2018

Investigation into the Predictive Power of Artificial Neural Networks and Logistic Regression for Predicting Default in Chit Funds

Ciara Kerrigan
Technological University Dublin

Investigation into the Predictive Power of Artificial Neural Networks and Logistic Regression for Predicting Default in Chit Funds

Ciara Kerrigan

D06108066

A dissertation submitted in partial fulfilment of the requirements of Dublin Institute of Technology for the degree of
M.Sc. in Computing (Data Analytics)

2018

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: 

Date: 13/06/2018

ABSTRACT

This study evaluated the performance of an artificial neural network (ANN) multi-layer perceptron model and a logistic regression logitboost (LR) model to predict default in chit funds. The two types of default investigated were late payment of 30 days and late payment of 90 days. The dataset was broken up into training and validation datasets using random sampling and K folds cross validation was used on the training dataset to assess performance of the tuning parameters. The validation dataset was used to compare performance of both algorithms. Principle component analysis (PCA) was used to reduce the feature set while still explaining 95% of the variance in the data. The classes were highly imbalanced and Synthetic Minority Oversampling Technique (SMOTE) and down sampling were used to overcome the class imbalance. 16 experiments were ran, 8 for each of the two defaults. The three key metrics that were measured for these experiments were balanced accuracy, Area under the ROC curve (AUC) and F1 score. After making Bonferroni's adjustment to the original p value statistical significance was set to 0.003 when comparing multiple experiments.

In these experiments the ANN model had the best results for balanced accuracy, AUC and F1score. Statistical analysis using a paired t test showed that there was a statistically significant difference in the results between ANN and LR. The results of these experiments also showed that there was very little difference in the contribution of the top 20 features to the first 30 principal components, which were used to predict default. These features included family id, income and address. Features that had little or no contribution to the principle components included Commission, Auction Amount, and type of relation the nominee is to the chit fund member. These findings are context specific and in this case the context is chit funds from a digital chit fund operator in India.

Key words: *chit funds, artificial neural networks, logistic regression, default, predicting default*

ACKNOWLEDGEMENTS

I would like to express my sincere thanks my dissertation supervisor Mohammed Mesabbah who guidance throughout the dissertation process was invaluable. I would also like to thank my family and friends for their support throughout.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

TABLE OF ABBREVIATIONS

ANN:	Artificial Neural Networks
AUC:	Area Under the receiver operating characteristic Curve
FN:	False Negative
FP:	False Positive
FPR:	False Positive Rate
LR:	Logistic Regression
KNN	K nearest neighbour
MLP:	Multi-Layer Perceptron
OLS:	Ordinary Least Squares
ROC:	Receiver Operator Characteristic
ROSCA:	Rotating and Saving Credit Association
TP:	True Positive
TN:	True Negative
TRP:	True Positive Rate

1 INTRODUCTION

This chapter presents the background of the research topic and discusses the research problem outlining its importance. This is followed by the research question including the research hypothesis, the aims and objectives and the research methodologies. It then outlines the scope and limitations of the study and ends with an outline of how the rest of the dissertation is organised.

1.1 Background

Chit funds have been used for over 60 years in India as a form of savings and loans often used by low income households and small businesses (Anderson, 1966; Rottenberg, 1968). Chit funds are a type of Rotating and Saving Credit Association (ROSCA) (Salamon, Kaplan, & Goldberg, 2009). ROSCAs are a rotating financial circle often run on a local basis.

Rottenberg (1968) asserts that in a chit fund a group of people come together and agree to pay a set amount into a fund on an agreed basis. At the end of each month the group meet and decide who will receive this monthly fund, picking this person either by lottery or allocating the fund to the lowest bidder. The fund operator is paid a commission and any unused funds are distributed evenly throughout the group. Every person in the chit fund is awarded the fund once and the group continue to meet until all fund members have received the fund.

Accurate prediction of loan default and late payment is important to lenders because it can negatively affect the liquidity of a lender (de Carvalho, 2015). This applies to chit funds and ROSCAs too, even though the lender in these cases is not a financial institution but the individual members of the ROSCA. Non-performing loans (NPLs) are loans that are in default and these can impact negatively on economic growth and increase the chance of financial crisis (Kelly & McCann, 2016; Li & Ng, 2013).

For modelling prediction of late payment and loan default, credit risk models such as Ohlsons's O-score, Altman's Z-score and the Merton distance to default model (MDDM) are regularly used by financial institutions (Castagnolo & Ferro, 2014).

Statistical algorithms such as multiple linear regression (MLR) and LR are also regularly used for prediction (Bardak, Tiryaki, Bardak, & Aydin, 2016; de Carvalho, 2015). For the past 20 years, machine learning algorithms such as support vector machines (SVM) and ANN have increased in popularity for many areas of prediction (Krogh, 2008; Noble, 2006).

1.2 Research Problem

Company A is an Indian digital company that have created a platform for chit fund companies. This project used transactional, family connection, demographic and referral history data provided by Company A to predict the factors that contributed to default.

One of the main challenges for chit companies is to decide if a new member can be admitted into a chit group. Prospective new members must provide personal ID information, photographs, reference letters and bank statements (if available). However, most chit funds still operate on a trust basis, chit funds trust their members not to default on their payments and the members trust the chit fund to keep their money safe.

Company A currently use domain knowledge of chit fund employees to determine the risk of a loan default or late payment of a chit fund subscriber. The problem is that this domain knowledge can take years to develop and is lost with employee turnover. To overcome this issue Company A need an automated solution that determines the factors that contribute to default so they can use this knowledge to screen potential chit fund customers more accurately.

1.3 Research question

Can an artificial neural network model outperform a logistic regression model when predicting risk of default and late payment in chit funds using demographic, transactional, referral history and family ties data?

Research Hypothesis

H_0 :Considering demographic, transactional, referral history and family connection factors, an artificial neural network model will not statistically enhance (using 95% CI), the prediction of risk of default as compared to considering the same factors with a logistic regression model using area under the ROC curve and balanced accuracy.

H_a :Considering demographic, transactional, referral history and family connection factors, an artificial neural network model will statistically enhance (using 95% CI), the prediction of risk of default as compared to considering the same factors with a logistic regression model using area under the ROC curve and balanced accuracy.

1.4 Research Aims and Objectives

The aims and objectives of the research are:

i. Critically analyse the literature regarding chit funds, default, prediction models and suitable configurations for the prediction models.

ii. Statistically analyse the factors that can predict default in chit funds.

iii. Evaluate the performance of an ANN and LR model for predicting default and late payment using real world chit fund data.

iv. Provide empirical evidence to accept or reject the null hypothesis based on determining if the difference in results is statistically significant using a 95% confidence level.

1.5 Research Methodologies

The research consisted of both a literature review phase and an experimental phase. The literature review phase was done first and consisted of a review of the related literature to determine what chit funds were, the most appropriate methods to predict default and

typical issues encountered. The experimental phase used real world data and consisted of these steps:

i. Principle component analysis was used to reduce the feature set so that 95% of the variance was explained.
ii. Parameter tuning was evaluated using K folds cross validation and a hold-out sample was used to assess and compare model performance.
iii. The model with the highest ROC and balanced accuracy on the validation dataset was considered the best model. The F1 score was also noted. The next step taken was to determine were the differences seen statistically significant and a paired t test was used. Bonferroni's adjustment was made to the original p value when comparing multiple experiments. When comparing the results from a single experiment the p value was kept at 0.05. The result of the paired t test determined if the null hypothesis could be rejected.

1.6 Scope and Limitations

The scope of this research included the statistical analysis of factors that predict default in chit funds. This research used demographic, family connection, transaction and referral history data to determine which machine learning algorithm (LR or ANN) achieved a statistically significantly higher ROC and higher balanced accuracy when predicting default in chit funds.

In terms of limitations of the study, a number of years of data from one Indian financial company was used for this study so only limited generalisations outside of this can be made. Other limitations include the small number of cases of default in the data and sampling techniques were used to overcome this issue.

1.7 Document Outline

The literature review is presented in chapter 2 along with similar studies and empirical evidence. Chapter 3 will cover the experimental design and methodology used. Chapter 4 reviews the implementation and results and explains the tools and algorithm

parameters used. Analysis, evaluation and discussion are presented in chapter 5 which includes a comparison to other similar studies. The conclusions, contributions and future work are covered in chapter 6.

