Imperial College London
Department of Computing

MEng Individual Project (JMC)

THE TITLE

Author: Daren Sin
Supervisor: Dr. Panos Parpas
Second marker: Dr. Ruth Misener

May 28, 2017

Abstract

Acknowledgements

Contents

Chapter 1

Introduction

1.1 Schizophrenia and epigenetics

Schizophrenia is a complex mental disorder that displays an array of symptoms, including hallucination and depression. It is commonly perceived that schizophrenia is a hereditary disease that can be passed down within the family, but some individuals diagnosed with schizophrenia do not have a family member with the disorder [1].

Furthermore, there is 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 diseases, including psychiatric disorders [1, 2, 3]. This results in a hypothesis that the *epigenetics* of an individual might have a role to play in the development of schizophrenia (section 2.2) [4].

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" [5]. Simply put, epigenetics provides a "window" which allows one's external environment to influence his or her DNA. 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 0.5% to 1% in the general population, according to Carey. However, it begs the question: why is the other twin not diagnosed with the disease with 100% certainty? This then is epigenetics at work, as environmental factors, not the twins' genetic material, influences the likelihood of the twins' diagnoses of schizophrenia.

1.2 Using machine learning to predict schizophrenia

Using data from a recent study on epigenetics and schizophrenia (section 2.2), this project aims to use machine learning to elucidate any statistical regularity in the data, in hope that any insight into the data can help geneticists and psychiatrists understand the etiology of schizophrenia - and indeed, other psychiatric disorders - better.

Previous work on using machine learning on biological data (section 2.3) has always been plagued with the *curse of dimensionality*, where the number of biological samples is far lesser than the number of features (or dimensions) of the data (the "$p \gg n$" problem [6]).

In this project, the data has 847 samples (individuals) with 420374 features, resulting in about 2 gigabytes of data. Besides the high dimensionality faced by the data, we also encounter a potential problem of a similar nature: not all of the features in the data involved in the study would directly play a part in the classification of the disorder; some genes may only contribute a little to the outcome of the classification. It is also hypothesised that subsets of features, rather than individual features, contribute to the genesis of the disorder [2].

1.3 Importance of feature selection

The aforementioned problems inherent in the data set makes *feature selection* an important preprocessing step to select only the features that have significant contribution to the classification outcome (section 2.6). This project pays particular attention to the feature selection process and sees which, if any, feature selection algorithm is especially beneficial in analysing the aforementioned data set.

Recently, a paper published by Senawi *et. al* [7] proposes a new method, Maximum Relevance Minimal Multicollinearity (MRmMC), to quantify the relevance and redundancy (section 2.6.7) of the features of a data set. This project extends MRmMC by investigating if incorporating a genetic algorithm will enhance the performance of MRmMC. This project further contributes to the investigation by comparing the merits, drawbacks and performance of different feature selection methods that are commonly used in the feature selection literature.

Thus, although the problem of epigenetics and schizophrenia forms the context for this project, this project will primarily describe and evaluate feature selection methods, comparing them to the MRmMC method.

1.4 •

Moreover, the current research on psychiatric disorders do not receive as much attention as other illnesses such as cancer [8]. Thus, any insight generated from this project would be beneficial to helping us understand psychiatric disorders better.

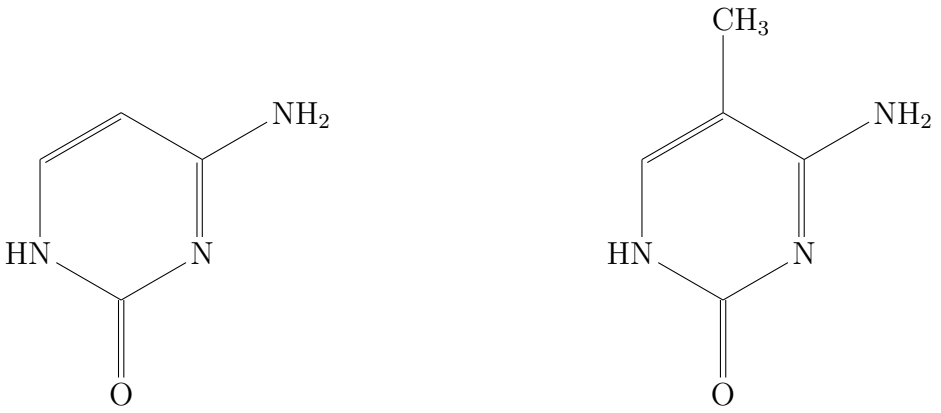
Chapter 2

Background

2.1 Molecular biology and definitions

This section outlines the necessary biology that will be relevant to the discussion in this project [9, 10, 11, 12].

- **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 then 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. This process is often called 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. differentiate between a brain cell and a skin cell).
- **Epigenetics:** The study of modifications in cells that are not influenced by changes in an individual's DNA.
- **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 alter the "on/off" action in gene expression and control the production of proteins. The epigenome arises naturally, but can be affected by external factors (e.g. environmental factors, disease), which might explain why even though twins have the same genome, it often happens that one twin inherits a disease, while the other does not [13].



(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.

- **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 which they are attached to 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 [14].
- **CpG island:** These DNA methylations mostly occur in CpG islands, which are areas in the DNA where the base cytosine is next to a guanine.

2.2 The dataset

In general, monozygotic (identical) twins allow us to investigate the link between genotype (our genetic makeup) and phenotype (our physical characteristics) [15, 16], 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 [17], Esteller *et. al* found that epigenetic modifications between twins become more disparate as the monozygotic twins become older. They 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 [18] 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.

In a similar vein, 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 genetic 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 [19], which utilised high-throughput methods that enable genomics

researchers to perform epigenome-wide association studies (EWAS).

In this study in particular, the researchers aimed to use these methods 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). In particular, we are interested in the data offered in "phase 2" of the study, where schizophrenia-associated differentially methylated positions (DMPs) (positions on the genome where there is a difference in DNA methylation patterns between two sets of genomes) were tested among 847 individuals, 414 of whom were schizophrenia cases.

Throughout this project, we shall identify the data produced from this study as *the data*.

2.3 Comparison with cancer classification

There is a significant amount of literature on cancer classification using gene expression data. These works primarily aim to uncover biological or medical insights using biological data obtained from microarrays, which are tools to measure the gene expression of thousands of genes simultaneously [20]. For example, using neural networks, gene expression data can be used to distinguish between tumour types, which helps in cancer diagnosis [21, 22]. [23] has even identified genes that can potentially predict 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 high dimensionality (*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) [22]. Furthermore, only a (small) subset of the features (genes) are relevant for the studies, as not all genes are useful in determining the type of cancer a patient has. This is known as biological noise [24]. As such, feature reduction on the data has to be performed to select only the relevant features for the classification problem. In other words, an ideal subset of pertinent features would ideally be sparse, as we seek to identify the features that are most relevant to the classification.

However, what is different about studies on psychiatric disorders and studies on cancer, is that the latter is observable, such that we can know for sure 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, [25] discusses culture-related issues in diagnosing schizophrenia: "the assessment of disorganized speech may be made difficult by linguistic variation in narrative styles across cultures". Furthermore, "ideas that appear to be delusional in one culture (e.g. witchcraft) may be commonly held in another". These highlight how diagnosing a psychiatric disorder like schizophrenia is not straightforward.

2.4 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 lists some popular machine learning classifiers used in the feature selection literature, and explains how some might or might not be useful for the purpose of this project.

2.4.1 Decision Trees

In our context, the task is to classify the data according to whether a sample (individual) 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 [26].

A decision tree algorithm is capable of sorting the instances - in our context, samples with different features - down the tree until the algorithm reaches a leaf node, during which a classification is given to the node. At each level of the tree, the intermediate node is split according to some attribute.

One variant of the decision tree algorithm is the ID3 algorithm [27]. The ID3 algorithm makes use of a statistical quantitative measure, the *information gain*, to determine the attribute to classify the samples with. Using definitions from [26], let S be the set of all the samples that we want to classify at a particular node. The samples can also be separated into two groups, those with a positive classification and those with a negative classification.

Definition 2.4.1.1 (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 2.4.1.2 (Information gain)**.** *The information gain with respect to the set S and an attribute (feature) A is defined as:*

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

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.

We then classify the samples in the node according to the attribute with the highest information gain. Intuitively, we want to choose the attribute that can give 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 [26] and the resulting decision tree can be easily interpreted, it might be difficult to classify samples according

to features that are highly correlated. This also happens to be a potential characteristic of our dataset, and we would expect some of the features to be correlated.

Furthermore, the features of our dataset are vectors of real numbers. We would thus need to discretise the range of real numbers into intervals. 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 2.6.11.

2.4.2 Random forest

The random forest method [28], a form of "ensemble learning", is an extension of the decision tree algorithm described above, and it has been used in areas such as multi-class object detection in images [29]. Overall, a random forest algorithm can be outlined as such:

- Split the dataset into distinct subsets.
- Using each subset, train a decision tree using a relevant algorithm, such as the ID3, as outlined above.
- Combine all the trees together to create a forest.
- Suppose we have an unseen sample $\boldsymbol{x}$. Put the $\boldsymbol{x}$ 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.

Even though random forests have been shown to outperform decision trees [30], the limitations of decision trees as described above would still be inherent in the random forest method. 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 been shown that the number of trees grow with the number of features that directly affect the classification outcome [31], 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, overall, 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 [30].

2.4.3 Lasso and Elastic net

In Section 1.2, we discussed how, in this project, not only are we seeking low classification errors, we also have to select features or variables in the data that are relevant in producing accurate predictions. An obvious, but naive, solution is to consider all the features in different combinations (section 2.6.6), but this solution is evidently computationally expensive, much less with data as large as the one we consider in this project.

One method to overcome this problem is by Lasso regression [32], which is a regularised least squares scheme that imposes an l_1-norm penalty on an error function that it tries to minimise. More importantly, in the context of big data and especially this project, the Lasso is an appealing solution because it produces a sparse solution, by shrinking the coefficients of insignificant features to 0.

However, Zou and Hastie [6] examined the limitations of the Lasso method, especially in the context of microarray data. In particular, Lasso has some limitations in variable selection if a subset of features have high correlation with one another.

As a result, Zou and Hastie proposed the *elastic net*, which imposes a linear combination (weighted) of the l_1-norm and the square of the l_2-norm. This method performs feature selection, presents a sparse solution and takes into account variables with high correlation, where groups of correlated variables are not known in advance [33]. Furthermore, Zou and Hastie showed that the elastic net method outperforms Lasso. As such, elastic net might be applicable for our data set.

Besides, we can also utilise the elastic net library in `scikit-learn` implemented in Python. This allows us to experiment with elastic net easily, to see if it would be suitable for our dataset.

2.5 Support vector machines

This section explains and discusses Support Vector Machines[1].

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, $y_i = \pm 1$ is the label of the i-th sample: $y_i = 1$ if the classification is positive (e.g. sample does not have schizophrenia) and $y_i = -1$ otherwise, for $i = 1, \ldots, n$.

Suppose we have data points that correspond to either class 1 or class 2. The Support Vector Machine (SVM) [34] uses a separating hyperplane to distinguish between data points that belong to class 1 and 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 ideal situation of a linear hyperplane perfectly separating training points that have different classification.

The hyperplane is parameterised with weight vector $\boldsymbol{w}$ and a bias b. We thus solve classification problems using linear models:

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

[1]Most of the Mathematics here is with reference to course notes from the Department of Computing, course CO496 - Mathematics for Inference and Machine Learning taught 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`.

