small dataset, the estimator using MOM may perform better compared to the MLE estimator [65].

After deciding the priors, we can get the posterior of each class for an unseen subject by multiplying the class prior and the likelihood. The class priors were approximated as the frequency of the class in the training data with Laplace smoothing. The posterior is written as

$$p(c' = y \mid n', k') = \binom{n'}{k'} \frac{B(k'+\alpha_y, n'-k'+\beta_y)}{B(\alpha_y,\beta_y)} \frac{N_y+1}{N+2}, y = 0,1 \tag{4.7}$$

where c', n', k' are the class label, number of unique TCRβs and number of phenotype-associated TCRβs of the novel subject respectively, and N_y is the count of the subjects with the label y in the training set. The prediction was made by the maximum a posteriori (MAP) classifier

$$\hat{c}(k', n') = \underset{y\in\{0,1\}}{\operatorname{argmax}}\, p(c' = y \mid n', k'). \tag{4.8}$$

To simplify the classification, we made the log-posterior odds ratio

$$F(k', n') = \log p(c' = 1 \mid k', n') - \log p(c' = 0 \mid k', n'). \tag{4.9}$$

Then classification could be made by the decision function

$$\hat{c}(k', n') = \begin{cases} 0, & F \leq 0; \\ 1, & F > 0. \end{cases} \tag{4.10}$$

Approaches to fitting the model

The paper [5] introduces the MLE method but does not mention how they determine the initial parameter values to optimize. Here we designed two functions, denoted as Function1 and Function2 respectively, to get an initial estimate of parameters. These two functions can be seen in Appendix A. Function1 approximates the probability of success within each class as $\hat{p}_l = \sum_{i:c_i=l} k_i / \sum_{i:c_i=l} n_i$ where $l = 0, 1$, and then it makes point estimate based on MOM. Function2 combines Function1 with the Jackknife sampling method. Supposing that there are n samples, the Jackknife sampling method calculates the estimate for each subset with the i-th sample omitted $(i = 1, 2, ..., n)$, and then averages these n estimates to get the final result [66]. After obtaining initial values, we can fit the model using MLE estimator.

In addition to the MLE estimator and the MOM estimator, we also employed the R package VGAM [67, 68] to fit the model. This package provides the function *vglm* that can fit vector generalized linear models (VGLMs) and the family function *betabinomialff* that can be used to fit the beta-binomial model using MLE method. A VGLM is defined as a vector of M linear predictors, each of which is $\eta_j = \beta_j^T x$ where $j = 1, 2, ..., M$, x is a vector of covariates (sometimes set as 1 for an intercept) and β is a vector of coefficients to fit. The M linear predictors model the M parameters of the distribution, and a parameter can be transformed using a link function. The function *betabinomialff* fits the probability of success p of the beta-binomial model, and the default models are $\eta_1 = \log(\alpha)$ and $\eta_2 = \log(\beta)$. [68]

4.2.2 Other algorithms

After reimplementing the baseline model, we explored new approaches for the project. Here we describe the process of developing new models.

After preprocessing the data as described in Section 4.1, we selected those features (i.e. TCRβ chains) that are most relevant to the target. First of all, we removed the TCRβs that only occur in one subject and thus would not contribute to the classification. It decreased the feature dimension of the CMV dataset from about 89 million to around 11 million, and that of the RA dataset is reduced from around 1.4 million to about 34,000. Secondly, the one-tailed Fisher exact test was performed on each of the remaining features. We were only interested in the TCRβs enriched in positive subjects. We kept the k ($k \in \mathbb{N}^+$) TCRβs that have the lowest

p-values, where k was chosen from a a set of candidates by performing 10-fold CV on the training set with the Logistic regression model evaluated by AUROC. Finally, we tried various feature selection methods, consisting of RFE, regularisation-based embedded and tree-based embedded methods. The subset of features that yielded the best evaluation score was selected.

After the feature selection, we trained different machine learning algorithms to fit the data. The algorithms we used were Linear regression with L1 and L2 regularisation, Random forest and LightGBM. The models were evaluated using LOOCV or 10-fold CV, and the AUROC was the evaluation metric. When LOOCV is performed, all the test point outputs (probability of phenotype-positive) are combined to create a single ROC curve, while when conducting 10-fold CV, the mean of the AUROCs of the 10 folds is calculated. To further improve the prediction ability, we selected one or several models among them that performed relatively well and optimised their model hyperparameters by Grid search.

For the CMV dataset, the final models we got could be assessed on the testing set to estimate the generalisation ability on unseen data. For the RA dataset, since there is no separate testing set and the size of the training set is relatively small, we used the LOOCV AUROC score as the criteria to assess the models.

4.3 Data Analysis

4.3.1 Cytomegalovirus (CMV)

The CMV data is composed of one training set and one testing set. The training set contains 666 subjects, where 352 subjects are CMV-negative, 289 subjects are CMV-negative and 25 subjects have unknown CMV-statuses. The training set has 89,872,238 unique TCRβ chains, with an average of 193,666 (80496 s.d.) unique TCR sequences per subject. The testing set contains 120 subjects with on average 202,918 (109,059 s.d.) unique TCRβ sequences per subject.

For a gene family or a gene, we compared its occurrence frequencies of two groups (phenotype-positive subjects and phenotype-negative subjects) by performing the one-tailed t-test [69], to see whether its occurrence frequency shows a statistical difference between different groups (whether the group of positive subjects has higher frequencies). There are only 2 J gene families, J01 and J02, and their occurrence frequencies do not differ by phenotype statuses. The V genes are more diverse and some of them show differences in occurrence frequencies between different classes. Figure 4.4 shows the mean occurrence frequencies of V gene families in the training data. From the figure, we can see that V05, V06, V07 are the V gene families that most often occur. Figure 4.5 presents the mean occurrence frequencies of the 10 most frequently appearing V genes in the training data.

