Imperial College London

Department of Computing

MEng Individual Project (JMC)

Feature selection methods in the context of epigenetics and schizophrenia

Author: Daren Sin

Supervisor: Dr. Panos Parpas

Second marker: Dr. Ruth Misener

June 21, 2017

Abstract

Schizophrenia is a mental disorder that is commonly perceived to be hereditary. However, there have been cases where patients with schizophrenia do not have a family member with the disorder. This hints that the epigenetics of an individual might have a significant role to play in causing an individual to have schizophrenia or other mental disorders. This project investigates an epigenetics data set and attempts to find any regularity or trend that might help us understand the relationship between epigenetics and schizophrenia.

In particular, this data set has about 400000 features, and understanding this data set will require feature selection to select only the relevant features. This project explores several feature selection algorithms and looks into their effectiveness in selecting the useful features from this data set.

This project also focuses on the maximum-relevance minimum-multicollinearity (MRmMC) algorithm and points out several of its limitations when it is applied to the epigenetics data set. We then attempt to extend the MRmMC algorithm by incorporating genetic algorithms to avoid these limitations.

Through numerical experiments, we show that incorporating genetic algorithms with MRmMC is able to overcome its limitations when applied to the epigenetics data set. Despite this achievement, we also recognise some flaws in this method.

By performing several feature selection algorithms on the epigenetics data set, we were able to identify some features that were selected by most of the algorithms.

This project concludes with some extensions that might enhance the performance of the feature selection process.

Acknowledgements

I would like to thank:

- Dr. Panos Parpas, for his helpful tips and advice which have guided this project in the right direction.
- Dr. Karim Malki, for his fervent help and guidance in this project. His expertise in epigenetics and biostatistics has also been tremendously valuable.
- Dr. Ruth Misener, for her constructive feedback, especially in the initial stage of the project.
- The High Performance Computing (HPC) team at Imperial College, for providing prompt and valuable assistance in using the HPC facilities.
- Dr. Fariba Sadri, my personal tutor, for her patient guidance and sagely advice, especially in my last year of studies.
- The Singapore community at Imperial College, for making my time at Imperial so memorable.
- My friends, family and course mates, for their support throughout my 4 years of education at Imperial.

Contents

Chapter 1

Introduction

Overall, this project aims to apply feature selection algorithms and supervised machine learning on epigenetics data to investigate any association between schizophrenia and epigenetics.

1.1 Schizophrenia and epigenetics

Schizophrenia is a complex mental disorder that displays an array of symptoms, including hallucination and depression. While schizophrenia is commonly perceived as a hereditary disease, there have been cases where individuals diagnosed with schizophrenia do not have a family member with the disorder [1].

There is also a strong indication that, besides genetic factors, environmental factors such as tobacco smoke and one's diet also have an influence in the development of psychiatric disorders [1, 2, 3]. This results in a hypothesis that *epigenetics* (section 2.2) plays a role in the development of schizophrenia [4]. This hypothesis forms the context of this project.

1.2 Using machine learning to predict schizophrenia

This project uses data from a recent study on epigenetics and schizophrenia (section 2.3) that distinguishes individuals diagnosed with schizophrenia (case) from those who do not (control). This project aims to find any statistical regularity in the data. Any insight on the data might help geneticists and psychiatrists better understand the relationship between epigenetics and schizophrenia.

Previous work on using supervised machine learning on biological data (section 2.4) has often been plagued with the *curse of dimensionality*, where the number of biological samples is far smaller than the number of features (or dimensions). This is commonly known as the $p \gg n$ problem [5]. In this project, the data faces a similar challenge with 847 samples and 420374 features, resulting in about 2 gigabytes of data. This also leads to other issues such as *overfitting*.

Besides the high dimensionality faced by the data, we also encounter another complication: not all of the features in the data play a part in the classification of the disorder. In other words, some features may only contribute a little to the classification, while some features might be redundant.

1.3 Contributions

The aforementioned difficulties inherent in the data set make *feature selection* an important preprocessing step to select only the features that have significant contribution to the classification outcome. This project focuses on the feature selection process and sees which, if any, feature selection algorithm is beneficial in analysing the data set.

Furthermore, a paper recently published by Senawi *et. al* [6] proposes a new method, *Maximum Relevance Minimum Multicollinearity* (MRmMC), to quantify the relevance and redundancy (section 2.7.7) of the features in a data set. This project will pay particular attention to this algorithm.

Overall, the primary contributions of this project include:

- Comparing the merits and drawbacks of some feature selection methods that are commonly used in the feature selection literature (section 2.7),
- Identifying ways to understand and process the epigenetics data (section 3.1),
- Identifying several shortcomings of the MRmMC method with respect to the epigenetics data set (section 4.5),
- Investigating if incorporating a genetic algorithm will enhance the performance of MRmMC with respect to the data set (section 4.6),
- Comparing the performance of MRmMC with the other feature selection algorithms discussed in this project (chapter 5),
- Identifying features in the epigenetics data set that are frequently selected by the feature selection algorithms (section 5.6).

Although epigenetics and schizophrenia form the motivation for this project, we will primarily describe and evaluate feature selection methods, paying particular attention to MRmMC.

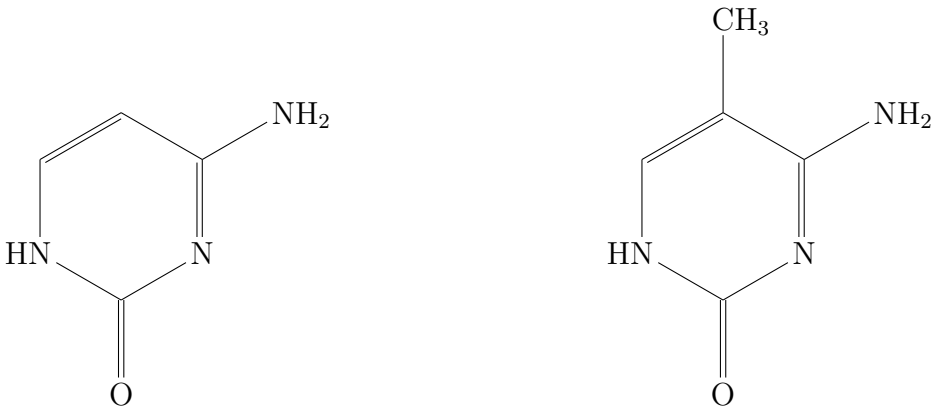
Chapter 2

Background

2.1 Molecular biology and definitions

This section outlines the biological terms that will be relevant to this project [7, 8, 9, 10].

- **DNA:** Deoxyribonucleic Acid, also known as DNA, is a molecule that contains all the hereditary material in all living things. It serves as the fundamental unit of heredity.
- **DNA bases:** The hereditary information stored in DNA molecules are made up of four bases - Adenine (A), Thymine (T), Cytosine (C) and Guanine (G). These bases pair up in a specific way: A with T and C with G. Along with other types of molecules, these pairs form a nucleotide. Nucleotides are arranged in a double helix structure.
- **Genes:** A gene is the fundamental building block of heredity. Genes consist of DNA, and encode instructions to produce proteins. These instructions are used to produce proteins through the process of transcription and translation. These processes form the *central dogma of molecular biology.*
- **Gene expression:** Gene expressions behave like a "switch" to determine when and what kind of proteins are produced by cells. All cells in a human being carry the same genome. Thus, gene expression allows cells to specialise into different functionalities (e.g. brain cell and skin cell).
- **Epigenome:** The epigenome is a set of chemical compounds and modifications that can alter the genome, and thus alter DNA and the proteins that it produces. The epigenome can thus change the "on/off" action in gene expression and control the production of proteins.
- **DNA methylation:** A common chemical modification is DNA methylation, where methyl groups ($-CH_3$) are attached to the bases of DNA at a specific place of cytosine (see Figure 2.1). These methyl groups switch off the gene to which they are attached in the DNA, and thus no protein can be generated from that gene. However, despite this chemical modification, an individual's underlying genetic sequence remains unchanged [11].



(a) DNA base Cytosine.

(b) DNA base 5-methylcytosine, with an additional $-CH_3$ group attached to one of its atoms.

Figure 2.1: Chemical representation of how DNA methylation modifies the DNA base cytosine on the molecular level.

- **CpG island:** DNA methylations mostly occur in CpG islands, which are areas in the DNA where a cytosine molecule is next to a guanine molecule.

2.2 Epigenetics

As Nessa Carey puts it in *The Epigenetics Revolution*, epigenetics is manifested when "*two genetically identical individuals are non-identical in some way we can measure*" [12]. Epigenetics provides a "window" which allows one's external environment to influence his or her DNA.

According to Carey, monozygotic (identical) twins have the same genetic material, but if one twin has schizophrenia, the other twin has a 50% chance of being diagnosed with schizophrenia, compared to only 0.5% to 1% in the general population. However, it begs the question: why is the other twin not diagnosed with the disease with 100% certainty, since they have the same genetic material? It is hypothesised that this is epigenetics at work, as environmental factors, not just the twins' genetic material, influence the likelihood of having schizophrenia.

Monozygotic twins also allow us to investigate the link between genotype (our genetic makeup) and phenotype (our physical characteristics) [13, 14], since monozygotic twins are genetically identical. Twin studies have also been conducted to investigate the extent to which the environment changes our epigentic makeup. In [15], Esteller *et. al* found that epigenetic modifications between twins become more disparate as they become older. Esteller *et. al* posit that this observation arises because the older twins are exposed to environmental factors for a longer period of time compared to their younger counterparts.

Similar twin studies also allow us to explore the extent to which epigenetics influences our chance of being afflicted with a disease. For example, a study [16] that focuses on monozygotic twins and their susceptibility to disease found that the genes that make up an individual cannot fully explain how likely he or she would be diagnosed with a disease.

2.3 The dataset

Similarly to the twin studies described above, this project aims to investigate the relationship between epigenetic data and the chance of being afflicted with schizophrenia. In other words, can we use one's epigenetic data to predict his or her chance of having schizophrenia?

This project makes use of data from a recent genetic-epigenetic analysis of schizophrenia, conducted in 2016 [17]. At the time of writing, there is no published work that deals with this data set.

In this study, the authors aimed to identify positions in the genome that display DNA methylation associated with environmental exposure and disease. We thus want to find out if there are statistically significant differences in DNA methylation between individuals diagnosed with schizophrenia (case) and those who were not (control). Blood samples of 847 individuals, 414 of whom were schizophrenia cases, were taken.

Furthermore, the data set quantifies the degree of DNA methylation in specific positions along the genome. These positions are CpG islands, and are labelled as `cgXXXXXXXX` (see Figure 3.1). The degree of DNA methylation in a CpG island of an individual is also recorded as a continuous value in the data set. This data set features 420374 CpG islands for 847 individuals. Thus, one can visualise the data set as a 420374×847 matrix. More details of understanding the data are discussed in chapter 3.1.

Throughout this project, we shall identify the data from this study as *the data*. The data differentiates between samples that are cases (individuals with schizophrenia) and samples that are controls (individuals without schizophrenia). These classifications are indicated through binary *target labels*.

2.4 Comparison with cancer classification

There is a significant amount of literature on cancer classification using gene expression data. These works aim to uncover biological or medical insights using biological data obtained from microarrays, which can measure the gene expression of thousands of genes simultaneously [18]. For example, using neural networks, gene expression data can be used to distinguish between tumour types, which helps in cancer diagnosis [19, 20]. [21] has even identified genes that can potentially predict cancer outcomes based on survival data. We can draw lessons from these studies to apply to this project.

What is similar about this project and previous work on cancer classification is that the data for both cases are plagued with the curse of dimensionality. For example, in cancer studies, microarrays produce data with a large number of genes (features) but a small number of samples (observations) [20]. Furthermore, only a small subset of the features are relevant for the studies, as not all features are useful in determining the type of cancer a patient has. This is regarded as biological noise [22]. As such, feature reduction on the data has to be performed to select only the relevant genes for the classification problem. Moreover, an ideal subset of features would be sparse, as we seek to eliminate the features that are redundant with respect to other features.

However, what is different between studies on psychiatric disorders and studies on cancer is that the latter is observable, such that we can ascertain that an individual has cancer using medical methods, such as conducting a blood test; it is not as obvious that an individual has a psychiatric disorder, as its symptoms might not be straightforward. For example, [23] discusses culture-related issues in diagnosing schizophrenia: "*the assessment of disorganized speech may be made difficult by linguistic variation in narrative styles across cultures*". In other words, diagnosing an individual with schizophrenia is not as "obvious".

2.5 Machine learning classifiers

Even though this project focuses on the feature selection process, the choice of the machine learning classifier is relevant as well. This section compares some popular machine learning classifiers used in the feature selection literature, and explains how some might or might not be useful for this project.

2.5.1 Decision Trees

In our context, the task is to classify the data according to whether a sample has schizophrenia or not. In other words, the classification task is binary. An intuitive solution is to use decision trees as problems with discrete output values can be solved using decision trees [24].

At each level of the tree, the intermediate node is split according to some attribute. A decision tree algorithm is capable of sorting the samples down a tree until it reaches a leaf node, where a classification is given to the node.

One variant of the decision tree algorithm is the ID3 algorithm [25]. ID3 makes use of *information gain*, to determine the attribute to classify the samples with. Using definitions from [24], let S be the set of samples that we want to classify at a particular node. The samples can be separated into two groups, those with a positive classification and those with a negative classification.

Definition 2.5.1.1 (Entropy)**.**

$$Entropy(S) = -p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)}$$

where $p_{(+)}$ and $p_{(-)}$ represents the proportion of samples with positive and negative classification respectively.

Definition 2.5.1.2 (Information gain)**.** *The information gain with respect to a set S and an attribute A is defined as:*

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where $Values(A)$ is the set of all possible values of attribute A, and $|S_v|$ is the number of elements in the set S with value v for its attribute A.

The samples in the node are classified according to the attribute with the highest information gain. Intuitively, we want to choose the attribute that gives the most distinct separation between the positive and negative classification, instead of choosing an attribute that, say, splits the samples into half according to their classification.

Although the decision tree algorithm is said to be robust to errors [24] and the resulting decision tree is easily interpreted, it might be difficult to classify samples according to features that are highly correlated. This might be a characteristic of our dataset, as we would expect some of the features to be correlated.

Furthermore, our data set consists of continuous values. We would thus need to discretise the range of real numbers of a feature into intervals, in order to compute the summation in definition 2.5.1.2. This would then give rise to another problem of defining a suitable interval for these values. This is similar to the problem of using mutual information in feature selection, which will be discussed in section 2.7.11.

2.5.2 Random forest

The Random Forest method [26] is a form of *ensemble learning* and is an extension of the decision tree algorithm. It has been used in areas such as multi-class object detection in images [27]. The Random Forest algorithm can be outlined as such:

- Split the dataset into distinct subsets.
- For each subset, train a decision tree using a decision tree algorithm, such as the ID3.
- Combine all the trees together to create a *forest*.
- Suppose we have an unseen sample $\boldsymbol{x}$. Put $\boldsymbol{x}$ through each tree, and obtain the resulting classification for each tree.
- Based on a "majority vote" system, determine the final classification of $\boldsymbol{x}$; that is, choose the classification that is the most popular among the decision trees.

Random Forest has been shown to outperform decision trees [28]. However, the limitations of decision trees described above would still be inherent in the Random Forest method, since Random Forest uses decision trees. Besides, Random Forest requires more parameters in general. For example, we would need to determine the number of trees to be trained. This would require numerical experiments.

It has also been shown that the number of trees required grows with the number of features that directly affect the classification outcome [29], and we do not know beforehand what these features are. As such, we might potentially have to train a lot of trees, which will require a lot of memory and time.

So, the decision tree and random forest methods might not be the best methods for our context, even though they are considered to be popular machine learning techniques [28].

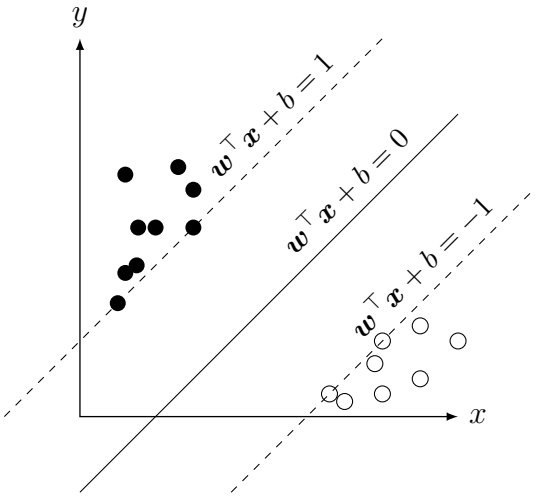


Figure 2.2: Diagram showing training points belonging to different classes (hollow versus solid dots) separated by a hyperplane $\boldsymbol{w}^\top \boldsymbol{x} + b = \pm 1$. The LaTeX script for this graph was originally created by Peng Yifan.

2.6 Support vector machines

This section explains and discusses Support Vector Machines[1]. This discussion will be limited to binary-class problems. That is, samples belong to two classes, say either class 1 or class 2.

Consider our dataset that comprises m features and n samples. Then, the data can be written as a set: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^m$ is an m dimensional vector that corresponds to the i-th sample. Moreover, for $i = 1, \ldots, n$, $y_i = \pm 1$ is the label of the i-th sample. $y_i = 1$ if the sample belongs to class 1 and $y_i = -1$ if it belongs to class 2.

Suppose we have data points that correspond to either class 1 or class 2. The Support Vector Machine (SVM) [30] uses a separating hyperplane to distinguish between data points that belong to class 1 from those in class 2. It finds a hyperplane with the largest margin between the two classes. This is shown in the graph[2] in Figure 2.2, which shows the situation of a linear hyperplane perfectly separating training points.

The hyperplane is parameterised with weight vector $\boldsymbol{w}$ and bias b. We thus solve the classification problem using:

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b \tag{2.1}$$

Finding the hyperplane with maximum margin amounts to solving the following optimi-

[1]Most of the Mathematics here is with reference to course notes from the Department of Computing, CO496 - Mathematics for Inference and Machine Learning by Prof. Stefanos Zafeiriou and Prof. Marc Deisenroth.

[2]The LaTeX script for this graph was originally created by Peng Yifan at `http://blog.pengyifan.com/tikz-example-svm-trained-with-samples-from-two-classes`.

sation problem:

$$\min_{w,b} \quad \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \tag{2.2}$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 \tag{2.3}$$

for $i = 1, \ldots, n$, where $y_i = \pm 1$, depending on the classification of the vector of features $\boldsymbol{x}_i$. Equation 2.3 is equivalent to:

$$\begin{aligned} \boldsymbol{w}^\top \boldsymbol{x}_i + b \geq 1 \quad &\text{if } y_i = 1 \\ \boldsymbol{w}^\top \boldsymbol{x}_i + b \leq -1 \quad &\text{if } y_i = -1 \end{aligned}$$

2.6.1 Dual representation

To solve the primal optimisation problem in (2.2) subject to the conditions in (2.3), we can formulate the following Lagrangian equation:

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \sum_{i=1}^{n} a_i(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1) \tag{2.4}$$

where $\boldsymbol{a}$ is the n-dimensional vector containing the Lagrangian multipliers ($a_i \geq 0$) corresponding to the inequality conditions in (2.3). The primal optimisation problem in (2.4) can be written as:

$$\min_{\boldsymbol{w},b} \max_{\boldsymbol{a} \geq 0} L(\boldsymbol{w}, b, \boldsymbol{a}) \tag{2.5}$$

This can be written as its dual equivalent:

$$\max_{\boldsymbol{a} \geq 0} \min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{a}) \tag{2.6}$$

It can be shown that:

- The equation L in (2.4) is convex, and thus any optimal solution found is guaranteed to be the global optimal solution [31].
- The primal (2.5) and dual (2.6) problems have the same optimal solutions, if any [32].

To solve the problem in (2.6), we first minimise L with respect to $\boldsymbol{w}$ and b for fixed $\boldsymbol{a}$. To do this, we can take the derivative of L with respect to $\boldsymbol{w}$ and b. Then, set the derivatives to 0.

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} a_i\, y_i \stackrel{!}{=} 0$$
$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} a_i\, y_i\, \boldsymbol{x_i} \stackrel{!}{=} 0 \tag{2.7}$$

This results in the constraints $a_i \geq 0$ and $\sum_{i=1}^{n} a_i\, y_i = 0$. Applying the above on equation 2.4, we have:

$$\begin{aligned} L(\boldsymbol{a}) &= \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \sum_{i=1}^{n} \left(a_i\, y_i \boldsymbol{w}^\top \boldsymbol{x_i} + a_i\, y_i\, b - a_i\right) \\ &= \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \boldsymbol{w}^\top \underbrace{\sum_{i=1}^{n} a_i\, y_i\, \boldsymbol{x_i}}_{=\boldsymbol{w}} - b \underbrace{\sum_{i=1}^{n} a_i\, y_i}_{=0} + \sum_{i=1}^{n} a_i \\ &= \sum_{i=1}^{n} a_i - \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} \end{aligned}$$

We would then obtain an expression of L with respect to $\boldsymbol{a}$ that we maximise:

$$L(\boldsymbol{a}) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j$$

Combining the above together with the constraints $a_i \geq 0$ and $\sum_{i=1}^{n} a_i y_i = 0$, we would get the following quadratic optimisation problem:

$$\begin{aligned} \max_{\boldsymbol{a}} \quad & \mathbf{1}^\top \boldsymbol{a} - \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K}_y \boldsymbol{a} \\ \text{subject to} \quad & a_i \geq 0, \quad i = 1, \ldots, n \\ & \boldsymbol{a}^\top \boldsymbol{y} = 0 \end{aligned}$$

where $\mathbf{1}$ is an n dimensional vector of ones, and $\boldsymbol{K}_y = \left[y_i\, y_j\, \boldsymbol{x}_i^\top \boldsymbol{x}_j\right]$.

From equation 2.7,

$$\boldsymbol{w} = \sum_{i=1}^{n} a_i y_i \boldsymbol{x}_i$$

Substituting this into the linear model equation in (2.1), we get:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i y_i \boldsymbol{x}_i^\top \boldsymbol{x} + b \tag{2.8}$$

In order to determine the classification of a new point $\boldsymbol{x}$, we simply have to determine

the sign of $f(\boldsymbol{x})$ in (2.8).

2.6.2 Mapping to higher dimensional space

When the relationship between data points in the input space is not linear, the linear SVM above would not be able to learn these non-linear relations. This would result in underfitting.

As such, we would need to map the input data points into a higher dimensional feature space. We can then build an SVM based on this higher dimensional space such that the points in this feature space is linearly separable.