2 LITERATURE REVIEW

This chapter presents the literature review. ROSCAs and group lending are discussed and that is followed by explanation of what default is and prediction methods related to it. The next section presents Machine learning algorithms and the following two sections focus on ANN and LR. This is followed by a discussion on how models are evaluated, typical issues seen in modelling and how to overcome them. The final section deals with information about the modelling tool used.

2.1 Chit funds and default

2.1.1 Rotating and Saving Credit Association (ROSCA)

A ROSCA is "an association formed upon a core of participants who make regular contributions to a fund which is given, in whole or in part, to each contributor in rotation" (Ardener & Burman, 1995). In the 60's ROSCAs were viewed by anthropologists as a useful intermediate stage for developing economies where individuals learnt the benefits of goal orientated saving (Geertz, 1962). Now ROSCAs are recognised both for the financial and social support they can bring to group members (Salamon, Kaplan, & Goldberg, 2009).

ROSCAs are similar in concept to group lending but in group lending (also called joint liability lending) the members borrow as a group from a microfinance institution, the members are self-selected and all the members of the group are jointly liable for the loan so if one cannot pay, the rest pay instead (Al-Azzam, Carter Hill, & Sarangi, 2012).

2.1.2 Definition of Default

Late payment is normally measured in the number of days late that a customer makes a payment. Kelly & McCann's, (2016) study shows that late payment is often called days past due and default in financial institutions is defined as 90 days past due. From January 2018 financial institutions used a new accounting standard called the International

Financial Reporting Standard 9 (IFRS 9) which also defines default as 90 days past due[1]. However it included new stages of impairment with a default of 90 days defined as a stage 3 impairment and a default of 30 days defined (along with other considerations) as a stage 2 impairment.

No discussion on default would be complete without explaining expected credit loss (ECL). Featherstone, Roessler, & Barry (2006) assert that PD is the probability of default, EAD is the exposure at default, LGD is the loss given default and ECL is defined as:

$$ECL = PD * LGD * EAD \quad [1]$$

Here is a worked example:

Original home value	200,000
Loan To Value	80%
Loan Amount	160,000
Outstanding loan	140,000
Current home value	130,000
liquidation cost	5,000
LGD	$\frac{size\ of\ likely\ loss\ on\ the\ exposure}{EAD}$
EAD	-140,000
Received when house sold	130,000
liquidation cost paid	-5,000
Size of likely loss on exposure	-15,000
LGD	$\frac{-15{,}000}{-140{,}000}$
LGD	0.11
PD (making this assumption as house in negative equity)	50%
ECL	PD * LGD * EAD
ECL	7,500

[1] EBA Report on results from the 2nd EBA IFRS9 IA.pdf. (n.d.). Retrieved from https://www.eba.europa.eu/documents/10180/1720738/EBA+Report+on+results+from+the+2nd+EBA+IFRS9+IA.pdf

2.1.3 Predicting Default and late payment

There are many studies that measure default and late payment (Kolodinsky & Roche, 2009; Li & Ng, 2013). However the author found no studies on measuring default and late payment for chit funds or ROSCAs. Group liability lending studies were the closest studies found. Ahlin & Townsend, (2007) found that in group lending cooperation (which is measured by the amount of sharing in the group) and group members being related were both negatively associated with repayment.

Credit risk models such as Ohlsons's O-score, Altman's Z-score and the Merton distance to default model (MDDM) are regularly used by financial institutions to measure default (Castagnolo & Ferro, 2014). Castagnolo & Ferro (2014) reported that studies do not agree on which of these financial models is the best model to predict default. They tested these models and found the O-score achieved the best results because it had the best balance between type I and type II errors (which will be explained in section 10 in this chapter). Statistical models such as ordinary least squares (OLS) are also used to predict default (Kolodinsky & Roche, 2009). OLS is the typical model used for simple linear regression and fits a line to the data points by ensuring the distance between the data points and the line is minimised.

There are many studies on factors that affect mortgage default and some of these found that credit history, credit score and loan to value ratio have been shown to effectively predict mortgage default or late payment (Li & Ng, 2013; Kolodinsky & Roche, 2009). Credit score systems are usually better predictors of default than subjective systems especially when they include local economic factors and individual events (Li & Ng, 2013) or amount of mortgage down payment and income (Kolodinsky & Roche, 2009). Credit scoring is the process of scoring customers usually using a model to determine the risk of lending to that customer (Bellotti & Crook, 2009). Different statistical and machine learning processes have been used since the 1930s for credit scoring (Thomas, Edelman, & Crook, 2002, Section 1.3).

2.2 Machine learning for prediction

2.2.1 Machine Learning Algorithms

Classification is the process of assigning a class or label to new instances during the data mining process i.e. predicting the correct class that the instance belongs to (Yadav & Shukla, 2016). A machine learning model is used to do this and the model will first need to be trained and validated on labelled data e.g. when emails are noted as spam that's giving them a label or category. Mallapragada, Jin, Jain, & Liu (2009) defines supervised learning as training a model with labelled data and non-supervised learning as training a model with unlabelled data, which is often done using clustering. Machine learning is when a model learns to perform a task by using a training set of examples (Louridas & Ebert, 2016). It has been around since before the 1970's and consists of both supervised and unsupervised learning. It's used in many areas such as fraud detection, pattern recognition and image classification.

There are many different machine learning algorithms that can be used for the machine learning process (Louridas & Ebert, 2016). Nonparametric machine learning algorithms such as artificial neural networks, Bayesian neural networks and support vector regression have been shown to outperform parametric models such as liner regression in benchmark tests when predicting credit default swaps spread (Son, Byun, & Lee, 2016) and credit default prices for certain time series predictions (Gunduz & Uhrig-Homburg, 2011). A parametric model has a fixed number of parameters and tends to run faster than a non- parametric model but it makes assumptions about the data such as that the data is linearly related (Cai, 2014[2]). A non-parametric model has a flexible number of parameters and makes less assumptions about the data so is less restrictive. See figure 2.1 for a diagram showing the different machine learning algorithms and machine

[2] Cai, E. (2014, January 15). Machine Learning Lesson of the Day – Parametric vs. Non-Parametric Models. Retrieved May 5, 2018, from https://chemicalstatistician.wordpress.com/2014/01/14/machine-learning-lesson-of-the-day-parametric-vs-non-parametric-models/

learning approaches. Louridas & Ebert, (2016) defines supervised learning as including both parametric and non-parametric models that are trained on labelled data. Unsupervised learning is training a model on unlabelled data and involves looking for unknown patterns in the data.

Bellotti & Crook (2009) assert that support vector machines (SVMs) have been used for credit scoring since the late 1990's and when compared to other methods in a study achieved good performance for predicting credit card default. However there are numerous prediction studies that show LR or ANN achieved the highest prediction accuracy and some of these studies were applied to predicting consumer loan default and late payment (Bardak et al., 2016; Son, Byun, & Lee, 2016; Stallkamp, Schlipsing, Salmen, & Igel, 2012).

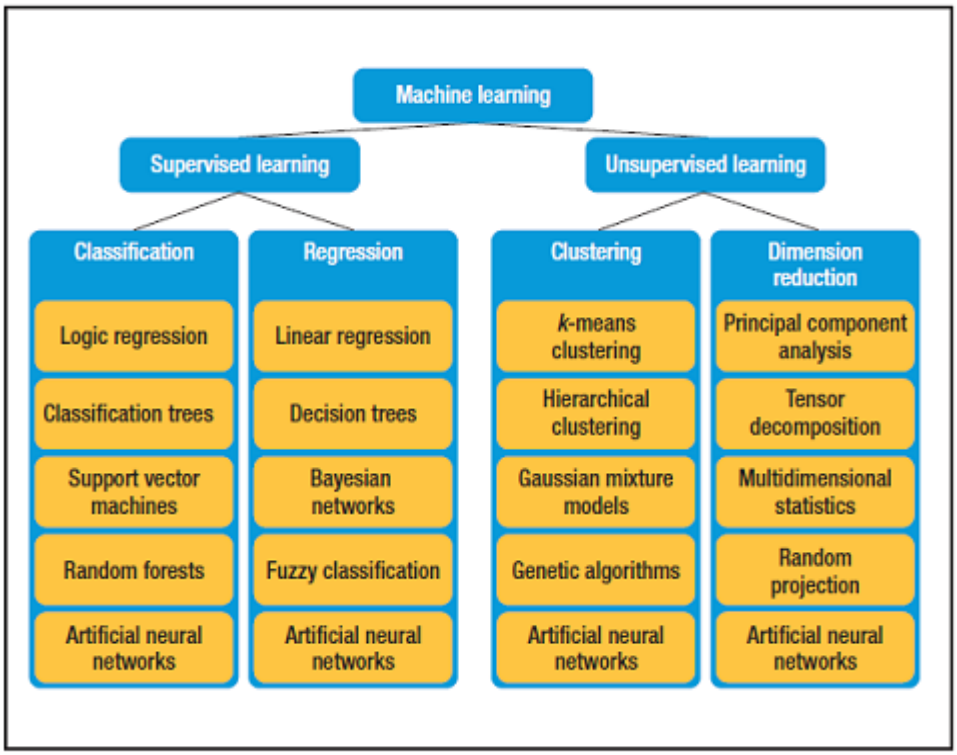


Figure 2.1: Machine Learning Approaches

Diagram showing the different machine learning algorithms and machine learning approaches (Louridas & Ebert, 2016)