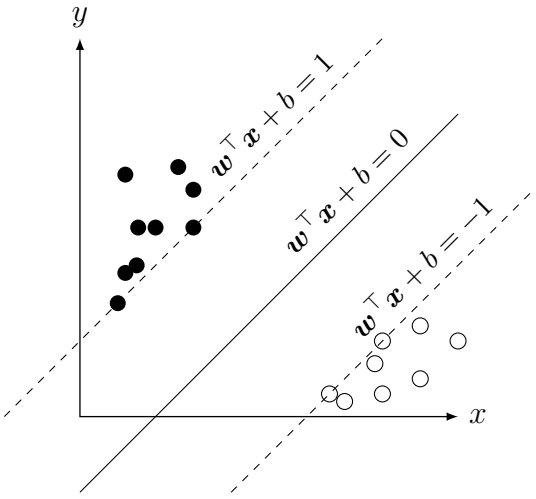


Figure 2.2: Diagram showing training points of different classes (hollow and solid dots) separated by a hyperplane due to $\boldsymbol{w}^\top \boldsymbol{x} + b = \pm 1$. The LaTeX script for this graph was originally created by Peng Yifan.

Finding the hyperplane with maximum margin amounts to solving the following optimisation 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, as defined above, $y_i = \pm 1$, depending on the classification of the vector of features $\boldsymbol{x}_i$. We then get the result:

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

2.5.1 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 [35].
- The primal (2.5) and dual (2.6) problems have the same optimal solutions, if any [36].

To solve the problem in (2.6), we must 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. Doing this would result in the constraints $a_i \geq 0$ and $\sum_{i=1}^{n} a_i y_i = 0$.

We would then obtain an expression of L with respect to $\boldsymbol{a}$ that we wish to 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 $\boldsymbol{K}_y = y_i \, y_j \, \boldsymbol{x}_i^\top \boldsymbol{x}_j$.

While setting the derivatives of L with respect to $\boldsymbol{w}$ and setting to 0, we would obtain the following condition:

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

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

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i y_i \boldsymbol{x}_i^\top \boldsymbol{x} + b \tag{2.7}$$

In order to determine the classification of a new point $\boldsymbol{x}$, we simply have to determine the sign of $f(\boldsymbol{x})$ in (2.7).

2.5.2 Mapping to higher dimensional space

When the relationship between data points in the input space is not linear, the (linear) SVM with the description 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 space (feature space). We can then build an SVM based on this high dimensional space such that the points in the 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 $\forall \boldsymbol{x}, \boldsymbol{y} \in X$,

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

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 (2.7), which determines the classification of a new data $\boldsymbol{x}$, can be written as:

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

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

2.5.3 Slack variables

Real-life data may not be perfectly linearly separable in the feature space $\phi(\boldsymbol{x})$, especially due to the presence of noise [37]. 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 [38], defined as such:

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

This is often referred to as the 1-norm soft margin constraint in the literature [37]. Now, we would need to maximise the margin of the hyperplane, while penalising the misclassified points. We can thus formulate our problem as such:

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{w} + C \sum_{i=1}^{n} \xi_i \tag{2.8}$$

$$\text{subject to} \quad y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \text{for } i = 1, \ldots, n \tag{2.9}$$

where $C > 0$ is the *penalty parameter*. Similarly, to state the dual of (2.8) subject to the conditions in (2.9), 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 derivative of the above with respect to $\boldsymbol{w}$, b and ξ_i to get the following dual problem:

$$\min_{\boldsymbol{a}} \quad L(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^\top \boldsymbol{K}_y \boldsymbol{a} - \boldsymbol{a}^\top \boldsymbol{1} \tag{2.10}$$

$$\text{subject to} \quad \boldsymbol{a}^\top \boldsymbol{y} = 0, \quad 0 \leq a_i \leq C \tag{2.11}$$

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

Now, suppose we choose to map the input space into a higher dimensional feature space, we simply modify K in the above problem:

$$K_y = [y_i\, y_j\, \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)]$$

We then need to solve the quadratic optimisation problem in (2.10).

2.5.4 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 and $\gamma > 0$ 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 [39] noted that the RBF kernel has less numerical difficulties than the polynomial kernel. We note that, $\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in X$,

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

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. We should then focus on the linear and RBF kernels.

Although the RBF kernel is more commonly chosen than the rest of the kernels listed above, the study also revealed that if the number of features is large, using the linear kernel may just suffice, as the nonlinear 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, we should view this conclusion with scepticism and still proceed to investigate the better kernel method (between linear and RBF) by using cross-validation, as the conclusion made in [39] might be data-dependent.

2.5.5 Implementing classification with SVM

Chang and Lin [40] developed LIBSVM, a library for SVMs. LIBSVM utilises the Sequential Minimal Optimisation (SMO) algorithm to solve the quadratic optimisation problem in (2.10). SMO allows the problem to be solved analytically [38]. 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. Furthermore, the library uses LIBSVM internally to handle computations.

2.6 Feature selection

This section explores the literature of feature selection, and discusses certain 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.
- k: Number of features in S, where $k \ll M$. That is, $|S| = k$.

2.6.1 Purpose and motivation

Discriminant analysis, which is common in microarray data analysis and cancer classification (see section 2.3), is the key idea behind 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 or dimensions to predict the result of the classification, we use only a (small) subset of the features. These features should be representative of all the features. Feature selection is also an important step in other areas such as text categorisation [41]. We can also visualise the data set as a matrix, and tackling the feature selection problem is akin to finding a smaller matrix that can be used much more efficiently that the original matrix [42].

The process of feature selection can be described as finding a subset S of the original M features, such that the cardinality of S, $|S| = k$, where $k \ll M$. Furthermore, classification accuracy using these k features should not be significantly lower than using all M features. Finding ways to obtain a satisfactory S is the main concern of feature selection.

As mentioned in section 1.2, the data suffers a *curse of dimensionality*, where the number of features exceeds the number of samples available. This usually causes *over-fitting* on the classifier [43, 44], which refers to the situation where the classifier can classify the training data perfectly, but performs poorly with unseen data [26]. This is a common problem and challenge in classification, and reducing the dimensions of the original data

set often helps reduce overfitting [45]. 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 [46]:

- **Savings in computational cost.** Reducing the dimensions of our features by removing what Kudo and Sklansky call *garbage features* [47] can improve the computational cost of our algorithm, which includes the cost incurred for training a classifier, and using the classifier to perform predictions.
- **Interpretable results.** In contrast with feature extraction, feature selection retains the properties of the original features, instead of returning a combination of these features (see section 2.6.2). For this project, ideally, we want to be able to obtain a compact subset of features from which we can draw biological conclusions.
- **Improvements in classification accuracy.** More features might, due to experimental faults or the nature of the data, lead to more noise in our data set [42, 41]. Having a more compact subset to train our classifier might improve the performance of the classifier [48]. Furthermore, if selected well, a compact subset of features can generalise better than when all features are used [49].

2.6.2 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 new, compact set of features. This implies that the interpretation of the original features is lost through the transformation. Furthermore, the original features in the data might contain data that are irrelevant to the target class, or that might be redundant (see section 2.6.7). 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 [42]. Thus, in this project, we would want to obtain features that are biologically relevant, and can be interpreted by an expert [50]. This allows us to interpret the result of applying classifiers to the reduced set of features [51].

2.6.3 Classification of feature selection methods

Feature selection methods can be categorised into one of the following [52, 53]:

- **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 a subset of features. No learning is involved in filters; they

are independent of the classifier [54]. In general, filters considers how each feature contribute to the classification of the targets, but interaction between features is not considered [55] (see section 2.6.7 on relevance and redundancy). Filter methods include t-test, information gain, F-test and the χ^2-statistic.

- **Wrappers** regard the learning algorithm as a *black box* [46], and evaluate features based on the classification score of the learner. Wrappers usually incur more computational cost due to the cross validation procedure (training the classifier and using it for prediction) when evaluating the features.

- **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 [53].

- **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 [46].

2.6.4 Feature selection methods and trades-off

The categories described above have their own merits and drawbacks. For example, as filters are independent of the classifier, they would be computationally more efficient. However, most filters are univariate, and thus might not take into account feature redundancy (see section 2.6.7). One can use multivariate filters which consider subsets, but these incur more computational cost.

On the other hand, wrappers are reported to produce better classification accuracy than filters [42], 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 [54], 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.

Indeed, Bolón-Canedo *et. al* [42] and Sharma *et. al* [56] echo these trades-off, by pointing out that "the best method" does not exist and that no single algorithm stands out in all aspects. Furthermore, Hua *et. al* [54] performed a review of feature selection algorithms using different data sets, and came to the conclusion that none of the methods that were reviewed performed consistently well across all the data sets. This sentiment is also reflected in [57].

To this end, previous works on feature selection seek to find a method that reports good classification accuracy for a specific problem setting. This is evident in previous works in the literature - 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 the existing ones in literature for a particular problem. This is precisely the strategy this project will take in chapter 4.

2.6.5 Terminating feature selection algorithms

Feature selection algorithms can be terminated via the following ways [47]:

- **Specifying** k: The algorithm stops when the number of features we desire in S is obtained.
- **Best performance**: The algorithm terminates when the performance of S - for example, with respect to a classifier - exceeds a specified threshold.
- **No significant improvement**: The algorithm can terminate when the performance of S does not improve much [56].
- **Balance between performance and subset size**: Optimise both classifier performance and k.

It seems that from most works in feature selection, authors often specify a fixed value of k. Furthermore, in [58] by Tang *et. al*, experiments on different datasets conclude that their algorithm might be a good choice for a small number of samples, with large M and k. k must also be defined beforehand. The experimenters set this number to $k = 100$ for all the datasets that were explored, but later recommended different values of k for two of the datasets that were used.

This suggests that k is dependent on the dataset. If k is not a constraint of the problem (e.g. in bioinformatics problems, we can specify that we just require k of the most informative genes), we could perform a *grid search*. However, due to time and computational constraints, one will probably not be able to perform a grid search on all the features (i.e. $k \in X = [0, M]$), but only on an interval of X. So, even if we find an optimal k in that interval of X, it is most likely a local optimum.

2.6.6 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.

Suppose we have M features. An exhaustive search that searches through the space of all possible subsets of features would iterate through 2^M of these subsets [59]. 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 (call it S), $|S| \in [0, M]$. For completeness, we include the trivial cases of $|S| = 0$ (all M features do not contribute to the result of the classification task) and $|S| = M$ (all M features are important). Then, we want to prove that the number of subsets searched is 2^M:

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

where, $\binom{M}{k} = \frac{M!}{k!(M-k)!}$ is the number of different 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 would be:

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

where in the second line, the following results were used:

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

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

□

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"* [60]. Thus, most of the algorithms in the literature of feature selection trade the optimality guaranteed in exhaustive search for computational cost. These methods involve some kind(s) of heuristic to guide the search for S. Some examples would be discussed in the following sections.

2.6.7 Feature relevance and feature redundancy

Relevance and redundancy play important roles in the literature of feature selection. In [42], 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 and rank the relevance of the features. It is tempting to select the first k features as our final subset S, but, as

Peng *et. al* put it, "The best k features are not the k best features" [59], as the highly ranked features are likely to have similar discriminating power [42]. In other words, some 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 2.6.11). [7] 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).

Greedy searches, which will be explained in detail in the later sections, use relevance or redundancy to guide the search in the feature space. For example, it evaluates a subset of features based on, say, classification score of a machine learner, to determine the discriminating power of that subset.

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, the feature selection method 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 on these features.

2.6.8 Greedy search

As mentioned in section 2.6.1, 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, researchers have proposed several algorithms that 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 optimal solutions - thus classified as sub-optimal - which is very beneficial in the context of high dimensional data, and are more practical to execute [46, 59].

For example, the *sequential forward selection* (SFS) [61] 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, $J(\cdot)$, which, in this context, is the classification score of the learner from just using m alone (i.e. only 1 feature). The scores are then sorted, and the feature (call it f_1) with the highest score is selected. 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. Let f_2 be the feature where the set $S = \{f_1, f_2\}$ has the highest classification score. The process continues until we have our desired k features.

A variant is the *sequential backward selection* (SBS), which starts with all M features instead and SBS sequentially removes one feature. However, we note that SBS starts with the full set of M features and works backwards. This suggests 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 [62]. Furthermore, SBS might give us a better result, but we might end up with larger feature subsets and longer computational time [63].

Another variant is the *floating forward/backward search* strategy proposed by Pudil [64].

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. This floating variant tackles this problem.