We first define a mapping $\phi : X \to F$ where ϕ is a non-linear mapping from the input space X to a higher dimensional feature space F. We then define the *kernel* function K such that $\forall \boldsymbol{x}, \boldsymbol{y} \in X$,

$$K(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$$

The optimisation problem can then be written as:

$$\begin{aligned} \min_{\boldsymbol{a}} \quad & \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K}_y \boldsymbol{a} - \mathbf{1}^\top \boldsymbol{a} \\ \text{subject to} \quad & a_i \geq 0, \quad i = 1, \ldots, n \\ & \boldsymbol{a}^\top \boldsymbol{y} = 0 \end{aligned}$$

where $\boldsymbol{K}_y = y_i\, y_j\, \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) = y_i\, y_j\, K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Similarly, equation 2.8, which determines the classification of a new data vector $\boldsymbol{x}$, can be written as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b = \sum_{i=1}^{n} a_i y_i\, \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}) + b$$

Similarly to the linear case, we simply need to test the sign of $f(\boldsymbol{x})$ for a new vector $\boldsymbol{x}$. If $f(\boldsymbol{x})$ is positive, then we can classify it as class 1, and if $f(\boldsymbol{x})$ is negative we can classify it as class 2.

2.6.3 Slack variables

Our data may not be perfectly linearly separable in the feature space $\phi(\boldsymbol{x})$, especially due to the presence of noise [33]. Slack variables, ξ, are introduced to allow some form of error when training data points are misclassified. These slack variables allow data points to be classified on the wrong side of the decision hyperplane, but the further away a point is from the decision boundary, the larger the penalty imposed. We then need one slack variable per input data point [34], defined as such:

- $\xi_i = 0$: data point is correctly classified.
- $0 < \xi_i \leq 1$: data point lies inside the hyperplane margin, but is on the correct side of the boundary.

- $\xi_i > 1$: data point is wrongly classified.

This is referred to as the *1-norm soft margin* constraint in the literature [33]. Now, we would need to maximise the margin of the hyperplane, while penalising the misclassified points. We can thus formulate our problem as such:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{n}\xi_i \tag{2.9}$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \text{for } i = 1, \ldots, n \tag{2.10}$$

where $C > 0$ is the *penalty parameter*. Similarly, to state the dual of (2.9) subject to the conditions in (2.10), we need to compute the Lagrangian:

$$L(\boldsymbol{w}, b, \xi_i, a_i, r_i) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n} a_i(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} r_i\xi_i$$

where $a_i \geq 0$ and $r_i \geq 0$ are the Lagrangian multipliers.

Similarly, we compute the derivatives of the above with respect to $\boldsymbol{w}$, b and ξ_i to get the following dual problem:

$$\min_{\boldsymbol{a}} \quad L(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K}_y \boldsymbol{a} - \boldsymbol{a}^\top \mathbf{1} \tag{2.11}$$

$$\text{subject to} \quad \boldsymbol{a}^\top \boldsymbol{y} = 0, \quad 0 \leq a_i \leq C \tag{2.12}$$

where $K_y = [y_i\, y_j\, \boldsymbol{x}_i^\top \boldsymbol{x}_j]$ and C is the penalty parameter.

We then need to solve the quadratic optimisation problem in (2.11) subject to (2.12).

2.6.4 Model selection

There are 4 widely used kernels:

- Linear kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$
- Polynomial kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma\, \boldsymbol{x}_i^\top \boldsymbol{x}_j + r)^d$
- Radial basis function (RBF): $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$
- Hyperbolic tangent kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\gamma\, \boldsymbol{x}_i^\top \boldsymbol{x}_j + r)$

where r, d, $\gamma > 0$ are kernel parameters.

First, we would pick a kernel before the training process. Then, using a procedure such as k-fold cross validation, we would then find the most optimal kernel and penalty parameters (C). For example, if we choose the RBF kernel, we would perform cross validation to obtain the most optimal parameters γ and C.

A study in [35] noted that the RBF kernel has less numerical difficulties than the polynomial kernel. We note that, $\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in X$, for an RBF kernel,

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \geq 0 \quad \text{and} \quad \gamma > 0 \quad \Rightarrow \quad 0 < K(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq 1$$

On the other hand, for large d, the polynomial kernel might go to infinity or close to 0. Furthermore, the hyperbolic tangent kernel is not valid under certain kernel parameters [36].

Although the RBF kernel is more commonly chosen than the rest of the kernels listed above, [35] also revealed that if the number of features is large, using the linear kernel may suffice, as the non-linear mapping provided by the RBF kernel may not necessarily improve performance. Furthermore, using the linear kernel would mean that we only need to investigate the optimal C penalty parameter value.

Nevertheless, the conclusion made in [35] might be data-dependent and ideally, we should determine empirically which kernel suits our data the best.

2.6.5 Implementing classification with SVM

Chang and Lin [37] developed `libsvm`, a library for SVMs, which allows the quadratic optimisation problem in (2.11) to be solved analytically [34]. Furthermore, the library can be interfaced to other programming languages like Java, MATLAB and Python.

The Python library `scikit-learn` also allows SVMs to be trained for classification purposes (`SVC`) and allows various kernels to be used. `scikit-learn` also uses `libsvm` internally to handle computations.

2.7 Feature selection

This section explores the literature of feature selection, and discusses some algorithms and methodologies employed in the literature. The following terms will be used throughout this paper:

- D: Original data set.
- M: Number of features in D. That is, $|D| = M$.
- S: Reduced subset of features selected by feature selection algorithm.
- k: Number of features in S, where $k \ll M$. That is, $|S| = k$.

2.7.1 Purpose and motivation

Discriminant analysis is key to this project. In the context of data with high dimensions, one of the main concerns of discriminant analysis is feature selection: instead of using all of the features to predict the result of the classification, we want to use only a small subset of the features. This subset should be representative of all the features.

We can also visualise the data set as a matrix, with columns as features and rows as samples. Then, the feature selection problem is akin to finding a smaller matrix with fewer columns that can be used more efficiently than the original matrix [38].

The process of feature selection can be described as finding a subset S of the original M features, such that the cardinality of S is k, where $k \ll M$. It is essential that classification accuracy using these k features should not be significantly lower than using all M features. Finding S that satisfies these criteria is the main concern of feature selection.

As mentioned in section 1.2, the data suffers from the curse of dimensionality, where the number of features is much higher than the number of samples. This usually causes overfitting [39, 40]. This refers to the situation where the classifier can classify the training data perfectly, but performs poorly with unseen data [24]. This is a common challenge in classification, and reducing the dimensions of the original data set often helps reduce overfitting [41]. As such, feature selection will play an important preprocessing role in allowing us to understand the data better.

Overall, there are several benefits associated with feature selection that are often mentioned in literature [42]:

- **Savings in computational cost.** Reducing the number of features by removing what Kudo and Sklansky call *garbage features* [43] can improve the computational cost of our algorithm. This includes the cost incurred for training a classifier, and using the classifier to perform predictions.
- **Interpretable results.** In contrast to feature extraction, feature selection retains the properties of the original features. For this project, ideally, we want to be able to obtain a compact subset of features from which we can draw biological conclusions. This will be discussed in more detail in section 2.7.2.
- **Improvement in classification accuracy.** Due to experimental faults or the nature of the data, more features might lead to more noise in our data set [38, 44]. Having a more compact subset to train our classifier might improve the performance of the classifier [45]. Furthermore, if selected well, a compact subset of features can generalise better than when all features are used [46].

2.7.2 Feature selection versus feature extraction

In contrast to feature selection, feature extraction techniques, such as principal component analysis (PCA) and Linear Discriminant Analysis (LDA), result in a set of features that is a *transformation* or combination of the original features. In other words, the original features are transformed into a space with lower dimensions. For example, PCA employs a transformation to map features to a space spanned by its principal components.

Feature extraction reduces the dimensions of the data by generating a set of new features. This implies that the interpretation of the original features is lost through the transformation. Furthermore, the original features in the data might contain features that are redundant or irrelevant to the target class (see section 2.7.7). Such features should ideally be removed.

In contrast, feature selection methods preserve the original features by simply removing those that are not relevant to the classification task. In general, feature extraction is performed when model accuracy is more important than model interpretability [38]. However, in this project, we would want to obtain features that are biologically relevant,

and can be interpreted by an expert [47]. This allows us to extract biological conclusions from the reduced set of features [48].

Due to these reasons, this project will focus on feature selection instead of feature extraction.

2.7.3 Classification of feature selection methods

Feature selection methods can be categorised into one of the following [49, 50]:

- **Filters** rank features with either a univariate or multivariate measure. The former evaluates each feature individually, and selects only the top ranking features, while the latter evaluates subsets of features. No learning is involved in filters; they are independent of the classifier [51]. In general, filters consider how each feature contributes to the classification of the targets, although interaction between features is not considered [52]. Filter methods include t-test, information gain, F-test and the χ^2-statistic.
- **Wrappers** regard the learning algorithm as a black box [42], and evaluate features based on the classification accuracy of the learner. Wrappers usually incur more computational cost from training the classifier and using it for prediction.
- **Embedded** methods incorporate the feature selection process into the training of the classifier. These methods combine the search in the feature space and the space of learning hypotheses, but they are specific to the choice of the classifier [50].
- **Hybrid** methods are usually a combination of the above approaches. For example, a method can use a filter-like approach to individually rank features, and then use a classifier to evaluate subsets of features [42].

2.7.4 Feature selection methods and trade-offs

The categories described above have their own merits and drawbacks. For example, as filters are independent of the classifier, they usually incur less computational cost. Furthermore, scoring each feature is usually independent of the other features, which means that the filter process can be parallelised. However, most filters are univariate, and thus do not consider feature redundancy or feature interaction. One can use multivariate filters which consider subsets of features, but these incur more computational cost.

On the other hand, wrappers are reported to produce better classification accuracy than filters [38], since they directly use a classifier to select features. At the same time, this means that wrappers are closely associated with the choice of learning algorithm, and might not generalise to other classifiers. One can also consider employing a "two-stage" process, such as in [51], where a filter is first used to remove unimportant features, and a classifier-dependent method can then be used to further refine the remaining features. Yet, one also needs to consider the number of features the first filter step should remove.

Indeed, Bolón-Canedo *et. al* [38] and Sharma *et. al* [53] echo these trade-offs, by pointing out that "the best method" does not exist and that no single algorithm stands out in all aspects. Furthermore, Hua *et. al* [51] performed a review of feature selection algorithms

using different data sets, and concluded that none of the methods that were reviewed performed consistently well across all the data sets. This sentiment is also reflected in [54].

To this end, previous works on feature selection seek to find a method that reports good classification accuracy for a *specific problem setting*. When one proposes a novel method or modification of a feature selection algorithm, one would compare the accuracy scores with other existing algorithms in the literature, and determine if their proposed method has an improvement over existing ones for a particular problem. We adopt this strategy in this project.

2.7.5 Terminating feature selection algorithms

Feature selection algorithms can be terminated via the following ways [43]:

- **Specifying** k: The algorithm stops when the desired number of features in S is obtained.
- **Satisfactory performance**: The algorithm terminates when the performance of S exceeds a specified threshold. For example, we can quantify this performance using cross validation scores of a classifier.
- **No significant improvement**: The algorithm can terminate when the performance of S does not improve much [53].
- **Balance between performance and subset size**: Optimise both classifier performance and k simultaneously by maximising performance and minimising k.

From most works in feature selection, authors often specify a fixed value of k. In [55] by Tang *et. al*, the authors initially set $k = 100$ for all the datasets that were used, but later recommended different values of k for two of these datasets. This suggests that k is dependent on the dataset.

If k is not a constraint of the problem, we could perform a grid search to find an optimal k. However, due to time and computational constraints, one will probably not be able to perform a grid search on all possible values of k (i.e. $k \in [1, M]$), but only on an interval $\theta \subset [1, M]$. So, even if we find an optimal k in θ, it is most likely only a local optimum.

2.7.6 Exhaustive versus heuristic search

One obvious but naive way to find S would be to exhaustively consider all possible subsets of the original feature space, and select the one that gives the best classification performance.

With M features, an exhaustive search would iterate through 2^M subsets [56]. Intuitively, for each feature in M, it can either be inside the selected subset, or not. We can prove this formally via induction.

Proof. The exhaustive search would require us to search through the space of all subsets, where for each subset S, $|S| \in [0, M]$. We include the trivial cases of $|S| = 0$ (all

M features do not contribute to the classification) and $|S| = M$ (all M features are important). Then, we want to prove that the number of subsets searched is 2^M:

$$\begin{aligned}&\text{No. of subsets with 0 feature} + \text{No. of subsets with 1 feature}\\&\quad + \cdots + \text{No. of subsets with } M \text{ features}\\&= \binom{M}{0} + \binom{M}{1} + \cdots + \binom{M}{M}\\&= 2^M\end{aligned}$$

where, $\binom{M}{k} = \frac{M!}{k!(M-k)!}$ is the number of subsets with cardinality k.

For $M = 1$, the total number of subsets searched is:

$$\binom{1}{0} + \binom{1}{1} = 2^1$$

Suppose $M = k$ is true, for an arbitrary $k > 1$. Then, the number of subsets searched would be

$$\binom{k}{0} + \binom{k}{1} + \binom{k}{2} + \cdots + \binom{k}{k} = 2^k$$

For $M = k + 1$, the total number of subsets searched is:

$$\begin{aligned}&\binom{k+1}{0} + \binom{k+1}{1} + \binom{k+1}{2} + \cdots + \binom{k+1}{k} + \binom{k+1}{k+1}\\&= \binom{k}{0} + \left[\binom{k}{0} + \binom{k}{1}\right] + \left[\binom{k}{1} + \binom{k}{2}\right] + \cdots + \left[\binom{k}{k-1} + \binom{k}{k}\right] + \binom{k}{k} \qquad (2.13)\\&= 2\left[\binom{k}{0} + \binom{k}{1} + \cdots + \binom{k}{k}\right]\\&= 2\left(2^k\right)\\&= 2^{k+1}\end{aligned}$$

where in (2.13), the following results were used:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

$$\binom{k+1}{0} = \binom{k}{0} = 1 \quad \text{and} \quad \binom{k+1}{k+1} = \binom{k}{k} = 1$$

So, for any $M \geq 1$, the number of subsets searched by an exhaustive search is 2^M. □

Evidently, although an exhaustive search would guarantee an optimal subset, its runtime complexity would be prohibitively high. As Sima and Dougherty put it, *"A major impediment to feature selection is the combinatorial nature of the problem"* [57]. Thus, most of the algorithms in the literature of feature selection trade the optimality guaranteed

in exhaustive search for lower computational cost. These methods involve heuristics to guide the search for S. Some examples would be discussed in the following sections.

2.7.7 Feature relevance and feature redundancy

Relevance and redundancy play important roles in the literature of feature selection. In [38], Bolón-Canedo *et. al* assert that "*detecting the relevant features and discarding the irrelevant and redundant ones*" are processes that define feature selection.

In general, a feature is *relevant* if it is closely related to the target class; in other words, it has a large influence on the outcome of the classification of the samples. For example, one can evaluate the relevance of each feature through some measure such as the t-statistic and rank the relevance of the features. It is tempting to select the first k features as our subset S, but as Peng *et. al* put it, "*The best k features are not the k best features*" [56], as the highly ranked features are likely to have similar discriminating power [38]. In other words, some features among these k features might be redundant.

A feature f_1 is *redundant* with respect to another feature f_2, if f_1 is "similar" to f_2. This similarity can be quantified in many ways and mutual information is often used (see section 2.7.11). [6] describes the "ultimate feature redundancy" as two features having exact linear dependency. For example, f_1 and f_2 might be linked by the equation $f_1 = 2\,f_2$, and thus f_2 does not provide additional information in the presence of f_1 (or vice versa).

In the context of this project, there is a possibility that not all of the features in our data set have a role to play in the classification of schizophrenia. As such, quantifying relevance and redundancy is necessary to select the significant features and eliminate the others. Besides, once we obtain a compact subset S, it would be computationally cheaper to train our classifier based on just these features.

2.7.8 Greedy search

As mentioned in section 2.7.1, the cost of an exhaustive search on all possible subsets of all M features is exponential with M. Even though the exhaustive search guarantees an optimal solution, its computational cost does not scale with more features. As a result, many algorithms involve heuristics to guide the search for the optimal subset S. Although none of the methods which use heuristics can guarantee an optimal solution, these methods improve the computational cost of finding S, which is very beneficial in the context of high dimensional data, making these methods more practical to execute [42, 56]. These methods are thus classified as sub-optimal.

For example, the *sequential forward selection* (SFS) [58] is a wrapper method that involves a greedy search strategy. The method starts with $S = \emptyset$. For each feature m, we compute an evaluation function, $J(\cdot)$. In this context, $J(\cdot)$ is the classification score of the learner from just using 1 feature. The scores are then sorted, and the feature with the highest score (call it f_1) is selected. That is, if F represents the set of all features, then:

$$\forall f_k \in F, \quad J(f_1) \geq J(f_k)$$

f_1 is then paired sequentially with each of the remaining features to form a subset, and classification scores using just 2 features are obtained and sorted. That is, we obtain classification scores for the sets

$$\{f_1, f_2\}, \{f_1, f_3\}, \ldots, \{f_1, f_M\}$$

Let f_2 be the feature where the set $S = \{f_1, f_2\}$ has the highest classification score. We now pair another of the remaining features with S and find classification scores. This process continues until we have our desired k features.

A variant is the *sequential backward selection* (SBS), which starts with all M features instead. SBS sequentially removes one feature. That is, SBS starts with the full set of M features and works backwards. This means that SBS will take a longer time than SFS to give us our desired number of features k, especially when k is a small number [59], since we will need to iterate through each feature and fit a classifier using a large number of features. It is suggested that SBS might give us a better result, but we will end up with larger feature subsets and longer computational time [60].

Another variant is the *floating forward/backward search* strategy proposed by Pudil [61]. Pudil noted that both SFS and SBS suffered from a "nesting effect", where previously selected or discarded features cannot be considered again in SFS and SBS respectively. The floating variant tackles this problem.

Lastly, Deng [62] proposed the *Restricted Forward Selection* (RFS), which is an even greedier variant of SFS. Suppose we have a sorted list of the classification scores when each feature is considered singly. That is,

$$J(f_1) > J(f_2) > \cdots > J(f_M)$$

f_1 would be selected as the first feature. While SFS considers all of the remaining features, RFS only considers those features that perform well individually. We thus consider the pairs $\{f_1, f_2\}, \{f_1, f_3\}$ all the way to $\{f_1, f_{M/2}\}$, and select the best subset. That is, we only consider $M/2$ of the sorted features that perform best individually.

The fact that RFS considers only part of the sorted features makes it a greedy variant of SFS. The pseudocode of RFS is listed in algorithm 1. In algorithm 1, the list L represents the sorted list of $(feature, score)$ pairs where *score* is the cross validation (CV) score obtained by using 1 feature only.

2.7.9 Recursive feature elimination

Another flavour of search is the Recursive Feature Elimination (RFE), first proposed by Guyon *et. al* in [41] and it has been used in [63, 64, 65]. In particular, Guyon *et. al* focused on RFE when used with SVMs. RFE can thus be classified as a wrapper method.

From section 2.6, fitting an SVM with a linear kernel boils down to solving the problem:

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} - \sum_{i=1}^{n} a_i(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1)$$

Algorithm 1: Restricted Forward Selection(D, k)

Input: Data set D with M features.
Output: Subset S ($S \subset D$) of the M features such that $|S| = k$, $k \ll M$.

begin
    Initialise empty list $L \longleftarrow [\,]$
    Initialise empty list $S \longleftarrow [\,]$
    for $i = 1 \longrightarrow M$ **do**
        $s \longleftarrow$ CV score for i-th feature.
        $L \longleftarrow L \cup \{(i, s)\}$
    $w \longleftarrow$ feature that corresponds to max s in L.
    $S \longleftarrow S \cup \{w\}$
    for $i = 2 \longrightarrow k$ **do**
        $num\_iterations \longleftarrow M/i$
        Initialise empty list $scores \longleftarrow [\,]$
        $j \longleftarrow 0$
        while $|scores| \leq num\_iterations$ **do**
            if *feature in* $L[\,j\,] \in S$ **then**
                $j \longleftarrow j + 1$
                continue
            $S_j \longleftarrow S \cup \{$feature in $L[\,j\,]\}$
            $s \longleftarrow$ CV score for S_j.
            $scores \longleftarrow scores \cup \{(j, s)\}$.
            $j \longleftarrow j + 1$
        $w \longleftarrow$ feature that corresponds to max s in $scores$.
        $S \longleftarrow S \cup \{w\}$

where the weight vector $\boldsymbol{w}$ and bias term b are parameters of the hyperplane that separates the data points:

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Furthermore, to determine the classification of an input vector $\boldsymbol{x}$, we can simply compute:

$$y(\boldsymbol{x}) = sign(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

Thus, evidently, the input vectors $\boldsymbol{x}$ that have the highest corresponding weights $\boldsymbol{w}$ will contribute the most to the classification of $\boldsymbol{x}$ [63]. This motivates the RFE.

In RFE, a classifier is first trained using all of the features. The weight of each i-th feature, w_i, is then computed. RFE eliminates insignificant features by removing features with the lowest weights $\boldsymbol{w}$. This procedure is repeated until the desired number of features is obtained. In cases where features are eliminated one at a time, RFE can be viewed as a *feature ranking* procedure, since we rank features according to their weights. Otherwise, it is considered a *feature subset ranking* procedure.

Evidently, RFE is a greedy algorithm, as it progressively reduces the subset of features it considers, and does not consider features that have been eliminated. Thus, RFE considers *nested* subsets of features.

Furthermore, as RFE iteratively removes features, it works similarly to the sequential backward selection (SBS) method. As such, one disadvantage of RFE is that it might be computationally inefficient, as the algorithm trains a classifier using all of the features at first, and progressively reduces the number of training features. However, Guyon *et. al* suggested removing several features at a time, making RFE a feature subset ranking algorithm. Evidently, optimality will be compromised in this manner.

2.7.10 Optimal search by branch-and-bound

It is also worth mentioning the work of Narendra and Keinosuke [66], who proposed a branch-and-bound algorithm to demonstrate that an optimal solution can still be obtained without an exhaustive search. Even though the paper showed that substantial savings were made in terms of number of subsets evaluated, the algorithm does not scale to high-dimensional data [61] due to its exponential time complexity [59, 67]. For example, the paper demonstrated the use of the branch-and-bound algorithm to choose only 12 features from 24. As such, we will not look into this method any further, as the other types of search, though less optimal, do not involve exponential time complexity.

2.7.11 Use of mutual information in feature selection

A common heuristic to quantify relevance and redundancy is mutual information (MI) [24]. MI can be interpreted as a measure that quantifies the dependence between variables X and Y [68] and the amount of information X carries about Y [69].