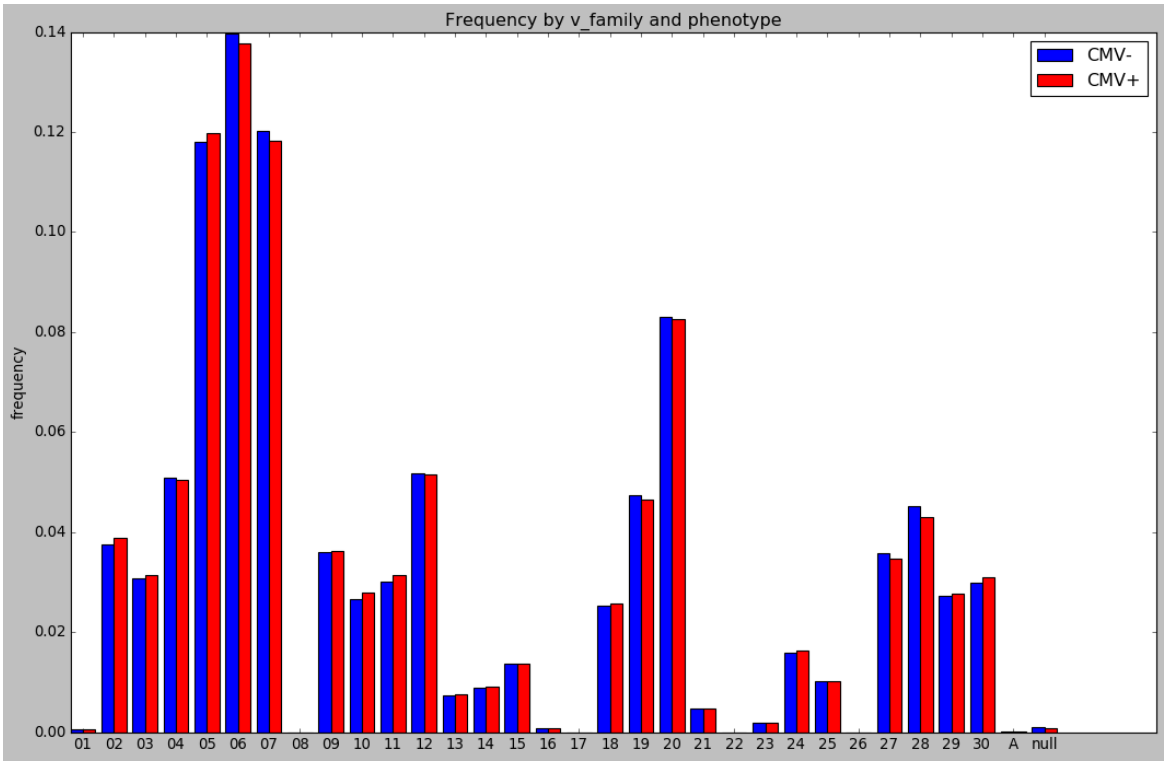


Figure 4.4: Mean frequencies of V gene families on the CMV traing set

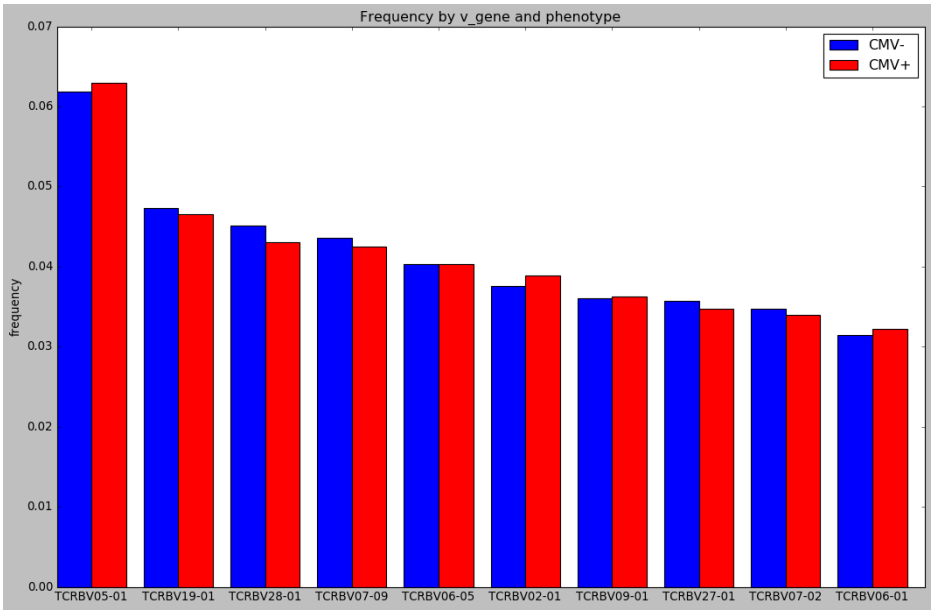


Figure 4.5: Mean frequencies of V genes on the CMV training set

The results of one-tailed t-test on each V gene family and the 10 most often occurring V genes show that the V02, V10, V17 gene families, and the V02.01 and V06.01 genes have statistical differences (p-value < 0.05) between CMV-positive and CMV-negative subjects (higher frequencies in the positive subjects).

4.3.2 Rheumatoid Arthritis (RA)

RA dataset contains a total of 1,193,150 unique TCRβ sequences, and the average number of unique TCRβs for each subject is 14639 (9213 s.d.).

Figure 4.6 and Figure 4.7 present the mean occurrence frequency of each V gene family and that of the 10 V genes with the highest values respectively. We can see that V05, V06 and V07 are the gene families with the highest mean frequencies, which is the same as the CMV case.

