Using Machine Learning to Investigate Anti-Discrimination Law in Credit Scoring

A study on variable importance

Submitted May 2015, in partial fulfilment of the conditions
of the award of the degree in Computer Science in Artificial Intelligence

William Heng Yijin
wyh04u
School of Computer Science and Information Technology
University of Nottingham

I hereby declare that this dissertation is all my own work except as indicated in the text:

Signature _____

Date _____/_____/_____

Abstract

Equal Credit Opportunity Act was signed in place as a preventative measure to ensure that credit scoring agencies do not take into account an individual's protected characteristics when calculating credit score. With new startups such as the likes of Wonga and Lenddo who use an individual's online social media presence to calculate their credit score, it is in the interest of this project to "peek into" the kind of models that will be trained based on a particular dataset. Thus, it is essential to look at which variables matter more and which matter less i.e. to study a dataset's variable importance. Machine learning techniques, specifically random forest and artificial neural network are used to study variable importance measures in this dissertation. A German credit dataset is used in this project to assess the influence of an individual's personal status on their credit score. The two variable importance heuristics provided by random forest (Mean Decrease in Accuracy and Mean Decrease in Gini) are compared against that of artificial neural network's two heuristics (Garson's algorithm and Connection Weights approach). Experiments have also been done to use and judge random forest as a feature selection technique. Variables that are ranked lowly in terms of variable importance are eliminated during the construction of the artificial neural network model.

Keywords: Equal Credit Opportunity Act, variable importance, random forest, artificial neural network, multilayer perceptron, mean decrease in accuracy, mean decrease in gini, Garson, connection weights, feature selection, credit scoring, German credit

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Rong Qu for her full support and guidance throughout the course of this project. Secondly, I would like to extend my gratitude to Dr. Ender Özcan for his occasional feedback during the presentation sessions.

I would also like to thank the open source community for their excellent work on maintaining and keeping the R programming language platform well and alive. To the maintainers of the libraries used in this project, namely `nnet`, `randomForest`, `caret`, `devtools`, `e1071`, and `NeuralNetTools`, thank you very much for the effort in creating and maintaining such wonderful tools for the research community.

Lastly, this project wouldn't be completed if it wasn't for the moral support from my dear friends and family. Thank you all very much.

Abbreviations

RF – Random Forest

ANN – Artificial Neural Network

MLP – Multilayer Perceptron

CART – Classification and Regression Tree

MDA – Mean Decrease in Accuracy

MDG – Mean Decrease in Gini

Contents

List of Figures

List of Tables

1. Introduction

Credit scoring is a statistical method employed by financial institutions to evaluate credit risk of loan applications. Credit score is a numerical expression of an individual's likeliness to default or become delinquent on one's loan [1]. An individual with a good credit score is likely to get his/her loan approved easily whereas one with a bad credit score might face obstacles while applying for a loan (having to present more documents than is usually required or having to get a guarantor), if he/she even gets to proceed to prove his/her creditworthiness at all.

Through loans made to borrowers previously, credit rating agencies typically construct a credit-scoring model or a "scorecard" which takes into account historical data of common borrower characteristics that are useful in predicting a loan's performance. A credit score is thus produced to rank loan applicants in terms of risk [1]. A good credit-scoring model typically give high scores to borrowers whose loans are predicted to perform well and low scores to applicants who are predicted to be more at risk to default. Machine learning techniques have been studied extensively to classify good and bad credit loans to assist credit rating agencies [2-4]. Machine learning classifiers are trained solely for the purpose of weeding out customers who are able to repay their loans from the many credit loan applications.

Anti-discrimination laws are enacted to give everyone an equal opportunity to access goods and services including credit scoring to secure a financial loan. In the United States, the Equal Credit Opportunity Act has been enacted in 1974 to prohibit creditors from discriminating against any applicant with respect to any transactions on the basis of race, colour, religion, national origin, sex, marital status, age, sexual orientation, or public assistance from the government. These characteristics are termed the protected characteristics of an individual. Similar laws are also enacted in other parts of the world such as the United Kingdom's Equality Act 2010 and Germany's General Act on Equal Treatment 2006. An individual's protected characteristics cannot play a role in determining an individual's rights to access goods and services.

1.1 Credit Scoring Background Information

1.1.1 Credit Scoring

Credit scoring is a method of evaluating credit risk of loan applications. Based on large historical samples of previous customers' credit history, credit scoring methods identify the connections among the consumers' characteristics and how their loans performed. By analysing the historical data on the performance of previously made loans, credit bureaus can build a scoring model that is useful in predicting whether a loan would perform well. A score is produced through the use of this method to rank the loan applicants in terms of risk.

Lenders utilise credit scores to aid them in granting consumer credit. It helps to assess risk in lending to a particular customer. It gives lenders a metric to judge who will get credit and how much credit can a loan applicant get. Typically, the higher a customer's credit score, the easier it is for the customer to get a loan and the higher the amount of credit available.

1.1.2 Anti-Discrimination Law

Signed in 1974 in the U.S., Equal Credit Opportunity Act aims to "...make credit available with fairness, impartiality, and without discrimination on the basis of sex..." The method for granting credit is defined in the Act such that "...a creditor [shall not] take sex...into account

in...connection with the evaluation of credit-worthiness of any applicant." [5] A credit applicant's personal attributes, or what is commonly termed as an individual's protected characteristics, cannot be included as a basis to calculate said individual's credit-worthiness. Similarly, the UK's Equality Act 2010 outlaws discrimination in credit granting on the grounds of race and sex. An individual's protected characteristics include but are not limited to:

1. Age,
2. Disability,
3. Gender reassignment,
4. Marriage and civil partnership,
5. Pregnancy and maternity,
6. Race,
7. Religion or belief,
8. Sex and,
9. Sexual orientation.

The list above is extracted from UK's Equality Act 2010 [6]. Depending on the specifications of the different laws of different countries, the list of protected characteristics can change. Also, depending on the circumstances, some protected characteristics may be included as a factor in the provisions of goods and services. An example may be that in European Union's Directive 2004/113/EC, insurers could use gender as a determining factor when assessing insurance risk as statistically, men are involved in a higher number of serious traffic accidents than women [7].

1.2 Problem

Credit scoring agencies are bound to comply with the anti-discrimination laws to not include protected characteristics within their credit-scoring models. This study intends to use machine learning methods to explore ways to "peek into" credit-scoring models that will be constructed based on certain datasets. This can be achieved by ranking how important each variable is in a dataset. This method can assist researchers to find out the sort of credit scoring models that will be built based on the datasets used in order to confirm that an applicant's protected characteristics are not essential to their credit score.

1.3 Motivation

With the recent spur in newly founded "start-up" companies trying to innovate conventional industries, credit scoring has found itself being "disrupted" [8]. Digital finance companies such as Lenddo and Wonga – they mostly focus on short-term loans - have made news by taking an unconventional method of determining an applicant's creditworthiness through said applicant's social media account information [8]. This method of incorporating an applicant's social media accounts to calculate one's creditworthiness, although creative, is nevertheless controversial. An individual's information such as his/her gender, religious views, political views, nationality, and et cetera can easily be found online. Irresponsible institutions scrutinise the public's digital footprints and try to rank people by crunching the information people put online. The question is: should the use of social media in the calculation of an individual's credit score be forbidden? Also, how can one know if an individual's protected characteristics were used in the calculation of said individual's credit score? This project is focused on studying variable importance in a dataset and judge the reliability of the different data mining techniques to discover whether the calculation of an individual's credit score involved certain protected characteristics.

1.4 Aims and Objectives

This study aims to explore methods to see if the personal status variable carries any influence to calculate an individual's credit score in the dataset used in this dissertation. Will one's credit score be severely affected if his/her protected characteristics are included in the calculation? Not only would this study further strengthens the point of the need of anti-discrimination enactments, it would also serve as a precaution to the public on the use of online lenders who engage in controversial methods to determine an individual's credit score. The objectives of this project would be to:

- Study the extent on the applicability of the anti-discrimination enactments on credit scoring and understand the motivation behind the regulations.
- Analyse the trend within the dataset if all dependent variables including protected characteristics are included in the credit scoring computation.
- Study the credit scoring result if all protected characteristics are excluded from the credit score's computation.
- Compare the accuracy of random forest and artificial neural network in the context of credit scoring.
- Experiment and compare the accuracy and precision of variable importance measures of different techniques.
- Use random forest as a feature selection technique and discuss the results of the experiment.
- Discuss and compare the different results obtained.

It is not in the interest of this dissertation to train a better or more accurate credit scoring binary classifier. The focus of this project would be on studying the many different variable importance measures of different techniques.

2. Literature Review

2.1 Random Forest (RF)

2.1.1 Introduction

Random forest (RF) is an ensemble learning algorithm - methods that generate many classifiers and aggregate their results - for classification and regression which are based on simple decision trees. It was introduced by Breiman [9] who is influenced heavily by Amit and Geman [10].In Geman et al. [10] on written character recognition, the authors define a set of geometric features and search over a random selection of said features for the best split at each node in the tree [10]. Breiman later combined bootstrap aggregating (bagging) [11] together with the random split and conceived the RF method.

RF essentially constructs a forest of decision trees (classification or regression) where each tree in the forest is grown on an independent bootstrap sample from the training data and each node of the tree is split using the best split among a subset of predictors randomly chosen at that node [12]. The bootstrap sample is produced through a random selection of *a subset* of features in the data set. Note that this is different from standard decision tress where each node is split using the best split among *all* variables of the dataset.

2.1.2 Classification and Regression Tree

Classification and Regression Tree (CART) analysis is a supervised learning technique that is based on decision trees. A decision tree is a hierarchical model where the local region is identified by splitting recursively through decision nodes with a test function. As opposed to decision tree that has the ability to perform a multi-split, CART analysis only grows tree with binary splits at every node of the tree.

CART is inherently a non-parametric model. In simpler terms, it makes no assumptions regarding the underlying model of the predictor variables [13]. It uses the Gini diversity index to split at a node (CART's test function). Gini diversity index is selected over information gain because Gini can be extended to include symmetrised costs and it can be computed more quickly than information gain [14]. The Gini diversity index indicates the partition purity of a dataset [14].

The Gini diversity index for a dataset S:

$$Gini(S) = 1 - \sum_{i=1}^{c} {p_i}^2 \quad (1)$$

where

- c is the number of classes,
- C_i represents classes for $i = 1, \dots, c$,
- s_i is the number of instances belonging to class C_i, and
- $p_i = s_i/S$ is a relative frequency of class C_i in the set.

$Gini_{split}$ is then calculated for all possible features. The feature with the lowest $Gini_{split}$ value is selected as the split point. The quality of a split on a feature into k subsets S_i is then computed as the weighted sum of the Gini indices of the resulting subsets:

$$Gini_{split} = \sum_{i=0}^{k-1} n_i \,/\, n \; Gini(S_i) \tag{2}$$

where

- n_i is the number of instances in subset S_i after splitting, and
- n is the total number of instances in the given node.

CART is robust to outliers and noisy data. It can also handle missing data and categorical variables effectively [15]. Besides that, the structure of the tree is invariant with respect to monotone transformations of the features. This means that an analyst can replace any feature with its logarithmic or square root value and the structure of the tree will still not change [14].

CART is not without its disadvantages. It may have unstable decision trees where insignificant modifications of learning samples such as eliminating several instances could lead to radical changes in a decision tree [14].

2.1.3 Ensemble Learning

Ensemble learning methods generate multiple classifiers and aggregate their results. Two well-known ensemble learning methods are the boosting and bagging of classification trees [11, 16]. In the boosting method, successive trees give extra weight to data instances that are incorrectly predicted while instances that are classified correctly lose weight by earlier predictors. Future trees that are grown will thus focus more on the instances that previous weak learners misclassified. At the end, a weighted vote is taken for prediction.

The bagging method tries to build a prediction model by combining the strengths of a collection of simpler base models. Bagging decreases error by decreasing the variance of unstable and noisy learners [14, 17]. Bagging in particular is important to RF as RF is essentially bagging with element of randomness introduced into it. In the bagging of RF, successive trees generated do not depend on previous trees. Each tree is constructed independently using a bootstrap sample of the data set. The mode (majority) is taken as a feature for prediction at the very end [17].

Bagging is only effective when using unstable nonlinear models where small changes in training data lead to significantly different classifiers and large changes in accuracy, which is the case that applies when using CART.

2.1.4 Definition

The definition of a RF from Breiman [9]:

A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, \ldots\}$ *where the* Θ_k *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* x.

Breiman added an additional layer of randomness to bagging. Apart from constructing each tree using a different bootstrap sample of the data, random forests change how the classification or regression trees are produced. Compared to standard trees where each node is split using the best split among all variables, random forest constructs trees by using the best among a subset of features randomly chosen at each node [12].

Random forest turns out to perform very well compared to other classifiers, including support vector machines and neural networks [18], and is robust against overfitting [9].

2.1.5 Algorithm

RF is very easy to use as it only has two parameters to tune: the number of variables in the random subset at each node and the number of trees in the forest.

Adapted from Liaw [12]:

1. Draw n_{tree} bootstrap samples from the original data set.
2. For each of the bootstrap samples, grow an unpruned classification or regression tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample m_{try} of the predictors and choose the best split from among those variables.
3. Predict new data by aggregating the predictions of the n_{tree} trees (i.e., majority votes for classification, average for regression).

There is an official implementation of the algorithm in the FORTRAN programming language by Breiman and Cutler which is published under the GNU General Public License [12].

2.1.6 Application of Random Forest in Credit Scoring

Agreeing with the opinion of Brown et al. [19], RF has yet to be thoroughly researched in the context of credit scoring and only limited work can be found. Fortunately, Breiman used the German credit dataset in his renowned paper to introduce RF [9]. He showed that the RF model trained can achieve a classification accuracy rate of 77.2%. However, the dataset used in Breiman's [9] is the numerical version of the dataset instead of the version with both numerical and categorical data that is used in this report.

In Wang [20], RF was used on the same dataset as well to show that RF can be a sensible choice in the application for credit scoring albeit using Weka (a Java-based machine learning tool). Wang successfully trained a RF model to achieve an average classification accuracy of 77.1% [20]. The accuracy of the RF model trained in Wang's [20] is comparable to that of Breiman's [9].