For discrete random variables X and Y, the MI between them, $I(X, Y)$, is given by:

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)\, p(y)} \right) \tag{2.14}$$

For continuous variables, the sum will simply change to an integral:

$$I(X,Y) = \int_{x \in X} \int_{y \in Y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)\, p(y)} \right) dy\, dx \tag{2.15}$$

For a combination of discrete and continuous random variables, we would simply use the sum or integral respectively. For example, we can calculate the MI between a target class C and a continuous feature X by evaluating:

$$I(X,C) = \int_{x \in X} \sum_{c \in C} p(x,c) \log_2 \left(\frac{p(x,c)}{p(x)\, p(c)} \right) dx$$

An example of a feature selection algorithm that makes use of MI extensively is the *minimal-redundancy maximal-relevance* (mRMR) algorithm [56]. mRMR maximises relevance between a feature set S and target class c by maximising the MI between them. That is,

$$\max\ D(S,c) = \frac{1}{|S|} \sum_{x_i \in S} I(x_i, c)$$

This heuristic maximises the discriminative power of each feature in S.

Furthermore, mRMR minimises redundancy between features in the feature set S by minimising the MI between features in S. That is,

$$\min\ R(S) = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i, x_j)$$

However, the main difficulty in using MI is the estimation of the joint and marginal probabilities ($p(x, y)$ and $p(x)$ or $p(y)$ respectively in equations 2.14 and 2.15). For discrete or categorical variables, estimating the probabilities would simply be obtained from frequency counts [70], although the estimation would only be accurate if we have enough samples [56]. Yet, with more classes and more states in each class, the estimation of the joint probability would become more difficult [42].

In the case of continuous variables, the computation of MI would be even harder [71], as we would need to compute the integral of the continuous probabilities [47]. A way to avoid computing integrals is to discretise the continuous variables. We can estimate the probabilities of continuous variables by discretising the values into *bins* (discrete intervals). A histogram can be created to investigate the distribution of the bins. However, we will have to determine an appropriate width for the intervals (or an appropriate number of bins), which will influence the estimates significantly [72]. We should also note that the features in the data for this project are continuous.

Another method to estimate these probabilities is by using the *Parzen Window* method, as proposed by Kwak and Choi in [70]. Although the work has shown that using the Parzen Window leads to better performances when combined with certain feature selection algorithms, one still has to specify certain parameters associated with the Parzen Window, such as the *window function* and the *window width*. These need to be correctly specified for the estimated probabilities to converge to the true probability [73]. Kwak and Choi used the Gaussian window with a fixed window width. However, these choices were not explained sufficiently. They have also made certain assumptions about the covariance of the variables, but these assumptions might not hold for all types of data sets. Nevertheless, with sufficient time, one could perform a search to find an optimal window width using different window functions.

Consequently, methods such as *maximum relevance minimum multicollinearity* (MRmMC) [6] avoid the complexity of estimating MI involving continuous variables by proposing other measures to quantify relevance and redundancy. This will be discussed in section 4.4.

2.8 Genetic algorithms in feature selection

Genetic algorithms (GAs) are a type of heuristic search method that explores solutions by repeatedly evolving and combining them. It is based on the theory of evolution and its concept of *natural selection* and *survival of the fittest* [74]. GAs search through a space of potential solutions by using probabilistic genetic operators to avoid local optima and produce better solutions [43, 75, 76]. Although GAs are reported to take more time to select features, they can explore a wider variety of subset features [52].

The following terms are often used in the context of GAs:

- A **chromosome** describes a solution [74, 77]. Usually, a chromosome uses a *binary encoding* to represent a solution. An array, say A, can be used to store a chromosome. In the context of feature selection, each element in the array, $A[\,i\,]$, is either 0 or 1, which means the i-th feature is not selected or a feature is selected, respectively.
- The i-th **generation** describes the i-th iteration of the GA.
- A **population** is the set of chromosomes (solutions) in a generation.

The following terms are inspired by evolution, and characterise the probabilistic aspect of a GA:

- **Fitness** quantifies how desirable a chromosome is with respect to the problem the GA is trying to solve. For example, [78] uses classification error as its fitness criteria, [79] uses both the classification error and the dimension of a chromosome, while [80] uses correlation between two variables.
- Two parents can perform a **crossover** to produce a pair of potentially fitter offsprings. There are several strategies for the crossover operator, such as one-point [80], two-point [81, 78] and uniform crossover [82]. These will be discussed in section 2.8.3.

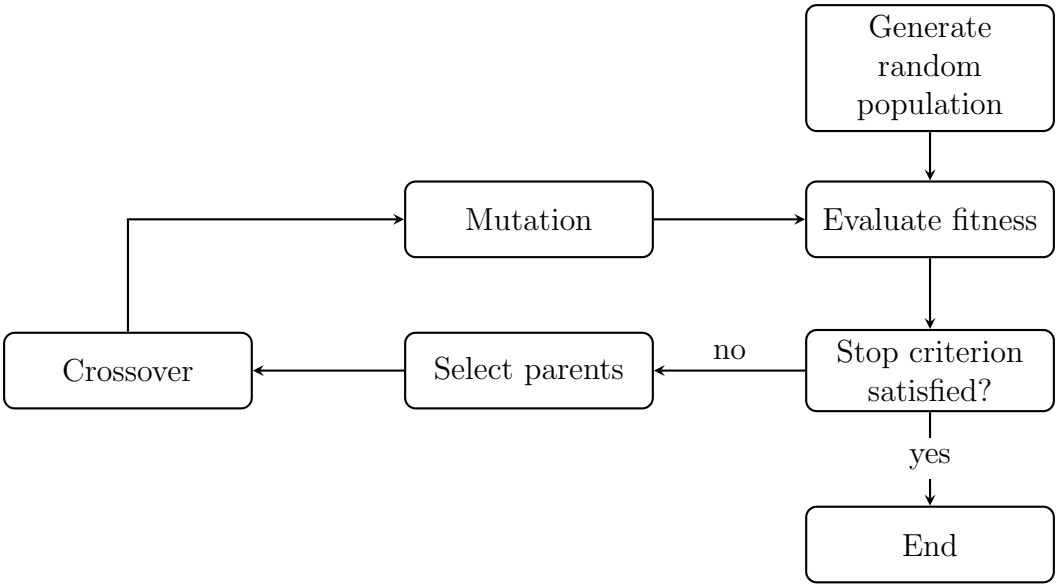


Figure 2.3: Flowchart showing the steps that are usually involved in a typical genetic algorithm.

- The **mutation** operator probabilistically selects and changes one or more elements of a chromosome. For example, [80, 78] use a uniform distribution to select these elements. In the context of feature selection, the chosen element will have its bit inverted.

- The **selection** process selects parents for the crossover process. Depending on the strategy that one employs, the parents might simply be the two fittest chromosomes or they might be randomly selected. One can also choose chromosomes with the worst fitness for crossover, in hope that this will raise the overall fitness of the population [83].

These operators are responsible for evolving a population and they help to increase the *diversity* of a population, which can be interpreted as a balance between *exploitation and exploration* [84]: changing a good solution encourages the algorithm to explore other solutions, possibly allowing us to escape a local optimum.

GAs encapsulate the concept of *survival of the fittest*, since the fitter a chromosome is, the more likely it is either copied to the next generation (*elitism/cloning* [81, 85, 86]), or selected as a parent for crossover [77]. Furthermore, Kudo and Sklansky also recommend GAs for "large-scale" problems that involve more than 50 features [43]. A typical genetic algorithm is illustrated in the flowchart in Figure 2.3.

Moreover, GAs do not require gradients to find an optimum solution. This avoids the numerical and computational complexity associated with other search methods [87]. GAs can also take advantage of parallelism to speed up computation (see section 4.7).

As an extension proposed by the authors, we investigate if incorporating a GA with the MRmMC method will improve its performance. This will be discussed in detail in section 4.6.

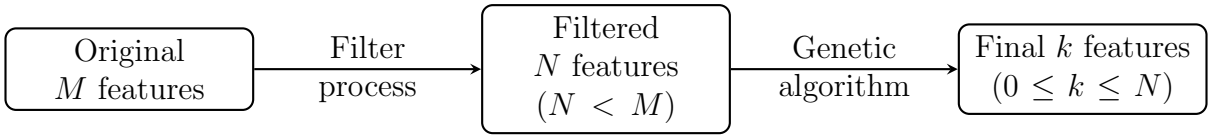


Figure 2.4: Flowchart depicting the two-stage genetic algorithm.

2.8.1 The two-stage genetic algorithm

There are several ways to implement GAs, and each has its merits and drawbacks. For example, a method described by Mitchell in [74] initially generates p solutions at random. This method is used in [88], where 20 solutions are randomly generated to form the initial population of the GA and each solution will contain M 0's and 1's where M is the total number of features available. We note that the GA in [88] was performed on a dataset of only 9 features. Thus, the array representing a solution will have 9 elements each. However, in the context of our project, storing 20 arrays of size over 400000 each might impose a large memory demand.

In contrast, several works such as [44, 89, 21, 79, 83] utilise a GA in a "two-stage" process. First, a filtering step removes features that might be irrelevant to the target class or redundant with respect to other features. For example, [44, 89, 83] use mutual information to conduct an initial filter on all the features. Second, the set of filtered features is then passed on to a GA to explore solutions in the reduced space. This idea is illustrated in Figure 2.4.

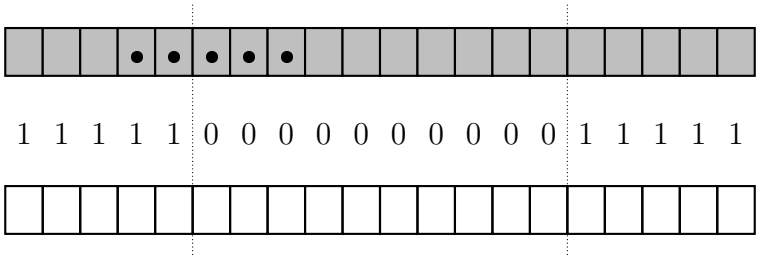
This two-stage process offers an improvement in terms of memory demand, as the filter process shrinks the dimension of the solution space for the GA. Instead of storing all 400000 features, we only need to keep track of the binary strings encoded for the reduced search space. Although [80] argues that the reduced search space still contains features that are the most promising, the GA implemented this way evidently suffers from a compromise in optimality.

2.8.2 Parameters and strategies of GAs

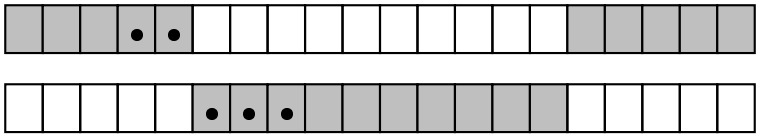
Even though previous works using GAs have reported positive experimental results, one difficulty of GAs is the number of parameters that need to be tuned [43], including population size, crossover rate, and mutation rate.

Moreover, different strategies can be used in certain parts of the GA. For example, in the selection process, the parents can either be the solutions with the worst fitness [83], or selected in proportion to their fitness (*Roulette Wheel selection*) [74, 86]. The mutation rate, which often takes a small value, say m, can range from 1% to 15% [88].

One would need to investigate what parameters and/or strategies would work best for the data set, and justify any parameters that seem arbitrary. This is the heart of one of the extensions listed in section 6.1.



(a) Before crossover operation. The array in the middle is the crossover mask, and the two arrays are the two parent chromosomes.



(b) After crossover operation. The two arrays shown are the offsprings of the above parent chromosomes.

Figure 2.5: A diagram demonstrating how a two-point crossover works.

2.8.3 Regarding crossover

The crossover operator is an important step in a GA, and is considered to be the "driving force" of the GA, as it allows the algorithm to explore more solutions over the search space [90, 91]. Two aspects of crossover can vary: its rate and its type, which include one-point, two-point and uniform crossovers.

In the *one-point* crossover, a position (crossover point) along a chromosome is randomly chosen. The segments partitioned by this position are swapped between the two parents to produce two offsprings. In the *two-point* crossover (see Figure 2.5), two positions are chosen and the segments in the parent chromosomes are swapped. In the *uniform* crossover, each position is chosen uniformly and is thus a potential crossover point.

The partition can be represented as a *crossover mask* [74]. Figure 2.5 shows how the two-point crossover works. The crossover mask is a binary string with two segments of 1's. For one of the offsprings, "1" might represent "copy bit from parent A" while "0" might represent "copy bit from parent B". The instructions will be inverted for the other offspring. On the other hand, for a uniform crossover, the crossover mask is generated randomly.

What the crossover and mutation operators do is to ensure that segments of information encoded in each chromosome (also referred to as *schemata* [92]), are sampled sufficiently. In [92], Picek and Golub discuss the performance of the aforementioned crossover operators. They found that GAs that do not use any crossover operators, or those that use one-point crossovers do not perform as well as those that use uniform or two-point crossovers. We note that the uniform and two-point crossover strategies disrupt segments of information in a chromosome more extensively than the one-point crossover. For example, in Figure 2.5a, a schemata is represented by dots. The crossover operation might disrupt this segment, as shown in Figure 2.5b.

The one-point and two-point crossover operators also introduce a *positional bias*. For

example, the extreme points of a chromosome are less likely to be swapped by the one-point crossover; shorter schemata are less likely to be disrupted than long ones in the two-point and uniform crossovers. On the other hand, while the uniform crossover does not have such positional bias, it disrupts the schemata of a chromosome most frequently, although this allows the GA to explore a larger space of solutions [92].

The fact that the *rate* of crossover also varies across problems is similarly brought up by Lin *et. al* in [93]. To tackle this variability, they have proposed an adaptive scheme that adjusts the mutation and crossover rates according to how the solutions perform in each generation, which eliminates the need for grid searches for these rates. The rates are adjusted based on the average increase or decrease in fitness of the offsprings. Empirical results have shown that this adaptive scheme improves the performance of GAs [93, 91].

Ultimately, Picek and Golub conclude that each crossover operator has its merits and drawbacks, and this echoes the discussion in section 2.7.4 where the strategies and parameters have to be tuned according to the problem setting, and no single strategy outperforms all other strategies.

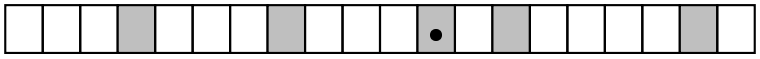
2.8.4 Regarding selection

Selection is another important genetic operator that might greatly influence the outcome of the algorithm. The purpose of the selection procedure is to select solutions as parents. In a way, the selection process creates an "intermediate population" [94], which the crossover and mutation operators change, to produce a new generation.

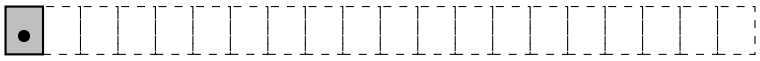
In [95], Whitley proposed that *diversity* and *selection pressure* are the primary motivations of the search for a solution, similar to the idea of exploitation and exploration. While the former refers to having a variety of solutions with different fitness values, the latter refers to how much we prefer fitter solutions. One needs to balance these two factors carefully. While having a generation with diverse solutions allows us to explore a larger area of the feature space, it might lead to slower convergence of the algorithm. On the other hand, a high selection pressure allows us to find an optimum quickly, but the algorithm might terminate prematurely, giving us a local optimum [87].

One popular selection method is the *fitness proportionate* method, also known as the *roulette wheel selection* method. As its name suggests, parents are chosen in proportion to their fitness with respect to the other solutions. Similarly, one can picture a roulette wheel, where each solution occupies a sector of the wheel with area proportional to its relative fitness. Consequently, the chance of picking a solution with high fitness will be higher. This method is relatively simple to implement. For example, the `numpy` [96] method `numpy.random.choice` can be used to pick two parents with a specified probability.

However, there are many problems that are commonly associated with roulette wheel selection. In [74], Mitchell describes a potential *crowding* problem. One can imagine that solutions with much higher fitness than the rest will get chosen as parents very often, possibly stagnating the algorithm. This might also result in a population having similar solutions, hence reducing the diversity of the population. This problem is also raised in [97]. If the solutions in a population have similar fitness, then the improvement obtained



(a) The initial population. Suppose that 5 of the solutions are chosen to participate in the tournament. These are represented as shaded boxes. In this case, $N_t = 5$. Suppose the dotted entry wins the tournament.



(b) The intermediate population. The shaded dotted entry from the previous figure fills up the intermediate population. The dashed entries represent part of the intermediate population that is yet to be filled.

Figure 2.6: A diagram demonstrating how tournament selection works.

by the next generation will be limited, since each solution is equally likely to be selected for the intermediate population [98].

An alternative is the *tournament selection* method. A pool of, say, N_t solutions is obtained from the population. From the pool, the solution with the highest fitness will be chosen with a certain probability to populate the intermediate population. This process is repeated until the intermediate population is filled up. Binary tournaments where $N_t = 2$ are frequently held [87, 99]. In this case, the solution with the better fitness is chosen with a probability p_t, where $0.5 < p_t \leq 1.0$. This is illustrated in Figure 2.6, where we pick $N_t = 5$.

[87] argues that the intermediate population will have, on average, better fitness than the original population, which contributes to an improvement in the overall fitness of the new population. This method is also able to avoid the pitfalls of the roulette wheel method. Moreover, because all of the solutions are equally likely to get selected for the tournament, the fitter solutions will less likely dominate the intermediate population [98].

Works such as [98, 100] conclude from empirical results that the tournament selection method is the better method between tournament selection and roulette wheel selection. However, as often mentioned in this report, there is no universal strategy or set of parameters in feature selection methods. We can investigate if switching between the roulette wheel and tournament selection methods will make a difference for our data set.

Chapter 3

Data exploration

3.1 Understanding the data

The data set[1] contains 2 giga bytes worth of data, with 847 samples and 420374 features. Due to the sheer size of the data set, it is not possible to open it on a typical program such as Microsoft Excel without waiting for a very long time.

To circumvent this issue, the Python library `pandas` [101] was used. `pandas` provides high performance and user-friendly data structures and tools to analyse data sets. In particular, `pandas` provides `read_csv` and `read_excel` methods that can read a large file. The data set is then cast into a `DataFrame`, a data structure which is also used in the *R* programming language.

One particular advantage of using `pandas` is that `DataFrame` has an `Index` object, that stores the labels of the data set. For example, from Figure 3.1, we can see how the data set is organised when cast into a `DataFrame` in `pandas`. It is also helpful that `pandas` indicates `[847 rows x 420374 columns]` which tells us that labels such as `101103430155_R06C01` indicate a sample (individual) while the label `cg00000029` refers to a feature (CpG island - see background in section 2.1).

`pandas` also works well with lists and arrays in the standard Python library, and with data structures in `numpy` [96]. However, calling `pandas.read_csv` and loading the data into a `DataFrame` every time we need the data set will impede efficiency. Instead, `pandas` offers the `HDFStore` class, which utilises `PyTables` [102] to allow fast read and write of `pandas` data structures. We can think of the `HDFStore` as a dictionary consisting of a key and a value. Therefore, each time we need the data set, or indeed any data that takes a long time to read, we simply have to retrieve it from the `HDFStore` with a key. The labels (classification) of the samples are also stored in this `HDFStore`.

Finally, data structures in `pandas` work well with the `scikit-learn` software [103], which was used extensively in this project, such as fitting an SVM and obtaining cross validation (CV) scores.

[1] Available from `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE84727`.

	cg00000029	cg00000108	cg00000109	cg00000165
101103430155_R06C01	0.458075	0.864368	0.825785	0.128369
3998567009_R05C02	0.405800	0.895983	0.704118	0.160892
100973330103_R01C02	0.461496	0.856618	0.779283	0.207591
101103430127_R05C01	0.387327	0.900614	0.799923	0.144362
100973330068_R02C02	0.416400	0.887695	0.735643	0.165496
3998567004_R03C02	0.407997	0.899651	0.739725	0.195290
3999215203_R06C01	0.344186	0.893728	0.746948	0.149002
3998928013_R01C02	0.410588	0.904365	0.777965	0.131979
3998567090_R03C02	0.364430	0.885433	0.797229	0.134324
3998567008_R02C02	0.381532	0.901766	0.732903	0.152917
3998634017_R01C01	0.468282	0.894842	0.803552	0.170385
101103430152_R02C01	0.425883	0.869845	0.730765	0.155393
3998634048_R02C02	0.354365	0.870555	0.780461	0.200983
3998567045_R01C01	0.399675	0.905126	0.786913	0.195889
3998952104_R01C01	0.437671	0.911549	0.752112	0.172296
100973330104_R05C02	0.364644	0.930791	0.799278	0.140339
101103430134_R03C02	0.428076	0.897993	0.786318	0.134738
3998567027_R03C02	0.441972	0.914435	0.780155	0.147537
101103430128_R02C01	0.271820	0.899177	0.810853	0.118240
3998928013_R06C02	0.415148	0.870553	0.809269	0.217449
101103430168_R03C02	0.338757	0.895134	0.748873	0.144406
3999215213_R01C02	0.393489	0.899325	0.711013	0.184176
3999215213_R03C02	0.395375	0.906487	0.762206	0.153578
3998568010_R01C01	0.396995	0.910559	0.790561	0.178585
101103430168_R01C02	0.420036	0.910173	0.776432	0.139460
101103430073_R05C02	0.436113	0.911736	0.799517	0.172902

Figure 3.1: A truncated output of the `DataFrame` that stores the (shuffled) data set. We can use this output to study the information available in the `DataFrame`, and how this information is structured.

3.2 Using SVMs

For all numerical experiments described in chapter 4, the SVM will be used, when there is a need for a classifier. For example, if the method used is a wrapper method, the SVM will be used to evaluate the performance of the feature subset. In contrast, in a filter method, the SVM will not come into play. This is similar to previous works such as [51], where the classifier is fixed, and attention is focused on the feature selection algorithms.

The SVM is fitted using the `scikit-learn` Python software [103]. Calling the `SVC()` method would create a classifier with default parameters, such as the default Radial Basis Function (RBF) kernel, and default error parameter $C = 1.0$ (see section 2.6.4). Alternatively, one can specify C and the type of kernel.

A practical guide to SVMs by Hsu *et. al* in [35] argues that when the number of features is much larger than the number of samples, using a linear kernel would suffice, so the linear kernel will primarily be used in the experiments in chapter 4.

However, it is not sufficient to simply assume the default penalty parameter $C = 1.0$. Instead, as Hsu *et. al* suggested, performing a grid search on C allows us to find an optimal C that might perform better than the default value. However, due to time constraints, the following experiments would stick to the default value of C. Using other kernels and parameters is proposed as an extension in section 6.1.

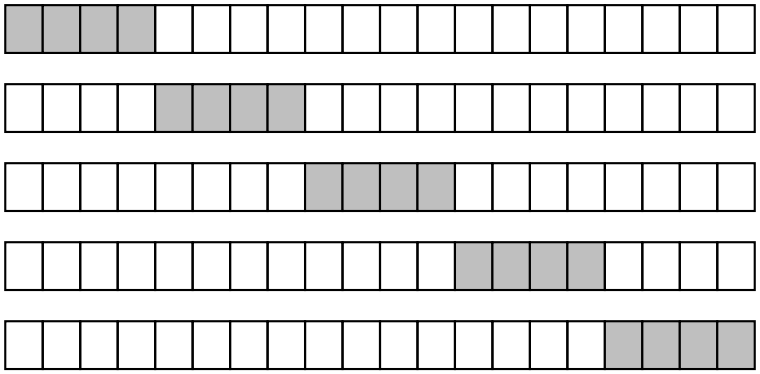


Figure 3.2: A diagram illustrating 5-fold cross-validation. In each iteration, 20% of the data is "held-out" as the test set (grey boxes), while the rest is used as the training set (white boxes). The overall CV score is the average classification accuracy across all 5 iterations.