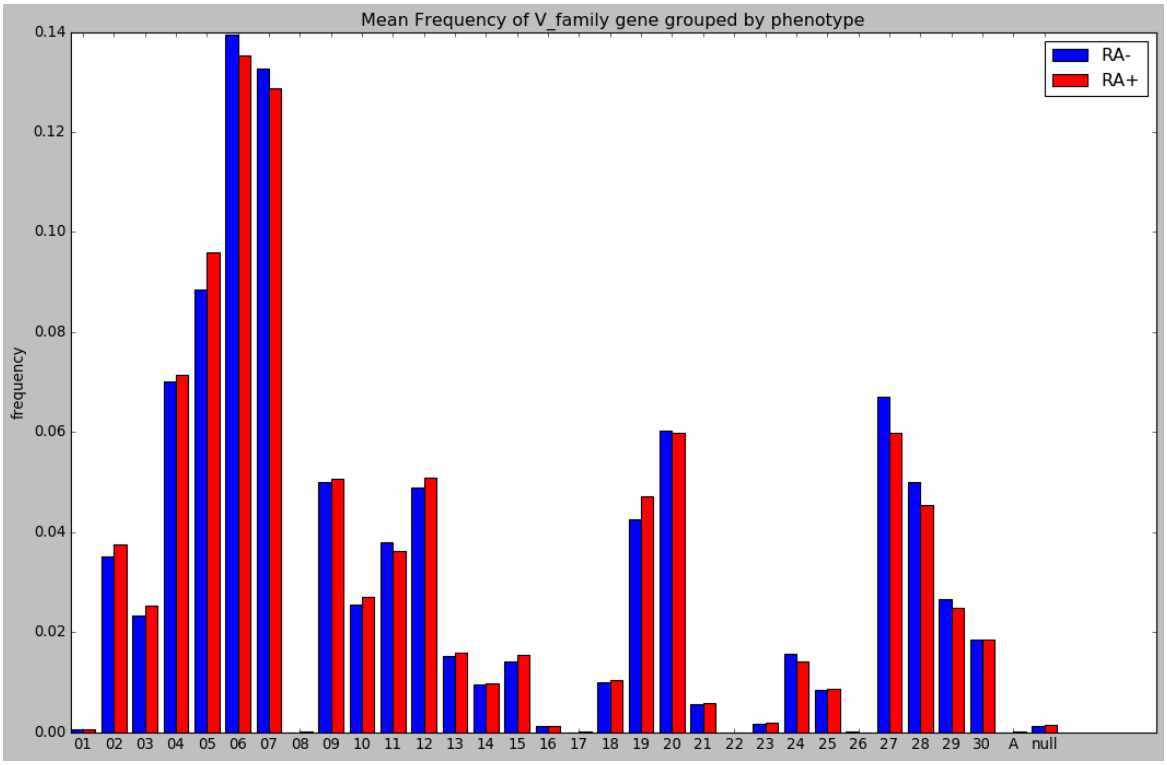


Figure 4.6: Mean frequencies of V gene families on the RA dataset

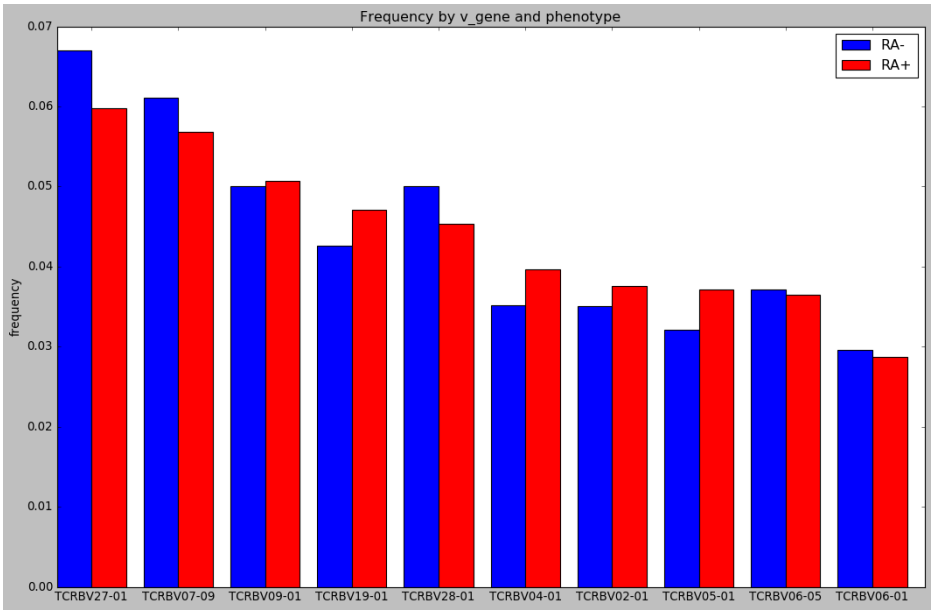


Figure 4.7: Mean frequencies of V genes on the RA dataset

Performing the one-tailed t-test on the V gene families and these 10 V genes, we can see that the V03, V5, V19 and V23 gene families, and the V19.01, V04.01, V02.01 and V05.01 genes show statistical differences between the positive class and the negative class.

4.4 Results

In this section, we present the experiment results. The applied machine learning algorithms consist of the Beta-binomal model (Baseline), Logistic regression, Random forest and LightGBM. Abbreviations of them in this section are: 'LR' - Logistic regression, 'RF'- Random forest, 'LightGBM' - Light Gradient Boosting Machine.

4.4.1 CMV

This section presents the results on the CMV dataset. When implementing the baseline model, we used the LOOCV AUROC to evaluate the performance on the training set. When applying other algorithms, the 10-fold CV AUROC is evaluated. The AUROC score and the Accuracy score are the metrics used on the testing set.

Baseline model

Firstly, we study the case where the TCRβ chains enriched in positive samples are the target features. The approximately optimal p-value threshold is obtained by minimizing LOOCV log loss (combining all the test point outputs to calculate a log loss) using the MLE estimator.