In a number of studies ANN achieved the best predictive performance, for example Son, Byun, & Lee, (2016) who used root mean squared error (RMSE) as the performance

measure and Tsai, Lin, Cheng, & Lin, (2009) who used misclassification and sensitivity as the performance measures.

2.2.2 Artificial Neural Networks

ANN algorithms work by mimicking the human learning processes and consists of layers and nodes similar to neurons in the brain (Were, Bui, Dick, & Singh, 2015). There are three types of layers and these are the input layer, the hidden layer and the output layer. The input layer is the first layer and this has nodes that send the values to the hidden layer. The hidden layer is between the input and output layer and this does a partial classification of the inputs and sends it on to the output layer (Krogh, 2008). The output layer gathers the partial classifications and uses these to determine a final classification. See figure 2.2 for an example of an artificial neural network.

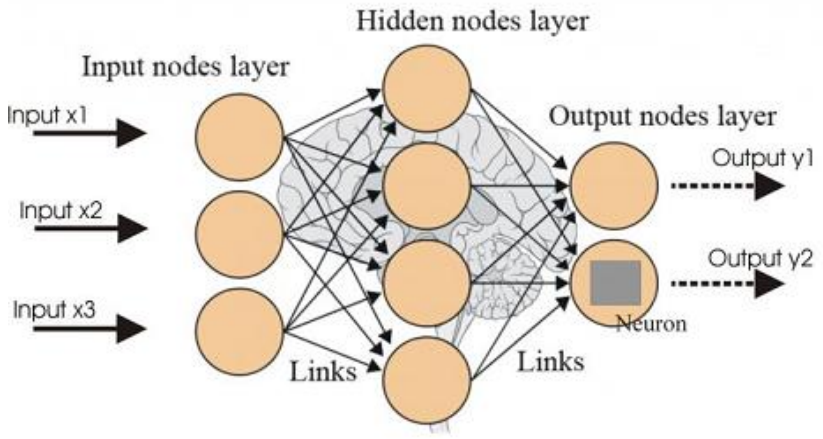


Figure 2.2: Example of an artificial neural network

(Says, 2016)[3]

[3] Says, D. V. R. M. (2016, August 3). The Evolution and Core Concepts of Deep Learning & Neural Networks. Retrieved May 1, 2018, from https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/

There are a number of different neural network algorithms and the multi-layer perceptron with back propagation is usually considered the most popular (Were, Bui, Dick, & Singh, 2015; Zekić-Sušac, Šarlija, Has, & Bilandžić, 2016). Krogh (2008) asserts that the back propagation algorithm uses feed forward neural networks with continuous output. At the start of the training all the weights in the network are assigned small random numbers. Then for each input value the network gives an output and the squared difference between the output and the required output is determined and is the error. The sum of all these differences for the training data is the total error of the network and the smaller this number, the better the network. The error is sent back through the network enabling the model to learn from its mistakes (Gschwind, 2007).

When modelling an ANN, the three main parameters that are configured are the number of neurons in the hidden layer, the number of hidden layers and the initialisation parameters (Krogh, 2008). Finding the number of hidden neurons is important for efficiency and its recommended to find this through experimental tests (Bardak, Tiryaki, Bardak, & Aydin, 2016). Zekić-Sušac et al. (2016) study ran tests using between one and thirty neurons and found the least error using 13 hidden neurons, while Tsai et al. (2009) achieved the best results with 4 hidden neurons. Bardak et al. (2016) study asserted that the correct number of hidden layers is important for efficiency and both they and Tsai et al., (2009) used one hidden layer in their experiments. One of the main purposes of training is to determine the optimum initialisation parameters including learning weights and the optimum value is normally determined by experimentation (Bardak et al., 2016).

2.2.3 Logistic Regression

LR is used when you want to predict a discreet or binary outcome, which is where there are a certain set number of outcomes (usually two) such as the customer will default or the customer will not default (Gschwind, 2007). Zekić-Sušac et al. (2016) concurs with this definition when they say that logistic regression is used when the outcome is categorical and that the outcome can be binary or multinomial. However they also point out that it can also be used when the outcome is ordinal which means the outcome is

ranked in a certain order. Linear regression is used when you want to predict a continuous outcome such as the height of an individual (Gschwind, 2007). LR is one of the most popular statistical modelling techniques for predicting financial distress. One of the advantages of LR is that it doesn't require the data to be normally distributed and both dependant and independent features do not have to be linearly related (Bardak et al., 2016; Qing Cao, Parry, & Leggio, 2011). Linear regression which is related to logistic regression has these data requirements so is more restrictive.

Sainani (2014) study shows that logistic regression works by trying to fit a line with an intercept and a slope to the data. The line is not fit directly to the data, instead it uses a transformation of the outcome called the log odds or Logit. This function shows how it is related to probability:

$$Logit = ln\left(\frac{p}{1-p}\right) \quad [2]$$

Where p is probability and ln is the natural logarithm.

Here is a worked example:

Let p be the Probability of passing a course	0.525
1-p	0.475
Logit	Ln(0.525/0.475)
Logit	0.100

Calculus is used to find the equation of the best fit line and an example of a logistic regression model to determine likelihood to pass a course is:

$$Logit\ (passing\ a\ course) = -3.5 + 0.26 * homework\ hours\ per\ week$$

Where -3.5 is the y intercept and 0.26 is the slope of the line. The slope shows that for every 1 hour of homework per week, the log odds of passing the course increases by 0.26. The slope is also called the beta coefficient or odds ratio and in this example the odds ratio for homework is calculated as:

$$\mathrm{OR}_{\mathrm{homework}} = \exp^{0.26}$$

Where exp is the exponent and in this example the odds ratio is calculated to be 1.3. This means that for every hour of homework per week, the odds of passing the course increase by 30%. If the model contained interaction terms i.e. a model contains more than one predictor and the predictors affected each other either positively or negatively, then determining the odds ratio is more complex. For example if gender was included in the

model then there would be two odds ratios for homework time, one for males and another for females.

2.3 Model Performance and performance issues

2.3.1 Evaluating Model Performance

Goldblatt, You, Hanson, & K Khandelwal (2016) study proposed that performance is measured on unseen data using one of these techniques: 1) A hold-out validation sample that is not used to train the model, 2) K folds cross validation which is where the dataset is divided into k parts and goes through k iterations of testing, 3) Bootstrapping which is where the dataset is sampled uniformly with the data and using replacement.

When evaluating model performance many researchers such as Tsai et al. (2009) and Bapna, Goes, Wei, & Zhang (2011) used measures such as classification accuracy and root mean squared error (RMSE). The accuracy of a classifier is the probability that it will classify a random set of samples correctly (Goldblatt et al. 2016). Pandey, Das, Pan, Leahy, & Kwapinskia (2016) defines mean squared error as:

$$MSE = \left(\sum_{i=1}^{n} \frac{(y_p - y_o)^2}{n}\right) \quad [3]$$

Where y_p is the mean predicted value, y_o is the actual value and n is the number of values. So in essence it is the sum of the differences between the predicted and the actual value divided by the number of the values. RMSE is the squared root of MSE.