3.3 Cross validation

The cross validation (CV) scores can be obtained via the `cross_val_score` method in the `sklearn.model_selection` package. The number of folds used throughout this project is 5. That is, 80% of samples will be used to train an SVM, while the other 20% will be used on the newly-fit SVM to obtain an accuracy score. This is repeated 5 times, where the training/testing set in each iteration is unique. The CV score across all the folds is then the average classification accuracy across all five iterations. The cross validation process is illustrated in Figure 3.2.

Fitting an SVM and obtaining CV scores using 5-fold cross validation with either the linear or RBF kernel took about 30 minutes each using the High Performance Computing cluster (see section 4.8) at Imperial College [104]. All of the features were used in both cases.

The choice of 5 folds was also made in other works such as [6], while other works such as [81] and [85] use 10 folds. More folds will allow us to train and test the classifier more extensively, obtaining scores with higher accuracy. However, we note that [81] investigated data sets with less than 20 features and the cross validation process will not take up a lot of time. On the other hand, due to the large number of features this project deals with, using 5 folds is a good balance between computational time and accuracy.

3.4 Data preprocessing

Before the data set can be used for cross validation, it needs to be preprocessed first. Hsu *et. al* also described the importance of *scaling* our data before fitting an SVM. In particular, scaling the data would prevent features with large values from dominating those with smaller values. Scaling the features such that each feature has zero mean and unit variance is also recommended when using the `SVC()` (support vector classification) method. This procedure is also carried out in other works such as [41]. In this project, the data was scaled with the `sklearn.preprocessing` package in `scikit-learn`, using the `scale` method.

Furthermore, the raw data was sorted based on the classification (target label) of each sample: the samples in the raw data set were ordered such that the cases (label 0) came first, followed by the controls (label 1). It is thus sensible to *shuffle* the data such that the target labels would not be in any particular order. The cross validation process also partitions the data set into folds, and the process of fitting a classifier using the data set will be affected if the samples are ordered, since this will introduce bias. For example, one of the test sets might only contain samples in a particular class.

The shuffling was done using the `sklearn.utils.shuffle` method. The method also allows us to set the reproducibility of the shuffling. Furthermore, as discussed in section 3.1, once the entire data set and labels are shuffled, these are stored in a `DataStore`, and is retrieved when needed.

3.5 Using all features

In the context of feature selection, it is only sensible to compare the performance of the feature selection algorithms to the situation where all the features are used in SVM classification.

Fitting an SVM using the RBF kernel with all the features produced a cross validation score of 0.834644, while with the linear kernel, the score was 0.929160. Nevertheless, this result does not necessarily mean that the linear kernel is more effective than the RBF kernel as suggested by Hsu *et. al*, since some subsets might perform better with the RBF kernel than with the linear kernel. However, due to time constraints, only the linear kernel was used. As an extension, we can also investigate the performance of the RBF kernel with different subsets of features (see section 6.1).

3.6 Investigating dependencies on other variables

As mentioned in section 1.1, epigenetics can be interpreted as a "window" through which the external environment of an individual can alter his or her DNA. Thus, in analysing the data, we should be aware of other variables, known as *confounding factors*, that might affect the classification of the samples. Confounding factors, such as gender and lifestyle, contribute to the dynamic nature of epigenetics [17].

From the relevant files[2], we can also extract the *gender* and *age* information for most of our samples. We thus need to check if these confounding factors influence the classification of our samples.

3.6.1 Effects of linear regression

Linear regression was performed on the data set, where we take age and gender as the independent variables and the feature values as the dependent variable. We want to

[2]For example, the files available from `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE84727`.

see if there is any relationship between age, gender and the features. Suppose we let $Y =$ feature and $X_1 =$ age, $X_2 =$ gender. Then, from [105], we have a linear model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

where ϵ is an unobservable, random error with mean 0 and unknown variance σ_ϵ^2. $\beta_0, \beta_1, \beta_2$ are the coefficients of the linear model.

The process of linear regression then finds the best estimates $\widehat{\beta}_i$ of β_i for $i \in \{0, 1, 2\}$ by the *method of least squares*, which minimises the square of ϵ for each sample:

$$\min \sum_{i=1}^{N} \epsilon_i^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2})^2$$

where N is the number of samples, and x_{ij} is the i-th sample of the j-th variable for $i \in [1, N]$ and $j \in \{1, 2\}$. The method of least squares was performed by using the `LinearRegression` module in Python.

After the linear regression procedure, we obtain a *fitted/predicted* value $\widehat{y}$ of y. That is,

$$\widehat{y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \widehat{\beta}_2 x_{i2}$$

3.6.2 Effects of the residuals

Intuitively, the method of least squares minimises the difference between the actual and fitted value of y. This leads to the definition of the *residual*:

$$e_i = y_i - \widehat{y}_i, \quad i = 1, 2, \ldots, N$$

From [106], the residual can generally be interpreted as the variation that is not explained by the fitted model. The resulting residuals are thus the variation in the features that is unexplained by age and gender. This means that we can use the residuals as our new "data set", which factors out age and gender from the original data set.

We again fit an SVM using the residuals:

	Linear	RBF
Original data set	0.929160	0.834644
Residuals	0.851158	0.799983

From the table above, we can see that the CV scores do not differ significantly (about 3% to 7% difference). With the residuals, the SVM seems to have slightly lower classification accuracy. This suggests that age and gender only have a small influence on the classification of the samples.

We should also note that in the original data, the ages of about 200 individuals were missing. These individuals were removed from the samples for the linear regression procedure, resulting in about 600 samples in the residuals data set. Because it is preferable to have more samples when fitting SVMs, and because there is little difference in using the residuals, we will use the *original* data set instead.

Chapter 4

Methods and approaches

This chapter outlines the feature selection algorithms with which this project has experimented. Chapter 5 will then present and discuss the numerical results that these algorithms have produced.

4.1 t-test

We first explore one popular feature selection method, the t-test. The t-statistic is commonly used to determine if the means of two populations are equal. In the context of feature selection, it can be used to quantify the *relevance* of each feature with respect to the target labels.

The t-statistic and its corresponding p-value can be computed using the `ttest_ind` method in the `scipy.stats` [107] library. For example, we can calculate the t-statistic between two groups, 0 and 1. Using this method, the null hypothesis is that the means of groups 0 and 1 (μ_0 and μ_1 respectively) are equal. That is,

$$H_0 : \mu_0 = \mu_1 \quad \text{vs.} \quad H_1 : \mu_0 \neq \mu_1$$

`ttest_ind` will then perform a two-sided test, and return the value of the t-statistic and its p-value. Given two groups of samples X_0 and X_1, the t-statistic T is:

$$T = \frac{\bar{x}_0 - \bar{x}_1}{\sqrt{\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}}}$$

where, for $i \in \{0, 1\}$, $\bar{x}_i$ denotes the mean value of X_i, n_i denotes the number of samples in X_i, and s_i^2 denotes an estimate of the variance of X_i. Then, under the null hypothesis, T has a t-distribution, with ν degrees of freedom [105], where

$$\nu = \frac{\left(\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}\right)^2}{M_0 + M_1}$$

$$M_0 = \frac{\left(s_0^2/n_0\right)^2}{n_0 - 1} \quad \text{and} \quad M_1 = \frac{\left(s_1^2/n_1\right)^2}{n_1 - 1}$$

4.1.1 Interpretation of t-test

In the context of feature selection, for each feature, we have to group the samples according to their classification under this feature. This means that for each feature, samples whose classification is 0 will belong to group 0, and samples whose classification is 1 will belong to group 1.

Under the null hypothesis $H_0 : \mu_0 = \mu_1$, the means of these two groups are the same. This implies that the feature values have no influence on the classification of the samples. Using `ttest_ind`, a low p-value means that we have enough evidence to *reject* H_0. Rejecting H_0 means that we are certain that a feature has significant contribution to the classification and is therefore highly relevant to the target labels. Thus, this method will select features whose p-values are significantly *low*.

Once we have the p-values of all the features, we can decide how to use these values to select the features that we desire. We can either select the first k features that have the *lowest* p values (i.e. most certain that $\mu_0 \neq \mu_1$), or choose a certain threshold and discard those features whose p-values exceed this threshold.

4.1.2 Relevance and redundancy

Evidently, since the SVM is not involved in the feature ranking process and since each feature is considered one at a time, we can classify this method as a *univariate filter* method. As mentioned in section 2.7.3, univariate filters are computationally efficient. In this case, the method iterates through each feature only once. The efficiency of this method can be further enhanced by tapping on its inherent parallelism, since we can compute the t-statistic of each feature independently from the others (see section 4.7).

Using `ttest_ind`, the t-statistic and p-values were appended to a list L, and sorted in ascending order of p-value. Once we have obtained the sorted list, selecting k features means we simply need to select the top k features, an operation that is computationally cheap.

However, one stark drawback of this method is that t-test only considers each feature singly. Even though its design is simple compared to multivariate filters or wrappers, it is not possible to take redundancy into account [54], at least not in its basic implementation. t-test ranks the features according to the t-statistic (or the p-value), but redundant features are known to have similar rankings [38]. As such, we might end up selecting highly ranked, but similar, features in our classification. For example, [108] describes how redundant genes (features) might belong to the same biological pathway, when actually only a subset of those genes is needed to achieve the same prediction accuracy.

4.1.3 Variation of t-test: A lazy version of Sequential Forward Selection

It is interesting to see if ranking each feature according to its individual cross validation (CV) score will differ significantly from the t-test method. In other words, instead of ranking features based on their t-statistic, we rank them based on their *individual* CV

score. In this context, individual CV score refers to the CV score obtained by the SVM when it is fitted with only one feature. Since this is a greedier and lazier version of Sequential Forward Selection (SFS), we call this method lazy SFS. Furthermore, unlike t-test, this variation of SFS can be classified as a wrapper method, since it uses the SVM to evaluate the features.

First, we iterate through each feature and fit an SVM using just that feature. So, the SVM is trained on one feature at a time only. Then, a list of CV scores is obtained. We then sort this list in descending order. Similarly to t-test, selecting k features means selecting the top k features in the sorted list. We also note that the lazy SFS also suffers from the same shortcoming as t-test, as both do not take redundancy of features into account.

4.2 Recursive feature elimination

Next, we consider the RFE method, which was relatively straightforward to implement, using the `RFE` method in the `sklearn.feature_selection` package. This method provides a convenient way to specify the number of "steps" that the algorithm should take when eliminating features that are deemed unimportant. This parameter specifies the number of features to eliminate in each iteration. As what was discussed in section 2.7.9, specifying a step size larger than 1 will cause the algorithm to consider nested subsets in each iteration.

The Python `feature_selection` package in `sklearn` offers an RFE method:

```
RFE(estimator, n_features_to_select=None, step=1, verbose=0)
```

to perform RFE on our data set. We can specify the `step` parameter to be larger than 1 to ensure that the algorithm does not take too long to compute. With step size of 1, the algorithm will iterate through each feature, and remove only one feature at a time. It will then eliminate the feature with the lowest weight. This is obviously computationally intensive, especially if we specify a small `n_features_to_select` parameter. Thus, the step size will vary according to the `n_features_to_select` parameter. This is discussed in detail in section 5.2.

4.3 Restricted Forward Selection (RFS)

The pseudocode of the RFS algorithm [62] is listed in algorithm 1 on page 20. The RFS is a greedy heuristic search that only searches the features that are the most promising.

While SFS incrementally selects the best feature to include in our subset of selected features S by considering *all* remaining features that are not in S, RFS only considers a *part* of these remaining features. The features that RFS consider in each iteration are those that produce the best CV score individually.

Suppose there are M features in total. Firstly, RFS iterates through each feature and records its CV score by fitting an SVM with just one feature. A list L is used to keep track of these scores. L is then sorted. The feature with the best CV score is selected

Iteration	Subsets considered	Subset selected S
1	$f_1, f_2, \ldots, f_M$	$\{f_3\}$
2	$\{f_3, f_1\}, \{f_3, f_2\}, \{f_3, f_4\}, \ldots, \{f_3, f_{M/2}\}$	$\{f_3, f_1\}$
3	$\{f_3, f_1, f_2\}, \{f_3, f_1, f_4\}, \ldots, \{f_3, f_1, f_{M/3}\}$	$\{f_3, f_1, f_5\}$

Table 4.1: A diagram demonstrating how the first three iterations of RFS work. Suppose the features f_3, f_1, f_5 are the top three features.

as the first feature (call it f_1). Then, in the second iteration, RFS iterates only half of L and pairs each remaining feature with f_1 to form a pair, and an SVM is fitted using this pair of features. The best pair is then selected. In the third round, RFS will iterate through $M/3$ of these features. The best triple is selected and the process continues until k features are selected. An example demonstrating this concept is shown in Table 4.1.

Since the RFS algorithm uses the SVM, it can be classified as a *wrapper* method. Furthermore, since RFS uses the CV score as a heuristic to guide its search, it does not explicitly quantify relevance and redundancy.

4.3.1 Runtime

In the first iteration of the algorithm, all M features are evaluated. In the second iteration, $M/2$ features are evaluated. Suppose this continues for 2^n iterations.

Then, we can calculate the number of features the algorithm evaluates:

$$M + \frac{M}{2} + \frac{M}{3} + \cdots + \frac{M}{2^n} = M\left[1 + \frac{1}{2} + \cdots + \frac{1}{2^n}\right] = M\sum_{j=1}^{2^n}\frac{1}{j}$$

It is a well-known result that the series $\sum_{i=1}^{\infty} 1/i$ diverges, but we can attempt to obtain an upper bound of the sum above. We can split the sum above into n groups, where the number of elements in the p-th group is 2^{p-1}.

$$\begin{aligned}
\sum_{j=1}^{2^n}\frac{1}{j} &= 1 + \underbrace{\left[\frac{1}{2} + \frac{1}{3}\right]}_{\text{Group 2}} + \underbrace{\left[\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}\right]}_{\text{Group 3}} + \cdots + \underbrace{\left[\frac{1}{2^{n-1}} + \frac{1}{2^{n-1}+1} \cdots + \frac{1}{2^n - 1}\right]}_{\text{Group } n} + \frac{1}{2^n} \\
&< 1 + \left[\frac{1}{2} + \frac{1}{2}\right] + \left[\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}\right] + \cdots + \left[\frac{1}{2^{n-1}} + \cdots + \frac{1}{2^{n-1}}\right] + \frac{1}{2^n} \\
&= 1 + 1 + 1 + \cdots + 1 + \frac{1}{2^n} \\
&= n + \frac{1}{2^n}
\end{aligned}$$

Now, let $k = 2^n$, so $k > 0$ and $n = \log k$ (the logarithm here implicitly has base 2).

$$\sum_{j=1}^{k} \frac{1}{j} < \log k + \frac{1}{k} < \log k$$

Thus, the total number of evaluations made by RFS is

$$M \sum_{j=1}^{k} \frac{1}{j} < M \log k$$

where k is the number of features that the algorithm selects. So, the RFS algorithm is executed in $\mathcal{O}(M \log k)$ (ignoring the complexity of the SVM classification).

In contrast, the SFS algorithm iterates through the whole feature set in each iteration. So, its complexity would be $\mathcal{O}(Mk)$. In the limit, if we want SFS to iterate through all the features, instead of terminating with k features, the runtime complexity of SFS would be $\mathcal{O}(M^2)$ while that of RFS is $\mathcal{O}(M \log M)$.

It is evident that since RFS is a greedier variant of SFS, the solution returned by RFS would most likely be less optimal than that by SFS. However, we need to find a balance between computational complexity and optimality of our solution. As M is large in this case, the time complexity of the RFS would be helpful in selecting k features within a reasonable amount of time despite its compromise in optimality.

4.4 Maximum relevance minimum multi-collinearity (MRmMC)

Recently, MRmMC [6] was proposed as a filter method to overcome the drawbacks and computational efforts required when using mutual information as a heuristic to quantify relevance and redundancy. MRmMC also does not require any parameter. Unlike RFE and RFS, MRmMC does not directly use CV scores to quantify relevance and redundancy.

4.4.1 Relevance

MRmMC uses a correlation coefficient, first used in [109], to quantify the relevance of each feature to the target class. For random variables X and Y, this correlation coefficient is:

$$r_{qn}(X, Y) = \left(1 - \frac{E_Y[\mathrm{var}_X(X|Y)]}{\mathrm{var}(X)}\right)^{\frac{1}{2}}$$

This measure is motivated by some definitions and properties of conditional probabilities:

Definition 4.4.1.1 (Expectation)**.** *The expectation of a discrete random variable X is:*

$$E[X] = \sum_{x \in X} xP(X = x)$$

Definition 4.4.1.2 (Marginal expectation)**.** *The marginal expectation of a discrete random variable X, in terms of its conditional expectation given another discrete random variable Y, is:*

$$E[X] = \sum_{y \in Y} P(Y = y)\, E_X[X \,|\, Y = y]$$

where E_X denotes expectation with respect to X. That is,

$$E_X[X \,|\, Y = y] = \sum_{x \in X} x\, P(X = x \,|\, Y = y)$$

Definition 4.4.1.3 (Marginal variance)**.** *The marginal variance of X is:*

$$var(X) = E[X^2] - (E[X])^2$$

The definition of marginal variance can be extended to the conditional variance.

Definition 4.4.1.4 (Conditional variance)**.** *The conditional variance of X, given Y, is:*

$$var_X(X \,|\, Y) = E_X[X^2 \,|\, Y] - (E_X[X \,|\, Y])^2$$

Since the conditional variance is a random variable with respect to Y, we can find the expectation of the quantity:

$$E_Y[\mathrm{var}_X(X \,|\, Y)] = E_Y E_X[X^2 \,|\, Y] - E_Y[E_X(X \,|\, Y)]^2 \tag{4.1}$$

By definition 4.4.1.1, we can show that:

$$\begin{aligned} E_Y E_X[X \,|\, Y] &= \sum_{y \in Y} P(Y = y)\, E_X[X \,|\, Y = y] = E[X] \qquad (4.2) \\ \Rightarrow E_Y E_X[X^2|Y] &= E[X^2] \end{aligned}$$

From equation 4.1,

$$\Rightarrow E_Y[\mathrm{var}_X(X \,|\, Y)] = E[X^2] - E_Y[E_X(X \,|\, Y)]^2 \tag{4.3}$$

Now, we also consider the variance with respect to Y of the conditional expectation:

$$\begin{aligned} \mathrm{var}_Y(E_X[X \,|\, Y]) &= E_Y\left(E_X[X \,|\, Y]\right)^2 - \left(E_Y E_X[X \,|\, Y]\right)^2 \\ &= E_Y\left(E_X[X \,|\, Y]\right)^2 - (E[X])^2 \qquad (4.4) \end{aligned}$$

where we used the result from equation 4.2. Then, rearranging equation 4.4 and combin-

ing with equation 4.3, we get:

$$\begin{aligned} E_Y[\text{var}_X(X \mid Y)] &= E[X^2] - \text{var}_Y(E_X[X \mid Y]) - (E[X])^2 \\ &= \text{var}(X) - \text{var}_Y(E_X[X \mid Y]) \\ \Rightarrow \text{var}(X) &= \text{var}_Y(E_X[X \mid Y]) + E_Y[\text{var}_X(X \mid Y)] \end{aligned}$$

where we used the definition of marginal variance in definition 4.4.1.3.

Thus, as discussed in [6], the variance of the random variable X is made of two parts: $\text{var}_Y(E_X[X \mid Y])$ and $E_Y[\text{var}_X(X \mid Y)]$. The latter describes the average variability with respect to different targets: on average, how much do the feature values vary with different targets?

Using X as our feature and Y as the target label, $E_Y[\text{var}_X(X|Y)]$ is used in measuring the relevance of the feature and its classification.

4.4.2 Properties of correlation coefficient

A mathematically convenient property of the correlation coefficient

$$r_{qn}(X,Y) = \left(1 - \frac{E_Y[\text{var}_X(X|Y)]}{\text{var}(X)}\right)^{1/2}$$

is that $0 \leq r_{qn}(X,Y) \leq 1$. First, we define independence between two variables:

Definition 4.4.2.1 (Independence)**.** *Two random variables X and Y are independent iff*

$$P(X,Y) = P(X)\,P(Y) \tag{4.5}$$

where $P(X,Y)$ denotes the joint probability of X and Y.

In $r_{qn}(X,Y)$,

$$\begin{aligned} E_Y[\text{var}_X(X|Y)] &= E[X^2] - E_Y\left(E_X(X|Y)\right)^2 \quad \text{(by eqn 4.3)} \\ &= E[X^2] - \sum_{y\in Y} P(Y=y)\left[E_X(X|Y=y)\right]^2 \quad \text{(by defn 4.4.1.2)} \\ &= E[X^2] - \sum_{y\in Y} P(Y=y)\left[\sum_{x\in X} x\,P(X=x|Y=y)\right]^2 \\ &= E[X^2] - \sum_{y\in Y} P(Y=y)\left[\sum_{x\in X} x\,\frac{P(X=x,Y=y)}{P(Y=y)}\right]^2 \end{aligned}$$

Assuming X and Y are independent, we can apply equation 4.5:

$$\begin{aligned}
E_Y[\text{var}_X(X|Y)] &= E[X^2] - \sum_{y\in Y} P(Y=y)\left[\sum_{x\in X} x\,\frac{P(X=x)\,P(Y=y)}{P(Y=y)}\right]^2 \\
&= E[X^2] - \sum_{y\in Y} P(Y=y)\left[\sum_{x\in X} x\,P(X=x)\right]^2 \\
&= E[X^2] - \underbrace{\sum_{y\in Y} P(Y=y)}_{=1}\,[E(X)]^2 \\
&= \text{var}(X) \\
\Rightarrow r_{qn}(X,Y) &= 0
\end{aligned}$$

So, $r_{qn}(X,Y)$ attains its lower bound when X and Y are *independent*. On the other hand, $r_{qn}(X,Y)$ approaches 1 when X and Y are strongly correlated, according to [6]. In the limit, suppose X is a function of Y: $Y = f(X)$. Then, given $Y = y$, we can find the value of $X = x$ and vice versa. In other words, we can deterministically find the value of one variable given the other. Thus,

$$\begin{aligned}
E_Y[\text{var}_X(X|Y)] &= E_Y[E_X(X^2|Y) - (E_X(X|Y))^2] \quad \text{(by defn 4.4.1.3)} \\
&= \sum_{y\in Y} P(Y=y)\left[E_X(X^2|Y=y) - (E_X(X|Y=y))^2\right] \\
&= \sum_{y\in Y} P(Y=y)\left[E_X\left(x^2\right) - \left(E_X\left(x\right)\right)^2\right] \\
&= \sum_{y\in Y} P(Y=y)\left[x^2 - x^2\right], \quad \text{since } x \text{ is non-random} \\
&= 0 \\
\Rightarrow r_{qn}(X,Y) &= 1
\end{aligned}$$

As such, MRmMC aims to find features in the set F, that are strongly relevant to the target class, by *maximising* r_{qn} between each feature f and the target labels c. That is,

$$\underset{f\in F}{\arg\max} \quad r_{qn}(f,c)$$

4.4.3 Computing relevance

Now, we see how the terms in r_{qn} can be computed using the raw data. From equation 4.3,

$$\begin{aligned}
E_Y[\text{var}_X(X\,|\,Y)] &= E[X^2] - E_Y[E_X(X\,|\,Y)]^2 \\
&= E[X^2] - \sum_{y\in Y} P(Y=y)\,[E_X(X|Y=y)]^2
\end{aligned}$$

In the context of our data, X represents the features and Y represents the target labels. For our data, binary target classes are used: $Y = \{0, 1\}$. So, we can expand the above:

$$E_Y[\text{var}_X(X \mid Y)] = E[X^2] - P(Y = 1)\left[E_X(X|Y = 1)\right]^2 - P(Y = 0)\left[E_X(X|Y = 0)\right]^2$$

The expectations above and the variance term in the denominator of r_{qn} can be computed using unbiased estimators of expectation and variance respectively:

$$E(X) = \bar{x} \approx \frac{1}{n}\sum_{x \in X} x \quad \text{and} \quad \text{var}(X) \approx \frac{1}{n-1}\sum_{x \in X}(x - \bar{x})^2$$

To compute the conditional expectation $E_X(X|Y = y)$, we simply compute the expectation of those realisations of X whose corresponding realisation of Y is y. To compute $P(Y = y)$, we simply have to count how frequent y appears as a realisation of Y throughout the cases.