Lastly, Deng [65] 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. Suppose the corresponding features are $\{f_1, f_2, \ldots, f_N\}$. That is, $J(f_1) > J(f_2) > \cdots > J(f_N)$. f_1 would be selected as the first feature. Then we consider $\{f_1, f_2\}, \{f_1, f_3\}$ all the way to $\{f_1, f_{M/2}\}$. That is, we only consider $M/2$ of the sorted features. 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. RFS is performed on our data set, and will be discussed is more detail in section 4.4.

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 ⟵ [ ]
    Initialise empty list S ⟵ [ ]
    for i = 1 ⟶ M do
        s ⟵ CV score for ith feature.
        L ⟵ L ∪ {s}
    w ⟵ feature that corresponds to max L.
    S ⟵ S ∪ {w}
    for i = 2 ⟶ k do
        num_iterations ⟵ M/i
        Initialise empty list scores ⟵ [ ]
        j ⟵ 0
        while |scores| ≤ num_iterations do
            if L[j] ∈ S then
                j ⟵ j + 1
                continue
            S_j ⟵ S ∪ {L[j]}
            s ⟵ CV score for S_j.
            scores ⟵ scores ∪ {s}.
            j ⟵ j + 1
        w ⟵ feature that corresponds to max scores.
        S ⟵ S ∪ {w}
```

2.6.9 Recursive feature elimination

Another flavour of search is the Recursive Feature Elimination (RFE), first proposed by Guyon *et. al* in [45] and it has been used in different contexts [66, 67, 68]. In particular, the paper by Guyon *et. al* focused on RFE when used with SVMs. RFE can be classified as a wrapper method.

From section 2.5, the problem of fitting an SVM with a linear kernel boils down to solving the optimisation problem:

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

where the weight vector $\boldsymbol{w}$ and bias term b are parameters of the equation 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 $\boldsymbol{x}$ [66]. 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 (or the cross validation score exceeds a threshold). In cases where features are eliminated one at a time, RFE can be viewed as a *feature ranking* procedure. 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, discussed briefly in section 2.6.8. As such, one disadvantage of this method 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, although, naturally, optimality might be compromised in this manner.

2.6.10 Optimal search by branch-and-bound

It is also worth mentioning the work of Narendra and Keinosuke [69], 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 [64] due to its exponential time complexity [62, 70]. 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.

2.6.11 Use of mutual information in feature selection

A common heuristic to quantify relevance and redundancy is mutual information (MI) [26]. MI can be interpreted as a measure that quantifies the dependence between variables [71] and the amount of information X carries about Y [72].

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

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)\, p(y)} \right) \tag{2.12}$$

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

$$I(X,Y) = \int_{x \in X} \int_{y \in Y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)\, p(y)} \right) dy\, dx \tag{2.13}$$

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:

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

For example, the *minimal-redundancy maximal relevance* (mRMR) scheme [59] makes use of MI extensively to select features. This scheme 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.

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 2.12 and 2.13). For discrete or categorical variables, the estimation of the probabilities would simply be obtained from frequency counts [73], although the estimation would only be accurate if we have enough samples [59]. Yet, with more classes and more states in each class, the estimation of the joint probability would become more difficult [46].

In the case of continuous variables, the computation of MI would be even harder [74], as we would need to compute the integral of the continuous probabilities [50]. We should also note that the features in the data for this project are continuous. One possible way to estimate the probabilities of continuous variables is 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 the width of the intervals, and the widths (or the number of bins) will influence the estimates significantly [75].

Another method to estimate the probabilities in equations 2.13 and 2.14 is by using the *Parzen Window* method, as proposed by Kwak and Choi in [73]. 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 densities to converge to the true density [76]. 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 matrix of the variables, but these assumptions might not hold for all types of data sets. Nevertheless, if time is not of the essence, one could perform a search to find the most optimal window width using different window functions.

Consequently, methods such as *maximum relevance minimum multi-collinearity* (MRmMC) [7] avoid the complexity of estimating MI between continuous variables by proposing other measures to quantify relevance and redundancy. As the measures used in this work will be used on our data set, more details will be discussed in section 4.5.

2.7 Genetic algorithms in feature selection

Genetic algorithms (GA's) are a type of heuristic search method to explore solutions by repeatedly changing and combining them. It is based on the theory of evolution and its concept of natural selection and *survival of the fittest* [77]. GA's search through a space of potential solutions by using probabilistic genetic operators to avoid local optimas and produce better solutions [47, 78, 79]. Although GA's are reported to take more time to select features, they can explore a wider variety of subset features [55].

The following terms are often used in the context of GA's:

- A **chromosome** describes a solution [77, 80]. Usually, a chromosome uses a binary encoding to represent a solution. In the context of feature selection, an array, say A, can be used to store a chromosome, and each element in the array, $A[i]$, is either 0 or 1, which means a feature is not selected or a feature is selected, respectively.
- The ith **generation** describes the ith 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, [81] uses classification error as its fitness criteria, [82] uses both the classification error and the dimension of a chromosome, while [83] uses correlation, which might not be feasible when we need to measure the correlation between a discrete and a continuous variable.

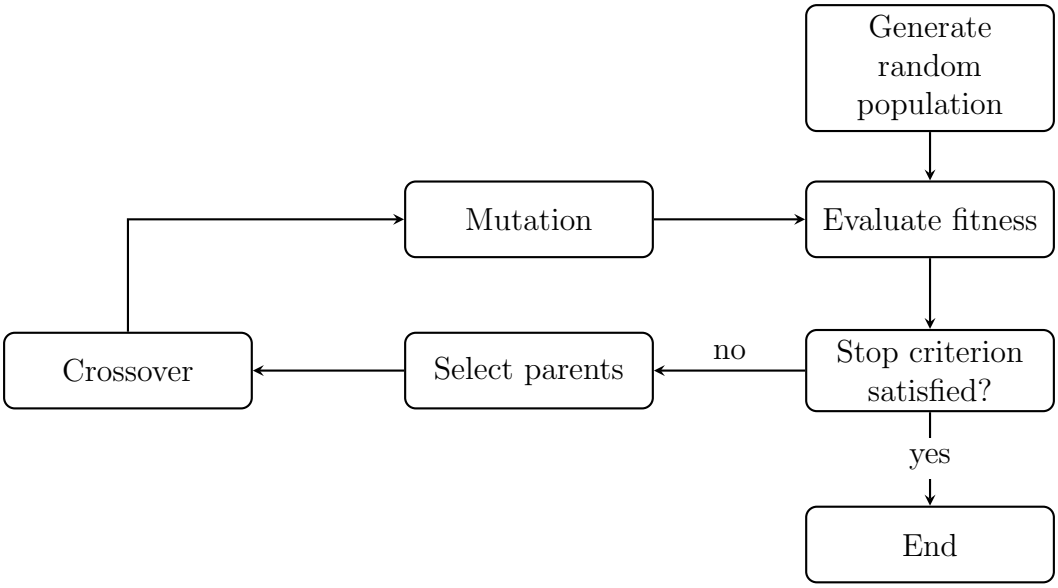


Figure 2.3: Flowchart showing the steps that are usually involved in a typical genetic algorithm.

- 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 [83], two-point [84, 81] and uniform crossover [85].

- The **mutation** operator probabilistically selects and changes one or more elements of a chromosome. For example, [83, 81] use a uniform distribution to select these elements.

- The **selection** process selects the fittest chromosome(s) of a population. Depending on the strategy that one employs, the selected chromosome(s) will either be selected as parents for crossover, or they will be retained and moved to the next generation (*elitism/cloning* strategy [84, 86, 87]). One can also select the chromosomes with the worst fitness for crossover, in hope that this will raise the overall fitness of the population [88].

These operators are responsible for evolving a population and help to increase the *diversity* of a population [89], which can be interpreted as a balance between *exploitation and exploration* [89]: risking a good solution encourages the algorithm to explore other solutions, possibly allowing us to escape a local optimum.

GA's encapsulate the concept of *survival of the fittest*, as the fitter a chromosome is, the more likely it is either copied to the next generation, or selected as a parent for crossover [80]. Furthermore, Kudo and Sklansky also recommend GA's for "large-scale" problems that involve more than 50 features [47]. The main and typical steps in a genetic algorithm can be found in the flowchart in Figure 2.3.

Moreover, genetic algorithms does not require gradients to find an optimum solution. This avoids the numerical and computational complexity associated with other search methods [90]. The GA can also make use of parallelism to speed up computation (see section 4.7).

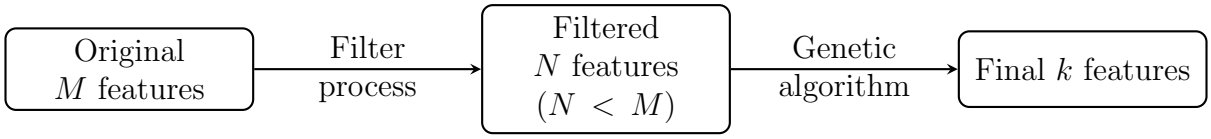


Figure 2.4: Flowchart depicting the two-stage genetic algorithm.

2.7.1 The "two-stage" genetic algorithm

There are several ways to implement GA's, and each has its own merits and drawbacks. For example, in a method described by Mitchell in [77], it initially generates p (defined by the user) solutions at random. For example, this method is used in [91], where 20 solutions are randomly generated to form the initial population of the GA. However, we note that the GA in [91] was performed on a dataset of only 9 features. 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 [41, 92, 93, 82, 88] 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, [41, 92, 88] 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. However, although [83] 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.

2.7.2 Parameters and strategies

Even though previous works using GA's have reported positive experimental results, one difficulty of GA's is the number of parameters that need to be tuned [47], including but limited to:

- Population size
- Maximum number of generations
- Crossover rate
- 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 [88], or selected in proportion to their fitness (*Roulette Wheel selection*) [77, 87]. The mutation rate, which often takes a small value (e.g. 1 to 15%) [91], say m, will uniformly choose m percent of a chromosome, and these elements would have their bits inverted.

One would need to take the time to investigate what parameters and/or strategies would work best for the data set, and justify any parameters that seem arbitrary.

Nevertheless, as an extension proposed by the authors, it is interesting to investigate if incorporating a GA with the MRmMC method (briefly discussed in section 2.6.11) will improve the classification score. This will be discussed in detail in section 4.6.

2.7.3 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 [94, 95]. Interestingly, two aspects of crossover can vary: its rate and its type, which come in different flavours, including one-point, two-point and uniform crossovers.

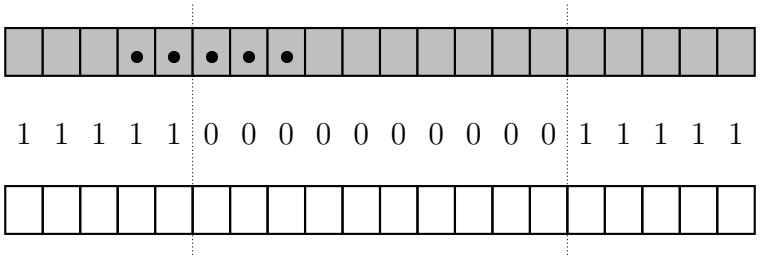