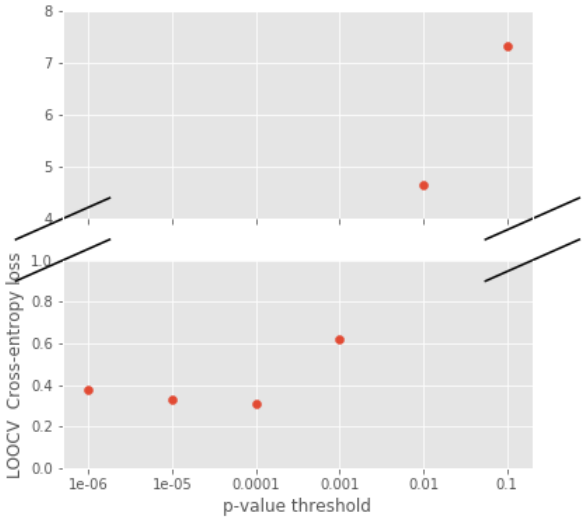
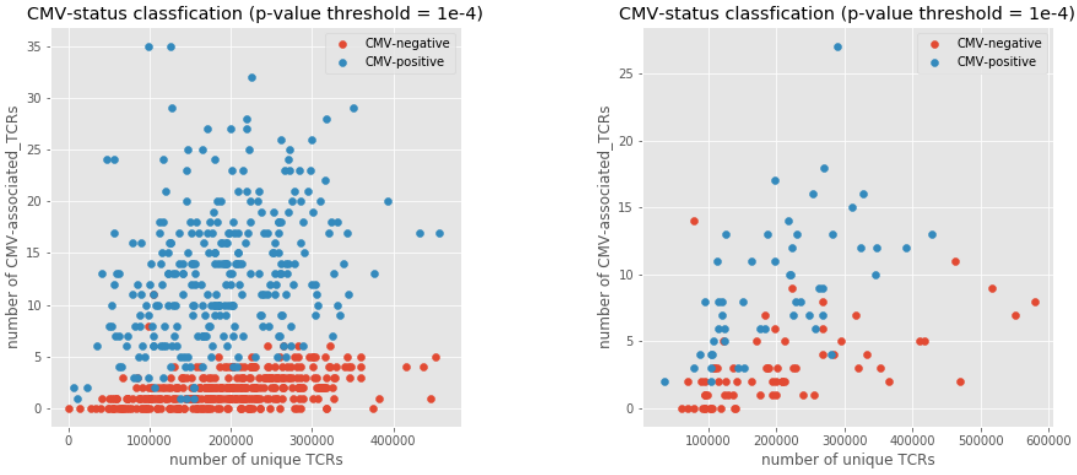


Figure 4.8: LOOCV log loss of p-value threshold candidates

From Figure 4.8, we obtain that the optimal threshold is 1e-4. Then we can draw the scatterplot that shows the relationship between the amount of CMV-associated TCRβs and the total amount of unique TCRβs, as Figure 4.9 shown. It presents the distribution in the training set and the testing set separately, and we can see that the subjects of different classes are approximately linear separable.



(a) Training set

(b) Testing set

Figure 4.9: Scatterplot - the relationship between the amount of CMV-associated TCRβs and the total amount of unique TCRβs

The table 4.2 shows the results of using different estimators or different parameter initialization methods. 'MLE(Function1)' is the MLE estimator with the priors

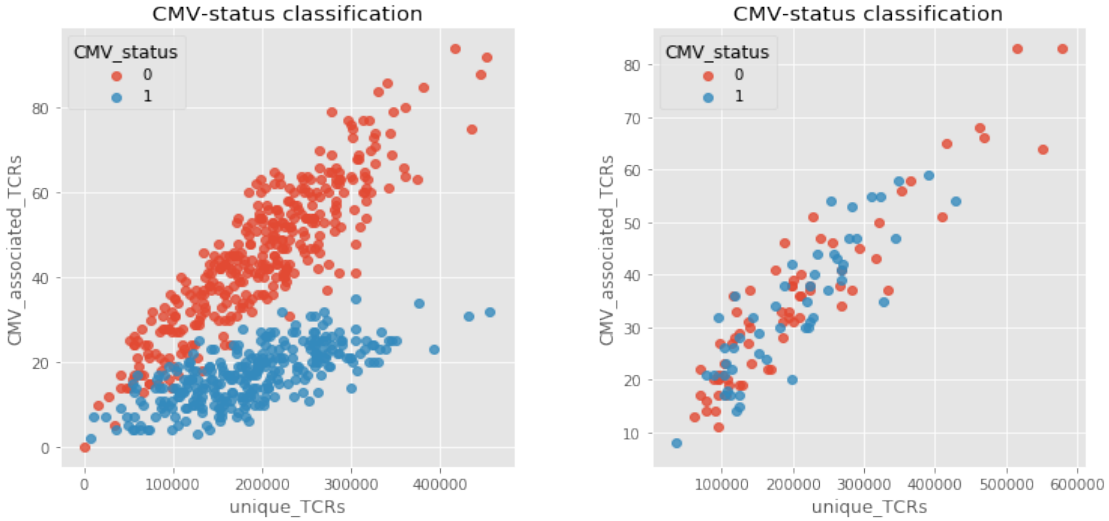
initialized by Function1, 'MLE(Function2)' is the MLE estimator with the priors initialized by Function2, 'MOM' is the MOM estimator, and 'VGLM' is the VGLM model fitted using VGAM package. 'LOOCV AUROC' denotes the LOOCV AUROC score on the training set, 'AUROC', 'Accuracy', 'Sensitivity' and 'Specificity' are the results on the testing set. The parameters of the model fitting on the training set are also shown.

	MLE(Function1)	MLE(Function2)	VGLM	MOM
LOOCV AUROC	0.943	0.943	0.944	0.944
AUROC	0.940	0.940	0.939	0.940
Accuracy	0.883	0.883	0.892	0.892
Sensitivity	0.882	0.882	0.902	0.902
Specificity	0.884	0.884	0.884	0.884
$(\alpha_{c0}, \beta_{c_0})$	(30.8, 3242047.2)	(31.2,3294640.2)	(32.1, 3380430.7)	(5.7, 601748.9)
$(\alpha_{c1}, \beta_{c_1})$	(5.8, 78523.6)	(5.8,78530.3)	(4.0, 51767.5)	(2.4, 30896.7)

Table 4.2: Comparison of the baseline models - CMV

Comparing the results, we can see that different estimators or different MOM initialization methods can yield similar scores, even though the parameters of the fitted models are possibly different.

We also apply the baseline model where using the Fisher exact test to identify the TCRβs that occur more frequently in negative subjects. When the p-value threshold is set as 10^{-3}, there is a separation line between different classes on the training set, but there is no obvious difference in the testing set. Figure 4.10 shows the scatterplot of the class distribution. The AUROC score on the testing set is 0.520. The result shows that this converse way can not make predictions about the phenotype statuses. And from the perspective of immunology, the result matches the TCR mechanism.



(a) Training set

(b) Testing set

Figure 4.10: Scatterplot - the relationship between the amount of CMV-associated TCRβs to the total amount of unique TCRβs (CMV-associated TCRs enriched in the negative samples)

Other algorithms

By performing 10-fold CV on the training set with each of the k value candidates, we got the optimal number of kept TCRβs is 150. The 150 TCRβs with the lowest p-values on the training set were selected. Then we constructed pipelines consisting a feature selection method (L1-based feature selection, Tree-based feature selection or RFE) and a model to train, and adopted Grid search to tune the parameters of the pipelines.