A simple example below shows how we could compute the probabilities discussed above:

Case	X	Y
1	0.5	1
2	0.2	0
3	0.1	1

In this case,

$$\begin{aligned}
E_X(X|Y = 1) &= \frac{1}{2}(0.5 + 0.1) = 0.3 \quad \text{(Expectation of } X \text{ over cases 1 and 3)} \\
P(Y = 1) &= \frac{2}{3} \\
E[X^2] &= \frac{1}{3}(0.5^2 + 0.2^2 + 0.1^2) = 0.1 \\
E[X] &= \frac{1}{3}(0.5 + 0.2 + 0.1) \approx 0.2666 \\
\text{var}(X) &= E[X^2] - (E[X])^2 \approx 0.02892
\end{aligned}$$

As such, we can employ the same method to compute the required expectations and variances for our data. To make this task simpler, we utilise methods in the `numpy` [96] package: the `mean` and `var` methods are used to compute expectation and variance respectively.

4.4.4 Redundancy and multicollinearity

Multicollinearity, in the context of feature selection, refers to a strong correlation between features [110].

MRmMC uses the *squared multiple correlation coefficient* (call it R^2) to quantify the degree of redundancy between features. The R^2 term between a dependent variable and an independent variable can be interpreted as the *proportion of the variance of the dependent variable explained by the independent variable* [111]. In other words, if R^2 is

high, then a high proportion of the dependent variable's variance can be extracted from the independent variable. Therefore, the dependent variable is highly redundant.

Assume there exists a set of vectors $Q = \{q_1, q_2, \ldots, q_k\}$ where the vectors are *pairwise orthogonal*. That is, $\forall i, j = 1, \ldots, k$,

$$q_i \cdot q_j = q_i^\top q_j = 0, \quad i \neq j$$

A useful property of R^2 is that, the R^2 term between a variable f and Q is the *sum* of R^2 between each $q \in Q$ and f [111]. That is,

$$R^2(Q, f) = \sum_{i=1}^{k} R^2(q_i, f), \quad q_i \in Q$$

This is useful when computing R^2 between one variable and a *group* of pairwise orthogonal variables.

In [6], the R^2 term used in MRmMC is defined as such: Suppose there exists a feature set S with features $\{f_1, \ldots, f_{k-1}\}$, with corresponding orthogonal components $Q = \{q_1, \ldots, q_{k-1}\}$, where $f_i, q_i \in \mathbb{R}^n$ for $i = 1, \ldots, k-1$. Suppose we are considering a new feature f. Then, for any orthogonal component $q \in Q$,

$$R^2(f, q) = \frac{\left(\sum_{i=1}^{n} f_i q_i\right)^2}{\left(\sum_{i=1}^{n} f_i^2\right)\left(\sum_{i=1}^{n} q_i^2\right)} \tag{4.6}$$

where f_i, q_i refers to the i-th element of the vectors f and q respectively, where $i = 1, \ldots, n$.

Since the components of Q are pairwise orthogonal, the R^2 term of the new vector f and all the components of Q would be the sum of the R^2 terms:

$$R^2(f, Q) = \sum_{i=1}^{k-1} R^2(f, q_i), \quad q_i \in Q$$

Furthermore, R^2 can be interpreted as the square of the *Pearson correlation coefficient* ρ of vectors $x, y \in \mathbb{R}^n$:

$$\rho(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $\bar{x}$ and $\bar{y}$ denote the mean of x and y respectively. We can see that the square of ρ would equal to the R^2 term in equation 4.6 when the means of f and q are zero. Thus, we need to ensure that f and q have zero mean for the above interpretation to work.

As mentioned above, since we would like to reduce the redundancy between features, we would need to *minimise* R^2 between f and the previously selected features.

Therefore, for an orthogonal set Q and target labels c, the algorithm will select feature

f that maximises the following criterion:

$$J(f, Q) = r_{qn}^2(f, c) - \sum_{i=1}^{k-1} R^2(f, q_i), \quad q_i \in Q$$

where the orthogonal set Q is obtained by orthogonalising the set of selected features.

Note that the r_{qn} term is squared so that both terms in the equation are squared. Overall, the pseudocode for MRmMC can be found in Algorithm 2.

Algorithm 2: MRmMC(D, k)

Input: Data set D with M features.
Output: Subset S ($S \subset D$) of the M features such that $|S| = k$, $k \ll M$.

begin
- Initialise empty list $S \longleftarrow [\,]$
- Initialise empty list $Q \longleftarrow [\,]$
- **for** $i = 1 \longrightarrow M$ **do**
  - Compute $r_{qn}^2(f_i, c)$ for i-th feature, f_i and target labels c.
- $w \longleftarrow$ feature with maximum r_{qn}^2.
- $S \longleftarrow S \cup \{w\}$
- $Q \longleftarrow Q \cup \{w\}$
- **for** $h = 2 \longrightarrow k$ **do**
  - **for** *each feature* f_i *not in* S **do**
    - Compute $J(f_i) = r_{qn}^2(f_i, c) - \sum_{j=1}^{h-1} R^2(f_i, q_j)$, where $q_j \in Q$.
  - $w \longleftarrow$ feature with maximum J.
  - $S \longleftarrow S \cup \{w\}$
  - $Q \longleftarrow Q \cup \{w\}$
  - Orthogonalise Q.

4.4.5 Computing redundancy

The MRmMC paper [6] derives the orthogonal vector q_k of f_k with respect to a set of orthogonal vectors $Q = \{q_i : i \in [1, k-1]\}$ via:

$$q_k = f_k - \frac{f_k^\top q_1}{q_1^\top q_1} q_1 - \frac{f_k^\top q_2}{q_2^\top q_2} q_2 - \cdots - \frac{f_k^\top q_{k-1}}{q_{k-1}^\top q_{k-1}} q_{k-1}$$

This is evidently the *classical Gram-Schmidt* (CGS) process. However, CGS is notable for its numerical instability because there is usually a loss of orthogonality in the resulting q_k vectors [112]. Instead, other methods such as the *modified Gram-Schmidt* (MGS) or *Householder transformations* are preferred.

The computation of Q was performed using the `numpy.linalg.qr` method. This method returns an *orthonormal* matrix and an upper triangular matrix $\boldsymbol{R}$ that does not play a role

in computing R^2. An orthonormal matrix has orthonormal columns, which are orthogonal columns with a magnitude (2-norm) of 1. This would not affect the computation of R^2:

Proof. Suppose we have an arbitrary vector $q \in \mathbb{R}^n$ and a normalised vector $\tilde{q} \in \mathbb{R}^n$ such that $\tilde{q} = q/\|q\|_2$. Then,

$$R^2(f,\tilde{q}) = \frac{\left(\sum_{i=1}^n f_i\tilde{q}_i\right)^2}{\left(\sum_{i=1}^n f_i^2\right)\left(\sum_{i=1}^n \tilde{q}_i^2\right)} = \frac{\left(f^\top \tilde{q}\right)^2}{(f^\top f)(\tilde{q}^\top \tilde{q})} = \frac{\frac{1}{\|q\|_2^2}\left(f^\top q\right)^2}{(f^\top f)\cdot\frac{1}{\|q\|_2^2}(q^\top q)} = R^2(f,q)$$

□

Thus, the computation of the R^2 term using orthonormal vectors will still be valid.

4.4.6 Implementing MRmMC

Two versions of MRmMC were implemented. The first incorporates the idea of Restricted Forward Selection (RFS). This truncated version only considers part of the features that have the highest r_{qn} (i.e. the top k features with the best individual r_{qn}), whereas the full version considers every feature that has not been already selected.

The truncated version considers all of the features in its first iteration, half of the features in the second iteration, one third of the features in the third iteration and so on. This is similar to RFS. Evidently, this will cause optimality to suffer, but the computation process will terminate much quicker.

The results are discussed in chapter 5.

4.5 MRmMC issues

The challenges of implementing MRmMC arise mainly due to the high dimensionality of the data set. In the MRmMC paper [6], the data sets that were experimented on had at most 216 features, and had many more samples than features. For example, one data set had 2000 samples and 216 features.

4.5.1 Problems with computing redundancy

From algorithm 2, we can see that the matrix of orthogonal vectors Q expands with the number of features considered in the step $Q \longleftarrow Q \cup \{w\}$. For example, after selecting the first feature, $Q \in \mathbb{R}^{n\times 1}$, where n is the number of samples in the data set. In the next iteration, a vector is appended to Q such that $Q \in \mathbb{R}^{n\times 2}$ and Q is converted to an orthonormal matrix. This continues until we reach the desired number of features k.

However, from [113], we note that this orthogonalisation process can be interpreted as constructing an *orthonormal basis* for a *subspace* of $\mathbb{R}^n$.

Suppose the set Q contains n feature vectors. That is, $Q \in \mathbb{R}^{n \times n}$. Then, the vectors of Q can be interpreted as a basis for the space $\mathbb{R}^n$. In other words, any vector $u \in \mathbb{R}^n$ can be formed by a linear combination of the vectors in Q.

Consequently, attempting to append another vector $v \in \mathbb{R}^n$ and orthogonalising the resulting matrix will render v redundant, as the space $\mathbb{R}^n$ can be constructed solely based on the previous n vectors in Q, without the additional v. Thus, we will not be able to orthogonalise a matrix $Q \in \mathbb{R}^{n \times (n+1)}$. This problem will occur if the number of features to be selected by the feature selection algorithm exceeds the number of samples in the data set. Indeed, we see that in the original MRmMC paper, MRmMC was performed on data sets which have fewer features than samples, unlike the case for our data set. So, this problem will not occur for the data sets used in the MRmMC paper.

This means that, for our data set, the algorithm should only select at most 847 features. The result of choosing 847 features is discussed in chapter 5. This limitation will put a heavy restriction on the feature selection process. Thus, instead of using the R^2 measure for redundancy, we can instead incorporate genetic algorithms while using the r_{qn} quantity.

4.5.2 Computational complexity

We first note that the MRmMC method is a filter method since it does not employ the SVM at all. A common benefit of filter methods is its computational efficiency. For example, the t-test only needs to compute the t-statistic of each feature once. Despite its classification as a filter method, MRmMC is not as computationally efficient, especially when a lot of features are involved.

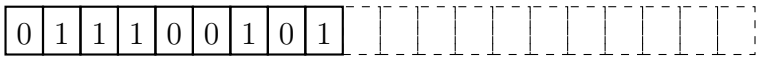
Computational effort is mostly spent in computing the QR factorisation of the selected features. This is for the purpose of quantifying the redundancy among the selected features. According to [113], the CGS algorithm used to compute the orthgonal vectors has complexity $\mathcal{O}(mn^2)$ for a matrix $Q \in \mathbb{R}^{m \times n}$. That is, the complexity of the algorithm grows quadratically with features, versus the linear time complexity offered by t-test. So, with more features, the algorithm will take a longer time to compute the QR factorisation of the matrix Q.

Thus, not using the R^2 term will allow us to avoid this computational complexity, which gives us a stronger reason to use genetic algorithms instead.

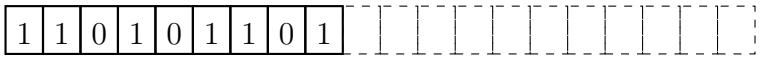
4.6 Integrating genetic algorithm with MRmMC

As mentioned in section 2.8, genetic algorithms (GAs) are popular in the literature of feature selection. The GA implemented in this project defines the *fitness* of the chromosomes based on their CV scores. The r_{qn} measure is also used in a two-stage variant of the GA.

(a) First stage of GA. Suppose the entire array represents all the features. The solid boxes represent those features that have high r_{qn} and thus considered for the GA, while the dotted boxes are those features that are not considered for the GA. Hence, only the features represented by the solid boxes will appear in a solution of the GA.



(b) Second stage of GA. A solution is generated by assigning either 0 or 1 to each feature. This diagram shows how $k = 5$ features are initially assigned "1". This means that the feature is selected. Evolution of a solution is thus equivalent to inverting the bits of the features.



(c) An example of how the solution in Figure 4.1b might evolve. Note that the number of features selected is no longer 5, as the genetic operators do not necessarily ensure that there are always $k = 5$ features selected.

Figure 4.1: A diagram demonstrating how the two-stage GA works.

4.6.1 Parameters and strategies

Due to the large number of features in the epigenetics data set, this project uses the "two-stage" GA, as illustrated in Figures 2.4 and 4.1.

We first use the correlation coefficient r_{qn} to choose the best features in the first stage of the GA. Alternatively, we could also use the classification score of each feature on an SVM as a guide to eliminate features that might not be relevant to the target class, as used in RFS. We choose the former. Once we have a list of the r_{qn} values, we sort it, and obtain the top s features. This set of features is the result of the initial filtering of the algorithm. This forms the first stage of the GA, as shown in Figure 4.1a.

Then, from the top s features, we randomly select k features that will form the solutions in the first generation of the GA. The presence of a feature is encoded by a '1', while its absence is encoded by a '0'. This is done until there are G solutions in the generation. This is shown in Figure 4.1b.

Figure 4.1 demonstrates this two-stage GA with 20 features in total. In this case, $M = 20$, $s = 9$ and $k = 5$.

Next, the *elitism/cloning* strategy will be used when creating a new generation of chromosomes, as performed in [85]. This means that we copy the best solutions in each generation to the next. This guarantees that the maximum fitness of the next generation will not decrease. The idea of elitism is enforced as follow:

- Clone a fraction p_e of the solutions that have the best fitness in each generation to the next generation.
- Select the next p_r of the solutions with the best fitness, mutate each solution and add to the next generation.

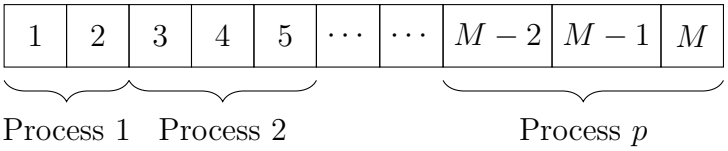


Figure 4.2: An illustration of how the work of evaluating M independent features can be distributed among p processes.

- Remove p_{elim} of the solutions with the worst fitness.

Crossover is then done on the solutions that remain. This will be done based on the *Roulette Wheel selection* and the *tournament selection*.

The literature also offers different stopping criteria for the GA. For example, [81] stops the GA after a fixed number of generations, while [85] stops the algorithm when there is no change in the fitness function in 30 generations. We experiment with the latter, where the algorithm terminates if the fitness increases by less than t for T generations.

In summary, the parameters used in our GA are summarised in Figure 5.3 in chapter 5. The pseudocode of our GA is listed in algorithm 3.

4.7 Utilising parallelism

In [114], an *embarrassingly parallel* problem is described as one that can be "*divided into components that can be executed concurrently*". Similarly, we find that we can utilise the fact that the search space in some of the methods above can be decomposed into several independent subproblems.

For example, the t-test method described in section 4.1 iterates through each feature to compute the t-statistic and p-value for the feature. Thus, the computation for one feature is independent of the computation for all other features. Hence, we can take advantage of this parallelism, by splitting the work into parts. This concept is illustrated in Figure 4.2, where p processes each computes a group of features.

For example, in the computation of the t-statistic of each feature, 10 processes were used to distribute the computational work. A pool of worker processes was created with the `Pool` class in the `multiprocessing` package in Python. Next, a `partition_list` method was written to divide the search space into groups of features for the processes to work on. The `Pool.map` method was then used. This maps a method - in this case, the computation of the t-statistic - on each of the partitions. The result is an improvement in runtime, from 263 seconds to 94 seconds.

As far as possible, this idea was extended to the other methods, where it is possible to conveniently divide the search space into independent segments. For example, RFS (see algorithm 1) can be parallelised in both of the `for` loops.

Algorithm 3: Genetic algorithm(D, k, s, G, T, P_e, P_m, P_r, P_{elim})

Input: Data set D. See section 4.6.1 for explanation of GA parameters.
Output: Subset S ($S \subset D$) which produces the best fitness.

begin
- Initialise empty list $R \longleftarrow [\,]$
- **for** $i = 1 \longrightarrow M$ **do**
  - $r \longleftarrow$ Calculate r_{qn} for the ith feature.
  - $R \longleftarrow R \cup \{r\}$
- Sort R in descending order.
- $initial\_candidates \longleftarrow$ top s elements of R.
- Initialise empty list $gen\_solution \longleftarrow [\,]$
- Initialise empty list $initial\_fitness \longleftarrow [\,]$
- **for** $i = 1 \longrightarrow G$ **do**
  - $g \longleftarrow$ Array of k random 1's. Fill the rest of the array with 0's.
  - $f \longleftarrow$ Evaluate fitness of g.
  - $initial\_fitness \longleftarrow initial\_fitness \cup \{f\}$
  - $gen\_solution \longleftarrow gen\_solution \cup \{g\}$
- $best\_fitness \longleftarrow$ Top fitness in $initial\_fitness$.
- $consecutive\_gen \longleftarrow 0$
- **while** $consecutive\_gen < T$ **do**
  - Initialise empty list $next\_gen\_solution \longleftarrow [\,]$
  - $e \longleftarrow$ Top p_e of solutions in $gen\_solution$.
  - $r \longleftarrow$ Next p_r of solutions in $gen\_solution$.
  - Perform *mutate* on each solution in r.
  - $next\_gen\_solution \longleftarrow next\_gen\_solution \cup \{e, r\}$
  - Remove last p_{elim} of solutions in $gen\_solution$.
  - **while** $|next\_gen\_solution| < G$ **do**
    - Select 2 parents from the rest of solutions in $gen\_solution$.
    - Perform *crossover* and produce 2 offsprings.
    - Append offsprings to $next\_gen\_solution$.
  - $current\_best\_fitness \longleftarrow$ Top fitness in $next\_gen\_solution$.
  - **if** $|\frac{current\_best\_fitness - best\_fitness}{best\_fitness}| \leq t$ **then**
    - $consecutive\_gen \longleftarrow consecutive\_gen + 1$
  - **else**
    - $consecutive\_gen \longleftarrow 0$
  - $best\_fitness \longleftarrow current\_best\_fitness$
  - $gen\_solution \longleftarrow next\_gen\_solution$
- $S \longleftarrow$ Features corresponding to the solution with $best\_fitness$.

4.8 High performance computing

Most of the computation for this project was performed using the High Performance Computing (HPC) cluster at Imperial College [104]. In particular, the `cx1` general purpose computer cluster was used and the `PBSPro` queueing system was used to manage the jobs submitted by users. The `cx1` system provided a convenient and highly efficient means to perform computations that required a long time or a large memory demand.

Users of the HPC cluster are provided with three directories: `$WORK`, `$HOME` and `$TMPDIR`. `$WORK` and `$HOME` are directories that allow users to store data and files. On the other hand, `$TMPDIR` is a directory for submitting jobs to the PBS system.

A job is submitted to the system by submitting a job script that first specifies the walltime and memory that the job expects. The job script then lists bash commands, which, for example, copies files and runs Python programs. Most of the Python programs in this project write numerical results to a text file. These text files are then copied to the `$WORK` directory, from which we can copy to our local machines.

Furthermore, the cluster allowed *serial jobs* to be conducted. That is, jobs that require multiple executions can be submitted and the program will be run for a specified number of times automatically. This feature was especially convenient for the execution of the GAs. 10 runs of the GA were performed for each of the selection strategies (roulette wheel and tournament), and serial execution made this task much easier.

Chapter 5

Experimental results and evaluation

5.1 t-test

Plots were obtained from t-test using different experimental settings:

- Sample k in broad intervals to investigate the general trend of t-test (Figure 5.1).
- Sample $k \in [10, 2000)$ with step size of 10 (Figure 5.2).
- Magnified view of Figure 5.2, where $k \in [10, 2000)$ (Figure 5.3).

where k is the number of features selected.

When plotting such graphs, we also investigate if the selected subset of features S causes the SVM to suffer significant performance degradation as compared to using all the features. To this end, a horizontal dotted line can be found in the graphs. This dotted line represents a *threshold*, which is the linear SVM's performance when using all of the features.

5.1.1 Broad intervals of k

The first experiment broadly samples k from a range of values. The values of k and the corresponding cross validation (CV) scores are shown in Table 5.1. For this table of values, we sampled k from 1 to 420374 (the full set of features).

As mentioned in section 4.1, the t-test allows us to generate a list of p-values and we then sort the p-values in ascending order. For each k, we will select the top k features from the sorted list, fit an SVM with the k features, and obtain CV scores. For example, when $k = 500$, we will select the top 500 features in the sorted list.

From Figure 5.1, we notice that the CV score peaks at around $k = 15000$ with a score of 0.950448, and after this value of k, the CV score exhibits a decreasing trend. Evidently, due to the coarse granularity of k that we have sampled, this method might have missed out on some local maxima after $k = 15000$. However, because the time required to fit an SVM and obtain CV scores increases with the number of features, it is perhaps sufficient to obtain a broad, general trend of t-test.

k	1	5	10	20	50	80	100
CV score	0.7001	0.837137	0.848937	0.863068	0.846479	0.861801	0.867704
k	250	500	750	1000	2000	5000	8000
CV score	0.882004	0.893754	0.907942	0.907921	0.93035	0.943376	0.949279
k	10000	**15000**	20000	40000	80000	100000	150000
CV score	0.950441	**0.950448**	0.946877	0.949258	0.946891	0.942171	0.939783
k	250000	300000	400000	420374			
CV score	0.938628	0.937444	0.932703	0.92916			

Table 5.1: Table of CV scores with the number of features (k) selected for classification. The maximum CV score and its corresponding k are highlighted in bold.