Other researchers such as Kim & Kang (2016) and Ząbkowski & Szczesny (2012) proposed area under the ROC curve as a suitable measure. This is often called AUC or AUROC. ROC stands for receiver operator characteristic and is a popular classification performance method (Wray, Yang, Goddard, & Visscher, 2010). An ROC curve plots the true positive rate (also called TP, TPR or sensitivity) against the false positive rate (also called FP, FPR or 1- specificity) for different cut off points. True positives are when a classifier predicts an outcome such as a patient has lung cancer and that patient does actually have lung cancer. False positives occur when a classifier predicts an outcome such as a patient has lung cancer but in reality the patient does not have lung

cancer. See figure 2.3 for an example of an ROC curve and the actual curve is the red line. High accuracies will have a curve line that is close to the upper left hand corner.

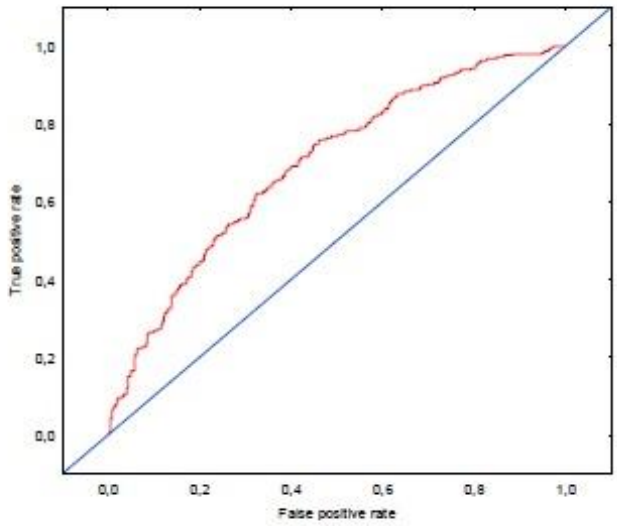


Figure 2.3: Example of an ROC curve

Shows the true positive rate against the false positive rate (Wray et al., 2010)

Bardak et al. (2016) reported that the mean absolute percentage error (MAPE) and regression coefficient (R^2) are also used. R^2 is also called goodness of fit and R^2 of 0.91 means that in a regression model such as:

$$Height = sex + weight * sex$$

91% of the variation of the response (in this case height) can be explained by the predictors (in this case sex and weight). A R^2 value of 0.90 or greater shows that the model is able to explain a large amount of the variation in the response while a R^2 of 0.82 to 0.90 would be a model that has relatively poor performance.

MAPE is considered an important comparison measurement as it is an easy to understand generic percentage term. A MAPE of less than 10% is considered a high accuracy prediction, a MAPE of between 10 and 20% is considered a good predictor, a MAPE of between 20 and 50% is considered a reasonable predictor and a MAPE of greater than 50% is an inaccurate predictor.

No discussion on performance would be complete without explaining a confusion matrix (see figure 2.4 for an example of a confusion matrix). Deng, Liu, Deng, & Mahadevan, (2016) study says that a confusion matrix is a 2 dimensional matrix containing results showing the actual and predicted classifications. The actuals are normally shown along the side and the predicted on the top. Using the same example as before, a true negative is where a classifier predicts that a patient does not have cancer and that is in fact the case and a false negative is where a classifier predicts that a patient does not have cancer when in fact the patient does.

Accuracy is a measure of the total number of correct predictions divided by the total number of predictions:

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad [4]$$

The misclassification rate is 1 - accuracy or the total number of incorrect predictions divided by the total number of predictions.

$$\text{Misclassification} = \frac{(FP+FN)}{(TP+TN+FP+FN)} \quad [5]$$

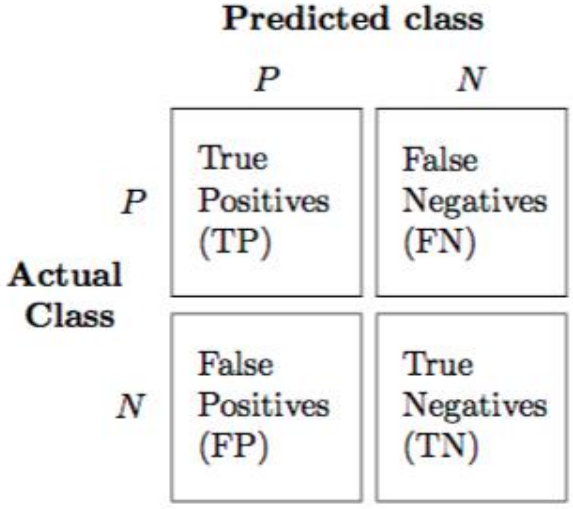


Figure 2.4: Example of a confusion matrix

For a 2 classification problem (Raschka, 2014[4])

[4] Raschka, S. (2014, January 1). Confusion matrix - mlxtend. Retrieved May 6, 2018, from https://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/

F1 score is another useful metric and is often defined as the harmonic mean of precision and recall (Zhang, Wang, Zhao, & Wang, 2015). Precision is what proportion of true positives were actually correct and is defined as:

$$Precision = \frac{TP}{TP+FP} \quad [6]$$

Recall is what proportion of actual positives were identified correctly and is defined as

$$Recall = \frac{TP}{TP+FN} \quad [7]$$

F1 score is widely used in text classification as a performance metric because it can be more informative and useful in class imbalance situations and these often occur with text classification.

A paired t Test compares means from two different sets of results from the same set of data and indicates if the means are different and how statistically significant the difference is (Salzberg, 1997). A one tailed test is where values from the alternative hypothesis are tested to determine are they greater or less than values from the null hypothesis (Pereira & Leslie, 2009). A two tailed test is where a test is made to check for differences in either direction but the direction is not specified beforehand.

Recommendations for performance include regularly assessing the performance of the models to ensure they include relevant measures (Kolodinsky & Roche, 2009). Yadav & Shukla (2016) claim that for classification the highest accuracy is the best performance measure provided the time to train is reasonable.

2.3.2 Overfitting

Van der Aalst et al. (2010) defines overfitting as when models "allow for only that which has been observed". In essence this means they have poor generalisation capabilities and the results vary greatly when new data is introduced to the model and their prediction accuracy reduces. This can happen when too much data is used to train the model and is mitigated by ensuring that enough data is used to test the model, either by using a hold-out validation sample that is not used to train the model or using a process such as k folds cross validation which will be explained in the next section. When using a hold-out sample different proportions are sometimes used to determine the optimum

performance, such as 70% for training and 30% for validation for one test and then 60% for training and 40% for validation for the next test. See figure 2.6 for an example of overfitting.

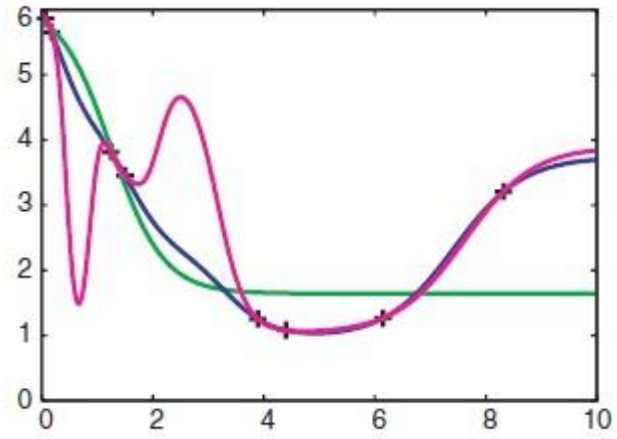


Figure 2.5 Example of overfitting

A model with 8 data points represented by the small black lines, the green line is a model that under fits the data that only passes through three of the data points, the purple line is an example of overfitting the data and the blue line is the best model that makes good approximations between the data points (Krogh, 2008).

2.3.3 K Folds Cross Validation

K folds cross validation is a popular technique used to evaluate performance of classification (Wong & Yang, 2017; Yadav & Shukla, 2016). It works by randomly dividing a dataset into k folds of approximately equal size. Every fold is used to test the model with the other k-1 folds are used to train the model for each iteration and this is repeated until all folds have been used to test the model and the other folds used to train the model. The accuracy is achieved by averaging across the k iterations (Goldblatt et al., 2016). See figure 2.7 for a diagram that explains the process.