The tables below demonstrate the results on the TCR presence/absence data, TCR count data and TCR frequency data respectively.

Model	10-fold CV AUROC	AUROC	Accuracy	Sensitivity	Specificity
LR	0.966	0.904	0.808	0.647	0.928
RF	0.965	0.920	0.867	0.784	0.928
LightGBM	0.939	0.901	0.858	0.745	0.942

Table 4.3: Results on the TCR presence/absence data

Model	10-fold CV AUROC	AUROC	Accuracy	Sensitivity	Specificity
LR	0.969	0.908	0.858	0.804	0.899
RF	0.964	0.930	0.858	0.824	0.884
LightGBM	0.939	0.889	0.850	0.765	0.913

Table 4.4: Results on the TCR count data

Model	10-fold CV AUROC	AUROC	Accuracy	Sensitivity	Specificity
LR	0.973	0.926	0.833	0.627	0.986
RF	0.966	0.947	0.900	0.824	0.957
LightGBM	0.933	0.902	0.833	0.686	0.942

Table 4.5: Results on the TCR frequency data

From the results, we can see that all the algorithms can make moderate predictions. Random forest performs best, while LightGBM is worse compared with the other two models. The best prediction is given by the Random Forest classifier on the TCR frequency data: the 10-fold CV AUROC on the training set is 0.966, and the AUROC score on the testing set is 0.947 with an accuracy score of 0.900. Compared with the baseline model, the other models tend to produce lower values of sensitivity and higher values of specificity. And it is likely that their generalization abilities are not better than the baseline model since they yielded higher CV AUROC on the training set but lower values of AUROC on the testing set.

4.4.2 RA

This section shows the results conducting experiments on the RA data. The LOOCV AUROC score is used to measure the prediction performance. Compared with the case of making predictions on the CMV dataset, the prediction using RA dataset is much more difficult because of the smaller sample size and the much smaller amount of unique TCRβ chains.

Baseline Model

We selected a set of values from 0.1 to 0.5 as the p-value candidates. When the p-value threshold is set as some certain candidates, one of the following conditions may happen: the VGLM model could not be fitted by the VGAM package, the parameters of the fitted model were extremely large, the prior initialization functions possibly gave negative estimate of the model parameters, and the MOM estimator was also possible to yield a fitted model with negative parameters. Excluding those p-value candidates that make the model fit fail or the parameters of the fitted model improper, we chose 0.2 that produced the lowest LOOCV log loss score as a suitable threshold to use. The LOOCV AUROC scores are shown in Table 4.6.

Model	AUROC
MLE(Function1)	0.632
MLE(Function2)	0.620
VGLM	0.615
MOM	0.632

Table 4.6: Comparison of the baseline models - RA

Other algorithms

The other algorithms are not as good as the baseline model. Trying to perform feature selection, including filtering by Fisher exact test and using Lasso model, does not help to improve the prediction performance, even making the scores worse. The LOOCV AUROC score of logistic regression is lower than 0.5 for all the three versions of data, and the trained Random forest model classifies all the samples as positive, which is meaningless for our purpose to predict unseen samples. The best model is LightGBM without any feature selection. It gives LOOCV AUROC scores 0.709, 0.715 and 0.705 on the binary data, count data and frequency data respectively.

5 Discussion and Conclusion

The adaptive immune system generates immunological memory, and an individual's TCR repertotie can reveal his history of exposing pathogens. Thus it is likely to identity certain immunological phenotypes of an individual by his TCR repertoire. In this thesis, we attempted to predict immunological phenotypes based on immunosequencing TCRβ data using machine learning. Two datasets, the CMV dataset and the RA dataset, are studied in our work. We preprocessed the data in three different ways and tried various feature selection methods to identify TCRβs related to positive phenotypes. Then we applied different machine learning models to make predictions. A Beta-binomial model is built as the baseline model, and the other models we used are Linear regression, Random forest and LightGBM. We also validated the immunological foundation of our work by showing that we could not predict the target phenotypes by identifying TCRβs associated with negative phenotypes and then fitting the baseline model.

Our approaches show different performance on the two datasets we used. The CMV dataset contains a massive number of TCRβ sequences (over 89 million) and has a relatively large sample size (666 traing subjects and 120 testing subjects). In this case, the baseline model shows an excellent classification power. Its cross-validation result and its performance on the testing set are very similar, showing that the cross-validation strategy is a reasonable estimation of model performance. In addition, we applied different methods to fit the baseline model. The good estimation scores given by the MLE estimator show the feasibility of our methods of prior initialization, and the MOM estimator also yields a similar prediction result as the MLE estimator. Even though the MLE estimator and the MOM estimator have different model parameters, their model expectations $\mu = \frac{\alpha}{\alpha + \beta}$ are similar. The other algorithms also provided good classifications. They often yielded higher cross-validation AUROC scores but in the most of cases, their generalization abilities on unseen data are not as good as that of the baseline model. The Random forest trained on the frequency data outperformed the baseline model.

However, the case of the RA dataset is very different, and it is much harder to make predictions in this case. The depth of the immunosequenced TCR data is much lower (around 1.19 million), and the size of the samples is relatively small (85 subjects). All the models could not get results as good as those of the CMV dataset. We could not fit the baseline model under some certain p-value thresholds. The Logistic regression model gives LOOCV AUROC scores lower than 0.5, the Random forest model classifies all the samples as the positives. The baseline model The LightGBM model has the best classification performance, with the highest socre 0.715 on the count data.

The experiments and the experiment results indicates that the sample size and depth of immunosequencing affect the performance of classifying phenotypes based on TCR repertoire. We can make a good prediction on the CMV dataset, but it is

more difficult to learn from the RA dataset because of its insufficiency in the sample size and depth of immunosequencing. The lowest p-value of the TCR in the RA dataset is 0.06 (>0.05), and such a relatively high p-value can not give the conclusion that the TCR is associated with the phenotype status. To make a reliable prediction, we need deeper TCR sequencing, or maybe also more peripheral blood samples. If the dataset is big enough as the CMV data, the TCR repertoire could provide a sensitive and specific diagnostic. Another problem of the RA dataset is the class imbalance. There are only around 23.5% negative samples, therefore some classifiers are likely to predict everything as positive (the majority class).