In conclusion, it is in this work's interest to see more research done using RF in the context of credit scoring.

2.2 Artificial Neural Network (ANN)

2.1.1 Introduction

Artificial neural network (ANN) is a machine learning method that is inspired by an animal's brain. The brain is a highly complex, non-linear, and parallel information processing system and ANN is created to try to model the brain's computational process [14].

The first computational model for ANN was introduced by McCulloch and Pitts in 1943 [21]. Rosenblatt later introduced his work on single-layer perceptron [22] for learning on pattern-classification in the late 1950s. ANN has been studied for more than six decades since then.

ANN is an abstract computational model of the brain. The brain has many tiny units called neurons and these neurons are interconnected with links. By mimicking the model of a brain,

ANN is composed of abstract models of biological neurons and interconnections. ANN is a network structure with a number of nodes connected through directional links where each node represents a processing unit i.e. artificial neuron, and the links determine the causal relationship between the connected nodes. One can view such a network as a graph, where the artificial neurons represent the nodes and the interconnections as the edges.

Due to ANN's strength in processing non-linear relationships [23] and its low-sensitivity to noise [24], ANN has proven to be a popular algorithm in machine learning and data mining. There are varying types of ANNs such as the Multilayer Perceptron, the Radial Basis Function Network, Kohonen self-organising network and Recurrent Neural Network.

The following sections provide a brief overview starting from a simple model of an artificial neuron, gradually evolving to the more complicated single-layer perceptron and later, a multilayer perceptron. Network architecture shall be discussed in a succinct manner, followed by the learning process of ANNs and the optimal ANN structure. The final section provides a review of existing studies that employ ANN in credit scoring.

2.2.2 Definition

There are multiple definitions of ANN and here is just one of the many definitions available:

> *An ANN is a massive parallel distributed processor made up of simple processing units. It has the ability to learn experiential knowledge expressed through inter-unit connection strengths, and can make such knowledge available for use. [14]*

This project regards an ANN to consist of an input layer, an *n*-number of hidden layer, and an output layer. Each layer consists of an arbitrary number of nodes.

An artificial neuron is the most basic computational unit that is essential to the operation of an ANN. It is modelled according to an animal's biological neuron. A basic model of an artificial neuron can generally be identified with three basic elements [14, 25]:

1. A set of connecting links from different inputs, each of which is characterised by a weight or strength. The weights of artificial neurons generally lie in a range that includes negative as well as positive values.
2. A transfer function for combining the weighted input signals represented by the respective synaptic strengths, typically a summation function which constitutes a linear combiner.

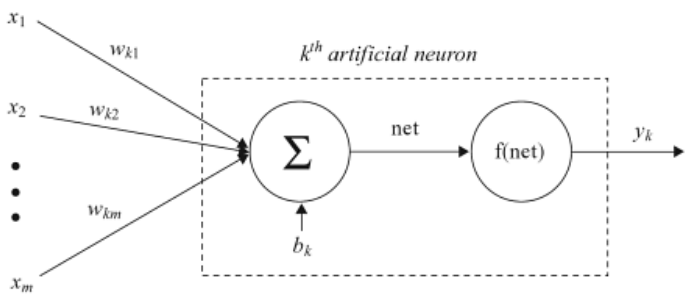

Figure 2.1: An artificial neuron. Adapted from Kantardzic [14]

An externally applied bias may be present, which has an effect of increasing or lowering the net input of the activation function depending on the bias's value.

To describe the neural model in Figure 2.1 in mathematical terms, we may write the pair of equations as below.

$$u_k = \sum_{i=1}^{n} w_{ki}\, x_i \tag{3}$$

and

$$y_k = \varphi(u_k + b_k) \tag{4}$$

where $x_1, x_2, \ldots, x_i$ are the input signals; $w_{k1}, w_{k2}, \ldots, w_{ki}$ are the respective weights of the neuron k; u_k is the sum of the weighted inputs; b_k is the bias; $\varphi(.)$ is the activation function; and y_k is the output signal of the neuron [25].

The bias b_k is an external parameter of neuron k. Its presence can be accounted for by considering it as yet another weight, w_{k0} with a synapse input of +1. The mathematical equations (3) and (4) can now be formulated as follows:

$$v_k = \sum_{i=0}^{n} w_{ki}\, x_i \tag{5}$$

and

$$y_k = \varphi(v_k) \tag{6}$$

2.2.3 Architecture of ANNs

The architecture of ANN is defined by the attributes of its nodes and the nodes' connectivity in the network. ANNs are generally classified into two categories: feedforward and recurrent.

In a feedforward network, data propagates from the input layer to the output layer unanimously without any loops or feedbacks in between. A layered representation of the feedforward ANN is generally used and there are no links between the nodes in the same layer. Outputs of nodes in one specific layer are always connected as inputs to nodes in succeeding layers. However, if there is a feedback link that forms a circular path then it is a recurrent network. The two different network architectures are shown in Figure 2.2.

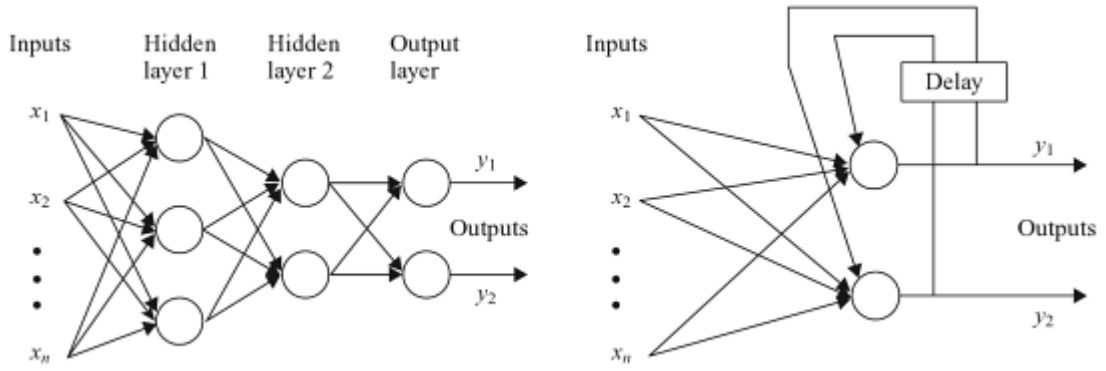

Figure 2.2: Feedforward network is shown on the left and a recurrent network is shown on the right. Obtained from Kantardzic [14].

Multilayer feedforward network with backpropagation-learning mechanism is the most widely used network architecture in terms of applications [14, 26]. The choice of a multi-layered ANN is made popular due to its ability to solve linearly-inseparable problems. A typical example of a linearly-inseparable function would be the famous XOR function where a zero-hidden layer network cannot draw a single linear separation of points that belong to different classes as shown in Figure 2.3. Minsky and Papert in their book *Perceptrons* showed that it was impossible for a simple perceptron i.e. zero-hidden layer network to learn the XOR function [27]. However, with the addition of hidden layers in between the input and the output layer, multi-layered feedforward network can now learn to solve non-linear problems. As in most real world cases where problems are not made idealistic and models are highly non-linear, multi-layered networks work better generally [14].

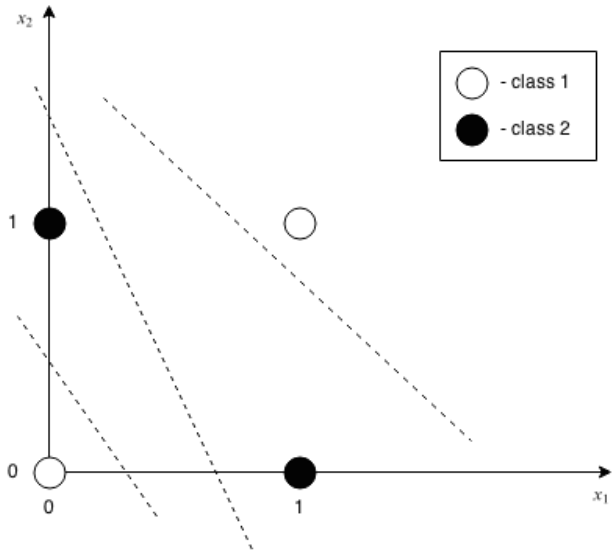

Figure 2.3: An XOR function that is linearly-inseparable.

2.2.4 Multilayer Perceptron (MLP) and Error-Backpropagation

Multilayer perceptron (MLP) is a form of feedforward ANN model. MLPs typically consist of a set of inputs which make up the input layer, one or more hidden layers, and an output layer. MLPs have three distinctive characteristics [14]:

1. Model of each neuron in the network is usually a non-linear activation function e.g. sigmoidal or hyperbolic

2. They contain one or more hidden layers made up of hidden neurons that are not a part of the input or output of the network. This enables MLPs to learn complex and non-linear tasks.
3. They exhibit a high degree of connectivity from one layer to the next one.

Data flow through the network in a forward direction, from left to right on a layer-by-layer basis. Generally, MLPs are trained in a supervised manner using the error-backpropagation algorithm. Error-backpropagation algorithm consists of two phases: forward pass and backward pass. During the training stage, instances are presented and passed through the network in a *forward* direction and if the network computes an output that matches the target, nothing is done. If there is an error (i.e. a difference between the output and the target), then the weights in the network are adjusted to reduce this error in a *backward* manner from the hidden-output weights, then adjusting weights layer-by-layer till the input-hidden weights. The essence of error backpropagation as worded by Russell et al. [24] is "to assess the blame for an error and divide it among the contributing weights."

2.2.5 Application of ANN in Credit Scoring

Unlike RF, ANNs have long been used in the field of credit scoring, be it academic or commercial. Nanni et al. [28] successfully trained a MLP with an average accuracy of 75.2% on the German credit dataset. In the case of Tsai et al. [26], the best result achieved is an average accuracy of 79.0%. Tsai managed to get this result using 200 epochs and 32 hidden nodes [26]. In West [29], a 73.3% average accuracy was achieved using the numerical version of the dataset. Although both Nanni [28] and Tsai [26] used the mixed version of the German credit dataset (containing both categorical and numerical data), the papers did not specify how pre-processing was done, specifically the encoding method that was taken to transform the categorical variables to a numerical form.

The reader is reminded that it is not the aim of this project to train a model that is better in terms of average classification accuracy. The findings and average classification accuracy rate found in other papers will be used as a benchmark so that an ANN model can be trained to achieve an acceptable performance so as to "extract" variable importance measure from the black-box model of ANN.

2.3 Variable Importance

It is important to define what variable importance refers to in this work. Terms such as "importance", "sensitivity", and "saliency" do not have precise and well-established meanings [30]. According to Sarle [30], there are two different contexts of importance. Predictive importance is concerned with the increase in generalisation error when an input is omitted from a model. Causal importance is concerned with situations where one can manipulate the value of the input in order to know how much the output will change. In the context and the scope of this project, the interest lies in knowing the predictive importance of variables i.e. how does including or excluding certain variables change the overall accuracy of prediction of a data mining model.

2.3.1 Variable Importance in RF

Variable importance measure is a by-product of the RF technique [31]. A random forest implementation by Liaw [12] is discussed whereby the algorithm estimates the importance of a variable by looking at how much prediction error increases when OOB data for that variable is permuted while all others are left unchanged. Liaw [12] implemented two

algorithms for calculating variable importance measures in the `randomForest` R package. The first heuristic is based on the Gini criterion. At each split, the decrease in the Gini node impurity is recorded for the variable x_j that was used to form the split. The average of all decreases in the Gini impurity in the forest where x_j forms the split yields the Gini variable importance measure. One thing to note is that Gini importance measure tends to favour variables that are continuous or categorical variable with a large number of levels [32].

The second heuristic calculates variable importance as mean decrease in accuracy using the out-of-bag (OOB) observations. Each tree is grown from a bootstrapped sample S_b taken from the original learning sample S, and about one-third of the instances in the dataset will not be used to grow the tree. These observations are the OOB observations for that tree. The OOB observations can be used to calculate variable importance as defined in Archer et al. [31]:

For bootstrap samples b = 1, …, B:

1. Identify the OOB observations, $S_{b,oob} = S \backslash S_b$;
2. Predict class membership for $S_{b,oob}$ using tree T_b and sum the number of times the tree predicts the correct class;
3. For independent attributes $j = 1, \ldots, p$:
    a. Permute the values of the independent variable x_j in $S_{b,oob}$;
    b. Use T_b to predict class membership for $S_{b,oob}$ using the permuted x_j values; sum the number of times the tree predicts the correct class for the $S_{b,oob}$ observations;
    c. Subtract the number of votes for the correct class in permuted OOB data from the number of votes for the correct class in the OOB data.

The average difference in accuracy of the OOB versus permuted OOB observations over the B trees is the variable importance measure for x_j. For a fixed number of trees, a variable that is ranked highly on the variable importance measure indicates that the variable is important for classification.

2.3.2 Variable Importance in ANN

Much research has been done in trying to explore techniques to extract variable importance measures from an ANN model especially in ecological studies [33-35]. In this project, two algorithms will be adopted in an attempt to extract variable importance measures from an ANN, namely (1) Connection Weights method and (2) Garson's algorithm.

Gevrey et al. [33] have made a comparison of multiple existing techniques that try to extract variable importance information from an ANN network. However, Olden et al. [35] argued that the experimental methodologies used in Gevrey et al. [33] are flawed. In Olden's own words, "…method comparisons by Gevrey et al. were based on an empirical dataset…which precludes the ability to establish any generalizations regarding the *true accuracy* and *precision* of the different approaches because the *true correlation structure* of the empirical data is unknown and therefore the *true importance of the variables* is unknown. [35]" In Olden et al. [35], simulated data is used (so the properties of the data are known and the *true importance* of the variables is known) and the authors recommend the use of the Connection Weights approach as it is the best performing algorithm out of the many algorithms compared.

Garson [36] proposed a method in 1991 that utilises an ANN model's connection weights to calculate and rank the importance of the different inputs in a network's input layer [34, 36]. This approach has been used extensively in ecological studies [34] but its accuracy and precision have always been in doubt [35]. Garson's algorithm proved to be the worst performing out of the many proposed methods studied in Olden et al. [35]. The relative popularity of the poor-performing Garson's algorithm in the field of ecological studies is undeserved and inexplicable. Olden et al.'s work [35] can also only provide a limited view on the issue of extracting variable importance from an ANN as more research can be done to thoroughly understand the different methods proposed.