In the *one-point* crossover, a position (crossover point) along a chromosome is randomly chosen. The segments partitioned by this position are swapped between the two parents to produce two offsprings. In the *two-point* crossover, two positions are chosen and the segments in the parent chromosomes are similarly swapped. In the *uniform* crossover, each position is a potential crossover point. The partition can be represented as a *crossover mask* [77]. An example of how the two-point crossover works can be found in Figure 2.5, which shows the crossover mask as a binary string with two segments of 1's. On the other hand, for a uniform crossover, each bit in the crossover mask will be generated randomly.

Ultimately, what the crossover, and also the mutation, operator does is to ensure that segments of information encoded in each chromosome (also referred to as *schemata* [96]), are sampled sufficiently. Suppose that there is a segment along the chromosome that improves the performance of the subset selected, or is crucial to good performance. For example, this segment resembles the global optimum. This is illustrated by dots in Figure 2.5a. The crossover might thus disrupt this segment, as shown in Figure 2.5b.

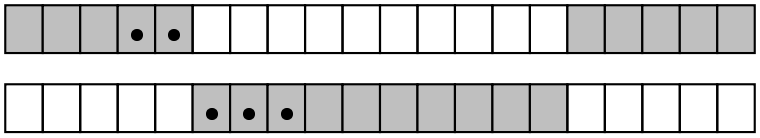
Consequently, the one-point crossover will less likely disrupt these segments compared to the two-point and crossover operators, and segments that perform well are less likely to be broken using one-point crossovers [96]. Furthermore, the one-point and two-points crossover operators introduce a *positional 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-points crossover. On the other hand, while uniform crossovers do 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 [96].

In [96], Picek and Golub discuss the performance of the aforementioned crossover operators. They found that GA's 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. However, ultimately, Picek and Golub conclude that each crossover operator has its merits and drawbacks, and this echoes the discussion in section 2.6.4 where the strategies and parameters have to be tuned according to the problem setting, and no single strategy outperforms all other strategies.

The fact that the *rate* of crossover also varies across problems is similarly brought up by Lin *et. al* in [97]. To tackle this variability, they have proposed a scheme that adjusts the



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



(b) After crossover operation. The two arrays shown are the offsprings of the above parent chromosomes. In this case, the 1 in the crossover mask indicates that the bit in the parent chromosome stays, while 0 indicates that the bit is swapped.

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

mutation and crossover rates according to how the solutions performs in each generation, which eliminates the need for grid searches for these rates. The rates are adjusted based on the average increase or decrease of the fitness of the offsprings. Empirical results have shown that this adaptive scheme improves the performance of GA's [97, 95].

Furthermore, in the classic genetic algorithm, the rate of crossover determines the proportion of the population chosen to be chosen as parents [77]. As mentioned above, the crossover operator might disrupt segments of good solutions, but it helps to improve the diversity of the population.

2.7.4 Regarding selection

Selection is another important probabilistic 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" [98], which the crossover and mutation operators change, to produce a new generation.

In [99], 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 discussed above. 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 [90].

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` [100] method `numpy.random.choice` allows one to modify the probability associated with sampling an arbitrary n elements from an array.

However, there are many problems that are commonly associated with the roulette wheel selection. In [77], Mitchell describes a potential *crowding* problem associated with this method. One can imagine that solutions with much higher fitness than the rest of the solutions will get chosen as parents very often, possibly making the algorithm stagnant. This might also result in a population having similar solutions, hence reducing the diversity of the population. This problem is also raised in [101]. On the other hand, if all the solutions in a population have similar fitness, then the improvement associated with the next generation will be limited, since each solution is equally likely to be selected for the intermediate population [102].

An alternative is the *tournament selection* method. A pool of, say N_t, solutions is randomly created from the population. From these N_t solutions, the solution with the highest fitness will be chosen to populate the intermediate population (mentioned above). This process is repeated until the intermediate population is filled up. Binary tournaments where $N_t = 2$ are frequently held [90, 103]. In this case, the solution with the better fitness is chosen with a probability p_t, where $0.5 < p_t \leq 1.0$. From [90], 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 have a chance to get selected to enter the tournament, the fitter solutions will be less likely to dominate the intermediate population [102].

Works such as [102, 104] conclude from empirical results that the tournament selection method is the better method between tournament selection and roulette wheel selection. However, as mentioned often in this report, there is no universal strategy or set of parameters in feature selection methods. It will be interesting to investigate if choosing between the roulette wheel or tournament selection method will make a difference for this project.

Chapter 3

Data exploration

3.1 Understanding the data

The data set[1] contains 2 giga bytes worth of data, with dimensions 420374×847. 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 long time.

To circumvent this issue, the Python library `pandas` [105] was used. `pandas` provides high performance and user-friendly data structures and tools to analyse data sets. In particular, `pandas` provide a `pandas.read_csv` method that reads the csv file. The data set can then be cast into a `DataFrame`, analogous to the data structure common in the *R* programming language.

One particular advantage of using `pandas` is that the `DataFrame` has an `Index` object, that is able to store the names of the labels along the axes. For example, by examining 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` indicates a test case (individual) while the label `cg00000029` refers to a feature (methylation site - see background in section 2.1).

`pandas` also work well with lists and arrays in the standard Python library, and data structures in `numpy` [100]. Furthermore, calling `pandas.read_csv` and loading the data into a `DataFrame` each time we need the data set will impede efficiency. Fortunately, `pandas` offers the `HDFStore` class, which utilises `PyTables` [106] to allow fast read and write of `pandas` data structures. Therefore, each time we need the data set, or indeed any data that takes a lot of time to read, we simply have to retrieve it from the `HDFStore`.

Finally, the `pandas` data structures work well with the `scikit-learn` software, which is used extensively in this project, such as fitting an SVM and obtaining cross validation scores (see section 3.2).

[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 which one would obtain when attempting to print the `DataFrame` that contains the (shuffled) data set. We can use this output to study the information available in the `DataFrame`.

3.2 Using SVM's

For all numerical experiments involving the different methods described below, the SVM will be used should there be a need for a classifier. For example, if the method used is a wrapper method, the cross validation error of fitting an SVM would be used to evaluate the algorithm. In contrast, in a filter method, the SVM would not come into play. This is similar to previous works such as [54], where the classifier is fixed, and attention is focused on the trends displayed by the feature selection algorithms.

The SVM is fitted using the `scikit-learn` Python software [107]. Calling the `SVC()` method would create a classifier based on `libsvm` [40] with default parameters, such as the default Radial Basis Function (RBF) kernel, and the error parameter $C = 1.0$ (see section 2.5.4). However, a practical guide by Hsu *et. al* in [39] explains that when the number of features is much larger than the number of samples, using a linear kernel would suffice; in other words, we might not need to map our data into higher dimensions.

This hypothesis was tested on our data set. Indeed, fitting an SVM using the RBF kernel with all the features, the cross validation score obtained was 0.834644, while with the linear kernel, the score was 0.929160. The cross validation scores can be obtained via the method `sklearn.model_selection.cross_val_score`. The number of folds performed performed is 5, which will be used throughout the experiments in this project. Furthermore, fitting an SVM with either the linear or RBF kernel took about 30 minutes using HPC [108].

Furthermore, 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 would allow us to find the optimal C. However, due to time constraints, the following experiments would stick to the default value of C. Moreover, this project focuses on the feature selection algorithms, not the parameters of the SVM.

Chapter 4

Methods and approaches

4.1 Data preprocessing

The aforementioned guide by Hsu *et. al* also described the importance of scaling our data before fitting an SVM on the data. In particular, scaling the data would prevent features with large values from dominating those with smaller values. Restricting the values of features such that each feature has zero mean and unit variance is also recommended when using the `SVC()` (support vector classification) package. This procedure is also conducted in other works such as in [45]. In this project, the data was scaled with the `sklearn.preprocessing` package in `scikit-learn`, using the `scale` method.

Furthermore, as the raw data was sorted based on the classification (target label) of each sample, it is sensible to *shuffle* the data such that the target labels would not be in a particular order. The cross validation method also partitions the data set into folds, and the process of fitting a classifier on a fold would be affected if all the target labels are the same. The shuffling was done using the `sklearn.utils.shuffle` method. The method also allows us to set the reproducibility of the shuffling. Furthermore, once the entire data set and labels are shuffled, these are stored in the `DataStore`, and retrieved if needed (see section 3.1).

4.2 t-test

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