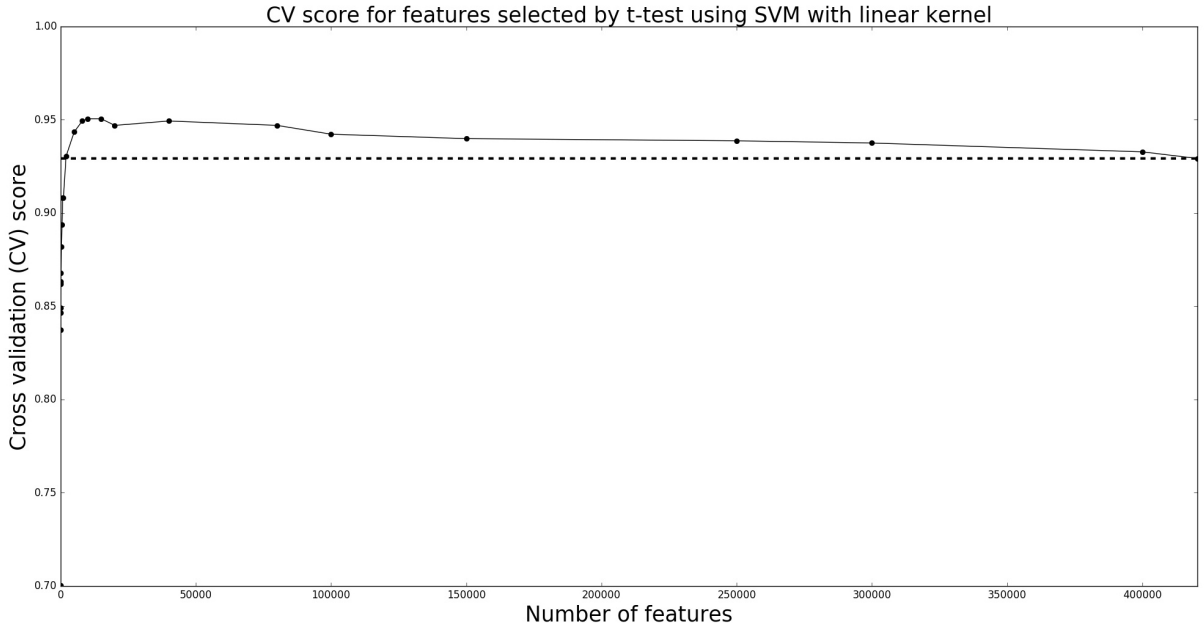


Figure 5.1: Plot of CV score against number of features, when an SVM with a linear kernel is fitted with the selected number of features.

5.1.2 Specific range of k

If we are interested in a specific range of k, we can easily plot a similar graph and sample k with smaller step sizes. This is shown in Figure 5.2, where k ranges from 10 to 2000 with a step size of 10. That is, we have $k = 10, 20, \ldots, 2000$. In this graph, we can see a similar trend, where the CV score increases with the number of features. The CV score peaks at around $k = 1500$.

To investigate where exactly the peak lies, we can zero in on a specific range of k and sample k with an even smaller step size. In Figure 5.3, k ranges from 1500 to 1600 with step size of 1. The peak is precisely $k = 1589$, where the corresponding CV score is around 0.932738. We can see that even with a small set of features, we can perform better than the threshold.

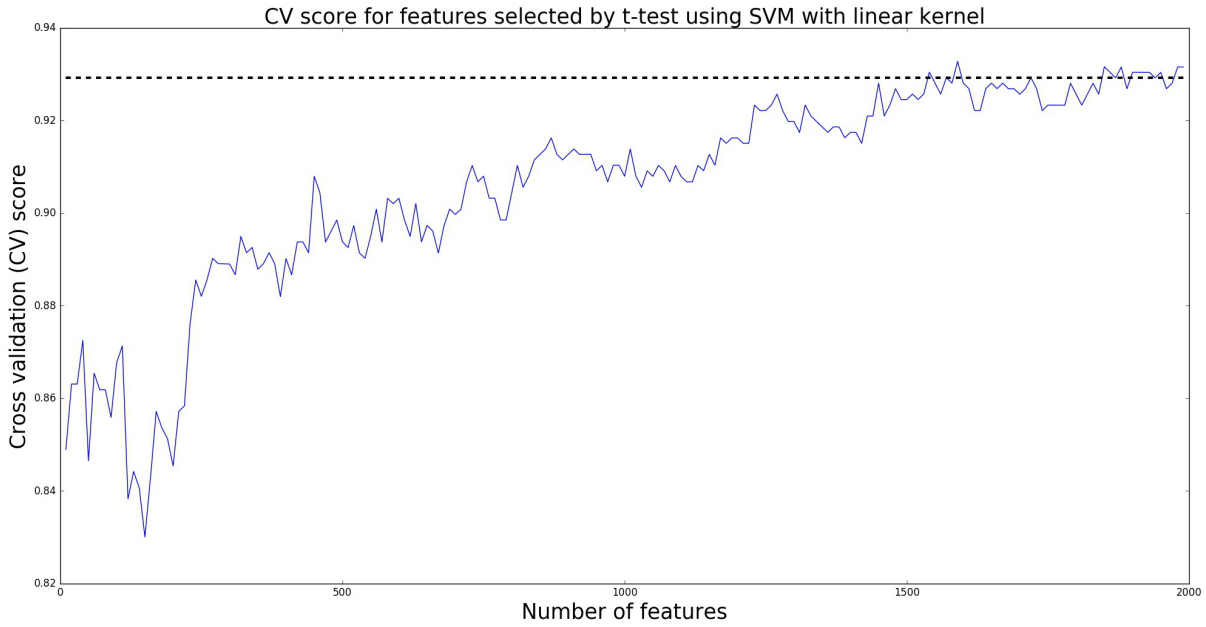


Figure 5.2: Plot of cross validation score against number of features, when an SVM with a linear kernel is fitted with the selected number of features, $k \in [10, 2000)$, with step size of 10.

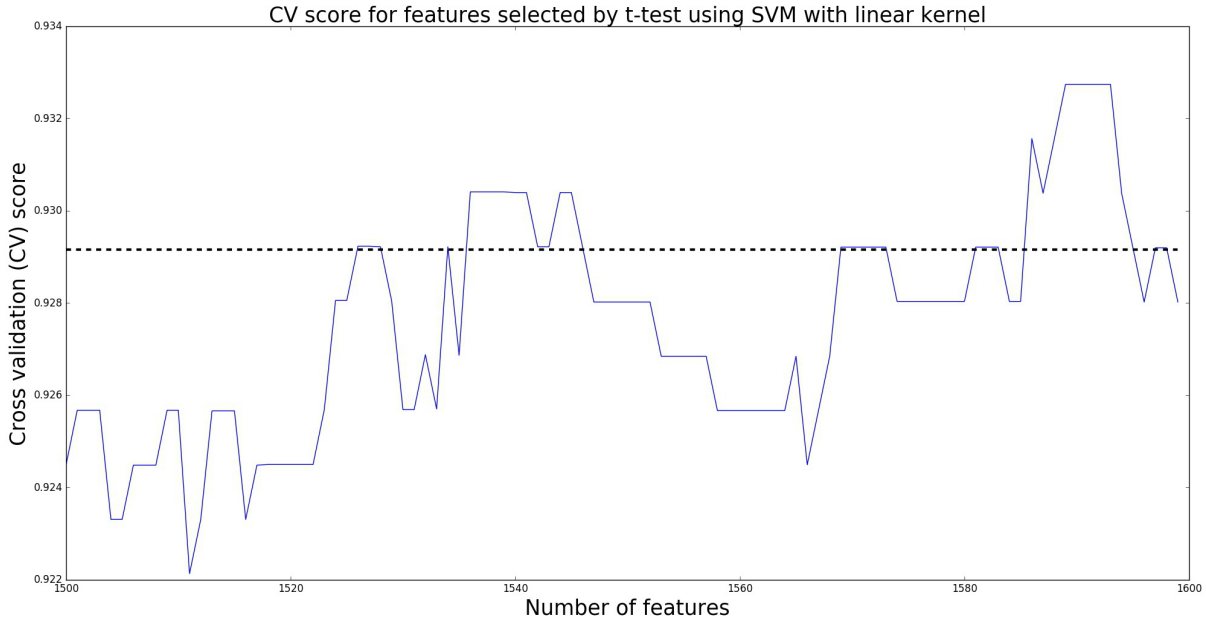


Figure 5.3: A magnified view of Figure 5.2. The peak is at $k = 1589$.

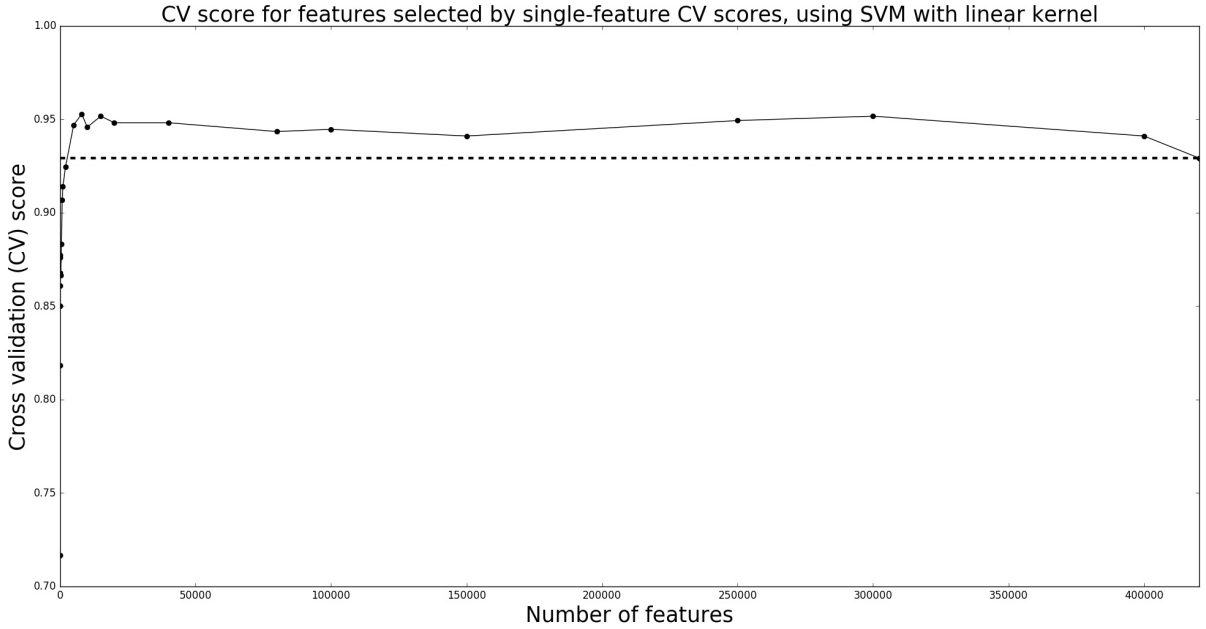


Figure 5.4: A graph of CV scores against number of features using a lazy variation of Sequential Forward Selection.

5.1.3 Lazy variation of SFS

Here, we investigate the lazy variation of SFS, as discussed in section 4.1.3. Instead of p-values, we sort the features according to their individual CV scores. We then obtain Figure 5.4.

We can see from the graph that at first, there is a similar trend to that in t-test in Figure 5.1: the CV score increases at a very fast rate and peaks at $k = 8000$ with CV score 0.952794. The CV score then decreases with more features. However, unlike the graph in Figure 5.1, the CV score peaks again slightly at around $k = 300000$. We can investigate this slight peak by plotting another graph shown in Figure 5.5, focusing on features between 250000 and 320000 with a step size of 1000. The maximum CV score in this range was 0.955161 with $k = 310000$.

5.1.4 Comparing t-test and lazy SFS

Using lazy SFS, the subset with 310000 features performs slightly better than that with 8000 features, but the former has about 40 times as many features as the latter. In this case, it might be sensible to compromise 0.3% of classification accuracy for about 300000 lesser features, as selecting only 8000 features for classification will impose much smaller computational and time constraints. Besides, it will be easier to make biological conclusions. This reflects the trade-offs faced in feature selection. We can also compare the lazy SFS and t-test clearly when we plot the two graphs in the same diagram in Figure 5.6.

Moreover, from the graph in Figure 5.6, we can see that both methods have similar performance when k is relatively small (i.e. $k < 150000$). However, beyond this value of

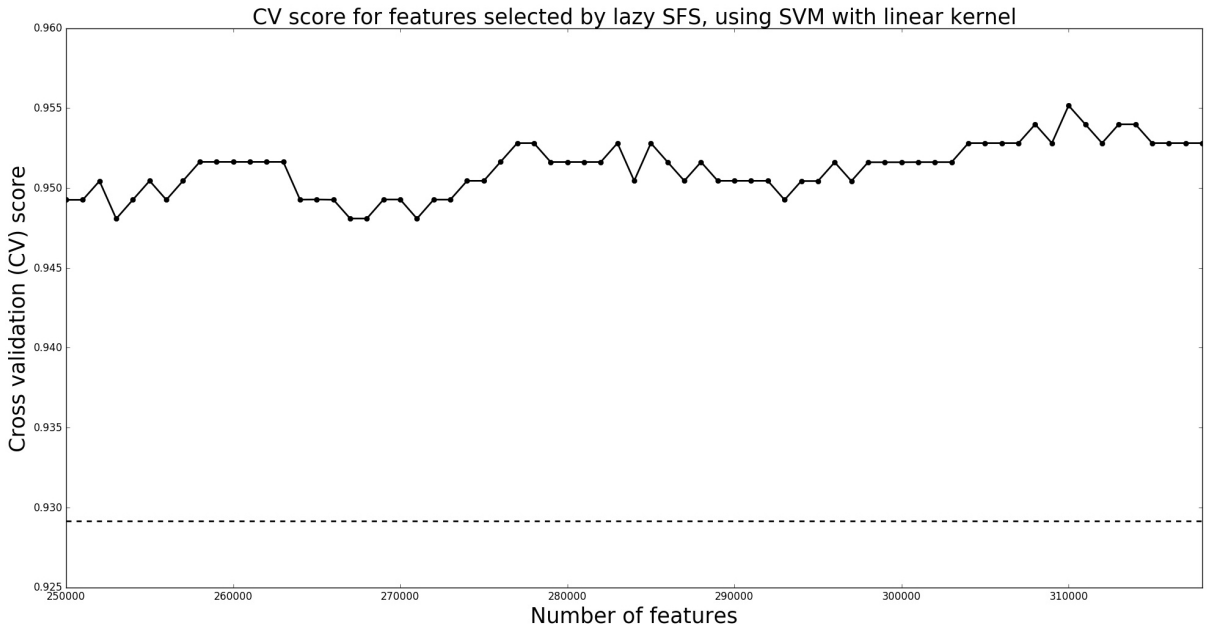


Figure 5.5: A "magnified" graph of CV scores against number of features using lazy SFS, focusing on the slight peak at around $k = 300000$.

k, the lazy SFS method seems to outperform the t-test method, although the difference in CV score is not significant (about 0.01).

Similarly to the discussion for the t-test method, the lazy SFS method only considers each feature singly, and does not take into account any form of redundancy; features that perform well individually might be redundant with respect to other features. Furthermore, features that do not perform well individually might perform well with other features, as discussed in [42].

Overall, both the t-test and the lazy SFS methods produce subsets with good performance relative to the threshold, but we also note that both methods do not take redundancy into account.

5.2 Recursive Feature Elimination

5.2.1 Varying step size

As mentioned in section 4.2, the RFE algorithm removes `step` number of features which have the lowest weights in each iteration. In implementing RFE, the step size was made to vary according to the number of features that we desire. This variation is summarised in Table 5.2. For example, suppose we wanted to select only $k = 5$ features. It will not be sensible to set the step size to a small number like 1, because the algorithm starts with the full set of features. Setting step size to 1 means that the algorithm will remove 1 feature at a time and thus the algorithm will take a very long time to terminate.

We then obtain CV scores based on the features selected by the algorithm. The CV scores can also be found in Table 5.2.

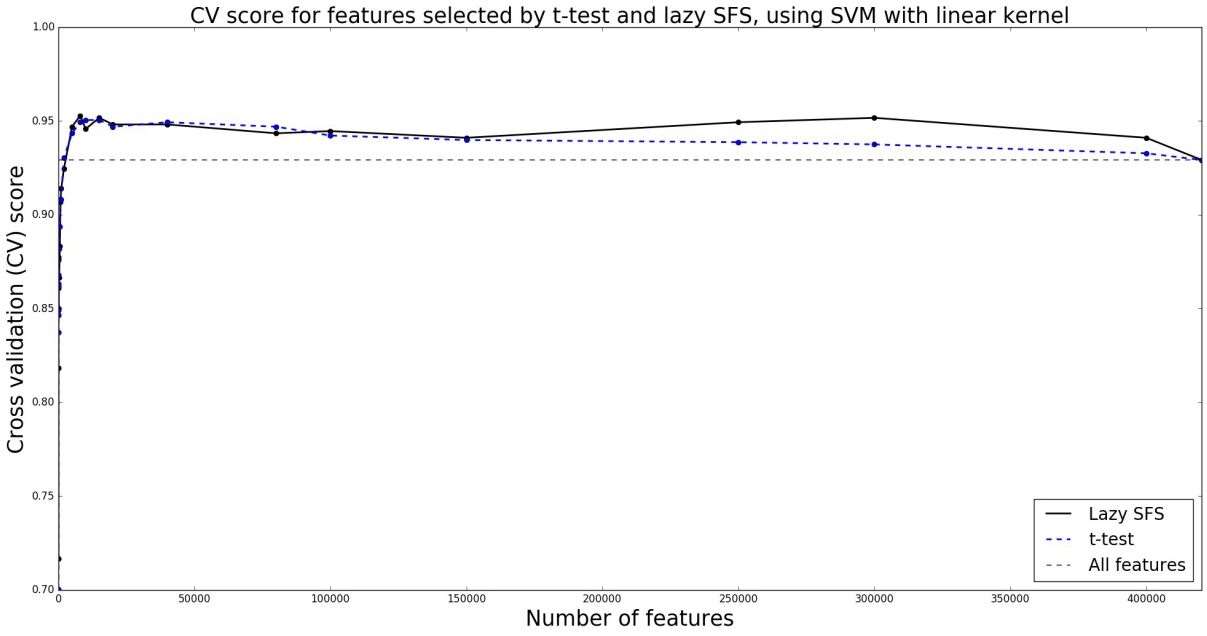


Figure 5.6: A graph of CV scores against number of features using both t-test and lazy SFS.

Next, we plot a graph of how the CV scores vary with the number of features selected. This graph can be found in Figure 5.7.

5.2.2 Performance of RFE

From Figure 5.7, we can see that there is an overall increasing trend in the CV scores. However, all the scores fall below the threshold. Evidently, using RFE to select features does not seem to benefit the feature selection method. We can compare this result to the t-test experiments performed earlier. We notice that using the t-test seems to produce better results than RFE. Besides, t-test is also computationally much less expensive than RFE.

Regardless, we also need to take into account how the `step` parameter works in the RFE method. At each iteration, the algorithm tries to fit an SVM on the features. Suppose it is considering F features, and the step size is f. It will remove f features that have the lowest weights until the number of features needed is achieved. Thus, the step size f here is important. From Table 5.2, we can see that the step sizes assigned to the algorithm are relatively large. For example, if we want only 10 features, the algorithm removes 10000 features at a time, which is a lot of features. This was done to allow the algorithm to terminate within a reasonable amount of time. Thus, this coarse step size might miss out on certain local optima which might be a cause of the trend we observe in RFE.

We also note that RFE suffers from the same *nesting effect* that Sequential Forward Selection and Sequential Backward Selection both experience, discussed in section 2.7.8. That is, once features are eliminated, they are not considered again by the algorithm.

From the above discussion, we can conclude that RFE might not be a good method to select features from our data set in terms of both computational cost and performance.

Step size	Num. of features	CV score
10000	1	0.5900
	5	0.7638
	10	0.8028
	20	0.8205
	50	0.8359
	80	0.8217
	100	0.8063
	250	0.7652
	500	0.7756
5000	750	0.8276
	1000	0.8158
	2000	0.8488
2000	5000	0.8831
	8000	0.8925
	10000	0.8972
1500	15000	0.9020
	20000	0.9043
	40000	0.9114
1000	80000	0.9091
	100000	0.9114
	150000	0.9150
	250000	0.9126
	300000	0.9126
	400000	0.9138

Table 5.2: Table of step size and corresponding chosen number of features and cross validation scores for Recursive Feature Elimination.

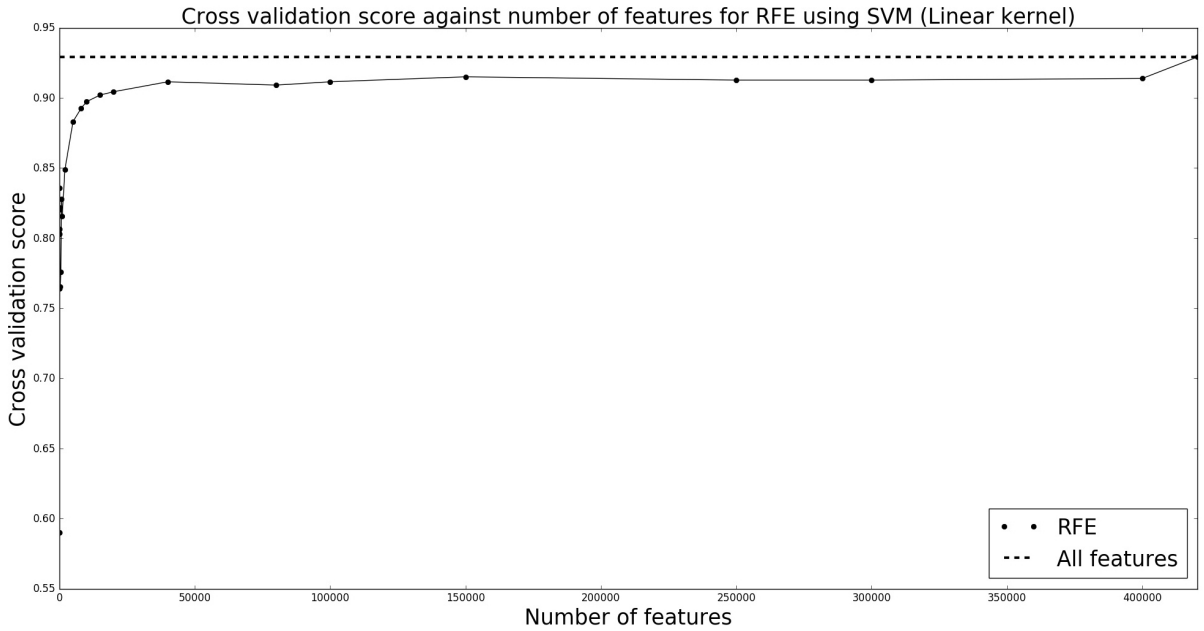


Figure 5.7: A graph of CV scores against number of features using both RFE.