Both Connection Weights approach and Garson's algorithm utilise the raw values of the connection weights to calculate the importance of an input. Connection Weights method calculates the product of the raw input-hidden and hidden-output connection weights between each input neuron and output neuron and sums the products across all the hidden neurons [34]. Garson's algorithm partitions hidden-output connection weights into components associated with each input neuron using absolute values of connection weights [33, 36]. The reader is recommended to refer to **Appendix 1** for a detailed example of how each of these algorithms works.

It is important to note that both algorithms only work with a simple MLP structure that has only a single hidden layer and one output neuron in the output layer. These requirements severely limit the use of the two algorithms to only binary classification and regression ANN models with a simple single-hidden layer structure.

2.4 Model Assessment Techniques

2.4.1 Confusion Matrix

To construct a confusion matrix, one needs to first build a classifier model based on a set of training data and prepare a set of testing data that the model has "never seen before" to judge the accuracy of a model. It is through this fresh set of testing data that will be used to be filled into the four different sections of a confusion matrix (four sections for a binary classifier):

1. True positive: instance is true and the model predicted that it is true,
2. True negative: instance is false and the model predicted that it is false,
3. False positive: instance is false but the model predicted it as true and,
4. False negative: instance is true but the model predicted is as false.

It is the researcher's responsibility to decide the ratio of which to split the original dataset into. The ratio used in this dissertation is a 7:3 split. With 1000 instances, 700 will be used for training while 300 instances will be used for testing.

		Actual Event	
		True	**False**
Predicted	**True**	True Positive (TP)	False Positive / Type I (FP)
	False	False Negative / Type II (FN)	True Negative (TN)

Table 2.1: A binary classifier's confusion matrix.

A confusion matrix is a table layout that is used to visualise the classification performance of a classifier. Table 2.1 shows a confusion matrix for a binary classifier in the context of credit scoring. A Type I error ("false positive") is made when the classifier predicts a credit application to be good when it is in fact bad. A Type II error ("false negative") is made when the classifier predicts a credit application as bad when it is actually good. Type I error is worse than Type II error in credit scoring i.e. it is worse to classify a credit application as good when they are bad than it is to classify a credit application as bad when they are good. This is because Type I error would result in a non-repayment by the customer which would mean loss in profit for loan-granting institutions. A Type II error would only mean a loss in business without costing the institutions.

There are a few metrics that can be calculated from the confusion matrix of a classifier that will be used in this project.

$$False\ Negative\ Rate = \frac{FN}{FN + TP} \tag{7}$$

$$False\ Positive\ Rate = \frac{FP}{FP + TN} \tag{8}$$

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN} \tag{9}$$

$$Specificty\ (Precision) = \frac{TP}{TP + FN} \tag{10}$$

The false positive rate is commonly mistaken for the Type I error rate and the false negative rate commonly mistaken for the Type II error rate as shown in Wang et al. [20] and Tsai et al. [26]. There is a subtle difference in between the two where Type I error rate is associated with the *a-prior* setting of the significance level by a researcher while the false positive rate is associated with the *post-prior* result. This report will not walk the same path as that of Tsai and Wang in using Type I/II error when showing the results obtained.

Sensitivity and specificity measure the quality of an information retrieval process such as that of a classification process. Sensitivity (also known as recall) indicates the completeness of information retrieval. It is defined as the portion of positive instances retrieved by the process versus the total number of existing positive instances. Specificity (also known as precision) describes the actual accuracy of the retrieval. It is defined as the portion of positive examples that exist in the total number of examples retrieved. Based on these two

measures, one can justify if a classifier is better than another at information retrieval. In the context of credit scoring, a classifier that has a low recall but high precision is preferred than a model with high recall but low precision.

It would be best to have a model that has a high recall and precision rate. However, if such a model is difficult to come by, credit-granting institutions can employ a two-stage screening strategy for credit scoring. The first stage would be to use a classifier model with high sensitivity but low specificity for an initial screening of all credit applications. This would ensure that the organisation would lose as little business as possible. The second stage would involve using a different classifier model with a high precision to be applied on the credit applications that have passed through the initial screening. Through the more precise classifier model, the credit-granting institution can make sure that it loses as little money as possible from all the potential businesses from the initial screening.

2.4.2 K-Fold Cross-Validation

If a researcher decides that there is not enough data to split into a training and testing dataset, *k*-fold cross-validation uses part of the available data to fit the model and a different part to test it. The original dataset is randomly partitioned into *k* equal sized subsamples. A single subsample is retained as the testing data to test the model while the remaining *k-1* subsamples are used as training data. This entire process is repeated *k* times with each of the *k* subsamples used exactly once as the testing data i.e. each instance of the dataset will be used once for testing and *k-1* times for training. The *k* results from the folds will then be averaged to produce a single model accuracy estimation.

Cross-validation only effectively estimates the average error of a model [17]. In this dissertation, *k* is set to 10. 10-fold cross-validation will mainly be used on ANN to judge the technique's accuracy. However, cross-validation will not be used on RF as RF has its own internal estimate of classification error (see **Section 4.2**).

3. Data

3.1 Data Analysis

A German credit dataset has been obtained from the University of California, Irvine Machine Learning Repository [37]. This dataset contains 1000 instances of credit loan applications with 20 different variables. There is no missing data in the dataset. There are two different versions of the credit dataset: (1) a mixed version containing categorical and numerical data, and (2) a purely numerical version. The numerical version of the dataset has been prepared for machine learning techniques that cannot process categorical data.

Each individual variable is scrutinised to have a thorough understanding which will make the task of data pre-processing easier. It is essential to note the characteristics of the different variables in order to correctly pre-process them.

	Feature	Variable name in dataset	Properties
1	Status of checking account	CheckingAccount	Ordered categorical
2	Duration in month	Duration	Numerical
3	Credit history	CreditHistory	Unordered categorical
4	Purpose	Purpose	Unordered categorical
5	Credit amount	CreditAmount	Numerical
6	Savings account/bonds	SavingsAccount	Ordered categorical
7	Present employment since	EmploymentSince	Ordered categorical
8	Instalment rate	InstallmentRate	Numerical
9	Personal status	PersonalStatus	Unordered categorical
10	Other debtors/guarantors	OtherDebtors	Unordered categorical
11	Present residence since	ResidenceSince	Numerical
12	Property	Property	Unordered categorical
13	Age	Age	Numerical
14	Other instalment plans	InstallmentPlans	Unordered categorical
15	Housing	Housing	Unordered categorical
16	Number of existing credits	ExistingCredits	Numerical
17	Job	Job	Unordered categorical
18	Number of dependents	Dependents	Numerical
19	Telephone	Telephone	Binary categorical
20	Foreign worker	ForeignWorker	Binary categorical

Table 3.1: Attributes in the German dataset, their variable names in the dataset, and their properties.

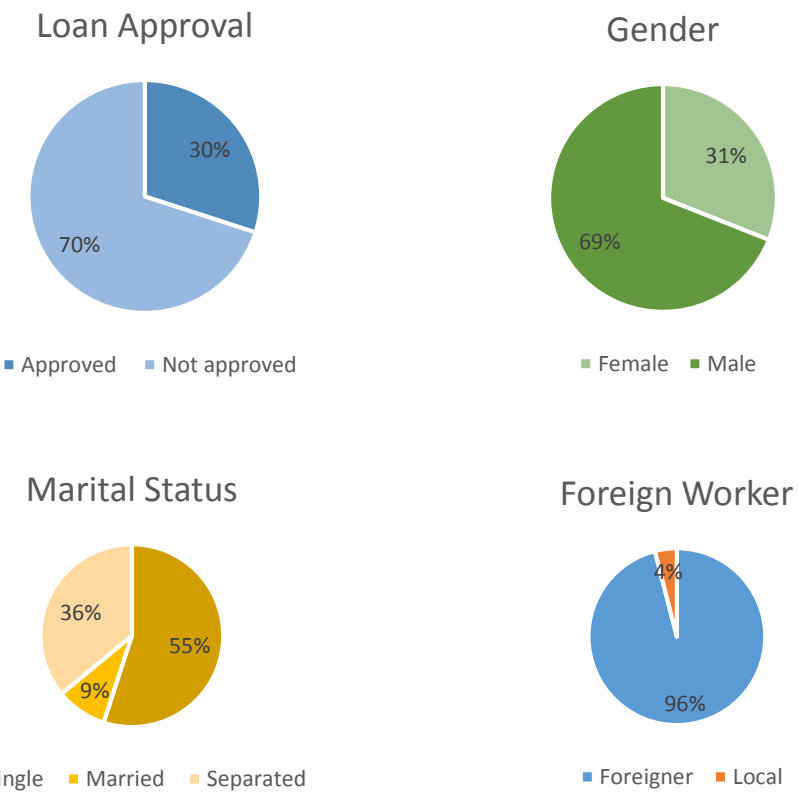

Figure 3.1: Pie charts showing the analysis of the dataset. From top to bottom, left to right: (1) Loan approval rate, (2) Gender composition of applicants, (3) Marital status of applicants, and (4) Applicant's nationality.

There are a few interesting points to note about the dataset. Out of the 1000 instances in the dataset, 70% has a positive outcome (success or good credit). The dataset is highly skewed towards data of foreign workers. Only 4% of the entire dataset are locals while a whopping 96% are credit loans applied for by foreigners. This presents a problem where the sample population is not representative of the real world.

Within the personal status category, there are more female than male applicants. And out of the entire dataset population, the majority are single. Due to the admittedly poor quality of the data, the females in the dataset are all categorised in a group no matter if they are divorced, separated or widowed. There are no single females in this dataset. Females represent 31% of the entire dataset with the rest are males. Out of the 69% of males in the dataset, 7.2% are divorced or separated, 13.6% are married or widows while the rest are single males. This dataset is heavily skewed towards single foreign male workers and cannot be representative of any population. As this dissertation is only interested in analysing the variable importance in this dataset, we can only find out what variables are important to the calculation of the credit score of the individuals in this dataset. The results in this report will not be able to generate a representing sweeping statement that is constructive to any other populations.

3.2 Data Pre-processing

Since RF - being an ensemble learning method - is a forest of DTs, it can process numerical as well as categorical data [14]. Only ANN requires data transformation on categorical data as it can only process numerical data. This section would focus on how categorical data is pre-processed into numerical data.

Numerical data does not need to be standardised. Sarle [30] argues that if input variables are combined linearly, as in an MLP (which is the case here), input standardisation is rarely necessary. This is because any rescaling of an input vector can be undone by changing the corresponding weights and biases [30]. However, Sarle [30] also notes that there are a variety of practical reasons in standardising the inputs to make training faster and reduce the chances of getting stuck in local optima.

There are a few methods to transform categorical data to numerical data:

1. Ordinal,
2. 1-of-N encoding,
3. 1-of-(N-1) encoding and,
4. Thermometer encoding.

An as example, assume that a variable with three categories is to be pre-processed. The ordinal raw values of the categories are low, medium, and high. The different methods of encoding are thus listed as below. x_i are derived variables that the encoding method will create to represent the original variable.

Ordinal raw value	Ordinal	1-of-N			1-of-(N-1)		Thermometer (N inputs)			Thermometer (N-1 inputs)	
	x_1	x_1	x_2	x_3	x_1	x_2	x_1	x_2	x_3	x_1	x_2
Low	1	1	0	0	1	0	1	0	0	0	0
Medium	2	0	1	0	0	1	1	1	0	1	0
High	3	0	0	1	0	0	1	1	1	1	1

Table 3.2: The different methods of encoding a categorical variable into numerical ones for ANN's purpose. x_i are the derived variables that the encoding method will create. The number of derived variables differ from method to method.

Although Fitkov-Norris et al. [32] argue that thermometer encoding is the most suitable method in transforming categorical data to numerical, Sarle [30] supports that 1-of-C encoding would do just as well. With only a 1% difference in the overall classification accuracy as shown in experiments done by Fitkov-Norris et al. [32], this dissertation agrees with the finding of Sarle [30] that the use of thermometer encoding does not result in a significant improvement in terms of the overall classification accuracy of the ANN model

In conclusion, the encoding approach that will be taken in this study is summed up as below:

1. If the input vector is numerical, leave it as it is.
2. If it is an ordered categorical input vector, an ordinal approach is taken.
3. If it is a binary categorical input vector, a binary encoding is taken.
4. If the input vector is unordered and categorical, a 1-of-C approach is taken.

There exists two different versions of the German credit dataset. One being a pure numerical version while the other is a mix of both categorical and numerical data (termed the mixed version). The numerical version of the dataset has been pre-processed by Strathclyde University by projecting the mixed data into a different, purely numerical hyperspace. This results in a dataset where the data are not categorised i.e. feature names are lost. Therefore, the numerical dataset is not suitable for the purpose of this project to rank variable importance.

4. Experimental Design

4.1 The Hypothesis

The hypothesis in mind when pursuing this project was simple: a loan applicant's protected characteristics affected the outcome of the loan in this dataset. To prove (or disprove) the hypothesis, one has to first find out how much the "personal status" variable contributes to the final outcome. Thus, the study of variable importance is made the focus in this dissertation.

As the main aim of this work is to study variable importance and in turn answer some of the questions posed, it is essential that the data mining models used are trained to be similar or better-performing in terms of overall accuracy compared to work done in the field. For RF, Breiman who invented RF used the numerical version of the German credit data in his paper [9]. Note that the dataset used here is the original version posted by Hofmann [37] where there is a mix of both categorical and numerical data which is slightly different from that of [9] where a numerical version of the dataset with 24 variables is used. Due to the different versions of dataset used, it is the aim of this dissertation to try to achieve accuracy comparable to that of [9] RF model.

As for ANN, it is the intention of this project to perform the same as it is for RF. An ANN model will first be trained to achieve a performance that is comparable to that of West's [29]. However, the same case applies here where the dataset used is the numerical version instead of the original dataset with mixed numerical and categorical data. It is suspected from the very start that the overall accuracy of the ANN model will be far worse off compared to that of [29] as ANN cannot handle categorical data. The data transformation process to convert categorical data to numerical will no doubt affect the performance of the ANN model.