The t-statistic and its corresponding p-value can be computed using a method in the `scipy` [109] software: `scipy.stats.ttest_ind`. 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 [110], where

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

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

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

As discussed in section 2.6.5, 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 $\mu_1 \neq \mu_2$), or choose a certain threshold and discard those features whose p values *exceed* this threshold.

4.2.1 Experimental results

We present plots obtained from t-test using different experimental settings:

- Sample k in broad intervals to investigate the general trend of t-test (Figure 4.1).
- Sample $k \in [10, 2000)$ with step size of 10 (Figure 4.2).
- Magnified view of Figure 4.2, where $k \in [10, 2000)$ (Figure 4.3).

When plotting such graphs, it is essential to keep in mind the purpose of feature selection: we want to choose a subset of features such that fitting an SVM with these features will not cause the SVM to suffer significant performance degradation as compared to using all the features. To this end, we have to compare the performance of the methods to how an SVM will perform when using the full set of features. This is indicated by plotting a horizontal dotted line in the graphs.

We include in Table 4.1 values that correspond to the graph in Figure 4.1. For this table of values, we sampled k broadly from 1 to 420374 (the full set of features). As mentioned

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

above, the t-test method allows us to generate a list of p-values for each feature, and we will sort the p-values in ascending order. Then, for each k in the range of k values we have, we will select the top k features from the list, fit it into an SVM, and obtain cross validation scores. For example, when $k = 500$, we will select the 500 features that have the lowest p-values, that is, the top 500 features in the sorted list.

From the resulting graph in Figure 4.1, we notice that the cross validation score peaks at around $k = 15000$ with CV score 0.950448, and after this value of k, the CV score seems to have a decreasing trend.

It is evident that due to the coarse granularity of the values of k that we have sampled, this method might miss out on some possible local maxima, especially after the maxima at $k = 15000$. 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 general trend of the t-test.

Furthermore, if we are interested in a range of values of k, we can easily plot a similar graph and sample k with smaller step sizes. This is precisely what was done in Figure 4.2, where k ranges from 10 to 2000 with a step size of 10. 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 $k = 1500$. This peak is precisely $k = 1589$, where the corresponding CV score is around 0.932738 (see Figure 4.3).

4.2.2 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 2.6.3, univariate filters are computationally efficient: in this case, the method only iterates through all the features once. The efficiency of this method can be further enhanced by tapping on the parallelism inherent in the problem (see section 4.7). 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 easy to achieve.

However, one stark drawback of this method is the fact that the t-test method only considers each feature singly. Even though its design is simple compared to multivariate

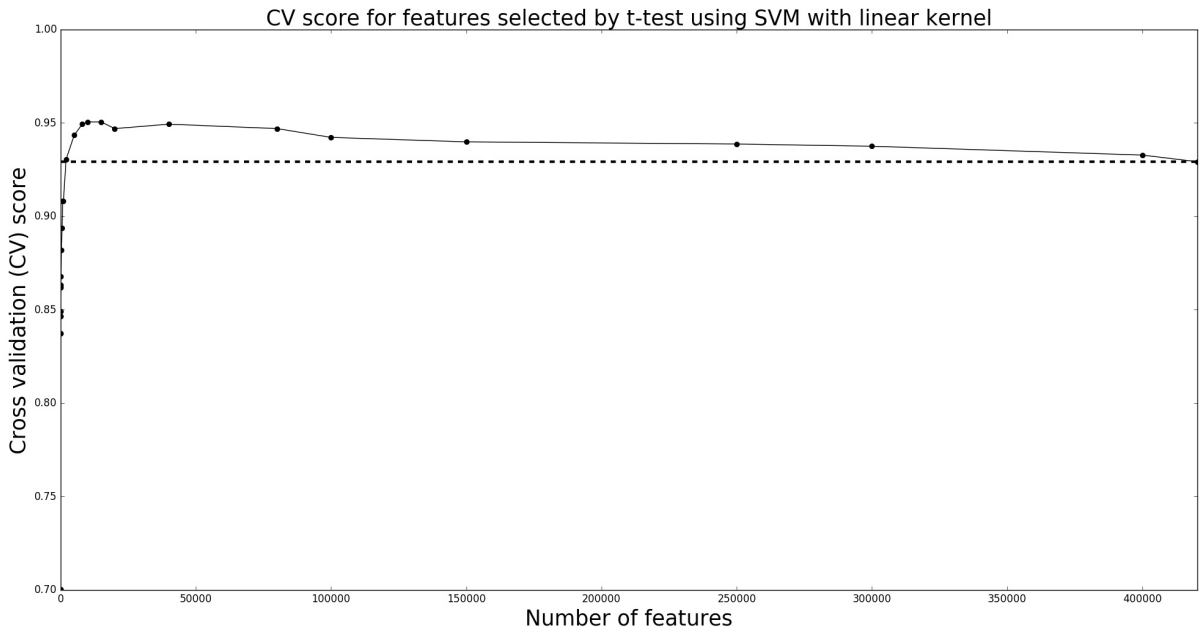


Figure 4.1: Plot of cross validation score against number of features, when an SVM with a linear kernel is fitted with the selected number of features. The dotted line represents the cross validation score with all 420374 features.

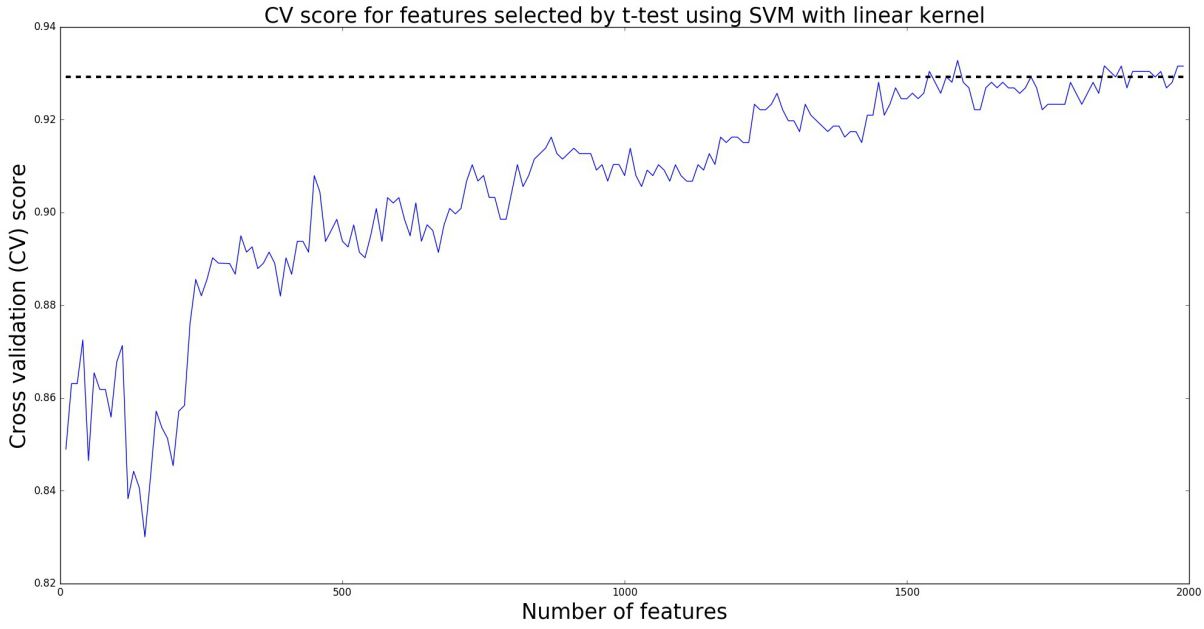


Figure 4.2: Plot of cross validation score against number of features, when an SVM with a linear kernel is fitted with the selected number of features, $k \in [10, 2000)$, with step size of 10. The dotted line represents the cross validation score with all 420374 features.

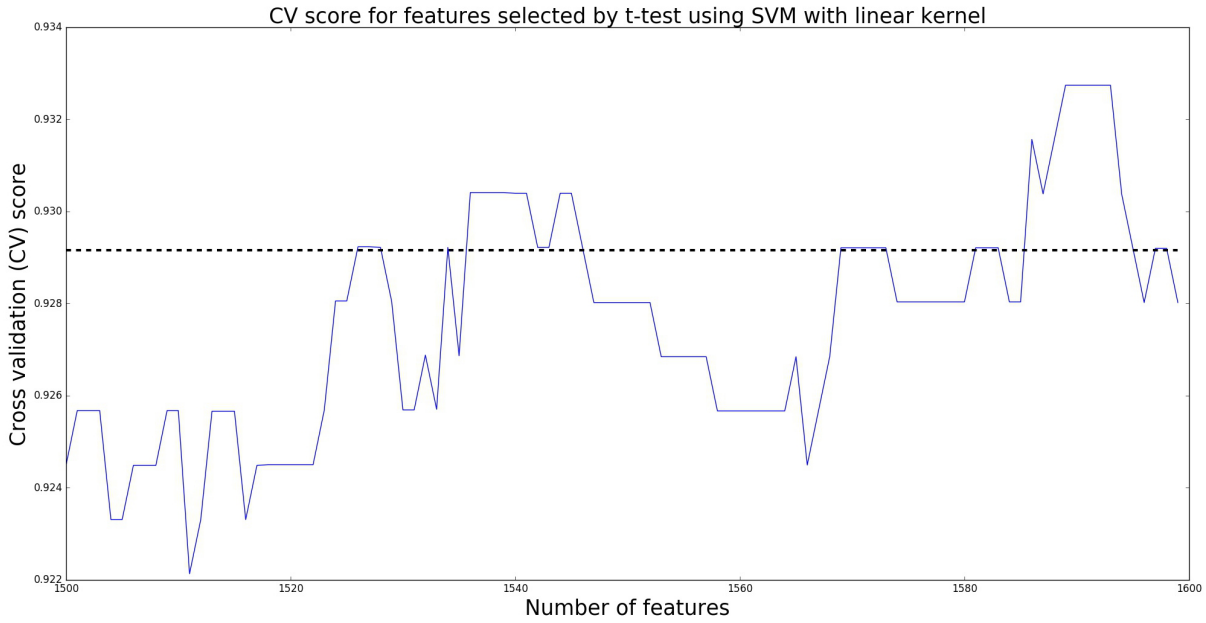


Figure 4.3: A magnified view of Figure 4.2. The peak is at $k = 1589$.

filters or wrappers, it is not possible to take redundancy into account [57], at least not in the basic implementation of t-test. The t-test method ranks the features according to the t-statistic (or alternatively, the p-value), but redundant features are known to have similar rankings [42]. As such, we might end up selecting highly ranked, but similar, features in our classification. For example, [111] 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.

4.2.3 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 score will differ significantly from the t-test method. Since this seems to be a greedier and lazier version of Sequential Forward Selection (SFS) (see section 2.6.8), we call this method "lazy" SFS.

First, we iterated through each feature and fitted an SVM using just that feature. So, the SVM is trained on one feature at a time only. Then, a list of cross-validation scores are obtained. We then sort this list in descending order. We then used the same range of k values as in Figure 4.1 in section 4.2. For each k value, we simply took the first k elements in the sorted list. That is, we fit an SVM using the k features that perform best individually. We then obtained the graph in Figure 4.4. This method also differs from the t-test method, as it can be evidently classified as a wrapper method, since the classifier is required to obtain the CV score.

We can see from the graph that at first, there is a similar trend to that in t-test in Figure 4.1: the CV score increases at a very fast rate, peaks at $k = 8000$ with CV score 0.952794, then the CV score decreases with more features. However, unlike the graph in Figure 4.1, the CV score peaks again slightly at around $k = 300000$. We can investigate this

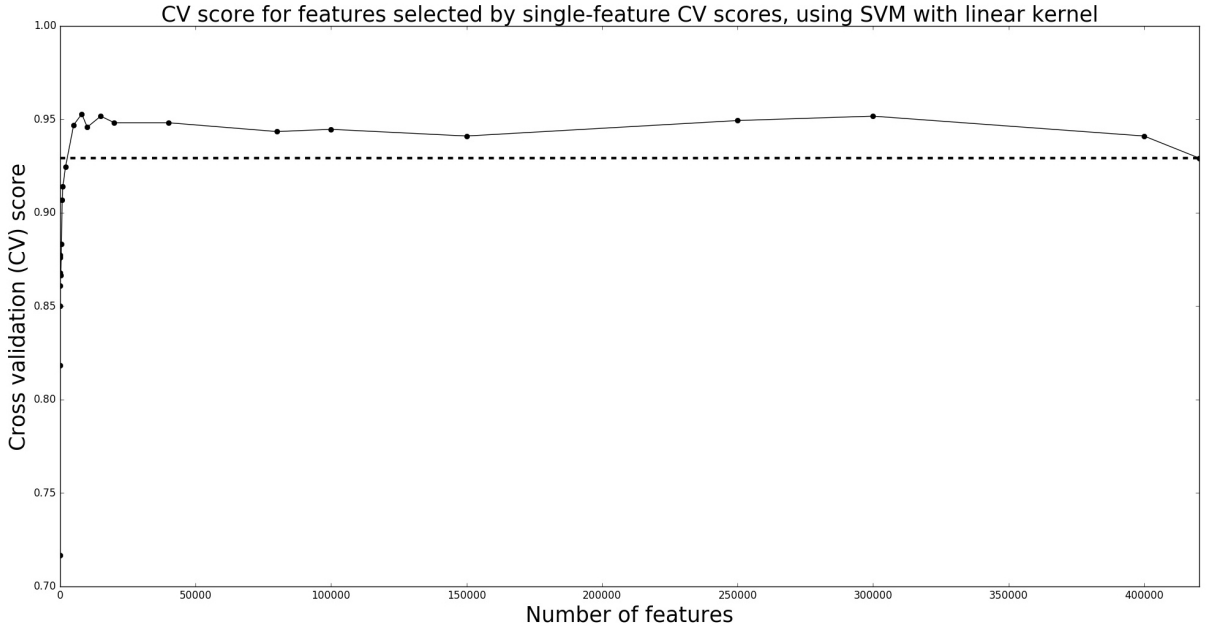


Figure 4.4: A graph of CV scores against number of features using a lazy variation of Sequential Forward Selection.

slight peak by plotting another graph shown in Figure 4.5, focusing on features between 250000 and 320000 with a step size of 1000. The maximum CV score in this range was 0.955161 with $k = 310000$.

We get a dilemma in this situation, as the subset with 310000 features performs slightly better than that with 8000 features. We think that it is only sensible to compromise 0.003 of classification accuracy for about 300000 lesser features, as using only 8000 features for classification will impose much smaller time constraints.

Furthermore, we can see the similarity and difference between the lazy SFS and the t-test methods clearly when we plot the two graphs in the same diagram in Figure 4.6.

Similarly to the discussion for the t-test method, this lazy SFS method only considers each feature singly, and does not take into account any form of redundancy, as features that perform well individually might contain redundancies with respect to other features. Furthermore, features that do not perform well individually might perform well with other features.

Moreover, from the graph in Figure 4.6, we can see that both methods have similar performance when k is small (i.e. $k < 150000$). However, beyond this value of k, the lazy SFS method seems to outperform the t-test method, although this difference in CV score is not significant (about 0.01).

It is also interesting to note that the t-test gives a maximum CV score of 0.950448 with $k = 15000$, while using the lazy SFS method (ignoring the peak at $k = 310000$), we get a maximum CV score of 0.952794 with $k = 8000$. Evidently, the lazy SFS method can produce a smaller subset that performs slightly better than the one selected by the t-test.

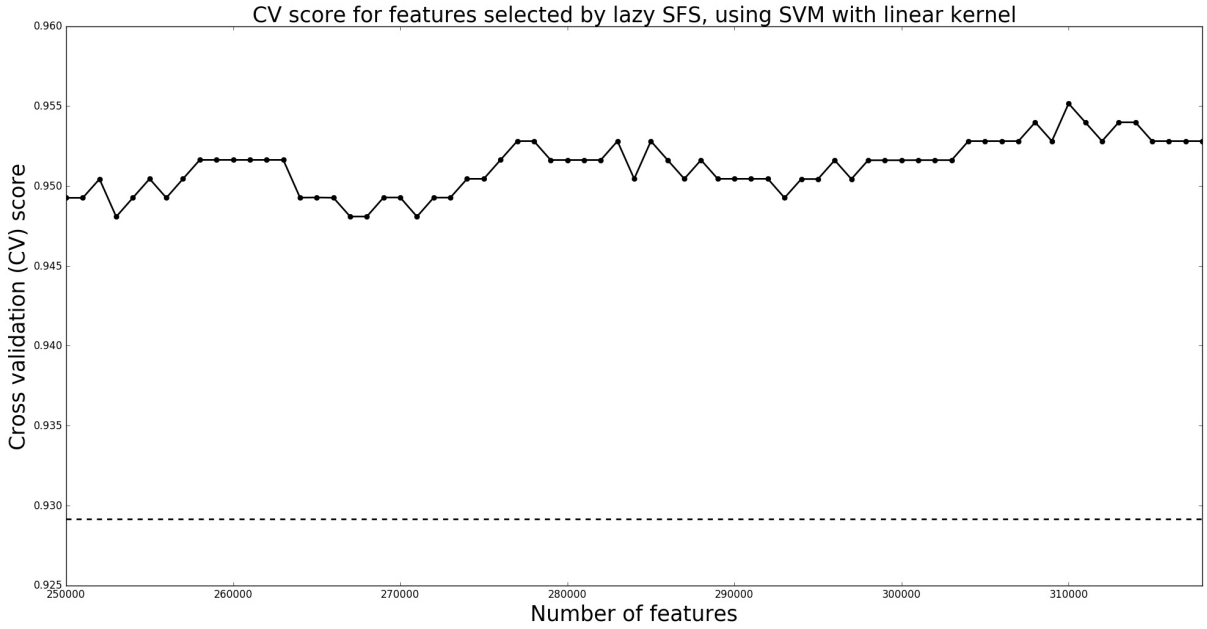


Figure 4.5: A "magnified" graph of CV scores against number of features using lazy SFS, focusing on the slight peak at around $k = 300000$.

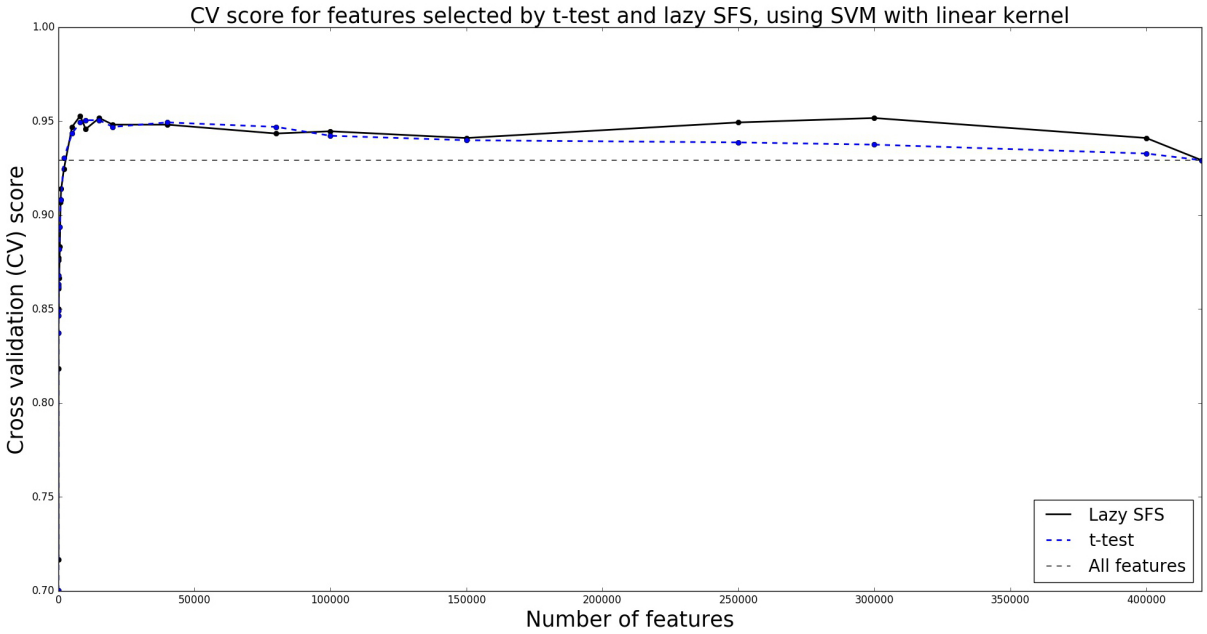


Figure 4.6: A graph of CV scores against number of features using both t-test and lazy SFS.

4.3 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 to be unimportant. As what was discussed in section 2.6.9, specifying a step size larger than 1 will cause the algorithm to consider nested subsets in each iteration.

4.3.1 Experimental results

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 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 one feature at a time (see section 2.6.9). It will then eliminate the feature with the lowest weight. This is obviously computationally intensive, especially if we specify a low `n_features_to_select` parameter, such as 5. Thus, the step size varies according to the `n_features_to_select` parameter. This variation is summarised in a table in Figure 4.7. Also, we can obtain cross validation scores based on the features that are deemed important by the algorithm. The CV scores can also be found in Figure 4.7.

Next, we plot a graph of how the cross validation scores vary with the number of features selected. This graph can be found in Figure 4.8. Also, it is instructive to plot the variation of the scores for just the number of features that were specified. This graph can be found in Figure 4.9.

From the graph in Figure 4.9, we can see that there is an overall increasing trend with respect to the cross validation scores. However, these scores fall below the horizontal dotted line that represents the CV score when we use all the features. That is, using RFE to select features does not seem to benefit the feature selection method. We can put this result in context of the t-test experiments performed earlier. We notice that using the t-test to perform feature selection produces better results than RFE. We also note that the t-test is a computationally less intensive algorithm 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 (see section 2.6.9). It will then produce a new set of features consisting $F - f$ features. Thus, the step size f here is important. Looking at the table in Figure 4.7, 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. This is so because of computational time and resource constraints. Thus, this coarse step size might be the reason why the algorithm might not perform as well as the t-test method.

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

Figure 4.7: Table of parameters used in genetic algorithm implementation of MRmMC.

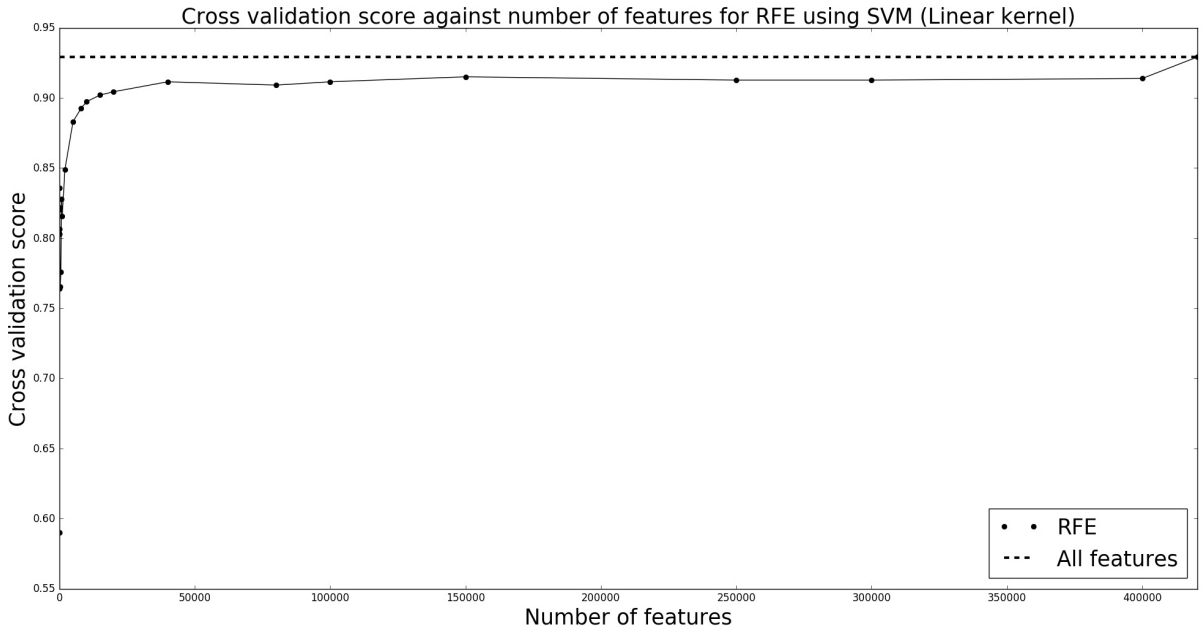


Figure 4.8: A graph of CV scores against number of features using both RFE.

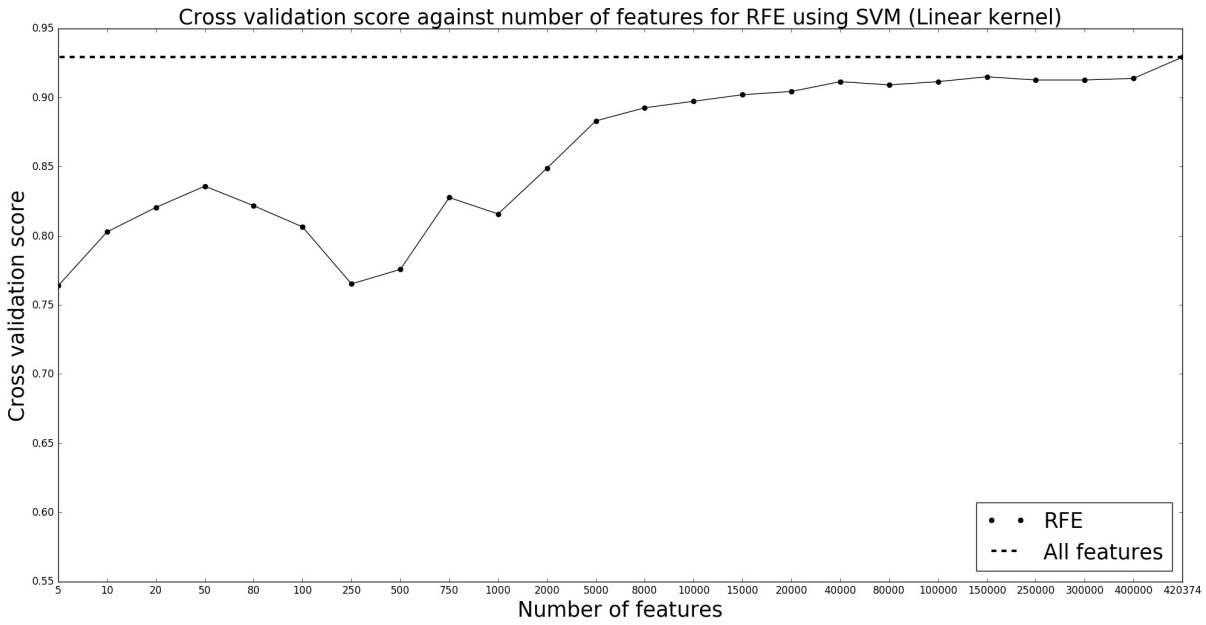


Figure 4.9: A graph of CV scores against number of features using both RFE, showing just the trend of the scores.

4.4 Restricted Forward Selection (RFS)