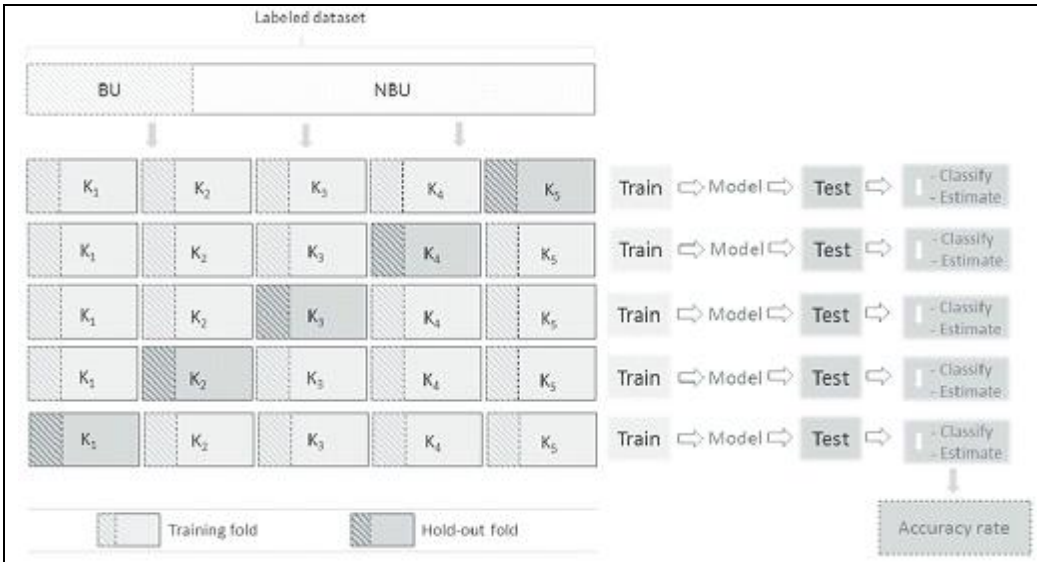


Figure 2.6: Example of k fold cross validation

In each iteration the hold out fold (which is the dark square) is left out of the training sample and used for testing instead (Goldblatt et al., 2016).

For large datasets (i.e. datasets with more than 100,000 instances), k folds cross validation has been shown to achieve higher accuracy than using a hold-out validation sample (Yadav & Shukla, 2016). Yadav & Shukla (2016) claim that the in k folds cross validation the data is normally split into two parts i.e.k=2. However studies such as Goldblatt et al., 2016 recommend using k of 5 or 10 to get less biased estimates. For large datasets using a hold-out sample to test is the typical recommended choice (Yadav & Shukla, 2016). However Yadav & Shukla (2016) study shows that for large datasets the time trade-off for using k folds validation over hold out validation is worth the improvement in accuracy it brings.

2.3.4 Type I and Type II errors

Pereira & Leslie (2009) define a hypothesis test as a framework that enables testers to make decisions based on a set of statistical methods. It is used to determine the probability that a given statement or hypothesis is true. There are a number of steps involved that start with the formulation of a null hypothesis, then the test statistic and a suitable significance level are chosen. This is followed by a calculation of the probability and lastly the probability is compared against the significance level to determine if the null hypothesis can be rejected.

A Type I error occurs when the null hypothesis is rejected by a statistical test when it shouldn't have been (Nicholson, 2014). This type of error is also known as false positive or the significance of the test. Nicholson (2014) defines a Type II error as occurring when the null hypothesis is not rejected when it should have been, while (Matuschek, Kliegl, Vasishth, Baayen, & Bates, 2017) defines a Type II error as "a loss in statistical power to detect the significance of fixed effects". This type of error is also known as false negative. A decrease in Type I error (often known as α and usually set at 0.05 for scientific experiments) will lead to an increase in Type II error (Matuschek, Kliegl, Vasishth, Baayen, & Bates, 2017). In essence this means that a Type II error (often known as β) is inversely proportional to Type I error i.e. as one increases the other decreases and vice versa. A Type II error is related to statistical power by:

$$Statistical\ Power = 1 - \beta \quad [8]$$

The Bonferroni adjustment is typically used to adjust the statistical significance of an experiment that consists of multiple tests (Salzberg, 1997). It consists of determining an adjusted value for statistical significance based on the number of tests in an experiment and is determined by

$$\alpha \geq 1 - (1 - \alpha^*)^n \quad [9]$$

Where α is the statistical significance, $\alpha^*$is the adjusted statistical significance level and n is the number of tests in the experiment. For example if α is set to 0.05 and there are 16 tests in the experiment then α^* should be 0.003.

2.3.5 Imbalanced Classes

Class imbalance occurs in datasets when there are many more examples of one class than the other (Blagus & Lusa, 2012; García & Herrera, 2009). Blagus & Lusa (2012) assert that class imbalance can lead to poor prediction accuracy for the minority class. Imbalanced data occurs in many areas such as fraud detection and medical identification (García & Herrera, 2009; Fernandez, Garcia, Herrera, & Chawla, 2018). Typically datasets have binary classifications i.e. just two classes and in imbalanced datasets the ratio of one class to the other can be as low as 1:100 or lower. For ratios of 1:100, the error of ignoring one class could be only 1% so its effect can be missed. García & Herrera, (2009) study says that imbalance ratio is the relation between the majority class and the minority class and is defined as

$$IR = \frac{N^-}{N^+} \quad [9]$$

Where IR is the imbalance ratio, N^- is the total number of occurrences of the majority class and N^+ is the total number of occurrences of the minority class. A dataset is imbalanced when IR is greater than 1 and datasets with an IR of 9 or above are considered to have a high IR.

The three main methods to deal with this are (García & Herrera, 2009):

1) Internal approaches at the algorithm level – they impose a bias on the minority class, i.e. class with the smaller set of examples, or use weights to improve the prediction performance.
2) External approaches - these are done at the data level and involve resampling the data in order to decrease the effect caused by the imbalance of data. This is usually done at the pre-process stage.
3) Boosting approaches - use weighting and replicate minority class instances to improve the performance of weak classification algorithms at the pre-process stage. The weighting aims to force the algorithm to give more attention to the minority class. They usually consist of a number of classifiers and the two main examples are SMOTEBoost and DataBoost-IM.

Brodersen, Ong, Stephan, & Buhmann (2010) argue that for imbalanced datasets accuracy can be a misleading performance measure. Imbalanced datasets can result in a classifier that is biased towards the class with the most instances. For highly imbalanced datasets if the classifier classified every instance as belonging to the majority class it could still achieve a very high accuracy. While sampling techniques can help to mitigate this they can still result in an optimistic accuracy been reported. One way to deal with this is to report the balanced accuracy instead. Balanced accuracy is the average accuracy obtained on either class so in essence it is

$$Balanced\ Accuracy = 0.5 * \left(\frac{TP}{TP+FP} + \frac{TN}{TN+FN}\right) \quad [10]$$

Classifiers that perform well on both classes have a similar accuracy and balanced accuracy. Kuhn (2016) study recommends to use another statistic called Kappa for imbalanced data or datasets where cross validation is used. They define Kappa as the measure of its agreement "relative to what would be expected by chance" and it is suitable for categorical data only. A positive Kappa shows there is an association

between the observed and predicted data. A value of 0 indicates a complete lack of agreement and a value of 1 shows complete agreement. Negative Kappa's can occur and indicate a negative association between the observed and predicted data. Kappa would be 0 in the case of an imbalanced dataset with 5% of one class label and 95% of the other, which predicted all instances to be the majority class.

2.3.6 Sampling Techniques

Down sampling (also called under sampling) will randomly sample a dataset so that all classes have the same frequency as the minority class and so it reduces the set of instances (García & Herrera, 2009). The aim is to balance datasets, to improve the classification accuracy and reduce bias for the minority class instances. Blagus & Lusa (2012) reported that generally under sampling improves prediction accuracy while oversampling does not. They also said that under sampling works by removing a subset of samples from the majority class so there is no class imbalance. Under sampling also has the benefit of reducing the training time as the set of instances to work with is reduced to match the minority class (Fernandez et al., 2018). Problems associated with under sampling are that the variance of the classifier is increased, useful examples for training may end up being discarded especially for high imbalance ratios and the generalisability of the classifier can be affected.