Readers of this dissertation might question the insistence to use the mixed version of the dataset instead of the numerical version if both Breiman [9] and West [29] use the numerical version of the dataset. This is due to the unnamed attributes of the numerical data which will influence the ultimate objective of this project which is to study which attribute is more important than another (see **Section 3.1**). If we cannot distinguish one attribute from another, it would be a waste even if models trained have performance similar to that of peer-reviewed dissertations such as that of Breiman's [9] and West's [29].

4.2 Model Tuning

There are mainly two variables that need to be tuned to obtain a model with a satisfactory performance:

1. Number of variables chosen during bootstrapping, m_{try} and,
2. Number of trees to construct, n_{tree}.

Breiman, Cutler and Liaw suggested in [9, 12, 15] that a starting value for m_{try} would be to take the square root of the number of variables. In this case, it would be $\sqrt{20} = 4.47 \approx 4$ variables. Experiments have to be done to determine which value of mtry would give the lowest OOB error rate. The initial value for n_{tree} will be controlled and set to 1000 so that only the value of m_{try} is varied. The value of n_{tree} is set to a large number due to the fact that OOB estimate of error rate is quite accurate given that enough trees are grown [12]. Once the value of m_{try} has been calculated to achieve an optimal performance, the number

of trees can then be varied to search for the number of trees to grow in order to get the lowest classification error.

As for ANN, it can be relatively more complex than RF to configure. There are more variables that affect the performance of ANN. Ultimately, there are three variables at play in designing the layout of an ANN:

1. Number of hidden layers,
2. Number of nodes in each hidden layer and,
3. Architecture of the ANN (feedforward or recurrent).

The scope of this work will be limited to only focus on an ANN model with a single hidden layer, or more definitively, a single hidden layer perceptron. There are two reasons behind this scope-limiting decision. Firstly, this will reduce the complexities that have to be dealt with, specifically, not having to deal with the number of layers in an ANN model. Secondly, Garson's algorithm and the Connection Weights approach can only be applied to a MLP with a single hidden layer. 10-fold cross-validation will be run 10 times (and the classification error averaged) to determine the appropriate number of nodes in the hidden layer in the model. The size of the single hidden layer is then selected by choosing the number of nodes with the lowest average classification error.

4.3 Experimental Setup

4.3.1 Comparison of Variable Importance Rankings

As RF's variable importance measure is more recognised in the field than Garson's algorithm or the Connection Weights approach for ANN, a RF model R_a, will first be trained to rank the importance of all 20 variables. R_a will be trained according to the m_{try} and n_{tree} values that are found to be optimal during the tuning of the RF models. R_a will then be used to look at the different variable importance measures. With the ranked variables based on variable importance, it will then be judged whether the "personal status" variable ranks highly in the calculation of the credit loans. If the results do indicate that the personal status column has a certain effect on the outcome of the credit application, it shall confirm the hypothesis proposed. However, if the opposite is true, the results will only prove that the personal status column wasn't used in the original equation to calculate the credit risk of applicants.

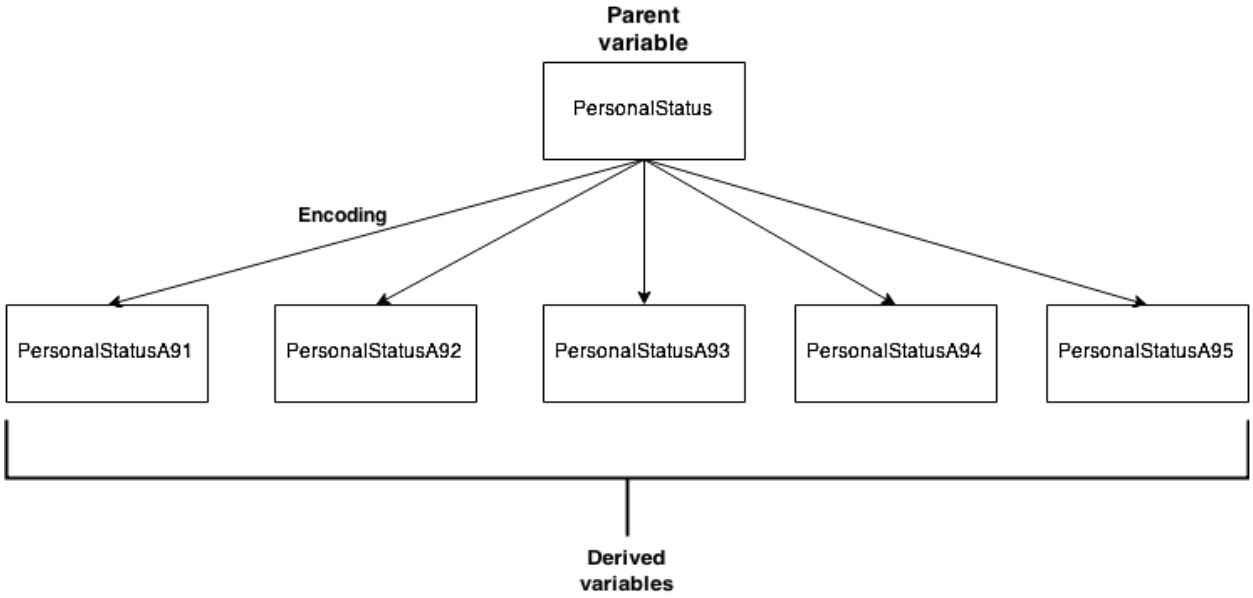

Figure 4.1: An example of ANN encoding method to transform a variable from the original dataset (termed the parent variable) into numerical variables (termed derived variables).

The variable importance rankings produced by Garson's algorithm and the Connection Weights approach for MLP will be compared against that of RF's. The two different algorithms produce a raw importance value for each input of a MLP and it is this raw importance value metric that the derived variables will be ranked by. Due to the encoding method taken, only the inputs to the MLP (i.e. derived variables) will be ranked and not the variables from the original dataset (i.e. parent variables). This poses a problem where the rankings of the variables in the original dataset are still unknown. Unfortunately, no ranking of any of the derived variables can dictate the ranking of the parent variable in the initial dataset (parent variable) i.e. the rankings of the parent variables in the original hyperspace are unknown. However, the different rankings of the derived variables that originated from the same parent variable in the initial dataset can let one speculate whether the parent variable ranks highly in terms of importance.

A method is proposed in this dissertation (termed the Rank Averaging method) to train 10 separate MLP models to obtain an average ranking of the derived variables and from this ranking, further average the result to obtain the rankings of the parent variables. This essentially permits the reverse calculation of variable importance of the parent variables from the rankings of each parent variable's derived variables.

$$DerivedVariables(p) = \{\, x \mid x \; is \; a \; derived \; variable \; of \; p \,\} \tag{11}$$

$$AverageRanking_i = \frac{\sum_{n=1}^{N} DerivedVariableRanking_{n,i}}{N} \tag{12}$$

where

- i is a derived variable e.g. PersonalStatusA91, PersonalStatusA92, … ,
- N is the number of runs made, i.e. number of models trained which in this project is 10 and,
- $DerivedVariableRanking_{n,i}$ is the index number in the ranking of a derived variable i in run n. The ranking of a derived variable depends on the raw importance value calculated either by Garson's algorithm or the Connection Weights approach.

$$ParentVariableRanking_j = Ranking\left(\frac{\sum_{k=1}^{|A|} Ranking(AverageRanking_k)}{|A|}\right) \quad (13)$$

where

- j is a parent variable e.g. PersonalStatus,
- $A = DerivedVariables(j)$, the set of derived variables of a parent variable as shown in Equation (7),
- $|A|$ is the cardinality of set A,
- $Ranking(x)$ returns the index number in the ranking of x.

One reason why the raw importance values produced by Garson's algorithm or the Connection Weights approach are not used in the calculation of the derived variable ranking is because the raw importance values are calculated from the raw weights value of a network which are dependent on the randomised weight values during a network's initialisation. Therefore, the raw weight values are independent from one network to another and thus the raw importance values are also independent. Also, the rankings of a parent variable are calculated from the sum of the rankings of its derived variables averaged over the number of its derived variables as shown in Equation (13). This is done to avoid introducing bias to a parent variable with a higher number of levels i.e. higher number of derived variables. The reader is referred to the **Appendix 2** section for a simple example of the Rank Averaging algorithm to obtain the rankings of a parent variable.

The 10 separate MLP models will be trained with all instances in the dataset. Garson's algorithm and the Connection Weights approach would then be applied to the MLP model to obtain the rankings of variable importance. The Rank Averaging method would then be applied on the results produced to obtain a ranking of the parent variables. It is highly suspected that Garson's algorithm's ranking would be more unreliable compared to that of RF's variable importance measuring heuristics and the Connection Weights approach. This is due to findings of Fitkov-Norris et al. and Olden et al. [32, 35] that showed Garson's algorithm's poor performance compared to other methods in trying to rank variable importance of a black-box ANN model.

4.3.2 Comparison of Model Accuracy

The dataset will be split into a randomised 7:3 ratio of training and testing data to test and compare the classification accuracy of RF and MLP. A model R_b will be trained with the same m_{try} and n_{tree} values of that of R_a. A MLP model will also be trained alongside but with encoding methods applied on the same training and testing data. This process will be repeated 10 times and the classification accuracy of the two models will be averaged. The comparison of R_b against MLP will effectively be the comparison of model accuracy of the two different techniques.

Process of comparing average classification accuracy of R_b against MLP:

```
for i := 1 to 10
  trainingData, testingData := randomise(dataset)
  Rb[i] := Random-Forest(trainingData)
  MLP[i] := MLP(preprocess(trainingData))
  RFError[i] := test(Rb[i], testingData)
  MLPError[i] := test(MLP[i], preprocess(testingData))
end
# Compare average RFError against average MLPError
```

Would taking away the personal status column have minimal to no impact on the outcome of the credit application? To answer the question, another model R_c will be trained with the personal status column taken out from the training and testing data. It is in the interest of this dissertation to see how R_c's performance will compare to that of R_b. This process is repeated 10 times and the average classification accuracy will shed some light on whether the personal status column will have a small or large impact on the performance of the classifiers.

Process of comparing average classification accuracy of R_b against R_c:

```
for i := 1 to 10
  trainingData, testingData := randomise(dataset)
  trainingDataRc := remove('PersonalStatus', trainingData)
  testingDataRc := remove('PersonalStatus', testingData)
  Rb[i] := Random-Forest(trainingData)
  Rc[i] := Random-Forest(trainingDataRc)
  RbError[i] := test(Rb[i], testingData)
  RcError[i] := test(Rc[i], testingDataRc)
end
# Compare average RbError against average RcError
```

Experiment	Model	Technique	Number of training instances	Number of testing instances	Number of variables
Variable Importance	R_a	Random Forest	1000	0	20
	MLP	Multilayer Perceptron	1000	0	48
Model Accuracy[1] (RF vs MLP)	R_b	Random Forest	700	300	20
	MLP	Multilayer Perceptron	700	300	48
Model Accuracy[2] (RF vs RF)	R_b	Random Forest	700	300	20
	R_c	Random Forest	700	300	19

Table 4.1: Summary of the different models to be trained and compared.

4.4 Using RF as a Feature Selection Technique

What happens if the lower-ranking variables in RF's variable importance measure are eliminated from the training of ANN as a dimensionality reduction method? It is also in the interest of this study to utilise the results from RF to try to reduce the number of variables fed into ANN. 10-fold cross-validation will be applied to see how well a MLP model will perform with the lower-ranking variables from RF eliminated from its training dataset. If the performance of the new MLP model is comparable to that of the old model where all variables are included, RF will be suggested as a candidate dimensionality reduction technique for ANN. Cross-validation will be used to find out the effective classification error of this new MLP model with reduced number of variables. However, without a comparison of RF as a dimensionality reduction technique against other more established dimensionality reduction techniques, this study cannot conclude saying that RF is an *effective* dimensionality reduction technique.

[1] Experiment to compare Random Forest and Multilayer Perceptron techniques' accuracy.
[2] Experiment to compare how the exclusion of the personal status variable affects Random Forest's accuracy.

5. Implementation

5.1 Choice of Platform

A declarative programming styled platform is preferred where programming is done with expressions as compared to sequential programming. There are two mainstream statistical programming languages to perform data mining applications: (1) MATLAB and (2) R programming language. Most of the statistical programming language syntaxes as offered by MATLAB and R are concise and have less room for error compared to sequential programming.

MATLAB is an implementation of a high-level language with a well-developed interactive development environment for technical computing. It is developed by MathWorks and has libraries that span across multiple disciplines such as image processing, control systems, and computational finance. The software is proprietary but it has documentation for each additional add-ons that MATLAB has to offer. MATLAB maintains a community where users of MATLAB can also implement libraries and offer it to other community members.

R is an open-source GNU project under the GPL license that attempts to implement the S programming language, a statistical programming language. Its purpose is mainly focused on statistical computing and graphics. It uses a command-line interface but there are several graphical user interface provided by other entities with the prominent one being RStudio developed by RStudio, Inc. (that is also open source). R, much like MATLAB, has a large amount of libraries (more than 5,800 additional packages) to offer that are written by the community of R's users at the Comprehensive R Archive Network (CRAN). Being a GNU project, most of the libraries are open-sourced to let users of the packages to look into their implementations.

	MATLAB	**R Programming Language**
Licensing	Commercial	Open source GNU project
Integrated Development Environment (IDE)	IDE comes with default installation of MATLAB	Command-line interface as default. Graphical user interface IDEs such as RStudio are available
Quality of Documentation	Paid product, has decent documentation	Quality of documentations of packages vary and can sometimes be lacking.
Technical Support Availability	Commercial product from a company, has technical support available for a fee. Help can also be sought from the user community.	Assistance is limited to the domain of internet. Mailing lists and active community are available for users to seek for help.

Table 5.1: A quick comparison of MATLAB and the R programming language.