The results also show that the baseline model can give a relatively good performance in both the CMV dataset and the RA dataset, and thus we can expect that it has the potential to be generalized to the other datasets. Random forest performed best on the CMV dataset but could not classify the subjects in the RA dataset into two different classes, while the LightGBM could learn the RA dataset best but could not predict the CMV dataset as well as the other algorthms. Except for the baseline model, we have not found another one model that outperform other algorithms on all the datasets.

The work for the project could still be further improved from multiple aspects. The functions we used to initialize the parameters of the baseline model are not so proper from the point of theory, and the more reasonable functions to perform prior initialization can be explored in the future. And in addition to the RFE and embedded methods, some other feature selection techniques also can be tried to see whether they could select features more efficiently. Another technique that has the potential to improve the prediction performance is feature engineering which we did little work on. As the data analysis conducted in Section 4.3, several V gene families and V genes show sample mean differences between different phenotype statuses, and therefore it is possible that they can help to identify phenotype statuses. Another idea is that multiple TCRs are likely to proliferate at the same time when the phenotype status is positive, and the higher frequency of their combination indicates the higher probability that the sample is positive. Combining them as new features may be helpful to improve the prediction accuracy. Adding new features related to the CDR3 region is also possible to make an improvement. CDR3 region is the most diverse element of the TCR, and it is the region that is the most relevant with antigen recognition. It is possible that some properties of the CDR3 region are closely related to phenotype statuses.

References

[1] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P. *Molecular Biology of the Cell.* 5. painos. New York: Garland, 2008. pp. 1539–1589. ISBN 978-0-8153-4106-2.

[2] Vanhanen, R., Heikkilä, N., Aggarwal, K., Hamm, D., Tarkkila, H., Pätilä, T., Jokiranta, T. S., Saramäki, J., Arstila, T. P. *T cell receptor diversity in the human thymus* Molecular Immunology, 2016. Vol. 76. pp. 116-122.

[3] Arstila, T. P., Casrouge, A., Baron, V., Even, J., Kanellopoulos, J., Kourilsky, P. *A Direct Estimate of the Human $\alpha\beta$ T Cell Receptor Diversity.* SCIENCE, 1999. Vol. 286. pp. 958–961.

[4] Zhao, Y., Nguyen, P., Vogel, P., Li, B., Jones, L. L., Geiger T. L. *Autoimmune susceptibility imposed by public TCR chains.* Scientific Reports, 2016. Vol. 6:37543. ISSN 2045-2322. DOI: 10.1038/srep37543.

[5] Emerson, R. O., DeWitt, W. S., Vignali, M., Gravley, J., Hu, J. K., Osborne, E. J., Desmarais, C., Klinger, M., Carlson, C. S., Hansen, J. A., Rieder, M., Robins, H. S. *Immunosequencing identifies signatures of cytomegalovirus exposure history and HLA-mediated effects on the T cell repertoire.* Nature Genetics, 2017. Vol. 49:5. pp. 659–665. ISSN 1061-4036. DOI:10.1038/ng.3822.

[6] DeWitt, W. S., Smith, A., Schoch, G., Hansen, J. A., Matsen, F. A., Bradley, P. H. *Human T cell receptor occurrence patterns encode immune history, genetic background, and receptor specificity.* eLife 2018; 7:e38358. DOI: 10.7554/eLife.38358.

[7] Nieuwenhove, E. V., Lagou, V., Eyck, L. V., Dooley, J., Bodenhofer, U., Goris, A., Baron, S. H., Wouters, C., Liston, A. *Machine learning identifies the immunological signature of Juvenile Idiopathic Arthritis.* bioRxiv, 2018. DOI: https://doi.org/10.1101/382499.

[8] Larran*ñ*aga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J. A., Armañanzas, R., Santaf*é*, G., P*é*rez, A., Robles, V. *Machine learning in bioinformatics.* Briefings in bioinformatics, 2005. Vol. 7. NO 1. 86-112. DOI:10.1093/bib/bbk007.

[9] Salzberg, S. *Localing protein coding regions in human DNA using a decision tree algorithm.* JComput Biol 1995;2: 473–85.

[10] Castelo, R, Guig*ó*, R. *Splice site identification by idlBNs.* Bioinformatics 2004;20(Suppl. 1):i69–76.

[11] Wang, S., Peng, J., Ma, J., Xu, J. *Protein secondary structure prediction using deep convolutional neural fields.* Scientific Reports, 2015. 6: 18962. DOI: 10.1038/srep18962.

[12] Kindt, T. J., Goldsby, R. A., Osborne, A. B., Kuby, J. *Kuby immunology.* 6 ed. W.H. Freeman, New York, 2007.

[13] Attaf, M., Huseby, E., Sewell, A. K. *$\alpha\beta$ T cell receptors as predictors of health and disease.* Cellular Molecular Immunology, 2015. Vol. 12, pp. 391–399.

[14] Roth, D. B. *V(D)J Recombination: Mechanism, Errors, and Fidelity.* Microbiol Spectr, 2014. Vol. 2(6). doi: 10.1128/microbiolspec.MDNA3-0041-2014.

[15] Sewell, A. K. *Why must T cells be cross-reactive?* Nature Reviews Immunology, 2012. Vol. 12:9. pp. 669–677. ISSN 1474-1733. DOI: 10.1038/nri3279.

[16] Murphy, K. *Janeway's Immunobiology (8th ed.).* Garland Science, 2012. ISBN: 978-0-8153-4243-4.

[17] Xu, J. L., Davis, M. M. *Diversity in the CDR3 Region of V_H Is Sufficient for Most Antibody Specificities.* Immunity, 2000. Vol. 13, pp. 37–45.

[18] Mitchell, T. M. *Machine Learning.* McGraw-Hill, Inc., New York, NY, USA, 1997.

[19] Marsland, S. *Machine Learning - An Algorithmic Perspective, second edition.* CRC Press, 2015. ISBN: 978-1-4665-8333-7.

[20] Zhou, Z. *Machine Learning.* Tsinghua University Press, 2016.