The pseudocode of the RFS algorithm [65] is listed in algorithm 1 on page 19. The RFS is a greedy heuristic search that searches through part of the space of features that are the most promising; that is, it only considers features that give the best cross validation results.

Suppose there are M features in total. Firstly, RFS iterates through each feature and records the cross validation result by fitting an SVM with just one feature. A list is used to keep track of these scores. It is then sorted. The feature with the best cross validation result is selected as the first feature (call it f_1). Then, in the second iteration, RFS iterates through this sorted list and pairs each remaining feature with f_1 to form a pair, and an SVM is fitted using this pair of features. RFS then continues down this list until $M/2$ features are formed. In the third round, RFS will iterate through $M/3$ of these features and so on.

Since the RFS algorithm uses the SVM to determine the classification score, it can be classified as a *wrapper* method. Furthermore, since this method directly compares the classification score of each subset of features, it does not explicitly quantify relevance and redundancy.

While the sequential forward search (SFS - section 2.6.8) incrementally selects the best feature to include in the existing set of 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 considers in each iteration are those that produce the best cross validation results individually.

4.4.1 Runtime

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

Then, we can calculate the number of evaluations the algorithm performs:

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

It is a well-known result that the series $\sum_{i=1}^{\infty} 1/i$ diverges. Yet, 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 $n = \log k$ (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 in this context is the number of features that we want to retain from the full feature set. 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 "round". So, its complexity would be $\mathcal{O}(Mk)$. In the extreme, if we want SFS to iterate through all the features, instead of terminating with k features, the runtime complexity of SFS would be $\mathcal{O}(M^2)$.

It is evident that since RFS is a greedier variant (it iterates only a fraction of the features each time) 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 manageable amount of time, although we have to be aware that the optimality of RFS is not guaranteed.

4.4.2 Experimental results

4.5 Maximum relevance minimum multi-collinearity (MRmMC)

Recently, MRmMC [7] 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 (see section 2.6.11). Furthermore, MRmMC does not require any parameter. Unlike RFE and RFS, MRmMC does not directly use cross validation scores to quantify relevance and redundancy.

4.5.1 Relevance

MRmMC uses a correlation coefficient, first used in [112], to quantify the relevance of each feature to the target class. This correlation coefficient is:

$$r_{qn}(X,Y) = \left(1 - \frac{E[\text{var}(X|Y)]}{\text{var}(X)}\right)^{\frac{1}{2}} \tag{4.1}$$

To understand the rationale behind this measure, we will need to lay out several definitions and properties of conditional probabilities:

Definition 4.5.1.1 (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 the expectation with respect to X. That is,

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

Definition 4.5.1.2 (Marginal variance)**.** *The marginal variance of X is:*

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

Furthermore, the definition of marginal variance can be extended to the conditional variance.

Definition 4.5.1.3 (Conditional variance)**.** *The conditional variance of X, given Y, is:*

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

We also note that the conditional variance is a random variable with respect to Y, and so we can find the expectation of the quantity:

$$E_Y[\text{var}_X(X \mid Y)] = E_Y E_X[X^2 \mid Y] - E_Y[E_X(X \mid Y)]^2$$

By definition of the marginal expectation (definition 4.5.1.1), we can show the following:

$$E_Y E_X[X \mid Y] = \sum_{y \in Y} P(Y = y)\, E_X[X \mid Y = y] = E[X] \tag{4.2}$$

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

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

$$\text{var}_Y(E_X[X \mid Y]) = E_Y\left(E_X[X \mid Y]\right)^2 - \left(E_Y E_X[X \mid Y]\right)^2 \tag{4.4}$$

$$= E_Y\left(E_X[X \mid Y]\right)^2 - \left(E[X]\right)^2 \tag{4.5}$$

where we used the result from equation 4.2. Then, rearranging equation 4.5 and combining with equation 4.3, we get:

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

where we used the definition of marginal variance in definition 4.5.1.2.

Thus, as discussed in [7], the variance of the random variable X is made of two parts:

- $\text{var}_Y(E_X[X \mid Y])$ describes the variability between the targets (or outcomes): how much do the targets change with the average of the feature values?
- $E_Y[\text{var}_X(X \mid Y)]$ describes the average variability with respect to different targets: on average, how much do the feature values vary with different targets?

The latter quantity, relative to the variance of the feature, is used in measuring the relevance of a feature and a target class in MRmMC in equation 4.1.

4.5.2 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 4.5.2.1. *Two random variables X and Y are independent iff*

$$P(X, Y) = P(X)\, P(Y) \tag{4.6}$$

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

Looking at the terms in r_{qn},

$$\begin{aligned}
E_Y[\text{var}_X(X|Y)] &= E_Y[E_X(X^2|Y) - (E_X(X|Y))^2] \\
&= E_Y E_X(X^2|Y) - E_Y\left(E_X(X|Y)\right)^2 \\
&= E[X^2] - E_Y\left(E_X(X|Y)\right)^2 \quad \text{(by eqn 4.2)} \\
&= E[X^2] - \sum_{y\in Y} P(Y=y)\left[E_X(X|Y=y)\right]^2 \quad \text{(by defn 4.5.1.1)} \\
&= E[X^2] - \sum_{y\in Y} P(Y=y)\left[\sum_{x\in X} x\, P(X=x|Y=y)\right]^2 \\
&= E[X^2] - \sum_{y\in Y} P(Y=y)\left[\sum_{x\in X} x\, \frac{P(X=x,Y=y)}{P(Y=y)}\right]^2
\end{aligned}$$

Assuming X and Y are independent, we can apply equation 4.6:

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

So, $r_{qn}(X,Y)$ attains its lower bound (i.e. $r_{qn}(X,Y)=0$) 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 [7]. In the limit, suppose X is a function of Y: $Y = f(x)$. Then, given $Y = y$, we can find the value of $X = x$, since we have $x = f^{-1}(y)$, implying that we can deterministically find the value of x given y. So,

$$\begin{aligned}
E_Y[\text{var}_X(X|Y)] &= E_Y[E_X(X^2|Y) - (E_X(X|Y))^2] \\
&= E_Y\left[E_X\left[\left(f^{-1}(Y)\right)^2\right] - \left(E_X\left[f^{-1}(Y)\right]\right)^2\right] \\
&= E_Y\left[\left(f^{-1}(Y)\right)^2 - \left(f^{-1}(Y)\right)^2\right] \\
&= 0 \\
\Rightarrow r_{qn}(X,Y) &= 1
\end{aligned}$$

As such, MRmMC aims to find features in the feature set F, that are strongly relevant (correlated) to the target class, by *maximising* r_{qn} between each feature, f, and the target

class labels, c. That is,

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

4.5.3 Computing relevance

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

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

For our data in particular, where binary target classes are used (i.e. $Y = \{0, 1\}$), we can expand the above:

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

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

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

To compute the conditional expectation $E_X(X|Y = y)$, we simply compute the expectation of those realisations of X for which the 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,

$$\begin{aligned} E_X(X|Y = 1) &= \frac{1}{2} (0.5 + 0.1) = 0.3 \\ P(Y = 1) &= \frac{2}{3} \\ E[X^2] &= \frac{1}{3} (0.5^2 + 0.2^2 + 0.1^2) = 0.1 \\ E[X] &= \frac{1}{3} (0.5 + 0.2 + 0.1) \approx 0.2666 \\ \text{var}(X) &= E[X^2] - (E[X])^2 \approx 0.07111 \end{aligned}$$

As such, we can employ the same computation method to compute the necessary expectations and variance for our data. To make this task simpler, we can utilise methods in the `numpy` [100] package to help us compute expectation and variance using the `mean` and `var` methods respectively.

4.5.4 Redundancy and multicollinearity

Now, we investigate how MRmMC seeks to minimise redundancy within the set of selected features. *Multicollinearity*, in the context of feature selection, refers to a strong correlation between features [113].

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 [114]. 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 redundant.

Another useful property of R^2 is that, if a group of independent variables, say $Q = \{q_1, q_2, \ldots, q_k\}$, is pairwise orthogonal, then the R^2 term between a dependent variable, say f, and these independent variables is the sum of R^2 between each independent variable and the dependent variable [114]. That is,

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

In [7], 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_k. Then, for any orthogonal component $q \in Q$,

$$R^2(f_k, q) = \frac{\left(\sum_{i=1}^{n} f_i q_i\right)^2}{\left(\sum_{i=1}^{n} f_i^2\right)\left(\sum_{i=1}^{n} q_i^2\right)} \tag{4.7}$$

Since the components of Q are pairwise orthogonal, the squared multiple correlation coefficient of the new vector f_k and all the components of Q would be the sum of the R^2 terms:

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

Furthermore, the R^2 term can be interpreted as the square of the *Pearson correlation coefficient* 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 4.7 when the means of f_k and q are zero. Thus, we need to ensure that f_k and q have zero mean so that the above interpretation works.

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

4.5.5 Computing redundancy

The only challenging aspect of computing R^2 is the orthogonalisation of the features. The MRmMC paper [7] 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:

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

which is evidently the *classical Gram-Schmidt* (CGS) process. However, CGS is notable for its numerical instability because there is usually a loss of orthogonality among the resulting q_k vectors [115]. Instead, other methods such as the *modified Gram-Schmidt* (MGS) are preferred. Although there are several algorithms in the literature of linear algebra, such as Householder transformations, that alleviate the problem of instability of CGS, this project will not delve further into these algorithms, but instead simply use MGS to compute the set of orthogonal vectors Q. Besides, [116] also notes that MGS is almost as numerically stable as algorithms that use Householder transformations.