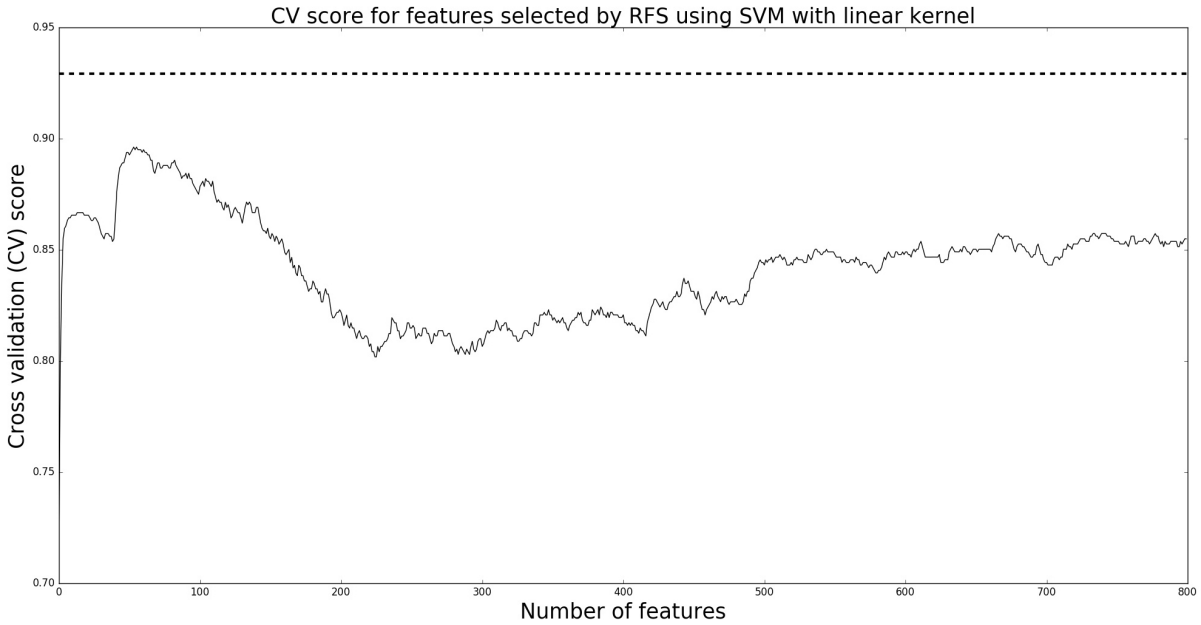


Figure 5.8: A graph of CV scores against number of features using Restricted Forward Selection.

5.3 Restricted Forward Selection

Due to time constraints, the RFS algorithm was only able to select 800 features, and the corresponding graph can be found in Figure 5.8. We see that across all 800 features, the graph does not exceed the threshold for the linear SVM.

We see that the graph peaks at $k = 53$ with a CV score of about 0.89611. After that, the CV score decreases and exhibits a slow increasing trend. However, the CV score does not exceed 0.89611 even with 800 features. We see that since there is an increasing trend in the CV scores, RFS might be able to select a subset that can perform better than the threshold. However, we cannot be absolutely certain about this until we use RFS to select more features. At the time of writing, RFS is being performed with $k = 5000$.

5.4 MRmMC

As mentioned in section 4.5.1, the MRmMC method only allows us to expand our feature subset S until $|S| = 847$. We present plots of the CV scores obtained based on the features selected by both the truncated (Figure 5.9) and full (Figure 5.10) versions of MRmMC. The CV scores presented in this section allow us to compare the effectiveness of incorporating GAs with MRmMC.

5.4.1 Comparison between truncated and full versions of MRmMC

The truncated version of MRmMC emulates RFS in that it only considers features that have high correlation coefficient r_{qn}, whereas the full version of the algorithm will consider

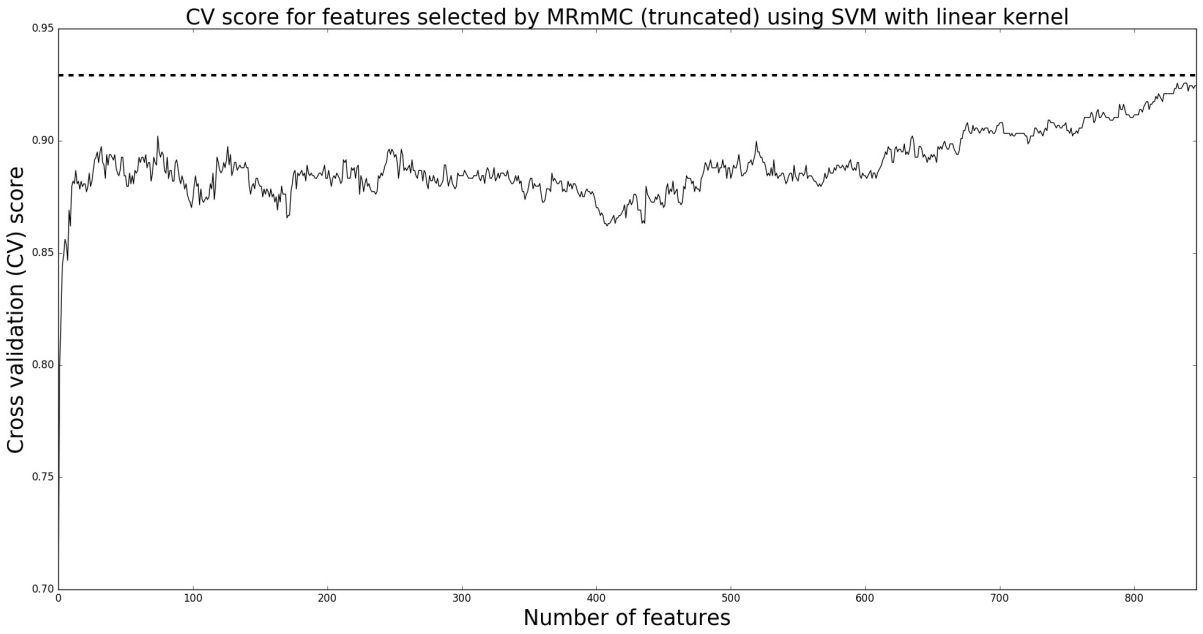


Figure 5.9: A graph of CV scores against number of features using the truncated version of MRmMC.

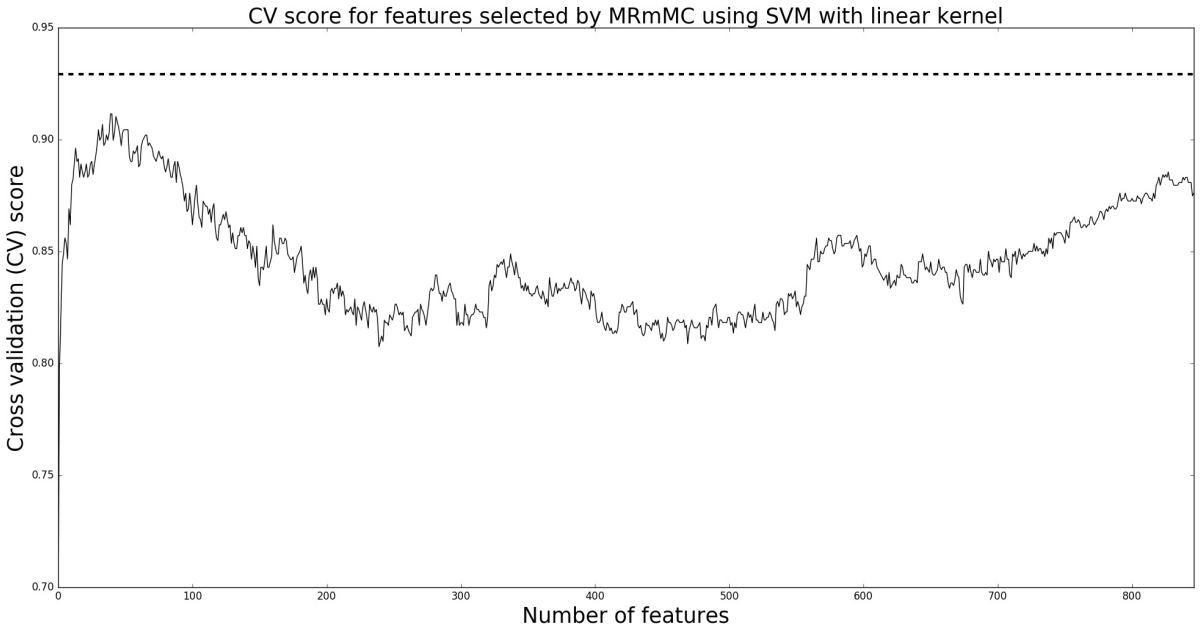


Figure 5.10: A graph of CV scores against number of features using the full version of MRmMC.

all features that have not been selected. We will thus expect the truncated version to suffer in optimality.

However, what we see from Figures 5.9 and 5.10 is that the truncated version is able to find a feature subset S that can perform better than that found in the full version. For the full version of MRmMC, the CV score peaks at $k = 41$ with a CV score of 0.911408. For the truncated version of MRmMC, the CV score peaks at $k = 832$ with a score of 0.925603. Thus, interestingly, the truncated version performs better than the full version *within this range of* k. We cannot make conclusions about values of k beyond this range.

5.4.2 Limitation of MRmMC

Evidently from both graphs, we can see that there is a general increasing trend in CV score as the number of features k increases, but both methods are not able to find features that can do better than the threshold. Had the algorithm allowed more features to be included in the set, we might obtain a feature subset S that can do better than the threshold.

However, we can also argue that we just require the CV score of S to be within a *range* of the threshold CV score. For example, we might just need the CV score of S to be within 5% of the threshold. This means that the minimum CV score that we can accept is 0.882702, which is 95% of the threshold CV score of 0.929160. Both the full and truncated versions satisfy this constraint.

Furthermore, previous methods, such as t-test, give us an approximate idea of the size of S that will give us better performance than the threshold. S should contain at least about 2000 features. Thus, we will not expect the subset that MRmMC chooses to outperform the set with all of the available features, since MRmMC can only choose 847 features from the data set. This is evident from the plots above.

5.5 MRmMC with Genetic Algorithm

Here, we modify MRmMC by removing the computation of R^2 and incorporating a genetic algorithm (GA). We also experiment with two selection strategies. Since the algorithm contains probabilistic elements, such as the mutation operator, the algorithm was run several times, and the average was taken. 10 runs were performed for each experimental setting. To obtain more accurate averages, we could perform more runs.

For these experiments, the number of features selected for the *first generation* of the GA, k is 8000, as illustrated in Figure 4.1b. We differentiate this value of k from the number of features eventually chosen by the algorithm in the subset S. We also select 16000 features that have the highest r_{qn} measure for the *first stage* of the GA, as illustrated in Figure 4.1a.

Parameters	Value
Number of features selected in first stage of GA k	8000
Number of top features to select with respect to r_{qn} heuristic in first stage of GA s	16000
Number of solutions in each generation G	50
Threshold number of generations to terminate algorithm T	5
Mutation rate p_m	0.05
"Runner-up" solutions to be chosen for mutation p_r	0.05
Cloning/elitism rate p_e	0.05
Rate of eliminating solutions with worst fitness p_{elim}	0.05
Number of solutions in tournament for tournament selection N_t	2
Probability of choosing winner in tournament selection p_t	0.8
Termination threshold for fitness t	0.001

Table 5.3: Table of parameters used in genetic algorithm implementation of MRmMC.

5.5.1 Roulette Wheel

First, Algorithm 3 was implemented with the Roulette Wheel seletion strategy. The parameters of the GA are listed in Table 5.3. The results obtained from the 10 runs were:

Run	CV score	Number of iterations	Number of features chosen
1	0.9551	6	8102
2	0.9587	8	8070
3	0.9563	9	7977
4	0.9563	6	8024
5	0.9586	6	7960
6	0.9563	7	8046
7	0.9586	7	8007
8	0.9551	10	7960
9	0.9563	10	7988
10	0.9575	8	7999
Average	0.9569	7.7	8013.3

On average, we see that the CV score is approximately 0.9569, slightly better than t-test in section 4.1. However, we note that in t-test, the best score of 0.950 was obtained with 15000 features, compared to about 8000 features with the GA, which is half of that in t-test. Furthermore, as the *elitism strategy* was implemented, the number of iterations required by the GA is relatively low, as the CV score for each generation is guaranteed to be non-decreasing.

We also note that the final number of features chosen by the GA is also roughly similar to the value k that we provide for the algorithm. We investigate if changing k drastically will make a difference to the outcome. To this end, we also set $k = 10, 50, 100$, while keeping the rest of the parameters the same as in Table 5.3.

For $k = 100$:

Run	CV score	Number of iterations	Number of features chosen
1	0.9610	19	7350
2	0.9598	16	6628
3	0.9563	14	6587
4	0.9563	18	7563
5	0.9563	19	7464
6	0.9563	22	7588
7	0.9551	15	6610
8	0.9563	15	6745
9	0.9575	14	6469
10	0.9587	22	7654
Average	0.9574	17.4	7065.8

For $k = 50$:

Run	CV score	Number of iterations	Number of features chosen
1	0.9575	15	6523
2	0.9586	15	6232
3	0.9551	11	4523
4	0.9539	14	6269
5	0.9563	14	6413
6	0.9586	19	7260
7	0.9598	16	6600
8	0.9575	14	6563
9	0.9563	15	6758
10	0.9539	18	7303
Average	0.9564	15.1	6482.9

For $k = 10$:

Run	CV score	Number of iterations	Number of features chosen
1	0.9563	16	6810
2	0.9586	19	7314
3	0.9574	17	6880
4	0.9539	17	7199
5	0.9551	12	4424
6	0.9551	9	4036
7	0.9575	11	5357
8	0.9574	20	6427
9	0.9575	14	7072
10	0.9563	17	7096
Average	0.9565	15.2	6261.5

We see that despite reducing k drastically, on average, the final number of features selected by the algorithm still remains in the thousands. Interestingly, for $k = 10$, the final number of features chosen is the lowest among the three experiments. The case of $k = 100$ also achieved the highest CV score so far of 0.9610.

5.5.2 Tournament selection

Here, we change the selection strategy to tournament selection, and construct similar tables to those in the previous section.

For $k = 8000$,

Run	CV score	Number of iterations	Number of features chosen
1	0.9599	15	8085
2	0.9575	7	8090
3	0.9598	15	7967
4	0.9575	6	7966
5	0.9563	7	7991
6	0.9575	5	8000
7	0.9610	14	8038
8	0.9563	11	8021
9	0.9610	10	7956
10	0.9610	11	7936
Average	0.9587	10.1	8005

For $k = 100$,

Run	CV score	Number of iterations	Number of features chosen
1	0.9598	18	7356
2	0.9563	15	6981
3	0.9575	14	6558
4	0.9587	11	6094
5	0.9563	11	5588
6	0.9586	15	7373
7	0.9587	11	5783
8	0.9575	14	5574
9	0.9622	19	7615
10	0.9634	17	7341
Average	0.9589	14.5	6626.3

For $k = 50$,

Run	CV score	Number of iterations	Number of features chosen
1	0.9586	17	7492
2	0.9633	16	7042
3	0.9575	17	7623
4	0.9563	17	7667
5	0.9610	15	6925
6	0.9610	12	5746
7	0.9586	14	6911
8	0.9586	16	7339
9	0.9563	15	6917
10	0.9575	14	6285
Average	0.9589	15.3	6994.7

For $k = 10$,

Run	CV score	Number of iterations	Number of features chosen
1	0.9563	16	6810
2	0.9586	19	7314
3	0.9574	17	6880
4	0.9539	17	7199
5	0.9551	12	4424
6	0.9551	9	4036
7	0.9575	11	5357
8	0.9574	20	6427
9	0.9575	14	7072
10	0.9565	17	7096
Average	0.9589	15.2	6261.5

5.5.3 Effects of k

Overall, we see that on average, the CV score, number of iterations and number of features chosen by the algorithm, do not change drastically with k. This seems to apply for both selection strategies. For all of the above settings, the CV scores fall between 0.95 and 0.96, the number of iterations rarely exceeds 20, and the number of features chosen by the GA is between 4000 to about 8000, regardless of the selection strategy and k.

5.5.4 Redundancy in genetic algorithm

In general, we see that the GA does not require many iterations to converge for this data set, and its CV scores are similar compared to t-test at about 0.95. At the same time, we see that for similar CV scores, the GA is able to form a subset that is much smaller than that produced by the t-test.

The GA was proposed to eliminate the need to compute R^2, but the problem of redundancy still arises in the GA, since we do not use a specific measure to quantify redundancy among the selected features. We might argue that for a CV score of about 0.95, the number of features selected by the GA is about half of those selected by t-test. We might have removed some redundancy from the features in t-test, but we cannot be absolutely certain, unless we explicitly check if each feature in the subset is redundant with respect to the other features.

This is proposed as an extension in section 6.1.

5.5.5 Evaluation of two-stage genetic algorithm

In the GA, we have employed the two-stage GA as discussed in section 2.8.1. The first stage calculates the r_{qn} quantity as specified in the MRmMC paper in [6] and chooses only the top 16000 features (Figure 5.3) that have the highest r_{qn}. We see that considering only features that have the best individual r_{qn} values does not seem to degrade the performance of the algorithm significantly, as indicated by the CV scores.

This has also allowed the algorithm to reduce the amount of memory needed, since we only need to keep track of the top 16000 features, instead of considering all 420374 features. Evidently, this is another parameter that can vary. We might be able to save even more memory by filtering out more features in the first stage of the GA. At the same time, we might also have missed out important features by truncating those features that do not have high r_{qn}. This issue will require further investigation into the parameters of the GA, as discussed in section 6.1.

5.6 Biological interpretation

In this section, we attempt to see if there are any features that repeatedly get selected by the feature selection algorithms. In particular, we inspect the features that are chosen in the following scenarios:

- t-test with $k = 15000$
- MRmMC with $k = 41$
- Truncated MRmMC with $k = 832$
- All features selected by MRmMC with GA where the CV score exceeds 0.96.

These settings are selected because the CV scores of the chosen feature subsets are relatively high.

We then see which features, if any, are selected in all or most of the above scenarios. To do this, we keep track of an array of 420374 zeros, and then count the number of times each feature appears in the above settings. In all of the experiments listed in this chapter, the features in the data set are represented by *indices*. So, if the algorithm selects a feature with index 50, we can find the feature by simply getting the 50th column of the data set.

We find that in out of the 9 settings listed above, the feature `cg13055685` appears in all 9 settings, 11 features appear in 8 of the settings and 138 features appear in 7 of the settings.

The features that are frequently selected can be mapped to a database that is publicly available. In particular, we extract information about our features through a database that contains methylation data in humans [115]. From this database, we can extract the features' gene names.

`cg13055685` corresponds to the gene name `MCF2L;MCF2L` and the 11 features that appear in 8 of the settings correspond to:

- `MAN2B2`
- `DAXX;DAXX;DAXX;DAXX`
- `SURF2`
- `SCAPER`
- `VPS53;VPS53`
- `MYH7B`

- CENPL;CENPL;CENPL;CENPL;DARS2;CENPL
- MCF2L;MCF2L
- DISP2
- CHERP

This discovery is made possible with the help of Dr. Karim Malki, who has also been giving advice for this project. At the time of writing, he is investigating the significance, if any, of the above genes.

Chapter 6

Conclusion

Overall, this project has explored the epigenetics data collected in [17]. It has also experimented with various feature selection algorithms and investigated the efficacy of these algorithms on the epigenetics data. In particular, we put our focus on the MRmMC algorithm and pointed out some characteristics which might not be applicable to our epigenetics data set. To mitigate these shortcomings, we have also extended the MRmMC algorithm with genetic algorithms under different experimental settings.

We can draw the following conclusions based on the work done in this project:

- The t-test method is able to select features that give good cross validation scores. Due to its limitations, these features might contain redundancies.
- Recursive Feature Elimination requires more fine-tuning in its parameters for it to be effective on the data set.
- MRmMC is not suitable for high dimensional data, specifically data with more features than samples.
- Incorporating genetic algorithms with MRmMC allowed us to find features that perform slightly better than those found by t-test and RFE. It has also allowed us to significantly reduce the number of features obtained.
- A way to measure redundancy is still lacking in the GA variant of MRmMC.
- Several features were repeatedly selected by the algorithms, and their corresponding genes were obtained.

6.1 Further work

One dilemma of greedy feature selection algorithms is that these algorithms only explore a subspace of the feature space, although they can find a local optimum in a shorter time than exhaustive algorithms.

Even though we have shown that some of the algorithms described here can find a subset S that performs better than the full set of features, we are not guaranteed the global optimum. In other words, even though we have found S that can perform better than

the full set of features, we do not know if there are other subsets that can perform even better.

The following extensions might help to enhance the performance of the algorithms discussed in this project.

6.1.1 Using different kernels and classifiers

In this project, we have limited ourselves to only the SVM. The kernel for the SVM was also mostly linear.

We can explore the effectiveness of using other types of kernels, such as the RBF kernel and the polynomial kernel. We can also try using other classifiers, as doing so might give us more insight into the data.

6.1.2 Exploring parameters and strategies

Parameters play an important role in feature selection algorithms. One of such parameters is the value of k, the number of features to be selected from the set of features.

The number of parameters and strategies also increases in GAs. As mentioned in section 2.8, there are many parameters in GAs that need to be tuned, and it might be instructive to experiment with parameters that are different from the ones used in this project. However, this might involve optimisation in many dimensions, where each dimension represents a parameter. Furthermore, different strategies can be used in GAs, such as different terminating conditions, crossover, selection and mutation strategies.

Thus, further work in this aspect will involve finding optimum parameters and/or strategies that can improve the search over the feature space.

6.1.3 Use of mutual information

As mentioned in section 2.7.11, mutual information (MI) is a highly popular measure to quantify the relationship between features and the target class (relevance) and between features themselves (redundancy). MI has also been used in many other works involving feature selection.

This project avoids the use of MI despite its popularity, simply because the data set uses continuous variables. MI also requires estimates of probabilities and it is computationally complex to estimate probabilities involving continuous variables.

Various methods have been proposed to estimate MI for continuous variables. For example, the Parzen Window [70] was proposed to estimate these probabilities. An extension can make use of the Parzen Window method to help estimate probabilities involving continuous variables. Parameters of the Parzen Window method also need to be optimised. This can then be combined with the algorithms used in this project. We can then determine if using the Parzen Window (or other methods) to estimate MI improves the feature selection process.