Up-sampling (also called oversampling) is where the minority class instances are replicated in the dataset so they ratios of one class to another are equal (García & Herrera, 2009). One of the most popular techniques is SMOTE which stands for Synthetic Minority Oversampling Technique and its aim is to improve random oversampling (Blagus & Lusa, 2012). It works by generating synthetic samples from the minority class using the information available in the data. In tests Blagus & Lusa (2012) found SMOTE to be less effective than random under sampling for high dimensional data with most classifiers. However Fernandez et al. (2018) study says that SMOTE is now the standard method to deal with imbalanced data because it has a simple design and robust when applied to different datasets. Another oversampling technique is Random Oversampling Examples (ROSE) that generates synthetic balanced samples but generally results is less accurate results than SMOTE (Lunardon, 2015).

2.4 Feature selection and model tools

2.4.1 Model Features

In terms of features to use in the model, Sharma & Zeller (1997) found some interesting results such as family ties, measured by how many relatives were in the group, have a negative impact on repayment of group loans and religiousness measured by how many times a day group members prayed had a positive impact on repayment rates. They also found that groups with a high percentage of phone ownership had a positive correlation with repayment rate. Ahlin & Townsend (2007) found that family ties were negatively associated with repayment rate in group lending and they found that areas that instigated a village wide shut down to a defaulter had a positive association with repayment rate. Bellotti & Crook (2009) used Support vector machines to find the most useful features to predict default and they also found that the type of product was an important predictor of default.

2.4.2 Correlation

Park & Allaby (2017) define correlation as a "measure of association between two variables". Two variables can be positively correlated which means as one increases the other also increases or negatively correlated, which is where one variable increases the other variable decreases. Pearson's correlation measures the linear relationship between two variables. The values are measured from 0 to 1 with values closer to 0 indicating a weak association and values closer to 1 indicating a strong association.

2.4.3 Principle Component Analysis

Principle Component Analysis (PCA) is used to reduce the number of features in a dataset while still capturing as much information as possible (Hooper, 2012; Polyak & Khlebnikov, 2017). The aim is to explain the majority of the variation while using a much smaller feature set. Polyak & Khlebnikov (2017) study reports that PCA is often used in pattern recognition, neurobiology and risk management. PCA works by finding the eigenvectors associated with the largest eigenvalues of a hyperplane, which are known as the principle components. Eigenvalues are created from a covariance matrix.

Allaby (2010) describes an eigenvector as the "loading of a feature on a component" which is measured by the correlation between the feature and the new component. See figure 2.8 for a visual example of PCA. The first principle component will explain the largest variability in the data and the second will explain the second largest and so on. In PCA the original features are replaced with uncorrelated features or components that are orthogonal to each other (Upton & Cook, 2014). PCA has some limitations such as it is sensitive to outliers and sensitive to uncertainties in the underlying data.

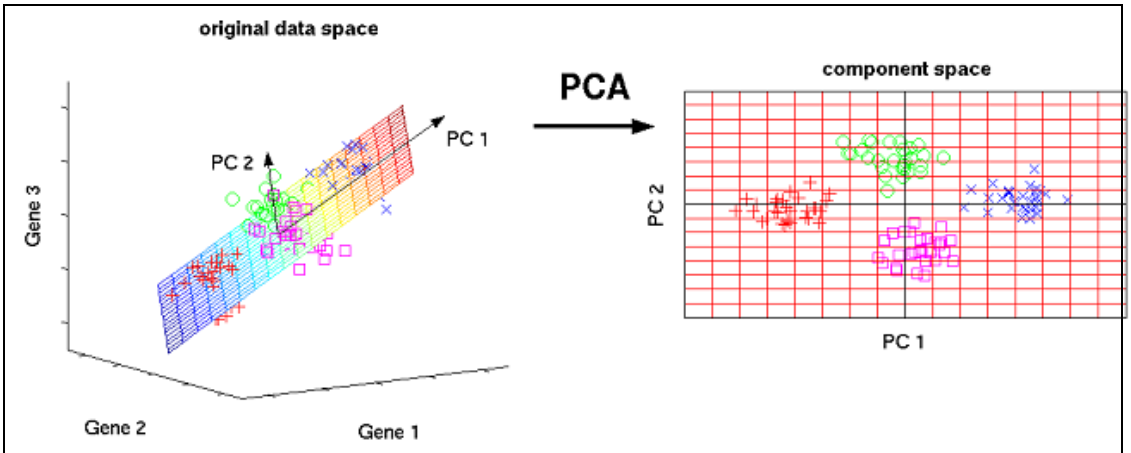


Figure 2.7: Example of PCA

In the original data space and the component space (Scholz, n.d.[5]). You can see PC1 is orthogonal to PC2.

A scree plot is often used to visually show the proportion of variation explained by the principle components (Upton & Cook, 2014). See figure 2.9 for an example of a scree plot.

[5] Scholz, M. (n.d.). PCA - Principal Component Analysis. Retrieved April 29, 2018, from http://www.nlpca.org/pca_principal_component_analysis.html

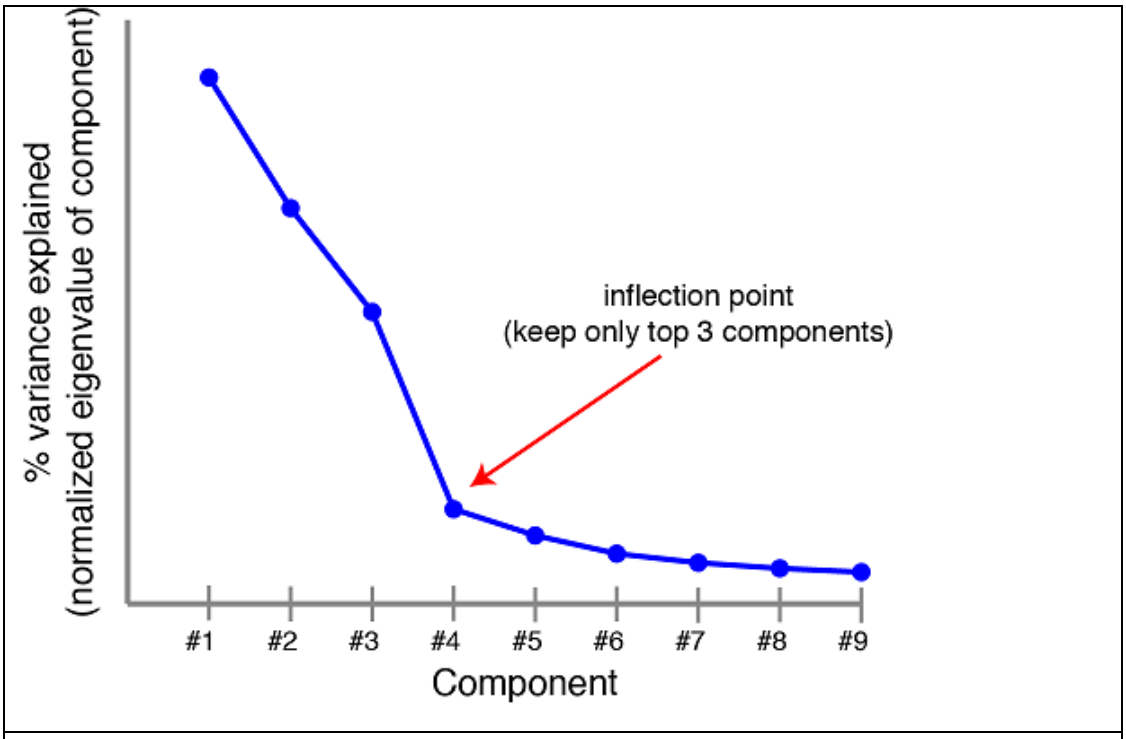


Figure 2.8: Example of a scree plot

In this example the first three components explain the majority of the data and the inflection point occurs after that (Williams, 2016).

Exploratory factor analysis (EFA) is a related technique however in EFA the aim is to find common factors by finding the latent structure of the dataset and to explore underlying theoretical constructs (Hooper, 2012).

2.4.4 Caret Package

The caret package is one of the most popular machine learning packages in R which is a statistical tool. Kuhn (2008) defines caret as classification and regression training and confirms that it contains many tools to develop predictive models. Its aim is to simplify the process to build different predictive models, to enable tuning of many of the parameters in the models and to build a framework that can be easily extended to enable parallel processing. The package contains functionality for data splitting, pre-processing, and modelling and comparing model results. The function createDataPartition can be used to create stratified random splits in a dataset and creates these within each class.