5.2 Packages

randomForest

randomForest is an R package implemented by Liaw and Wiener [12]. It provides an R interface to the random forest algorithm written in Fortran originally implemented by Breiman and Cutler. This package produces two additional pieces of information namely variable importance and proximity measure. The variable importance measure is especially important for this project. There are only two heuristics implemented out of the four different variable importance heuristics suggested by Breiman [9, 12].

nnet

The nnet package implements a feed-forward ANN with a single hidden layer. It is implemented by Venables et al. for the book *Modern Applied Statistics with S* [38]. This package is suitable for the aim of the project as the scope of this project is restricted to only a single hidden layer MLP. Similar to the randomForest package, nnet provides an R interface with most of the underlying code implemented in the C language. Therefore, the algorithm is quick to converge as it runs at native, compiled speed.

6. Results and Evaluation

6.1 Tuning of RF and MLP

6.1.1 Tuning of RF

With enough trees grown, we can get an acceptable value for m_{try}. Empirical experience tells us that n_{tree} = 1000 is a good enough value to estimate the OOB error. The OOB estimate of error rate is quite accurate given that enough trees have been grown. Otherwise, OOB error can bias upward [12, 39]. There is no need for cross-validation technique to be used on RF to obtain a classification error measure as RF already has its own internal estimate of test set error i.e. OOB error due to the way how RF works. An experiment has been run to determine the correct number of m_{try}. Each potential m_{try} value is run 10 times with 1000 trees grown and the average is shown below.

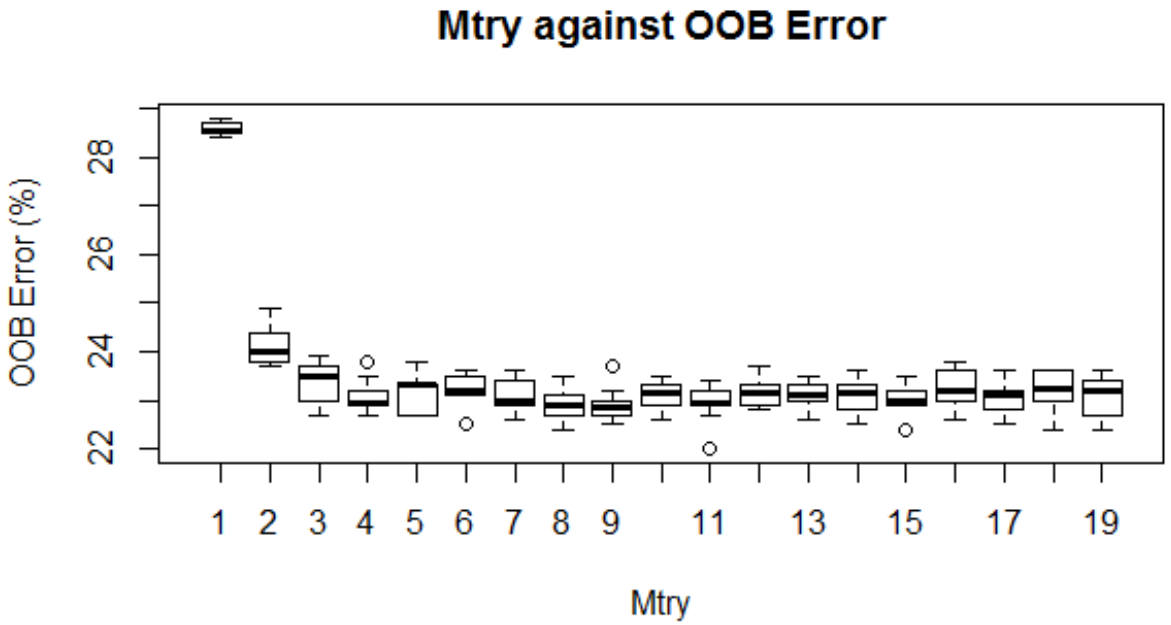

Figure 6.1: A box plot of 10 experiments of m_{try} value against OOB error with n_{tree}=1000 trees grown.

m_{try}	Mean OOB Error (%)	Median OOB Error (%)
1	28.58	28.55
2	24.10	24.00
3	23.38	23.50
4	23.07	22.95
5	23.14	23.30
6	23.24	23.20
7	23.10	23.00
8	22.94	22.90
9	22.91	22.85
10	23.10	23.15
11	22.94	22.95
12	23.15	23.15
13	23.10	23.10
14	23.07	23.15
15	22.98	23.00
16	23.24	23.20
17	23.04	23.10
18	23.20	23.25
19	23.10	23.20

Table 6.1: Mean and median value of OOB error with different m_{try} values. n_{tree} is set to 1000. Underlined values are best-performing, lower is better.

From Table 6.1, a m_{try} value of 9 seems to be the most appropriate with the lowest mean OOB error at 22.91% and the lowest median OOB error of 22.85%. With the m_{try} value now found, the correct number of trees to grow, n_{tree} has to be determined. Each potential n_{tree} value is run 10 times and the OOB error results are shown in the box plot below.

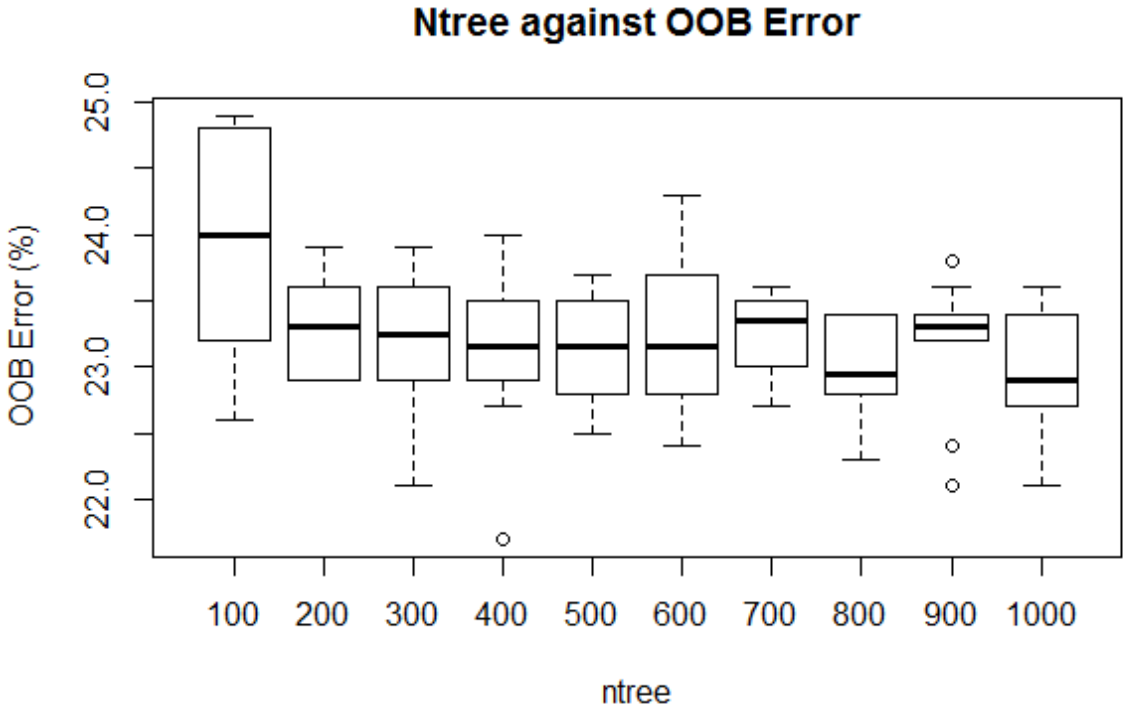

Figure 6.2: A box plot of 10 experiments of n_{tree} value against OOB error with m_{try} = 9.

n_{tree}	Mean OOB Error (%)	Median OOB Error (%)
100	23.94	24.00
200	23.32	23.30
300	23.17	23.25
400	23.13	23.15
500	23.15	23.15
600	23.21	23.15
700	23.24	23.35
800	23.00	22.95
900	23.16	23.30
1000	<u>22.96</u>	<u>22.90</u>

Table 6.2: Mean and median OOB error with different n_{tree} values. m_{try} is set to 9. Best-performing values are underlined, lower is better.

From Figure 6.2, a value of $n_{tree} = 800$ is chosen instead of $n_{tree} = 1000$. There are three reasons behind this choice. Firstly, a value of $m_{try} = 9$ will obviously result in a RF model with $n_{tree} = 1000$ having the lowest OOB error as the m_{try} value is minimised with n_{tree} set to 1000 originally thus a bias exists for $n_{tree} = 1000$. Secondly, even though the OOB error of 1000 is lower, $n_{tree} = 800$ has a smaller OOB error variance compared to when $n_{tree} = 1000$ which can help in producing more consistent results throughout this dissertation. Thirdly, with a lower number of trees grown, the time needed to train a RF model will be shorter.

6.1.2 Tuning of MLP

With data pre-processing, there is a huge surge in the number of variables from 20 to 48 that are fed to the MLP model due to the encoding methodologies employed in this study (see **Section 3.2** and Figure 4.1). This rise in the number of inputs is inevitable as ANN in general cannot process categorical data. MLP needs to first be tuned to find the correct size of hidden layer. 10-fold cross-validation is used to search for the most suitable number of nodes in the hidden layer. Results are shown in Figure 6.3.

A lowest classification error of 25.8% is obtained when a MLP model is trained with a 12-node hidden layer. The number of nodes in the hidden layer is thus decided to be set to 12 in future MLP model constructions. For the raw results shown in Figure 6.3, please refer to **Appendix 4**.

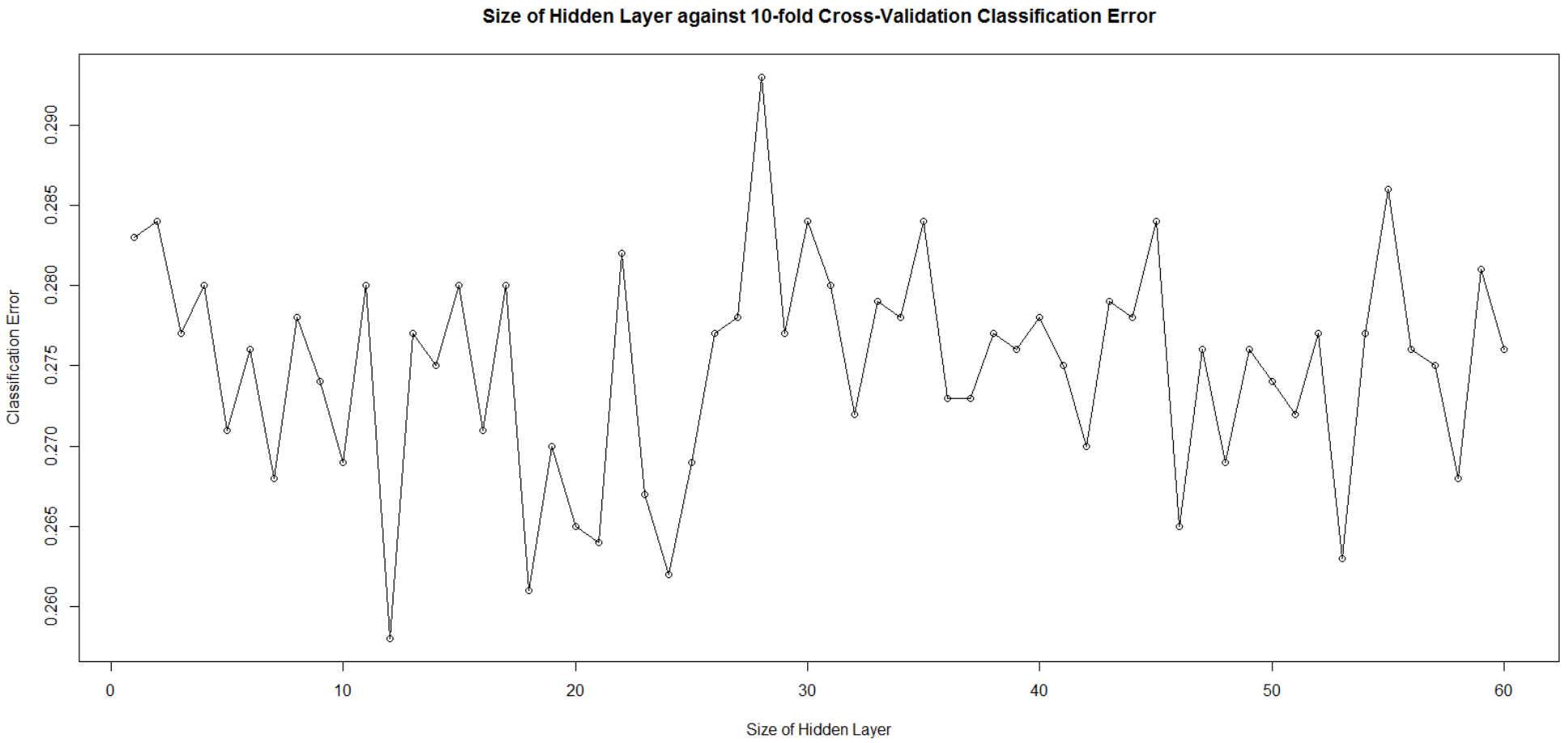

Figure 6.3: Size of hidden layer of MLP trained against classification error rate.

6.2 Comparison of Variable Importance

6.2.1 RF Variable Importance

A RF model R_a will then be tuned according to the m_{try} and n_{tree} values found to be most suitable. The entirety of the dataset will be used to train R_a. The variable importance measure is then extracted from R_a.