The pseudocode in algorithm 2, from [115], outlines the procedure for MGS. We also note that MGS produces an upper triangular matrix $\boldsymbol{R}$ that does not play a role in computing R^2. Furthermore, the original matrix A in algorithm 2 is overwritten by *orthonormal* columns, which are orthogonal columns with the added constraint that they have a magnitude (2-norm) of 1. This would not affect how R^2 is used as described above.

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.

Algorithm 2: Modified Gram-Schmidt Orthogonalisation(A)

Input: Matrix $A \in \mathbb{R}^{m \times n}$, with columns A_k where $k \in [1, n]$.
Output: Matrix $A \in \mathbb{R}^{m \times n}$ with orthonormal columns.

begin
  for $k = 1 \longrightarrow n$ **do**
    $R(k, k) = \|A_k\|_2$
    $Q_k = {}^{A_k}/_{R(k,k)}$
    for $j = k + 1 \longrightarrow n$ **do**
      $R(k, j) = Q_k^\top A_j$
      $A_j = A_j - Q_k \times R(k, j)$

4.5.6 Experimental results

4.6 Integrating genetic algorithm with MRmMC

As mentioned in section 2.7, genetic algorithms (GA's) are a popular method in the literature of feature selection. The GA implemented in this project has two variants, in terms of the evaluation function of the GA (the function that determines the *fitness* of the chromosomes). First, we can evaluate fitness based on the classification scores of an SVM. Second, fitness can be quantified based on the relevance and redundancy measure of MRmMC. It will be interesting to see if either method will outperform the other.

4.6.1 Parameters and strategies

First, we will use the "two-stage" GA process, as illustrated in Figure 2.4, due to the large number of features that we need to deal with in the epigenetics data set. To stick to the MRmMC method as closely as possible, we can use the correlation coefficient r_{qn} to choose the best few features to pass to 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. In our experiment here, 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 genetic algorithm.

Then, from the top s features, we randomly select k features that will form one of the solutions in the first generation of the algorithm. The presence of a feature is encoded by a '1', while its absence is encoded by a '0'. So, assume $s = 2000$ and $k = 1000$. Only 1000, out of the 2000 features selected by the first stage, will form a solution. This is done until there are G solutions in the generation.

Next, the *elitism/cloning* strategy will be used when creating a new generation of chromosomes, as performed in, for example, [86]. By copying the best solutions in each generation to the next, we can guarantee that the maximum fitness of the next generation will not decrease (see section 2.7.4). The idea of elitism is enforced in our experiment as follows:

- Clone a fraction of 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 append to the next generation.
- Remove p_{elim} of the solutions with the worst fitness.

Crossover is then done on the remaining solutions that are not affected by the elitism strategy. This will be done based on the Roulette Wheel selection method. Two parents will be randomly selected, in proportion to their relative fitness. This helps to boost the overall fitness of the offsprings, as there is a higher chance that a relatively fit parent is selected. At the same time, it helps to generate diversity, as parents with lower fitness

also have a chance to be selected for crossover. This is done until there are G solutions for the next generation.

The literature also offers different stopping criteria for the GA. For example, [84] stops the GA after a fixed number of generations, while [86] stops the algorithm when there is no change in the fitness function in 30 generations. We experiment with using the latter, where the algorithm terminates if the fitness increases by less than t for T generations.

In summary, the parameters used in the algorithm implemented are summarised in Figure 4.10. The pseudocode for the implementation of our genetic algorithm is listed in Algorithm 3.

4.6.2 Experimental results - Roulette Wheel

First, the algorithm listed in Algorithm 3 was implemented. We first experimented with the parameters listed in the table in Figure 4.10. Since the algorithm contains probabilistic elements, such as the mutation operator, the algorithm was run 10 times, to see if there was any interesting observations. The best scores obtained from the 10 runs were:

Run	Best 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 best cross validation score is approximately 0.9569, slightly better than t-test in section 4.2. However, we note that in t-test, the best score of 0.950 was obtained with 15000 features, compared here to about 8000 features on average, which is a big improvement from t-test. Furthermore, as the elitism strategy was implemented, the number of iterations of the genetic algorithm is relatively low, as the best cross validation score for each generation is guaranteed to be non-decreasing.

Lastly, we also note that the final number of features chosen by the genetic algorithm is also roughly similar to the value k that we provide for the algorithm. It will be interesting to investigate if reducing k drastically will make a difference in the outcome. To this end, we set $k = 10, 50, 100$, while keeping the rest of the parameters the same as in Figure 4.10.

For $k = 100$:

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 ($S \subset D$) such that the features in S produce 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 ascending 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 $next\_gen\_solution \longleftarrow [\,]$
  - $e \longleftarrow$ Select top p_e of solutions in $gen\_solution$.
  - $r \longleftarrow$ Select next p_r of solutions in $gen\_solution$.
  - Perform *mutate* on each solution in r.
  - $next\_gen\_solution \longleftarrow next\_gen\_solution \cup \{r, e\}$
  - 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**
    - $consecutive\_gen \longleftarrow 0$
  - $best\_fitness \longleftarrow current\_best\_fitness$
  - $gen\_solution \longleftarrow next\_gen\_solution$
- Evaluate fitness of $gen\_solution$.
- $S \longleftarrow$ Features corresponding to the solution with $best\_fitness$.

Parameters	Value
Number of features selected in initial generation 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
Termination threshold for fitness t	0.001

Figure 4.10: Table of parameters used in genetic algorithm implementation of MRmMC.

Run	Best 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	Best 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 $k = 10$:

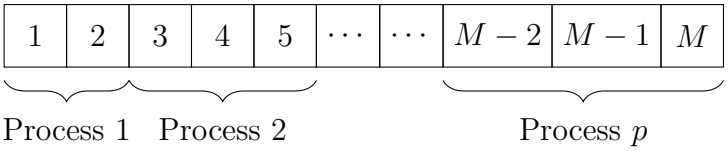


Figure 4.11: An illustration of how the work of evaluating M independent features can be distributed among p processes.

Run	Best 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

It is interesting to note that despite reducing k drastically, the magnitude of the final number of features selected by the algorithm still remains in the thousands, although we notice that, on average, for $k = 10$, the final number of features chosen is the lowest among the three experiments. The case of $k = 100$ has also achieved the highest CV score so far of 0.9610.

4.6.3 Experimental results - Tournament selection

4.7 Utilising parallelism

In [117], 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, for the t-test method described in section 4.2, the algorithm iterates through each of the features to compute the t-statistic and p-value. We note that the computation for one feature is independent of the computation of all of the other features. Thus, we can design our computation to take advantage of this parallelism. This concept is illustrated in Figure 4.11.

In particular, 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` method was written to divide the search space - in this case, all of the features - into partitions for the processes to work on. The `Pool.map` method can then `map` 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.

This idea can also be extended to the other methods, where the search space can be conveniently divided into independent segments. For example, in RFS, the algorithm (see algorithm 1) can be parallelised in both of the `for` loops.

Chapter 5

Conclusion

5.1 Further work

One serious limitation of the work presented in this project, is the fact that only a small percentage of the features in our reduced subset S is explored. The "optimal" solution generated from the feature selection algorithms presented here might be a local optimum.

If there were more time and resources, we could explore the possibility of including more features in the subset S, and find other local optima, if any.

5.1.1 Exploring parameters and strategies

As mentioned in section 2.7, there are many parameters in genetic algorithms (GA's) that need to be tuned, and it might be enlightening to experiment with parameters that are different from the ones used in this project. This might involve optimisation in many dimensions, each dimension representing a parameter. Furthermore, different strategies can be used in GA's, such as different terminating conditions, crossover strategies and selection strategies.

5.1.2 Use of mutual information

As mentioned in section 2.6.11, 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.

As our data set uses continuous variables, it is difficult to estimate probabilities involving these variables. The Parzen Window [73] seems to be a feasible way to estimate these probabilities. It will be interesting to use the Parzen Window method to help quantify relevance and redundancy. This can then be combined with the other methods and 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.

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. ID: TN_pubmed_central1392256.

[5] 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.

[6] 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. ID: TN_wj10.1111/j.1467-9868.2005.00503.x.

[7] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

[8] Heidi Ledford. If depression were cancer. *Nature*, (515):182–184, 2014. Available from `http://www.nature.com/news/medical-research-if-depression-were-cancer-1.16307`.

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

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

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

[12] 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`.

[13] University of Utah. Insights from identical twins. Available from `http://learn.genetics.utah.edu/content/epigenetics/twins/`.

[14] 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.

[15] Arturas Petronis. Epigenetics and twins: three variations on the theme. *Trends in Genetics*, 22(7):347–350, 2006. ID: TN_sciversesciencedirect_elsevierS0168-9525(06)00126-0.

[16] 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.

[17] 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.