Kuhn (2008) asserts that many models cannot run with predictors that have a single unique value, which are also known as zero variance predictors. When the data is partitioned into test and training sets this can also cause other features that were previously not zero variance to become zero variance or near zero variance, which can cause training to fail. Also when sampling techniques such as down sampling are used this can result in much less data been used and can cause features that had good variance to become near zero variance. There is a caret function called nearZeroVar that can return the index of a dataset column number that meets these criteria: 1) percentage of unique values is less than 20%, 2) ratio of the most frequent to the second most frequent is greater than 20.

Some models such as linear models and neural networks have poor performance when there is a high correlation between predictors i.e. when predictors are strongly associated (Kuhn, 2008). Principle component analysis removes correlations so can be used to reduce this affect. Once the set of predictors is finalised the values may need to be transformed. Some models such as support vector machines or neural networks need the predictors to be scaled and the pre-process function can be used to do this.

This chapter presented the literature review. It discussed group lending and ROSCAs and explained what default it and how is measured. Methods to predict default were discussed with the focus on two of the most popular current methods which are ANN and LR. Model performance and typical issues encountered such as overfitting and imbalanced data were discussed. Principle component analysis is used to reduce the feature selection to a manageable amount and that was presented along with information about the tool used to model the data. The next chapter will discuss the experimental design and methodology used.

3 DESIGN AND METHODOLOGY

This chapter presents the experimental design and methodology used. It describes the processes used, names some of the tools used for analysis and explains the main aim of each of the five steps.

3.1 Data Exploration

In Data exploration a data familiarisation exercise is completed and the data is searched for typical data quality issues such as date inconsistencies and categorical field inconsistencies. In this experiment, data exploration was done in two phases. The initial phase was done using Microsoft Excel and the second phase was performed using a data profiling tool called DataCleaner. The source files were in excel format and were opened in excel to undergo basic analysis such a visual check of the field names, the order of the features in the input files and the types of data in each field and the files were then converted into csv format. The main analysis was done in DataCleaner[6] and it looked for typical data quality issues such as inconsistency in field values.

3.2 Data Pre-processing

Data pre-processing is used to clean and transform the data so it is suitable for data modelling. Data pre-processing was carried out in two steps. Step one was completed in an open source extract, transform and load (ETL) tool called Kettle[7]. Here the categorical data was factorised and the source files were merged into one file. Company A advised that the data was inconsistent prior to 2010 so the data was analysed and all data that had a commencement date from 2010 was sent to the output file for modelling. The rest of the data was disregarded. The last thing that was done in Kettle was the creation of the two default features using data from a number of date features.

[6] DataCleaner is available at https://datacleaner.org/get_datacleaner_ce

[7] Kettle is available at https://help.pentaho.com/Documentation/5.3/0F0/0J0/030

The second part of the pre-processing was carried out using the CARET package in R Studio[8]. The data was checked for data quality issues such as null values and these were fixed. The data was broken up into training and validation datasets to enable parameter tuning using the training dataset and performance assessment using the validation dataset. The data was then centred and scaled to normalise it for the next stage which was principle component analysis (PCA). PCA was used to reduce the number of features to a more manageable form while still explaining most of the variance in the data.

3.3 Data Modelling ANN

The caret package in R studio was used for the data modelling element for ANN. Parameter tuning was used to tune the ANN model and the parameters tuned were the number of neurons in the hidden layer and the initialisation parameters. K folds cross validation was used in the training phase so part of the data for each the k folds was held out of the training fold and used for testing that fold. Sampling was used as the data was highly imbalanced and two different sampling techniques were used to determine the best one. Two separate models were created, one for each of the defaults.

3.4 Data Modelling LR

The caret package in R studio was also used for the data modelling element for logistic regression. The model was created using the most important components from the principle component analysis. Parameter tuning was used to tune the model and in this case the tuning done was on the number of iterations. Again two separate models were created, one for each of the defaults, a number of different sampling techniques were used and k folds cross validation was used.

[8] R studio is available at https://www.rstudio.com/products/rstudio/download/

3.5 Evaluation of Models

A number of statistical tests were run in R studio to determine the best model. The two key metrics were balanced accuracy and the AUC. The aim is to have balanced accuracy and AUC as high as possible. These tests were run on both the training and validation datasets but the comparisons were made on the results from the validation dataset. To determine if the differences between the models were statistically significant, a paired t test was run.

This chapter presented the design methodology for the experiment. Data exploration was done first to become familiar with the data and find typical data quality issues. Then the data underwent a pre-processing step to merge the data files and factorise the data. The data was now ready for modelling. Lastly the results from both of the models were compared and differences in the models calculated. The next chapter outlines the experimental implementation and results.

4 IMPLEMENTATION AND RESULTS

This chapter describes the data and the data quality issues encountered and how they were resolved. It then describes the implementation including parameter tuning and algorithms used. The results are then presented using tables and charts.

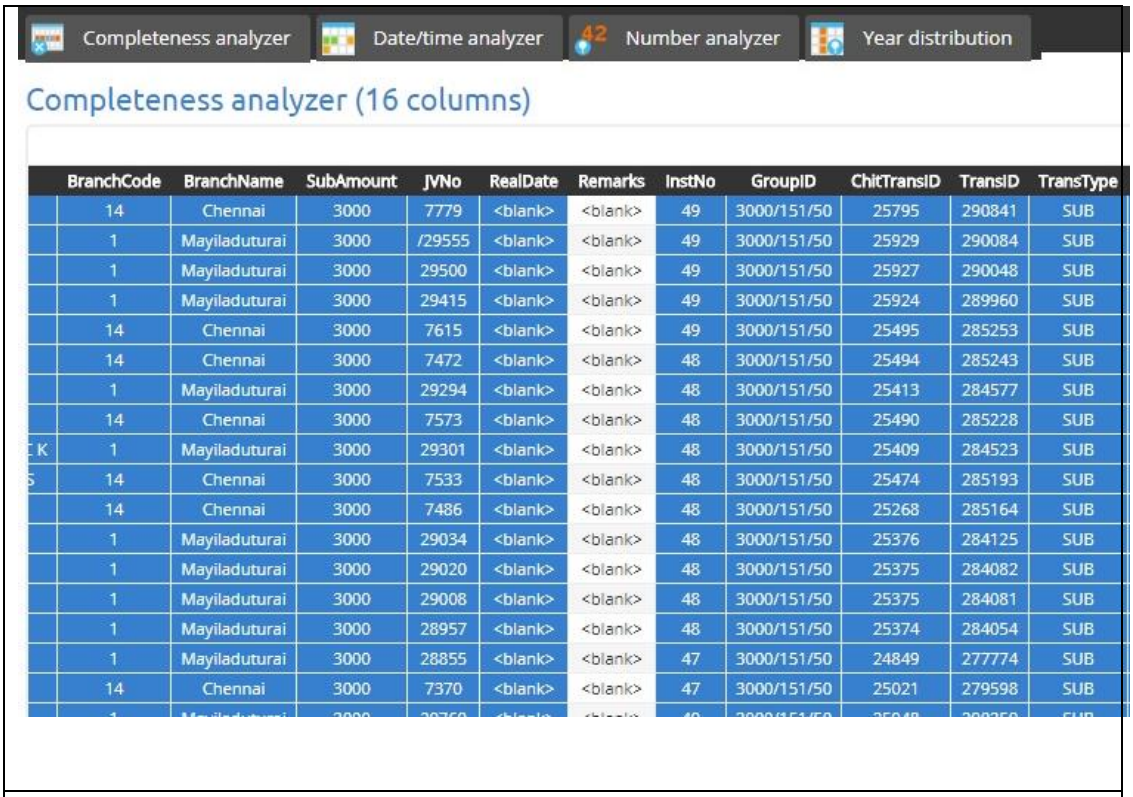
4.1 Data Overview

The data used for this project was real world data provided by Company A and consisted of data from January 2004 up to December 2017 when the data was provided. The data was initially spread over 110 features in 13 excel files, 8 of these files were transaction level files and the rest were master data such as auction master and subscriber master. After consultation with Company A it was agreed to use transactional data with a commencement date of 2010 onwards as the data was incomplete prior to 2010. See table A.1 in appendix A for a description of the data in the 13 files.