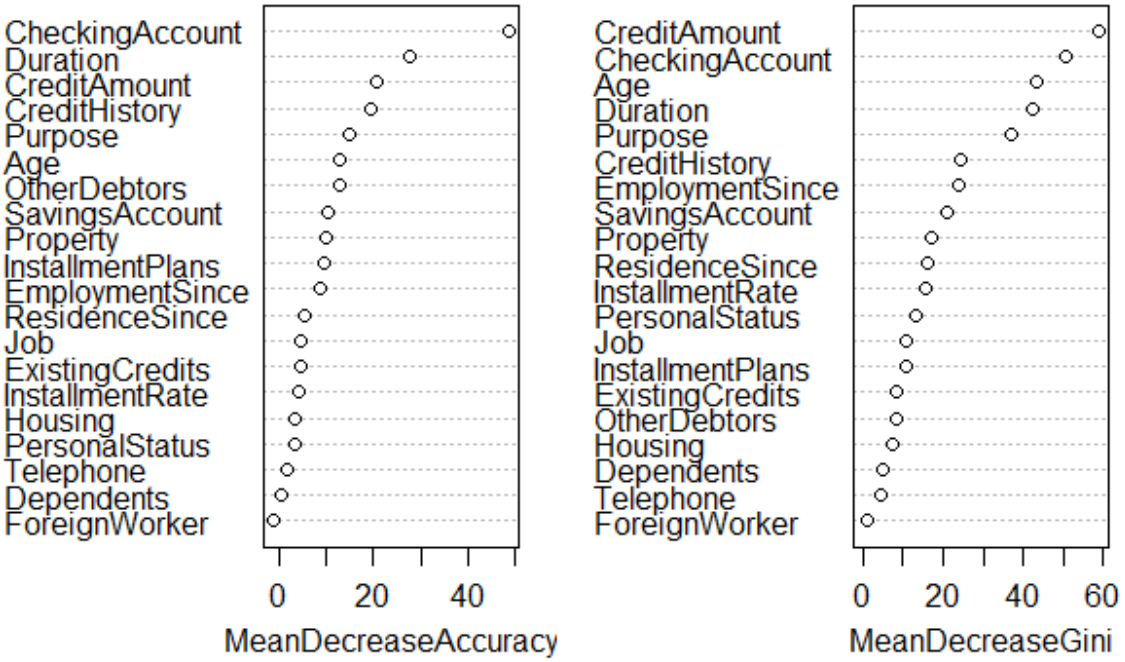

Figure 6.4: Variable importance measure extracted from RF model R_a. The figure shows the rankings of variables according to Mean Decrease in Accuracy (left) and Mean Decrease in Gini (right). Most important to the least important variables are ranked from top to bottom.

Two different heuristics of variable importance measure are available: mean decrease in accuracy (MDA) and mean decrease in Gini (MDG). From Figure 6.4, even though the two heuristics might disagree on the precise rankings of the variables, the top six most important variables in deciding the outcome of credit risk in the MDA is also in the top six ranked in MDG. The six most important variables regarded are listed below in no specific order:

1. Checking Account,
2. Duration,
3. Credit Amount,
4. Credit History,
5. Age and,
6. Purpose.

As for the personal status column, it is ranked lowly in both heuristics. From this experiment, we can know that RF thinks that an individual's personal status has minimal to no impact on the outcome of his/her credit application.

6.2.2 MLP Variable Importance

With Garson's algorithm, the Connection Weights approach and the method to obtain rankings of parent variables from their derived variables, the average rankings of the parent variables are shown in the table below. The MLP models are trained using all instances in the dataset as overfitting is assumed to not be an issue in extracting the variable importance of MLP.

Ranking	Garson's Algorithm	Connection Weights
1	CreditAmount	CheckingAccount
2	CheckingAccount	InstallmentRate
3	Age	ExistingCredits
4	Duration	Dependents
5	ResidenceSince	Duration
6	Telephone	CreditHistory
7	ForeignWorker	InstallmentPlans
8	ExistingCredits	Job
9	Dependents	PersonalStatus
10	EmploymentSince	OtherDebtors
11	InstallmentPlans	Housing
12	CreditHistory	Property
13	OtherDebtors	Purpose
14	Purpose	CreditAmount
15	Job	Age
16	PersonalStatus	ResidenceSince
17	Housing	SavingsAccount
18	Property	EmploymentSince
19	InstallmentRate	ForeignWorker
20	SavingsAccount	Telephone

Table 6.3: The rankings of the parent variables calculated using Garson's algorithm and the Connection Weights approach.

From Table 6.3, we can see that the personal status derived variables aren't ranked highly. Even with Garson's notorious unreliability, it also shows that the personal status variable has minimal to no impact on the success of the credit loan. It came as a surprise that the rankings calculated from the Connection Weights approach disagree with the rankings of RF severely. In Olden et al. [35], Connection Weights was the best performing algorithm out of the many different ANN variable importance measure compared. However, in this dissertation, Connection Weights seems to be the worst performing out of all the different variable importance measures.

6.3 Comparison of Model Accuracy

RF model R_b is first compared to MLP in terms of model accuracy. A 7:3 ratio of training and testing data are used to create a confusion matrix. This is repeated 10 times and the average classification accuracy is shown below.

Model	Average Accuracy (%)	False Negative Error (%)	False Positive Error (%)	Recall Rate (%)	Precision Rate (%)
R_b (RF)	<u>76.0</u>	<u>9.1</u>	58.8	<u>90.9</u>	41.2
MLP	67.6	24.0	<u>51.9</u>	76.0	<u>48.1</u>

Table 6.4: Comparison of R_b and MLP models. The model whose performance is best in the metric specified is underlined.

R_b model that represents RF performs better than MLP in terms of classification accuracy. However, RF has a higher false positive rate and a lower precision rate which might not make it a suitable model for making the final decision whether to grant a loan. With RF's high recall rate, RF is more suitable for an initial screening of credit applications whose output can then be fed to

a model with a higher precision rate. By using this method, loan-granting institutions will only miss a small amount of good-performing loan through the initial screening while the other more precise model will weed out the good-performing loans from the bad-performing ones during the second screening.

Next, comparison of RF models R_b and R_c is made with their results shown below. R_b is trained and tested with 20 variables while R_c is trained and tested with only 19 variables (personal status column taken out).

Model	Average Accuracy (%)	False Negative Error (%)	False Positive Error (%)	Recall Rate (%)	Precision Rate (%)
R_b[3] (RF)	77.2	<u>9.2</u>	56.7	<u>90.8</u>	43.3
R_c[4] (RF)	<u>77.3</u>	9.5	<u>55.4</u>	90.6	<u>44.6</u>

Table 6.5: Comparison of R_b and R_c models. The model whose performance is best in the metric specified is underlined.

Overall, it is safe to say that R_c performs better than R_b. R_c is more accurate and precise while its false negative rate and recall rate are similar to that of R_b. It is interesting to see such a result with the personal status variable left out from training. It can be safely said that this only further confirms the rankings of RF that the personal status variable is not considered as a factor during the original calculation of the credit risks. In fact, through the comparison of R_b and R_c, the inclusion of the personal status variable is actually introducing noise to the RF classifier.

The tables below show how the performance of the models used in this dissertation compare to models of other papers.

	Average Accuracy (%)	False Negative Error (%)	False Positive Error (%)	Data used (Mixed/Numerical)
Random Forest	77.2	9.2	56.7	Mixed
Breiman [9]	77.2	-	-	Numerical
Wang et al. [20]	77.1	9.5	54.3	Mixed

Table 6.6: Comparison of RF model used in this dissertation to that of other papers.

The RF model used in this dissertation is up to par of that of other papers. The RF model achieved an equal average accuracy rate to that of Breiman's original model in Breiman [9] and is better than the model trained by Wang et al. [20]. This shows that the m_{try} and n_{tree} values used in this dataset are reasonable as they produced a RF model with a good performance.

[3] RF model trained with all 20 variables in the German credit dataset.
[4] RF model trained with 19 variables in the German credit dataset. The personal status variable is taken out.

	Average Accuracy (%)	False Negative Error (%)	False Positive Error (%)	Data used (Mixed/Numerical)
MLP	67.6	9.2	56.7	Mixed
Nanni et al. [28]	75.2	47.6	26.8	Mixed
Tsai et al. [26]	79.0	44.3	9.5	Mixed
West [29]	73.3	13.5	57.5	Numerical

Table 6.7: Comparison of MLP model used in this dissertation to that of other papers.

The MLP model lacks a lot in terms of average accuracy when compared to other papers. It is suspected that this is caused by the data pre-processing method that was employed. It is regrettable that Nanni et al. [28] and Tsai et al. [26] did not include the encoding method used to pre-process their dataset. A separate experiment has been done on the numerical version of the dataset and the results are published in **Appendix 4**, to justify the size of the hidden layer and the implementation of the MLP algorithm are not the factors to be blamed for MLP's poor performance.

No matter what the accuracy of the different techniques are for this dataset, the reader is reminded that the focus of this study is not on model accuracy. The experiments done above to compare model accuracy are mainly just to judge the correctness of the different technique's implementation and also to validate the parameter tuning results of RF and MLP.

6.4 Using RF as a Feature Selection Technique for MLP

With the explosion in the number of variables RF can be used as a dimensionality reduction method for ANN. 10-fold cross-validation is run to get the classification accuracy of the MLP models whose inputs are reduced. There are two different heuristics where the rankings of the least important variables are different. For each heuristic, the number of variables are reduced by eliminating the *n*-least important variable and 10-fold cross-validation is run with a MLP model with a hidden layer with 12 nodes. The classification accuracy for each of the individual experiments are shown below:

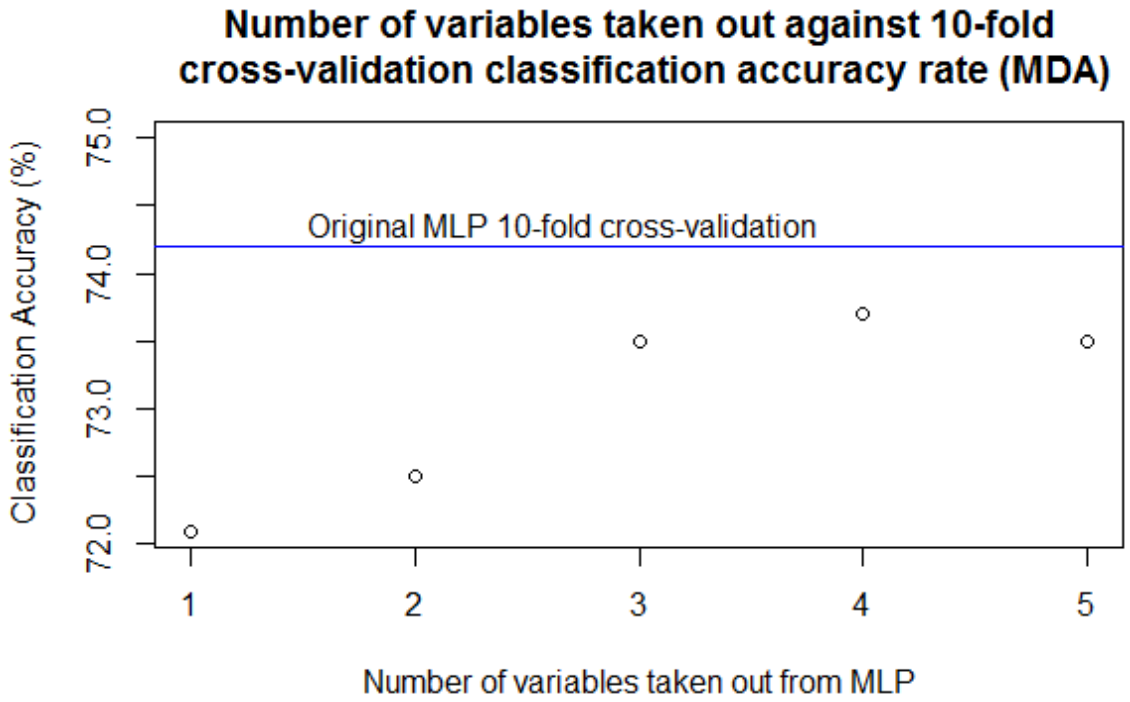

Figure 6.5: Bottom-most ranked variables according to MDA taken out from the original dataset against the 10-fold cross-validation classification accuracy of MLP.

Number of variables taken out (Variable Taken Out)	Effective number of variables taken out from MLP (according to MDA)	Classification Accuracy (%)
1 (ForeignWorker)	1	72.1
2 (Dependents)	2	72.5
3 (Telephone)	3	73.5
4 (PersonalStatus)	8	73.7
5 (Housing)	11	73.5

Table 6.8: Raw data values of 10-fold cross-validation classification accuracy as depicted in Figure 6.6. Effective number of input variables taken out from MLP are cumulative.

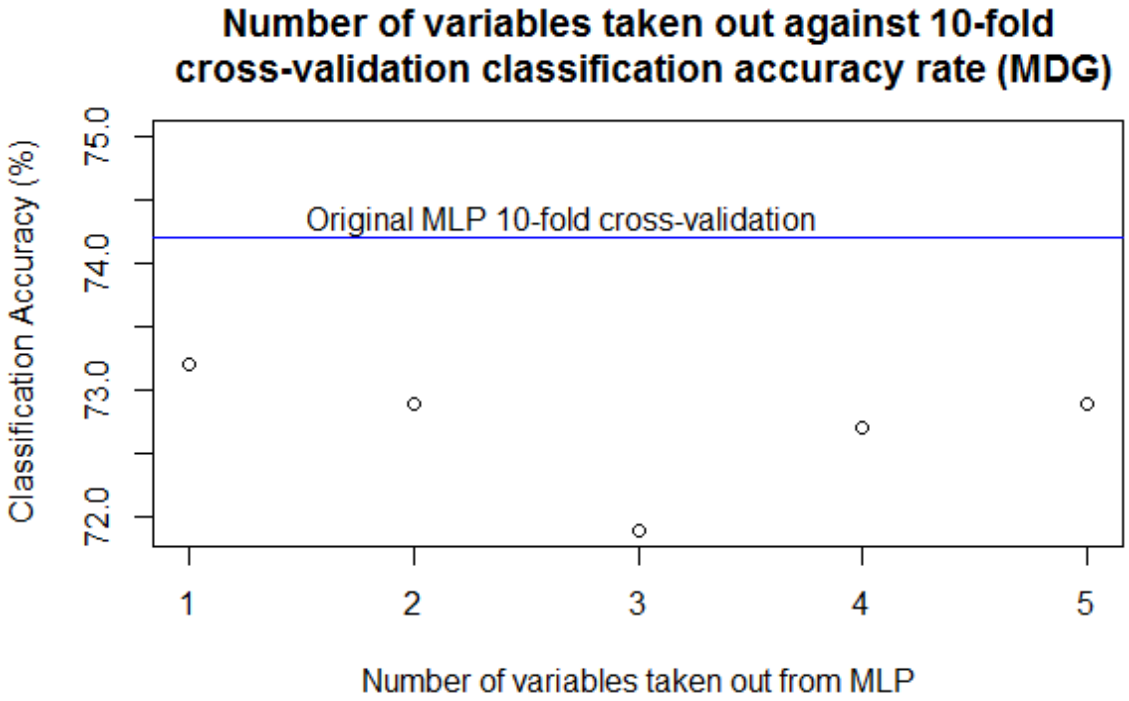

Figure 6.6: Bottom-most ranked variables according to MDG taken out from the original dataset against the 10-fold cross-validation classification accuracy of MLP.