[18] 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. ID: TN_medline17413848.

[19] 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. ID: Hannon2016.

[20] 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.

[21] 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. Compilation and indexing terms, Copyright 2016 Elsevier Inc.; T3: Proceedings of the 2010 International Conference on Signal and Image Processing, ICSIP 2010.

[22] 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. Compilation and indexing terms, Copyright 2016 Elsevier Inc.

[23] Qihua Tan, Mads Thomassen, Kirsten M. Jochumsen, Jing Hua Zhao, Kaare Christensen, and Torben A. Kruse. Evolutionary algorithm for feature subset selection in

predicting tumor outcomes using microarray data. In *4th International Symposium on Bioinformatics Research and Applications, ISBRA 2008, May 6, 2008 - May 9*, volume 4983 LNBI, pages 426–433. Springer Verlag, 2008.

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

[25] American Psychiatric Association. Diagnostic and statistical manual of mental disorders, 2013. ID: 44IMP_ALMA_DS5172131690001591.

[26] Tom M. Mitchell. *Decision Tree Learning*, pages 52–78. Machine learning. McGraw-Hill Education, international 1997 edition, 1997. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2143719110001591.

[27] J.R. Quinlan. Induction of decision trees, 1986. ID: RS_60168743381inductionofdecisiontrees.

[28] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ID: TN_springer_jour1010933404324.

[29] 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. Compilation and indexing terms, Copyright 2016 Elsevier Inc.; T3: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[30] 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.

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

[32] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, pages 267–288, 1996.

[33] C. De Mol, E. De Vito, and L. Rosasco. Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–30, 04 2009.

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

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

[36] Edwin K. P. Chong and Stainlaw H. Zak. *Duality.* An introduction to optimization. Wiley, Hoboken, New Jersey, 4th; fourth edition, 2013. Includes bibliographical references and index.; ID: dedupmrg214935921.

[37] 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. Includes bibliographical references (p. 173-186) and index.; ID: 44IMP_ALMA_DS2146175590001591.

[38] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006. Includes Bibliography: p. 711-728 and index; ID: 44IMP_ALMA_DS2144511690001591.

[39] 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.

[40] 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`.

[41] 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.

[42] 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. ID: TN_springer_series978-3-319-21858-8.

[43] 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. T3: Proceedings of the 2009 Second International Conference on Machine Vision (ICMV 2009);.

[44] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

[45] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

[46] 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.

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

[48] Richard O. Duda. *Problems of Dimensionality*. Pattern classification. Wiley, New York ; Chichester, 2nd / edition, 2001. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2141359100001591.

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

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

[51] Yvan Saeys, Iñ Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007. ID: TN_oxford10.1093/bioinformatics/btm344.

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

[53] Z. M. Hira and D. F. Gillies. A review of feature selection and feature extraction methods applied on microarray data, 2015. ID: 44IMP_DSP_DS10044/1/25192.

[54] 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.

[55] 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.

[56] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

[57] C. Huertas and R. Juarez-Ramirez. Filter feature selection performance comparison in high-dimensional data: a theoretical and empirical analysis of most popular algorithms. *17th International Conference on Information Fusion (FUSION)*, page 8, 2014.

[58] 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.

[59] 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.

[60] 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. LR: 20091104; GR: CA104620/CA/NCI NIH HHS/United States; JID: 9808944; ppublish.

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

[62] 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.

[63] Isabelle Guyon and Andre Elisseeff. *Introduction*, pages 1–25. Feature extraction: foundations and applications. Springer, 2006. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2142486550001591.

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

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

[66] 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. T3: Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. IIH-MSP 2009.

[67] 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. T3: 2016 Seventh International Conference on Intelligent Control and Information Processing (ICICIP). Proceedings.

[68] 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. T3: Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics. Proceedings of the 10th European Conference, EvoBIO 2012.

[69] 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.

[70] 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.

[71] Kari Torkkola. *Information-Theoretic Methods*, pages 168–185. Feature extraction: foundations and applications. Springer, Berlin, 2006. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2142486550001591.

[72] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

[73] 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.

[74] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.; T3: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

[75] 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. T3: 2014 IEEE 34th International Scientific Conference on Electronics and Nanotechnology (ELNANO).

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

[77] Tom M. Mitchell. *Genetic Algorithms*, pages 249–270. Machine learning. McGraw-Hill Education, international 1997 edition, 1997. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2143719110001591.

[78] 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.

[79] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, 3rd rev. and extended edition, 1996. Includes bibliographical references (p. 363]-382) and index.; ID: 44IMP_ALMA_DS2148128000001591.

[80] Juha Reunanen. *Search strategies*, pages 120–136. Feature extraction: foundations and applications. Springer, 2006. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2142486550001591.

[81] 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. T3: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD).

[82] 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. T3: 2015 International Conference on Computing, Communication Automation (ICCCA.

[83] 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. T3: Applications of Evolutionary Computation. 20th European Conference, EvoApplications 2017. Proceedings: LNCS 10199.

[84] 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. T3: 2016 IEEE Congress on Evolutionary Computation (CEC).

[85] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

[86] 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. T3: 2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI).

[87] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.; T3: Proceedings of the International Conference on Information Technology Interfaces, ITI.

[88] 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. T3: 2014 7th International Symposium on Computational Intelligence and Design (ISCID).

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

[90] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

[91] 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. T3: 2014 9th International Conference on Industrial and Information Systems (ICIIS).

[92] 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. T3: 2014 11th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP).

[93] 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. T3: Bioinformatics Research and Applications.Fourth International Symposium, ISBRA 2008.

[94] 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. T3: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406).

[95] G. Pavai and T. V. Geetha. A survey on crossover operators. *ACM Computing Surveys*, 49(4), 2016. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

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

[97] 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.

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

[99] 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.

[100] 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.

[101] 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. T3: Evolutionary Computing. AISB Workshop. Selected Papers.

[102] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.; T3: Proceedings of the World Congress on Engineering 2011, WCE 2011.

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

[104] 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, page 6 pp., 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.

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

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

[107] 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.

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

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

[110] Douglas C. Montgomery. *Engineering statistics*. Wiley ; John Wiley distributor], Hoboken, N.J. : Chichester, 4th edition, 2007. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2147792160001591.

[111] 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.

[112] 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.

[113] 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. Compilation and indexing terms, Copyright 2017 Elsevier Inc.

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

[115] Gene H. Golub. *Matrix computations*. Fourth edition, 2013. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2148558440001591.

[116] V. P. Sreedharan. A note on the modified gram-schmidt process. *International Journal of Computer Mathematics*, 24(3-4):277–90, 1988.

[117] Maurice Herlihy. *The art of multiprocessor programming*. Elsevier / Morgan Kaufmann, Amsterdam ; London, revised first edition, 2012. Includes bibliographical references and index.; ID: 44IMP_ALMA_DS2180263830001591.