6.1.4 Quantifying redundancy in MRmMC

Although implementing a GA with MRmMC gave good cross validation scores with a relatively small number of features, we cannot be sure if the features chosen contain any redundancy. An extension can thus be to check for redundancy within this chosen set of features.

The authors of the MRmMC paper in [6] also suggested the need for an alternative redundancy measure. This might be due to the limitations of the redundancy measure R^2 that was originally proposed.

Suppose we are successful in estimating probabilities of continuous variables using mutual information as suggested in section 6.1.3, we can use the redundancy measure discussed in section 2.7.11:

$$\min\ R(S) = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i, x_j)$$

This measure was originally proposed in [56].

6.1.5 Further metrics to evaluate algorithms

From section 5.5, the quantities listed in tables - best cross validation (CV) score, number of iterations and number of features chosen - can be used to evaluate the efficiency and efficacy of feature selection algorithms. In the previous chapter, we have evaluated the performance of the subset S based on CV scores. An alternative will be to consider the above quantities to evaluate S.

For example, we can propose a metric J to select a subset $\widehat{S}$ that takes into account the CV score C, the number of iterations I, and the number of features chosen F. Intuitively, we maximise C and minimise I and F. That is,

$$\widehat{S} = \arg\max\ J(S) = C - I - F$$

We can also decide to weigh C, I and F with non-negative coefficients:

$$\widehat{S} = \arg\max\ J(S) = \alpha C - \beta I - \gamma F$$

where $\alpha, \beta, \gamma \in \mathbb{R}$. This is similar to the fitness function used in [21].

An extension will thus use this metric to evaluate S, not just based on its CV score.

Bibliography

[1] National Institute of Mental Health. Schizophrenia, 2016. Available from `https://www.nimh.nih.gov/health/topics/schizophrenia/index.shtml`.

[2] Schizophrenia.com. Heredity and the genetics of schizophrenia, 2004. Available from `http://www.schizophrenia.com/research/hereditygen.htm`.

[3] Randy L. Jirtle and Michael K. Skinner. Environmental epigenomics and disease susceptibility. *Nature*, 8:253–262, 2007.

[4] Bob Weinhold. Epigenetics: The science of change. *Environmental health perspectives*, 114(3):A160–A167, 2006.

[5] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[6] Azlyna Senawi, Hua-Liang Wei, and Stephen A. Billings. A new maximum relevance-minimum multicollinearity (mrmmc) method for feature selection and ranking. *Pattern Recognition*, 67:47–61, 2017.

[7] Lister Hill National Center for Biomedical Communications. *Help Me Understand Genetics*. 2017. Available from `https://ghr.nlm.nih.gov/primer`.

[8] National Human Genome Research Institute. Epigenomics, 2016. Available from `https://www.genome.gov/27532724/`.

[9] YourGenome. What is gene expression?, 2016. Available from `http://www.yourgenome.org/facts/what-is-gene-expression`.

[10] Heidi Chial, Carrie Drovdlic, Maggie Koopman, Sarah Catherine Nelson, Angela Spivey, and Robin Smith. Essentials of genetics, 2014. Available from `http://www.nature.com/scitable/ebooks/essentials-of-genetics-8`.

[11] Nessa Carey. *Life as we know it now*, pages 54–74. The Epigenetics Revolution: How modern biology is rewriting our understanding of genetics, disease and inheritance. Icon Books Ltd, United Kingdom, 2012.

[12] Nessa Carey. *Introduction*, pages 1–10. The Epigenetics Revolution: How modern biology is rewriting our understanding of genetics, disease and inheritance. Icon Books Ltd, United Kingdom, 2012.

[13] Arturas Petronis. Epigenetics and twins: three variations on the theme. *Trends in Genetics*, 22(7):347–350, 2006.

[14] Nessa Carey. *Why aren't identical twins actually identical?*, pages 75–96. The Epigenetics Revolution: How modern biology is rewriting our understanding of genetics, disease and inheritance. Icon Books Ltd, United Kingdom, 2012.

[15] Mario F. Fraga, Esteban Ballestar, Maria F. Paz, Santiago Ropero, Fernando Setien, Maria L. Ballestar, Damia Heine-Suñer, Juan C. Cigudosa, Miguel Urioste, Javier Benitez, Manuel Boix-Chornet, Abel Sanchez-Aguilera, Charlotte Ling, Emma Carlsson, Pernille Poulsen, Allan Vaag, Zarko Stephan, Tim D. Spector, Yue-Zhong Wu, Christoph Plass, and Manel Esteller. Epigenetic differences arise during the lifetime of monozygotic twins. *Proceedings of the National Academy of Sciences of the United States of America*, 102(30):10604–10609, July 26 2005.

[16] Pernille Poulsen, Manel Esteller, Allan Vaag, and Mario F. Fraga. The epigenetic basis of twin discordance in age- related diseases. *Pediatric research*, 61(5):38R, 2007.

[17] Eilis Hannon, Emma Dempster, Joana Viana, Joe Burrage, Adam R. Smith, Ruby Macdonald, David St Clair, Colette Mustard, Gerome Breen, Sebastian Therman, Jaakko Kaprio, Timothea Toulopoulou, Hilleke E. Hulshoff Pol, Marc M. Bohlken, Rene S. Kahn, Igor Nenadic, Christina M. Hultman, Robin M. Murray, David A. Collier, Nick Bass, Hugh Gurling, Andrew McQuillin, Leonard Schalkwyk, and Jonathan Mill. An integrated genetic-epigenetic analysis of schizophrenia: evidence for co-localization of genetic associations and differential dna methylation. *Genome biology*, 17(1):176, 2016.

[18] W. P. Kuo, E. Y Kim, J. Trimarchi, T. K Jenssen, S. A. Vinterbo, and L. Ohno-Machado. A primer on gene expression and microarrays for machine learning researchers. *Journal of Biomedical Informatics*, 37(4):293–303, 08 2004.

[19] A. Bharathi and A. M. Natarajan. Microarray gene expression cancer diagnosis using machine learning algorithms. In *3rd IEEE International Conference on Signal and Image Processing, ICSIP 2010, December 15, 2010 - December 17*, pages 275–280, Chennai, India, 2010 2010. Bannari Amman Institute of Technology, Tamilnadu (State), India, IEEE Computer Society.

[20] Chang Kyoo Yoo and Krist V. Gernaey. Classification and diagnostic output prediction of cancer using gene expression profiling and supervised machine learning algorithms. *Journal of Chemical Engineering of Japan*, 41(9):898–914, 2008.

[21] Qihua Tan, M. Thomassen, K. M. Jochumsen, Hua Zhao Jing, K. Christensen, and T. A. Kruse. Evolutionary algorithm for feature subset selection in predicting tumor outcomes using microarray data. In *Fourth International Symposium, ISBRA 2008*, Bioinformatics Research and Applications, pages 426–33, Berlin, Germany, 6-9 May 2008 2008. Dept. of Biochem., Pharmacology Genetics, Odense Univ. Hosp., Odense, Denmark, Springer-Verlag.

[22] Y. Lu and J. Han. Cancer classification using gene expression data. *Information Systems*, 28(4):243–68, 06 2003.

[23] American Psychiatric Association. Diagnostic and statistical manual of mental disorders, 2013.

[24] Tom M. Mitchell. *Decision Tree Learning*, pages 52–78. Machine learning. McGraw-Hill Education, international 1997 edition, 1997.

[25] J.R. Quinlan. Induction of decision trees, 1986.

[26] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[27] Juergen Gall, Nima Razavi, and Luc Van Gool. An introduction to random forests for multi-class object detection. In *15th International Workshop on Theoretical Foundations of Computer Vision, June 26, 2011 - July 1*, volume 7474 LNCS, pages 243–263, Dagstuhl Castle, Germany, 2011 2012. Computer Vision Laboratory, ETH Zurich, Switzerland/Max Planck Institute for Intelligent Systems, GermanyESAT/IBBT, Katholieke Universiteit Leuven, Belgium, Springer Verlag.

[28] Songul Cinaroglu. Comparison of performance of decision tree algorithms and random forest: An application on oecd countries health expenditures. *International Journal of Computer Applications*, 138(1):37–41, March 2016.

[29] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, December, 2002.

[30] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–99, 1999.

[31] Edwin K.P. Chong and Stanislaw H. Zak. *Convex optimization problems.* An introduction to optimization. Wiley, Hoboken, New Jersey, 4th; fourth edition, 2013.

[32] Edwin K. P. Chong and Stainlaw H. Zak. *Duality.* An introduction to optimization. Wiley, Hoboken, New Jersey, 4th; fourth edition, 2013.

[33] Nello Cristianini and John Shawe-Taylor. *An introduction to Support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, Cambridge, U.K. ; New York, 2000.

[34] Christopher M. Bishop. *Pattern recognition and machine learning.* Springer, New York, 2006.

[35] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, 2016. National Taiwan University, Taipei 106, Taiwan.

[36] Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *submitted to Neural Computation*, pages 1–32, 2003.

[37] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[38] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. *Foundations of Feature Selection*, pages 13–28. Feature Selection for High-Dimensional Data. Cham, 2015.

[39] C. M. M. Wahid, A. B. M. S. Ali, and K. Tickle. Impact of feature selection on support vector machine using microarray gene expression data. In *2009 Second International Conference on Machine Vision (ICMV 2009)*, pages 189–93, Piscataway, NJ, USA, 28-30 Dec. 2009 2009. Sch. of Comput. Sci., CQ Univ., QLD, Australia, IEEE.

[40] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004.

[41] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[42] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(7-8):1157–82, 10/01 0015.

[43] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25–41, 01 2000.

[44] H. Uguz. A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowledge-Based Systems*, 24(7):1024–32, 10 2011.

[45] Richard O. Duda. *Problems of Dimensionality.* Pattern classification. Wiley, New York ; Chichester, 2nd / edition, 2001.

[46] Ioannis Tsamardinos and Constantin F. Aliferis. Towards principled feature selection: relevancy, filters and wrappers. In *AISTATS*, 2003.

[47] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3(7-8):1415–38, 10/01 2003.

[48] Yvan Saeys, Iñ Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[49] P. K. Ammu and V. Preeja. Review on feature selection techniques of dna microarray data. *International Journal of Computer Applications*, 61(12), 2013.

[50] Z. M. Hira and D. F. Gillies. A review of feature selection and feature extraction methods applied on microarray data, 2015.

[51] J. Hua, W. D. Tembe, and E. R. Dougherty. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3):409–24, 03 2009.

[52] E. Ke Tang, Ponnuthurai N. Suganthan, and Xin Yao. Gene selection algorithms for microarray data based on least squares support vector machine. *BMC bioinformatics*, 7(1):95, 2006.

[53] Alok Sharma, Seiya Imoto, and Satoru Miyano. A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):754–764, 2012.

[54] C. Huertas and R. Juarez-Ramirez. Filter feature selection performance comparison in high-dimensional data: a theoretical and empirical analysis of most popular algo-

rithms. *17th International Conference on Information Fusion (FUSION)*, page 8, 2014.

[55] E. K. Tang, PN Suganthan, and Xin Yao. Gene selection algorithms for microarray data based on least squares support vector machine. *BMC Bioinformatics*, 7:95, 2006.

[56] Hanchuan Peng, Fuhui Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–38, 08 2005.

[57] C. Sima and E. R. Dougherty. What should be expected from feature selection in small-sample settings. *Bioinformatics (Oxford, England)*, 22(19):2430–2436, Oct 1 2006.

[58] A. W. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, C-20(9):1100–3, 1971.

[59] A. Jain and D. Zongker. Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–8, 02 1997.

[60] Isabelle Guyon and Andre Elisseeff. *Introduction*, pages 1–25. Feature extraction: foundations and applications. Springer, 2006.

[61] P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–25, 11 1994.

[62] Kan Deng. *OMEGA: On-line memory-based general purpose system classifier*. PhD thesis, Georgia Institute of Technology, 1998.

[63] Xiangyan Zeng, Yen-Wei Chen, Caixia Tao, and D. van Alphen. Feature selection using recursive feature elimination for handwritten digit recognition. In *IIH-MSP 2009*, 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pages 1205–8, Piscataway, NJ, USA, 12-14 Sept. 2009 2009. Dept. of Math. Comput. Sci., Fort Valley State Univ., Fort Valley, GA, United States, IEEE.

[64] Hengbo Ma, Tianyu Tan, Hongpeng Zhou, and Tianyi Gao. Support vector machine-recursive feature elimination for the diagnosis of parkinson disease based on speech analysis. In *2016 Seventh International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 34–40, Piscataway, NJ, USA, 1-4 Dec. 2016 2016. Harbin Inst. of Technol., Harbin, China, IEEE.

[65] K. Kancherla and S. Mukkamala. Feature selection for lung cancer detection using svm based recursive feature elimination method. In *10th European Conference, EvoBIO 2012*, Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, pages 168–76, Berlin, Germany, 11-13 April 2012 2012. Inst. for Complex Additive Syst. Anal. (ICASA), New Mexico Inst. of Min. Technol., Socorro, NM, United States, Springer-Verlag.

[66] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–22, 1977.

[67] Il-Seok Oh, Jin-Seon Lee, and Byung-Ro Moon. Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1424–37, 11 2004.

[68] Kari Torkkola. *Information-Theoretic Methods*, pages 168–185. Feature extraction: foundations and applications. Springer, Berlin, 2006.

[69] Kenneth E. Hild II, Deniz Erdogmus, Kari Torkkola, and Jose C. Principe. Feature extraction using information-theoretic learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1385–1392, 2006.

[70] N. Kwak and Chong-Ho Choi. Input feature selection by mutual information based on parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1667–71, 12 2002.

[71] Walters-Williams Janett and Yan Li. Estimation of mutual information: A survey. In *4th International Conference on Rough Sets and Knowledge Technology, RSKT 2009, July 14, 2009 - July 16*, volume 5589 LNAI, pages 389–396, Gold Coast, QLD, Australia, 2009 2009. School of Computing and Information Technology, University of Technology, Kingston 6, JamaicaDepartment of Mathematics and Computing, Centre for Systems Biology, University of Southern Queensland, QLD 4350, Australia, Springer Verlag.

[72] M. Zhukov and A. Popov. Bin number selection for equidistant mutual information estimaton. In *2014 IEEE 34th International Conference on Electronics and Nanotechnology (ELNANO)*, pages 259–63, Piscataway, NJ, USA, 15-18 April 2014 2014. Phys. Biomed. Electron. Dept., Nat. Tech. Univ. of Ukraine, Kiev, Ukraine, IEEE.

[73] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[74] Tom M. Mitchell. *Genetic Algorithms*, pages 249–270. Machine learning. McGraw-Hill Education, international 1997 edition, 1997.

[75] Il-Seok Oh, Jin-Seon Lee, and Byung-Ro Moon. Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1424–37, 11 2004.

[76] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, 3rd rev. and extended edition, 1996.

[77] Juha Reunanen. *Search strategies*, pages 120–136. Feature extraction: foundations and applications. Springer, 2006.

[78] H. Boubenna and Dohoon Lee. Feature selection for facial emotion recognition based on genetic algorithm. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 511–17, Piscataway, NJ, USA, 13-15 Aug. 2016 2016. Dept. of Electron. Comput. Sci. Eng., Pusan Nat. Univ., Busan, Korea, Republic of, IEEE.

[79] S. Singh and S. Selvakumar. A hybrid feature subset selection by combining filters and genetic algorithm. In *2015 International Conference on Computing, Communication & Automation (ICCCA)*, pages 283–9, Piscataway, NJ, USA, 15-16 May

2015 2015. Dept. of Comput. Sci. Eng., Nat. Inst. of Technol., Tiruchirappalli, India, IEEE.

[80] C. De Stefano, F. Fontanella, and di Freca Scotto. Feature selection in high dimensional data by a filter-based genetic algorithm. In *20th European Conference, EvoApplications 2017*, volume pt. I of *Applications of Evolutionary Computation*, pages 506–21, Cham, Switzerland, 19-21 April 2017 2017. Dipt. di Ing. Elettr. e dell'Inf., Univ. di Cassino e del Lazio Meridionale, Cassino, Italy, Springer International Publishing.

[81] R. Altilio, L. Liparulo, A. Proietti, M. Paoloni, and M. Panella. A genetic algorithm for feature selection in gait analysis. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4584–91, Piscataway, NJ, USA, 24-29 July 2016 2016. Dept. of Inf. Eng., Univ. of Rome La Sapienza, Rome, Italy, IEEE.

[82] Xiao-Bing Hu and Ezequiel Di Paolo. An efficient genetic algorithm with uniform crossover for air traffic control. *Computers and Operations Research*, 36(1):245–259, 2009.

[83] Hong Ge and Tianliang Hu. Genetic algorithm for feature selection with mutual information. In *2014 7th International Symposium on Computational Intelligence and Design (ISCID)*, volume 1, pages 116–19, Los Alamitos, CA, USA, 13-14 Dec. 2014 2014. Sch. of Comput. Sci., South China Normal Univ., Guangzhou, China, IEEE Computer Society.

[84] John H. Holland. Genetic algorithms. *Scientific American*, 267(1):66–72, 1992.

[85] T. Maini, R. K. Misra, and D. Singh. Optimal feature selection using elitist genetic algorithm. In *2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI)*, page 5 pp., Piscataway, NJ, USA, 14-17 Dec. 2015 2015. IIT (BHU), Varanasi, India, IEEE.

[86] Andrej Taranenko and Aleksander Vesel. An elitist genetic algorithm for the maximum independent set problem. In *23rd International Conference on Information Technology Interfaces, ITI 2001, June 19, 2001 - June 22*, pages 373–378, Pula, Croatia, 2001 2001. PEF, University of Maribor, Koroska c. 160, SI-2000 Maribor, Slovenia, University of Zagreb.

[87] J. Yang and C. K Soh. Structural optimization by genetic algorithms with tournament selection. *Journal of Computing in Civil Engineering*, 11(3):195–200, 1997.

[88] R. C. Anirudha, R. Kannan, and N. Patil. Genetic algorithm based wrapper feature selection on hybrid prediction model for analysis of high dimensional data. In *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6, Piscataway, NJ, USA, Dec 2014. Dept. of Inf. Technol., Nat. Inst. of Technol. Karnataka, Mangalore, India, IEEE.

[89] Pan-Shi Tang, Xiao-Long Tang, Zhong-Yu Tao, and Jian-Ping Li. Research on feature selection algorithm based on mutual information and genetic algorithm. In *2014 11th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 403–6, Piscataway, NJ, USA, 19-21 Dec. 2014 2014. Sch. of Comput. Sci. Eng., Univ. of Electron. Sci. Technol. of China, Chengdu, China, IEEE.

[90] E. Falkenauer. The worth of the uniform [uniform crossover]. In *Congress on Evolutionary Computation-CEC99*, volume 1 of *Proceedings of the 1999*, pages 776–82, Piscataway, NJ, USA, 6-9 July 1999 1999. CAD Unit, Brussels Univ., Brussels, Belgium, IEEE.

[91] G. Pavai and T. V. Geetha. A survey on crossover operators. *ACM Computing Surveys*, 49(4), 2016.

[92] S. Picek and M. Golub. Comparison of a crossover operator in binary-coded genetic algorithms. *WSEAS Transactions on Computers*, 9(9):1064–73, 2010.

[93] Wen-Yang Lin, Wen-Yuan Lee, and Tzung-Pei Hong. Adapting crossover and mutation rates in genetic algorithms. *Journal of Information Science and Engineering*, 19(5):889–903, 2003.

[94] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[95] L. Darrell Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *ICGA*, volume 89, pages 116–123, 1989.

[96] der Walt van, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 03 2011.

[97] P. J. B. Hancock. An empirical comparison of selection methods in evolutionary algorithms. In *AISB Workshop*, Evolutionary Computing, pages 80–94, Berlin, Germany, 11-13 April 1994 1994. Dept. of Psychol., Stirling Univ., Stirling, United Kingdom, Springer-Verlag.

[98] Noraini Mohd Razali and John Geraghty. Genetic algorithm performance with different selection strategiesin solving tsp. In *World Congress on Engineering 2011, WCE 2011, July 6, 2011 - July 8*, volume 2, pages 1134–1139, London, United kingdom, 2011 2011. School of Mechanical and Manufacturing Engineering, Dublin City University, IrelandEnterprise Research Process Centre, Dublin City University, Ireland, Newswood Ltd.

[99] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93, 1991.

[100] Jinghui Zhong, Xiaomin Hu, Min Gu, and Jun Zhang. Comparison of performance between different selection strategies on simple genetic algorithms. In *Jointly with International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, Proceedings. 2006 International Conference on Intelligence For Modelling, Control and Automation, Los Alamitos, CA, USA, 28-30 Nov. 2005 2005. SUN Yat-sen Univ., Guangzhou, China, IEEE. CD-ROM; T3: Proceedings. 2006 International Conference on Intelligence For Modelling, Control and Automation. Jointly with International Conference on Intelligent Agents, Web Technologies and Internet Commerce.

[101] Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9, 2011.

[102] Francesc Alted, Ivan Vilata, et al. PyTables: Hierarchical datasets in Python, 2002.

[103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[104] Imperial college high performance computing service. doi: 10.14469/hpc/2232.

[105] Douglas C. Montgomery. *Engineering statistics*. Wiley ; John Wiley distributor], Hoboken, N.J. : Chichester, 4th edition, 2007.

[106] NIST/SEMATECH. Are the model residuals well-behaved? (e-handbook of statistical methods), 2013. Available from `http://www.itl.nist.gov/div898/handbook/pri/section2/pri24.htm`.

[107] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001.

[108] Jochen Jäger, Rimli Sengupta, and Walter L. Ruzzo. Improved gene selection for classification of microarrays. In *Pacific Symposium on Biocomputing*, volume 8, pages 53–64, 2002.

[109] Yutian Wang, Bingqi Tan, Yifeng Wang, and Jiangtao Wu. Information structure analysis for quantitative assessment of mineral resources and the discovery of a silver deposit. *Natural Resources Research*, 3(4):284–294, 1994.

[110] Alexandr Katrutsa and Vadim Strijov. Comprehensive study of feature selection methods to solve multicollinearity problem according to evaluation criteria. *Expert Systems with Applications*, 76:1–11, 2017.

[111] Herve Abdi. Multiple correlation coefficient. *The University of Texas at Dallas*, 2007.

[112] Gene H. Golub. *Matrix computations*. Fourth edition, 2013.

[113] Lloyd N. (Lloyd Nicholas) Trefethen. *Numerical linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[114] Maurice Herlihy. *The art of multiprocessor programming*. Elsevier / Morgan Kaufmann, Amsterdam ; London, revised first edition, 2012.

[115] Kasper Daniel Hansen. *IlluminaHumanMethylation450kanno.ilmn12.hg19: Annotation for Illumina's 450k methylation arrays*. R package version 0.2.1.