Number of variables taken out (Variable Taken Out)	Effective number of variables taken out from MLP (according to MDG)	Classification Accuracy (%)
1 (ForeignWorker)	1	73.2
2 (Telephone)	2	72.9
3 (Dependents)	3	71.9
4 (Housing)	6	72.7
5 (OtherDebtors)	9	72.9

Table 6.9: Raw data values of 10-fold cross-validation classification accuracy as depicted in Figure 6.7. Effective number of input variables taken out from MLP are cumulative.

Note that when a variable is taken out, there might be more than one input that is reduced when training the MLP model. This is due to the encoding method that is being employed in this project as shown in Figure 4.1. Termed the effective number of variables taken out, it represents the number of variables reduced from the training of the MLP model. As an example, if the personal status variable is decided to be taken out, the effective number of variables taken out from the training of the MLP model would be 5 as the personal status column has five different categories (as the encoding method derives five different inputs from this parent variable).

The original MLP that has been run with 10-fold cross-validation has a classification accuracy of 74.2%. The performance of the five different MLP models with the *n*-bottom-most ranked variables in MDA taken out decreased only by a slight 1-2%. It is surprising as the trend of the classification accuracy rate increases with more variables taken out. However, it is still important to note that the performance of all the different models are still worse off compared to the original MLP model.

Similar results can be seen by taking out the *n*-bottom-most ranked variables according to MDG. The performance of the different models are worse off than the original MLP model. The difference in classification accuracy rate is also only in the range of 1-2%. The best performing dataset would be the dataset with 4-bottom-most ranked variables taken out as it resulted in a MLP model with an accuracy of 73.7%.

7. Discussions and Conclusions

7.1 Discussion

This project regards the rankings of variable importance produced by the two different heuristics of RF as more accurate, precise, and reliable as the variable importance measure of RF is more established and recognised in the field [9, 12, 15, 40]. The MDA and MDG measures regard the same six variables as the most important variables although their ranks might differ slightly. However, there is a need to take into account that RF biases slightly towards a variable with a high number of levels (i.e., high number of categories) when calculating the variable importance measure [32, 40].

Rank	MDA	MDG	Garson's Algorithm	Connection Weights
1	CheckingAccount	CreditAmount	CreditAmount	CheckingAccount
2	Duration	CheckingAccount	CheckingAccount	InstallmentRate
3	CreditAmount	Age	Age	ExistingCredits
4	CreditHistory	Duration	Duration	Dependents
5	Purpose	Purpose	ResidenceSince	Duration
6	Age	CreditHistory	Telephone	CreditHistory
7	OtherDebtors	EmploymentSince	ForeignWorker	InstallmentPlans
8	SavingsAccount	SavingsAccount	ExistingCredits	Job
9	Property	Property	Dependents	PersonalStatus
10	InstallmentPlans	ResidenceSince	EmploymentSince	OtherDebtors
11	EmploymentSince	InstallmentRate	InstallmentPlans	Housing
12	ResidenceSince	PersonalStatus	CreditHistory	Property
13	Job	Job	OtherDebtors	Purpose
14	ExistingCredits	InstallmentPlans	Purpose	CreditAmount
15	InstallmentRate	ExistingCredits	Job	Age
16	Housing	OtherDebtors	PersonalStatus	ResidenceSince
17	PersonalStatus	Housing	Housing	SavingsAccount
18	Telephone	Dependents	Property	EmploymentSince
19	Dependents	Telephone	InstallmentRate	ForeignWorker
20	ForeignWorker	ForeignWorker	SavingsAccount	Telephone

Table 7.1: Comparison of ranking by RF's Mean Decrease in Accuracy (MDA), RF's Mean Decrease in Gini (MDG), ANN's Garson's algorithm, and ANN's Connection Weights approach.

As for the variable importance rankings produced by Garson's algorithm and the Connection Weights approach, this dissertation doubts the reliability of both algorithms, disagreeing with the findings of Gevrey et al. [33] and Olden et al. [35] who claimed that these algorithms can somehow shed light on the importance of variables in an ANN. If the variable importance measures produced by RF (MDA and MDG) are set as the benchmark, at first glance, Garson's algorithm might seem to produce a result similar to that of MDA and MDG. Out of the six most important variables in MDA and Garson's, Garson's has four of the same variables ranked in its top six most important variables (Credit Amount, Checking Account, Age, and Duration). Also, the four variables regarded by MDG to be the most important are ranked exactly in the same order by Gason's algorithm. However, Garson's algorithm seemingly decent performance is under suspicion when one looks at how important Garson's ranks the Telephone and Foreign Worker variables when all the other variable importance measures rank these two variables lowly.

The findings of this study contradicts the findings of Olden et al. [35] who have suggested and recommended the use of the Connection Weights approach to rank the importance of variables in a dataset. Connection Weights, like Garson's algorithm, uses the raw weights values of an ANN to calculate the rankings of the inputs to the ANN model. The rankings produced by Connection Weights differ significantly from the rankings produced by all other variable importance measures including that of Garson's. These results suggest that there needs to be further study on the accuracy and precision of the Connection Weights algorithm.

As to the comparison of both the RF and ANN (multilayer perceptron, MLP, in this case) techniques, results show that RF compare more favourably in this dataset than ANN, at least in terms of classification accuracy. RF beats MLP in most metrics and its performance does not lag far behind that of ANN in metrics that it did not do as well. With the unsatisfactory precision rate of RF, there needs to be more work done in the field to tune the precision rate of RF. With a low 41.2% precision rate and a high recall rate of 90.9%, it might seem like RF just classifies most of the instances as positive (good credit) blindly. However, the unbalanced dataset such as the German credit dataset that is used in this dissertation might have affected the precision of RF. This study recommends RF to be considered as a technique to be used in the context of credit scoring during the initial screening process of credit applications as described in **Section 2.4**.

MLP performed surprisingly bad at binary classification in this project. This might be due to the encoding method adopted in this dissertation to pre-process the categorical variables in the dataset into pure numerical only data that the MLP can process. When a MLP model with 12 hidden nodes are trained on the numerical version of the dataset (no pre-processing needed) as opposed to the mixed version of the dataset that contains both categorical and numerical data, MLP performed decently with an accuracy of 76.3% which is comparable to that of RF (for the results, please refer to **Appendix 3**). It is apparent that the technique is not at fault but it is suspected that the encoding method used is dragging down the performance of MLP.

RF was experimented as a feature selection technique for MLP. The focus is on selecting a subset of the most relevant features for use in model construction. Results show that the MLP model trained on the dataset with a reduced number of variables achieve a satisfactory performance when compared to the MLP model trained on the unaltered dataset. However, it is important to note that in even though it is in the academia's interest to trade-off classification accuracy for faster training and lower calculation cost through dimensionality reduction methods, it is the opposite in real world applications. Every percentage in a model's accuracy counts as there are immense economic ramifications to institutions that rely on credit scoring models' judgments to make a profit. Institutions that depend upon a model's accuracy rate would be willing to trade-off time and money for a model with a higher accuracy.

7.2 Criticism and Future Work

The tuning of RF has introduced a bias to the value of n_{tree} unexpectedly by fixing the value of n_{tree} to search for the value of m_{try}. However, this project regards the high number of trees as having little effect on the performance of the RF algorithm. As long as n_{tree} is set to an acceptable, high enough value which is supported by Liaw [12], the performance of RF will not be impacted. After all, the performance of the RF models trained in this study are comparable to that of Breiman's [9], the inventor of the algorithm. This high performance of RF is not surprising as the underlying implementation used in this study is the exact same implementation written by Breiman which is wrapped by an R programming language interface by Liaw [12]. It is fortunate that as long as the value of n_{tree} is high enough, the performance of RF would still be satisfactory. This project should have used multi-objective optimisation methods in an attempt to

search for the values of m_{try} and n_{tree} [41]. A suggestion for the process of optimisation would be to vary both n_{tree} and m_{try} values at the same time to search for the combination that would produce a RF model with the lowest classification error.

Another aspect that this dissertation did not discuss on was the effect of overfitting of a model on the rankings of variables in terms of variable importance. This concern is more directed towards ANN rather than RF as RF does not overfit as easily [9, 15]. As both the Garson's algorithm and the Connection Weights approach rely on the raw weights values of an ANN to produce the rankings of variable importance, the resulting overfitted model might produce an over-tweaked poorly-ranked outcome. It is in this dissertation's interest to see more work done to study the effect of the degree of overfitting on the accuracy and precision of the rankings produced by Garson's and Connection Weights.

The encoding method used in this project has introduced more variables that made the MLP suffer from the curse of dimensionality [42] which this dissertation attributes to the MLP model's poor classification performance. From the 20 variables of the original dataset, the encoding method increased it to 48 variables while the number of instances still remains the same. It was proven in a separate experiment that the bad classification performance of MLP is not the fault of the technique in the context of credit scoring after applying MLP on the numerical version of the dataset (see **Appendix 3**). The German credit dataset being a weakly non-linear problem [3], MLP should theoretically have no problem in achieving a good performance for credit scoring, especially the dataset used in this project as proven by Baesens et al. [3] who used the same dataset. The methods to pre-process categorical variables into numerical variables are few and research in this area is limited. It is this report's opinion that there needs to be further study on the effects of the encoding method adopted during the pre-processing procedure on the performance of an ANN classifier.

Due to the encoding method, there needs to be a method to project the rankings of the 48 inputs of the MLP produced by Garson's algorithm and the Connection Weights approach back to the original hyperspace of 20 variables to obtain a ranking for the original 20 variables. The Rank Averaging method has been proposed and it might be this projection that could have affected the rankings of the variables produced by Garson's and Connection Weights which might explain the discrepancies of the findings between this project and that of Olden et al. [35]. The Rank Averaging method average the multiple rankings obtained from several independent runs. It is obvious that this method is susceptible to anomalies. If a derived variable is somehow ranked lowly in one out of the ten runs, it would have had a huge impact on the ranking of its parent variable. There needs to be more work done to investigate the effects of variable importance ranking projection (from the inputs to the MLP to the original dataset) on the accuracy and precision of the ANN variable importance measures.

The original equation to calculate the credit risk of applicants is unknown and this project cannot compare the reliability of RF variable importance measures' rankings against that of the original equation. This report regards the rankings produced by RF as the benchmark and this can be a fundamental flaw if RF's rankings are proved to be unreliable. Also, Gower's algorithm (or something similar) should be used to easily compare and visualise how similar/dissimilar a measure's rankings are to another measure's rankings.

7.3 Conclusion

This project showed that the personal status column was not taken into account when calculating the approval or rejection of credit application in this German dataset. The variable importance

measure of random forest (RF) is concluded to be the most accurate as the personal status variable is ranked lowly on both measures of variable importance heuristics that RF has to offer. The two different variable importance measure heuristics of RF proved to be reliable as they agree with each other albeit there are slight differences in the rankings of the variables. The personal status variable's insignificance is corroborated by the finding of another experiment. When the personal status attribute is taken out from the training and testing data, the RF model trained achieved a slightly better performance, proving that the variable only introduces more noise to the training of a classifier model.

This study also tried to perform two variable importance "extraction" algorithms, namely the Garson's algorithm and the Connection Weights approach. They are applied on artificial neural network (ANN) models trained but the results differ significantly from that of RF's variable importance measure. Results indicate that both Garson's and Connection Weights are unreliable. However, there are too many unknowns that are introduced unexpectedly to the model of ANN trained in this dissertation to make any strong suggestion on the reliability, accuracy, and precision of Garson's algorithm and the Connection Weights approach. It is the interest of this project to see more study done to verify the reliability of these ANN variable importance algorithms.

Finally, RF was also tested as a feature selection technique for MLP. It is shown that RF was able to effectively eliminate 8 variables (out of 48) for MLP and still be able to achieve a performance that is 99.3% of the model trained on the original dataset. This study recommends RF to be considered as a candidate feature selection technique by eliminating variables that RF ranks lowly on the variable importance scale.

8. Bibliography

1. Mester, L.J., *What's the point of credit scoring?* Business review, 1997. **3**: p. 3-16.
2. Baesens, B., *Developing intelligent systems for credit scoring using machine learning techniques.* status: published, 2003.
3. Baesens, B., et al., *Benchmarking state-of-the-art classification algorithms for credit scoring.* Journal of the Operational Research Society, 2003. **54**(6): p. 627-635.
4. Huang, C.-L., M.-C. Chen, and C.-J. Wang, *Credit scoring with a data mining approach based on support vector machines.* Expert systems with applications, 2007. **33**(4): p. 847-856.
5. Chandler, G.G. and D.C. Ewert, *Discrimination on the basis of sex under the Equal Credit Opportunity Act*. 1976: Krannert Graduate School of Management, Purdue University.
6. Parliament, U., *Equality Act 2010.* The Stationery Office. London: UK Parliament, 2010.
7. di Torella, E.C. *On lies and statistics: the relationship between gender equality and insurance*. in *ERA Forum*. 2011. Springer.
8. Morozov, E. *Your Social Networking Credit Score*. 2013 30th January 2013 29th April 2015]; Available from: http://www.slate.com/articles/technology/future_tense/2013/01/wonga_lenddo_lendup_big_data_and_social_networking_banking.html.
9. Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.
10. Amit, Y. and D. Geman, *Shape quantization and recognition with randomized trees.* Neural computation, 1997. **9**(7): p. 1545-1588.
11. Breiman, L., *Bagging predictors.* Machine learning, 1996. **24**(2): p. 123-140.
12. Liaw, A. and M. Wiener, *Classification and regression by randomForest.* R news, 2002. **2**(3): p. 18-22.
13. Lewis, R.J. *An introduction to classification and regression tree (CART) analysis*. in *Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California*. 2000.
14. Kantardzic, M., *Data mining: concepts, models, methods, and algorithms*. 2011: John Wiley & Sons.
15. Cutler, A., *Trees and Random Forests.* 2013.
16. Schapire, R.E., et al., *Boosting the margin: A new explanation for the effectiveness of voting methods.* Annals of statistics, 1998: p. 1651-1686.
17. Hastie, T., et al., *The elements of statistical learning*. Vol. 2. 2009: Springer.
18. Caruana, R. and A. Niculescu-Mizil. *An empirical comparison of supervised learning algorithms*. in *Proceedings of the 23rd international conference on Machine learning*. 2006. ACM.
19. Brown, I. and C. Mues, *An experimental comparison of classification algorithms for imbalanced credit scoring data sets.* Expert Systems with Applications, 2012. **39**(3): p. 3446-3453.
20. Wang, G., et al., *Two credit scoring models based on dual strategy ensemble trees.* Knowledge-Based Systems, 2012. **26**: p. 61-68.
21. McCulloch, W.S. and W. Pitts, *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 1943. **5**(4): p. 115-133.
22. Rosenblatt, F., *The perceptron, a perceiving and recognizing automaton Project Para*. 1957: Cornell Aeronautical Laboratory.