[21] Russell, S. J., Norvig, P. *Artificial Intelligence: A Modern Approach, (3rd Edition).* Pearson Education, Inc., 2009. ISBN-13: 978-0-13-604259-4.

[22] Trunk, G. V. *A Problem of Dimensionality: A Simple Example.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1979. PAMI-1 (3): 306–307. DOI:10.1109/TPAMI.1979.4766926.

[23] Fawcett, T. *An introduction to ROC analysis.* Pattern Recognition Letters, 2006. 27, 861–874.

[24] Bewick, V., Cheek, L, Ball, J. *Statistics review 13: Receiver operating characteristic curves.* Critical Care , 2004. Vol. 8, No. 6. pp. 508-512.

[25] Kennet-Cohen, T., Bronner, S., Ben-Simon, A., Intrator, N. *Different Approaches for Combining Scores on a Test Battery for the Diagnosis of Learning Disabilities.* ResearchGate, 2008. ISBN:965-502-144-0.

[26] Raschka, S. & Mirjalili, V. *Python Machine Learning (2nd Edition).* Packt Publishing, 2017. ISBN: 978-1-78712-593-3.

[27] Richert, W. & Coelho, L. P. *Building Machine Learning System with Python.* Packet Publishing, 2013. ISBN: 978-1-78216-140-0.

[28] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., Rubin, D. B. *Bayesian Data Analysis (3rd Edition).* Chapman and Hall/CRC, 2013. ISBN-13: 978-1-4398-9820-8.

[29] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Edition).* Springer, 2016. ISBN-13: 978-0387848570.

[30] Bishop, C. M. *Pattern Recognition and Machine Learning.* Springer, 2006. ISBN-13: 978-0387-31073-2.

[31] Alpaydin, E. *Introduction to machine learning (3rd edition).* The MIT Press, 2014. ISBN: 978-0-262-02818-9.

[32] Han, H., Guo, X., Yu, H. *Variable Selection Using Mean Decrease Accuracy And Mean Decrease Gini Based on Random Forest.* 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016. pp. 219-224. DOI: 10.1109/ICSESS.2016.7883053.

[33] H. Pishro-Nik, *Introduction to probability, statistics, and random processes.* available at https://www.probabilitycourse.com, Kappa Research LLC, 2014.

[34] Myung, I. J. *Tutorial on maximum likelihood estimation.* Journal of Mathematical Psychology, 2003. Vol. 47, Issue 1, pp. 90-100.

[35] SCHUCKERS, M. E. *Using the Beta-binomial distribution to assess performance of a biometric identification device.* International Journal of Image and Graphics, 2003. vol. 3, no. 3.

[36] Bassett, R. & Deride, J. *Maximum a posteriori estimators as a limit of Bayes estimators.* Math. Program., 2018. https://doi.org/10.1007/s10107-018-1241-0.

[37] Tumer, K. & Ghosh, J. *Estimating the Bayes Error Rate through Classifier Combining.* Proceedings of the 13th International Conference on Pattern Recognition, Vol. 2, pp.695–699.

[38] Fukunaga, K. *Introduction to Statistical Pattern Recognition.* Academic Press Professional, Inc., 1990. ISBN 978-0-12-269851-4.

[39] Lee, J. *Classification using Bayes Rule in 1-dimensional and N-dimensional feature spaces.* https://www.projectrhea.org/rhea/images/3/33/Bayes_rule_by_jihwan.pdf

[40] Goodfellow, I., Bengio, Y., Courville, A. *Deep Learning.* MIT Press, 2016. https://www.deeplearningbook.org/. pp.114.

[41] Walpole, R. E., Myers, R. H., Myers, S. L., Ye, K. *Probability Statistics for Engineers Scientists (9th Edition).* Pearson, 2012. ISBN 13: 978-0-321-62911-1.

[42] Hamilton Institute. *The Binomial Distribution* October 20, 2010.

[43] Chapelle, O., Chang, Y. *Yahoo! learning to rank challenge overview.* JMLR: Workshop and Conference Proceedings, 2011.

[44] Frery, J., Habrard, A., Sebban, M., Caelen, O., He-Guelton, L. *Efficient Top Rank Optimization with Gradient Boosting for Supervised Anomaly Detection.* Machine Learning and Knowledge Discovery in Databases, 2017. DOI: https://doi.org/10.1007/978-3-319-71249-9_2.

[45] Rogojan, B. *Boosting and Bagging: How To Develop A Robust Machine Learning Algorithm.* https://hackernoon.com/how-to-develop-a-robust-algorithm-c38e08f32201

[46] Friedman, J. H. *Stochastic gradient boosting.* Computational Statistics Data Analysis, 2002. Vol. 38. pp. 367–378.

[47] Chen, T., Guestrin, C. *XGBoost: A Scalable Tree Boosting System.* 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, 2016.

[48] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157.

[49] Chandrashekar, G. & Sahin, F. *A survey on feature selection methods.* Computers and Electrical Engineering, 2014. Vol. 40, Issue 1, pp. 16-28.

[50] McDonald, J.H. *Handbook of Biological Statistics (3rd ed.).* Sparky House Publishing, Baltimore, Maryland, 2014. pp. 77-85.

[51] Weisstein, Eric W. *Fisher's Exact Test.* MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/FishersExactTest.html.

[52] Gholami, B., Norton, I., Tannenbaum, A. R., Agar, N. Y. *Recursive feature elimination for brain tumor classification using desorption electrospray ionization mass spectrometry imaging.* Conf Proc IEEE Eng Med Biol Soc, 2012. 2012:5258-5261. doi: 10.1109/EMBC.2012.6347180.

[53] Claesen, M. & Moor, B. De. *Hyperparameter Search in Machine Learning.* The XI Metaheuristics International Conference, 2015.

[54] Rasmussen, C. E. & Williams, C. K. I. "Model Selection and Adaptation of Hyperparameters." *Gaussian Processes for Machine Learning.* the MIT Press, 2006. ISBN: 026218253X.

[55] Bergstra, J., Bengio, Y. *Random Search for Hyper-Parameter Optimization.* Journal of Machine Learning Research, 2012. Vol. 13, pp. 281-305.