23. Hornik, K., M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators.* Neural networks, 1989. **2**(5): p. 359-366.
24. Russell, S., P. Norvig, and A. Intelligence, *A modern approach.* Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 1995. **25**.
25. Haykin, S.S., et al., *Neural networks and learning machines*. Vol. 3. 2009: Pearson Education Upper Saddle River.
26. Tsai, C.-F. and J.-W. Wu, *Using neural network ensembles for bankruptcy prediction and credit scoring.* Expert Systems with Applications, 2008. **34**(4): p. 2639-2649.
27. Minsky, M. and P. Seymour, *Perceptrons.* 1969.
28. Nanni, L. and A. Lumini, *An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring.* Expert Systems with Applications, 2009. **36**(2): p. 3028-3033.
29. West, D., *Neural network credit scoring models.* Computers & Operations Research, 2000. **27**(11): p. 1131-1152.
30. Sarle, W.S., *Neural network FAQ.* Periodic posting to the Usenet newsgroup comp. ai. neural-nets, 1997.
31. Archer, K.J. and R.V. Kimes, *Empirical characterization of random forest variable importance measures.* Computational Statistics & Data Analysis, 2008. **52**(4): p. 2249-2260.
32. Fitkov-Norris, E., S. Vahid, and C. Hand, *Evaluating the impact of categorical data encoding and scaling on neural network classification performance: the case of repeat consumption of identical cultural goods*, in *Engineering Applications of Neural Networks*. 2012, Springer. p. 343-352.
33. Gevrey, M., I. Dimopoulos, and S. Lek, *Review and comparison of methods to study the contribution of variables in artificial neural network models.* Ecological Modelling, 2003. **160**(3): p. 249-264.
34. Olden, J.D. and D.A. Jackson, *Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks.* Ecological modelling, 2002. **154**(1): p. 135-150.
35. Olden, J.D., M.K. Joy, and R.G. Death, *An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data.* Ecological Modelling, 2004. **178**(3): p. 389-397.
36. Garson, D.G., *Interpreting neural network connection weights.* 1991.
37. Blake, C. and C.J. Merz, *UCI Repository of machine learning databases [http://www. ics. uci. edu/~ mlearn/MLRepository. html]. Irvine, CA: University of California.* Department of Information and Computer Science, 1998. **55**.
38. Venables, W.N. and B.D. Ripley, *Modern applied statistics with S*. 2002: Springer Science & Business Media.
39. Bylander, T., *Estimating generalization error on two-class datasets using out-of-bag estimates.* Machine Learning, 2002. **48**(1-3): p. 287-297.
40. Strobl, C., et al., *Bias in random forest variable importance measures: Illustrations, sources and a solution.* BMC bioinformatics, 2007. **8**(1): p. 25.
41. Deb, K., *Multi-objective optimization*, in *Search methodologies*. 2014, Springer. p. 403-449.
42. Bellman, R., *Dynamic programming and Lagrange multipliers.* Proceedings of the National Academy of Sciences of the United States of America, 1956. **42**(10): p. 767.

9. Appendix

Appendix 1: Garson's algorithm and Connection Weights approach

Input-Hidden Connection Weights

X \ Y	Hidden A	Hidden B	Hidden C	Hidden D	Hidden E
Input 1	-0.93	-1.49	0.37	-0.91	0.37
Input 2	-0.57	-1.96	-0.14	1.18	1.26
Input 3	-0.85	1.74	-1.86	-0.05	0.10
Input 4	0.25	-3.01	-0.99	-1.34	-1.65
Input 5	-0.82	0.09	0.86	-0.41	-0.05

X

Hidden-Output Connection Weights

	Hidden A	Hidden B	Hidden C	Hidden D	Hidden E
Output	-1.75	-1.08	-1.13	-2.90	3.37

Connection Weight Products

X \ Y	Hidden A	Hidden B	Hidden C	Hidden D	Hidden E
Input 1	1.63	1.62	-0.42	2.64	1.24
Input 2	1.00	2.12	0.16	-3.43	4.25
Input 3	1.48	-1.89	2.10	0.14	0.34
Input 4	-0.43	3.26	1.12	3.90	-5.57
Input 5	1.43	-0.09	-0.97	1.18	-0.16

$$Input_X = \sum_{Y=A}^{E} Hidden_{XY}$$

	Importance	Rank
Input 1	6.71	1
Input 2	4.10	2
Input 3	2.18	3
Input 4	2.28	4
Input 5	1.38	5

Connection Weight Approach

$$Input_X = \sum_{Y=A}^{E} \frac{|Hidden_{XY}|}{\sum_{Z=1}^{5} |Hidden_{ZY}|}$$

	Importance	Rank
Input 1	0.88	4
Input 2	1.11	2
Input 3	0.94	3
Input 4	1.50	1
Input 5	0.57	5

Garson's Algorithm

Simulated data with 5 variables are generated to train a MLP. Each input is simulated to correlate with the output. The first variable is simulated to correlate with the output the most and in a linear fashion for all variables, the variables correlate lesser till the fifth variable which correlates with the output the least. The input-hidden layer's weights and the hidden-output layer's weights are shown above. As Garson's algorithm uses the absolute values of the Connection Weight products in its calculation, Garson's can produce unreliable ranking results at times.

Example to calculate the importance value of **Connection Weight Approach**:

$$Input_1 = 1.63 + 1.62 - 0.42 + 2.64 + 1.24 = 6.71$$

Example to calculate the importance value for **Garson's Algorithm**:

$$Input_1 = \frac{1.63}{1.63 + 1.00 + 1.48 + 0.43 + 1.43} +$$

$$\frac{1.62}{1.62 + 2.12 + 1.89 + 3.26 + 0.09} + \frac{0.42}{0.42 + 0.16 + 2.10 + 1.12 + 0.97} +$$

$$\frac{2.64}{2.64 + 3.43 + 0.14 + 3.90 + 1.18} + \frac{1.24}{1.24 + 4.25 + 0.34 + 5.57 + 0.16}$$

$$= 0.27 + 0.18 + 0.09 + 0.23 + 0.11$$

$$= 0.88$$

Appendix 2: Rank Averaging Example

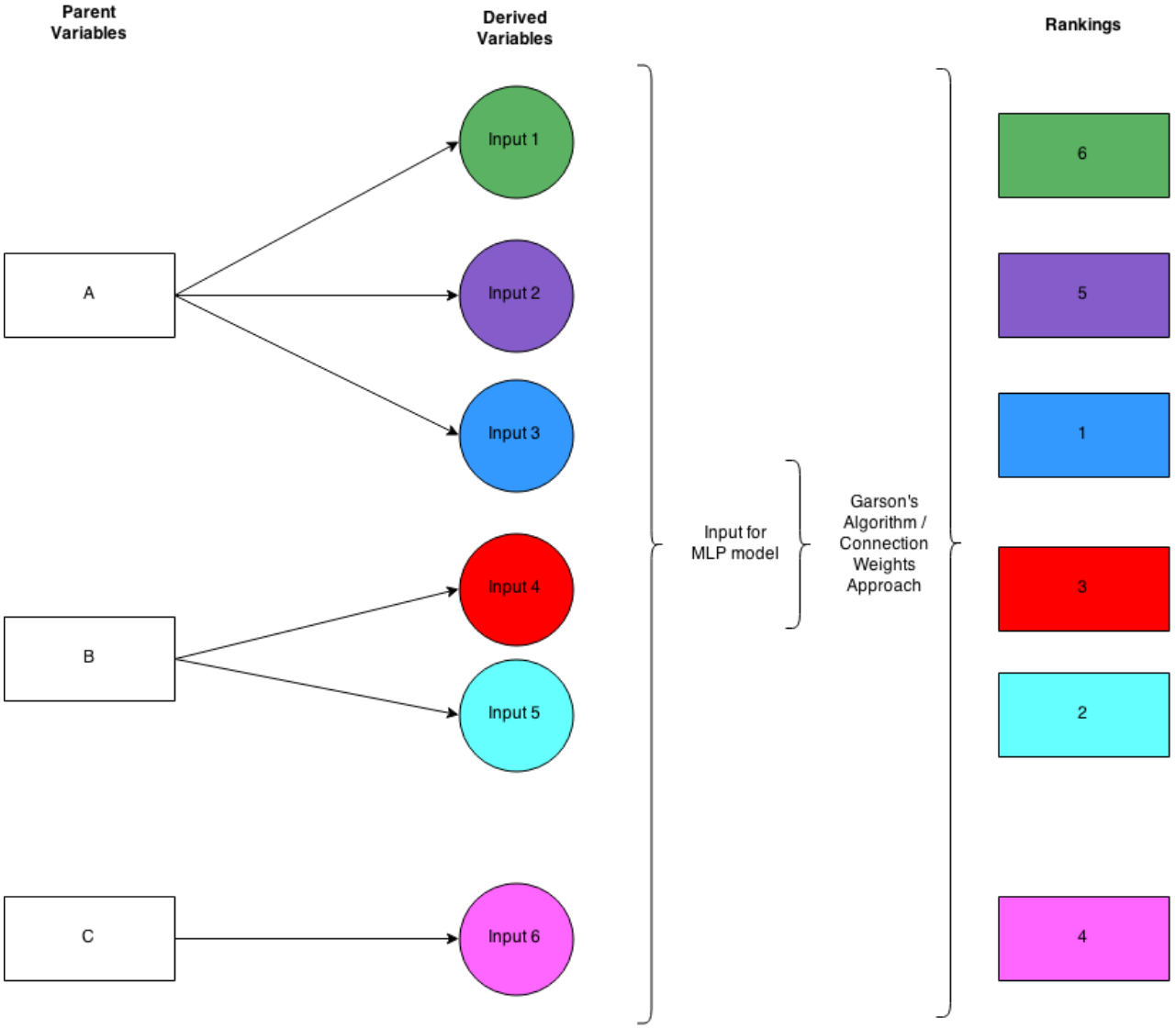

Figure 9.1: Rankings of inputs produced by applying Garson's algorithm or Connection Weights approach.

Variables in the Dataset	Type of variable
A	Categorical with 3 levels
B	Categorical with 2 levels
C	Numerical

Table 9.1: The different variables in the example dataset and their properties.

The example above shows how parent variables are encoded into their respective derived variables and how the derived variables are ranked by either Garson's algorithm or the Connection Weights approach. Imagine if the process above is run 3 times and the three different ranking results are shown below. All results are hypothetical.

Inputs to the ANN	Results of rankings across three runs	Average Rank	Ranking
Input 1	{6, 4, 6}	5.3	5
Input 2	{5, 6, 5}	5.3	5
Input 3	{1, 2, 1}	1.3	1
Input 4	{3, 3, 4}	3.3	3
Input 5	{2, 1, 2}	1.6	2
Input 6	{4, 5, 3}	4.0	4

Table 9.2: The rankings of the inputs across three runs. Average rank is obtained by summing the rankings of the three runs and divided by 3.

To get the parent variable's ranking:

Variables in the Dataset	Parent Variable Score	Parent Variable Ranking
Variable A	$\frac{5+5+1}{3} = 3.7$	2
Variable B	$\frac{3+2}{2} = 2.5$	1
Variable C	$\frac{4}{1} = 4$	3

Table 9.3: The rankings of the original variable after applying the Rank Averaging method.

From the Rank Averaging method, we find that variable B is most important, followed by variables A and C.

Appendix 3: Results of MLP trained with numerical version of the German credit dataset

Model	Average Accuracy (%)	False Negative Error (%)	False Positive Error (%)	Recall Rate (%)	Precision Rate (%)
MLP (numerical)	76.3	13.3	48.0	86.7	52.0

Table 9.4: Results of MLP model trained on the numerical version of German credit dataset.

Appendix 4: Raw result values of number of hidden nodes against 10-fold cross-validation error

Size of Hidden Layer	10-fold Cross-Validation Error (%)
1	28.3
2	28.4
3	27.7
4	28.0
5	27.1
6	27.6
7	26.8
8	27.8
9	27.4
10	26.9
11	28.0
12	25.8
13	27.7
14	27.5
15	28.0
16	27.1
17	28.0
18	26.1
19	27.0
20	26.5
21	26.4
22	28.2
23	26.7
24	26.2
25	26.9
26	27.7
27	27.8
28	29.3
29	27.7
30	28.4
31	28.0
32	27.2
33	27.9
34	27.8
35	28.4
36	27.3
37	27.3
38	27.7
39	27.6
40	27.8
41	27.5
42	27.0
43	27.9
44	27.8
45	28.4
46	26.5
47	27.6
48	26.9
49	27.6
50	27.4
51	27.2
52	27.7
53	26.3
54	27.7
55	28.6
56	27.6
57	27.5
58	26.8
59	28.1
60	27.6

Table 9.5: Raw results of the size of hidden layer in an MLP against 10-fold cross-validation error.