[56] Rosati, E., Dowds, C. M., Liaskou, E., Henriksen, E., Karlsen, T. H., Franke, A. *Overview of methodologies for T-cell receptor repertoire analysis.* BMC biotechnology, 2017. 17(1), 61. DOI:10.1186/s12896-017-0379-9.

[57] Savola, P., Kelkka, T., Rajala, H.L., Kuuliala, A., Kuuliala, K., Eldfors, S., Ellonen, P., Lagstro, S., Lepisto, M., Hannunen, T., Andersson, E.I., Khajuria, R.K., Jaatinen, T., Koivuniemi, R., Repo, H., Saarela, J., Porkka1, K., Leirisalo-Repo, M. Mustjoki, S. *Somatic mutations in clonally expanded cytotoxic T lymphocytes in patients with newly diagnosed rheumatoid arthritis.* Nature Communications, 2017. 8:15869. DOI: 10.1038/ncomms15869.

[58] Raulet, D. H., Garman, R. D., Saito, H., Tonegawa, S. *Developmental regulation of T cell receptor gene expression.* Nature (Lond) 1985;314:103–107.

[59] Snodgrass, H. R., Dembic, Z., Steinmetz, M., Boehmer, v. H. *Expression of T-cell antigen receptor genes during fetal development in the thymus.* Nature (Lond) 1985;315:232–233.

[60] Keane, C., Gould, C., Jones, K., Hamm, D., Talaulikar, D., Ellis, J., Vari, F., Birch, S., Han, E., Wood, P., Le-Cao, KA., Green, M. R., Crooks, P., Jain, S., Tobin, J., Steptoe, R. J., Gandhi, M. K. *The T-cell Receptor Repertoire Influences the Tumor Microenvironment and Is Associated with Survival in Aggressive B-cell Lymphoma.* Clinical Cancer Research, 2017. 23(7):1820-1828. DOI: 10.1158/1078-0432.CCR-16-1576.

[61] Basha, S. M., Rajput, D. S., Vandhan, V. *Impact of Gradient Ascent and Boosting Algorithm in Classification.* International Journal of Intelligent Engineering & Systems, 2018. Vol.11, No.1, pp. 41-49. DOI: 10.22266/ijies2018.0228.05.

[62] Siegrist, K. *Random: Probability, Mathematical Statistics, Stochastic Processes.* Chapter 6, Topic 2. http://www.randomservices.org/random/point/Moments.html.

[63] Young-Xu, Y., Chan, K. A. *Pooling overdispersed binomial data to estimate event rate.* BMC medical research methodology, 2008. Vol. 8 58. DOI: 10.1186/1471-2288-8-58.

[64] *Beta-Binomial Empirical Bayes.* 2015. http://www.probabilaball.com/2015/05/beta-binomial-empirical-bayes.html.

[65] Atwood, C. L. *Estimators for the truncated beta-binomial distribution.* Institute of Mathematical Statistics conference, 1980. https://www.osti.gov/servlets/purl/5359803.

[66] McIntosh, A. I. *The Jackknife Estimation Method.* http://people.bu.edu/aimcinto/jackknife.pdf

[67] Yee, T. W. *Vector Generalized Linear and Additive Models: With an Implementation in R.* Springer, 2015. ISBN: 978-1-4939-2818-7.

[68] Yee, T. W. *The VGAM package for R.* https://www.stat.auckland.ac.nz/~yee/VGAM/.

[69] Welch, B. L. *The generalization of "Student's" problem when several different population variances are involved.* Biometrika, 1947. Vol. 34, Issue 1–2, pp. 28–35. https://doi.org/10.1093/biomet/34.1-2.28.

A Prior initialization functions

```
import numpy as np
import statistics


def estimate_beta_params_MOM_arr(arr):
    """Estimating Beta binomial parameters by MOM (input is an array)
    """
    K = np.sum(arr[:,1])
    N = np.sum(arr[:,0])
    N2 = np.sum(arr[:,0]**2)

    # (theta)^ = alpha/(alpha+beta)
    theta = K/N
    p = arr[:,1]/arr[:,0]
    K_var = np.sum((arr[:,0]**2)*((p-theta)**2))

    temp = (N2/N)-1
    temp2 = N*theta*(1-theta)
    temp3 = K_var-N*theta*(1-theta)

    alpha = theta*temp*(temp2/temp3)
    beta = (1-theta)*alpha/theta

    if alpha < 0 or beta < 0:
        alpha, beta = 0, 0

    return [alpha, beta]


def Function1(train):
    train_c0 = train[train['phenotype_status']==0]
    train_c1 = train[train['phenotype_status']==1]

    # array of [n, k] of the negative class
    c0_arr = train_c0[['unique_TCRs','phenotype_associated_TCRs']].values

    # array of [n, k] of the positive class
    c1_arr = train_c1[['unique_TCRs','phenotype_associated_TCRs']].values

    init_params_c0 = estimate_beta_params_MOM_arr(c0_arr)
    init_params_c1 = estimate_beta_params_MOM_arr(c1_arr)

    return [init_params_c0, init_params_c1]
```

```
def Function2(train):
    train_c0 = train[train['phenotype_status']==0]
    train_c1 = train[train['phenotype_status']==1]
    n_c0 = train_c0.shape[0]
    n_c1 = train_c1.shape[0]

    # array of [n, k] of the negative class
    c0_arr = train_c0[['unique_TCRs','phenotype_associated_TCRs']].values

    # array of [n, k] of the positive class
    c1_arr = train_c1[['unique_TCRs','phenotype_associated_TCRs']].values

    jackknifing_params_c0 = list()
    jackknifing_params_c1 = list()

    for i in range(n_c0):
        i_c0_arr = np.delete(c0_arr, i, 0)
        init_params_c0 = estimate_beta_params_MOM_arr(i_c0_arr)
        jackknifing_params_c0.append(init_params_c0)

    for i in range(n_c1):
        i_c1_arr = np.delete(c1_arr, i, 0)
        init_params_c1 = estimate_beta_params_MOM_arr(i_c1_arr)
        jackknifing_params_c1.append(init_params_c1)

    init_params_c0 = np.mean(jackknifing_params_c0, axis=0).tolist()
    init_params_c1 = np.mean(jackknifing_params_c1, axis=0).tolist()

    return [init_params_c0, init_params_c1]
```