4.2 Data Quality issues encountered

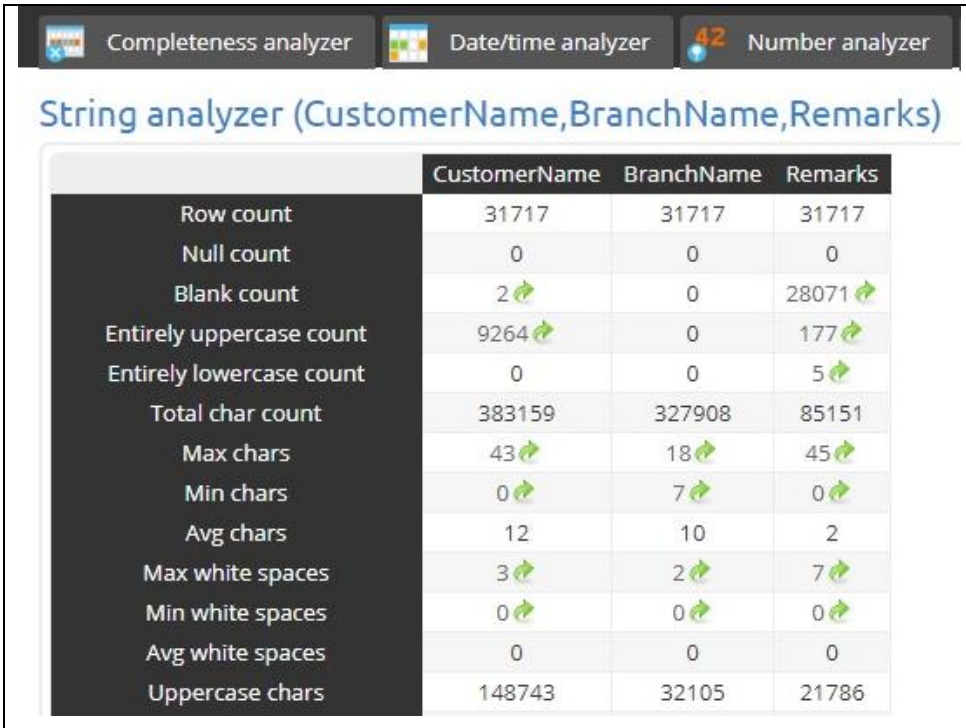
Microsoft excel was used to do the first checks on data quality, checking for inconsistent field values and missing values and no issues were found using this visual check.

Then a data quality and data profiling tool called DataCleaner was used to do a more in depth check for issues. This tool ran five checks on the data. The first was a completeness analyser that checked for incomplete records and output them to a file. See figure 4.1 for sample output that was run on 2500-111 to 2500-A.xlsx. This showed that there were 14,606 incomplete records and this was due to a number of features such as RealDate and Remarks containing missing or blank values and these were removed from the final dataset. The next check was a string analyser that was run on the three string features and sample output from 2500-111 to 2500-A.xlsx showed useful information like the number of instances with null values, the number of instances with blank values and the max characters on nominated string features. See figure 4.2 for sample output. These results show that there were no instances of nulls for customer name, branch name or remarks but there were 28,071 instances were remarks were blank.



BranchCode	BranchName	SubAmount	JVNo	RealDate	Remarks	InstNo	GroupID	ChitTransID	TransID	TransType
14	Chennai	3000	7779	<blank>	<blank>	49	3000/151/50	25795	290841	SUB
1	Mayiladuturai	3000	/29555	<blank>	<blank>	49	3000/151/50	25929	290084	SUB
1	Mayiladuturai	3000	29500	<blank>	<blank>	49	3000/151/50	25927	290048	SUB
1	Mayiladuturai	3000	29415	<blank>	<blank>	49	3000/151/50	25924	289960	SUB
14	Chennai	3000	7615	<blank>	<blank>	49	3000/151/50	25495	285253	SUB
14	Chennai	3000	7472	<blank>	<blank>	48	3000/151/50	25494	285243	SUB
1	Mayiladuturai	3000	29294	<blank>	<blank>	48	3000/151/50	25413	284577	SUB
14	Chennai	3000	7573	<blank>	<blank>	48	3000/151/50	25490	285228	SUB
1	Mayiladuturai	3000	29301	<blank>	<blank>	48	3000/151/50	25409	284523	SUB
14	Chennai	3000	7533	<blank>	<blank>	48	3000/151/50	25474	285193	SUB
14	Chennai	3000	7486	<blank>	<blank>	48	3000/151/50	25268	285164	SUB
1	Mayiladuturai	3000	29034	<blank>	<blank>	48	3000/151/50	25376	284125	SUB
1	Mayiladuturai	3000	29020	<blank>	<blank>	48	3000/151/50	25375	284082	SUB
1	Mayiladuturai	3000	29008	<blank>	<blank>	48	3000/151/50	25375	284081	SUB
1	Mayiladuturai	3000	28957	<blank>	<blank>	48	3000/151/50	25374	284054	SUB
1	Mayiladuturai	3000	28855	<blank>	<blank>	47	3000/151/50	24849	277774	SUB
14	Chennai	3000	7370	<blank>	<blank>	47	3000/151/50	25021	279598	SUB

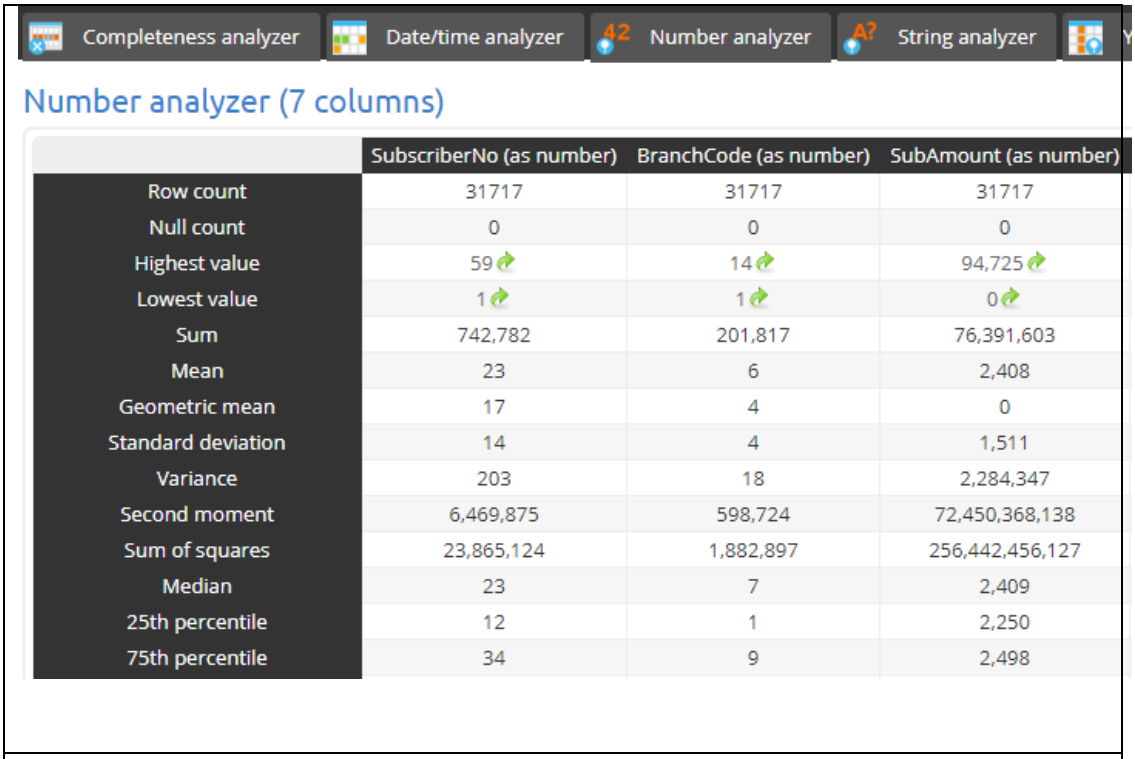
Figure 4.1 : DataCleaner Completeness Analyser output



	CustomerName	BranchName	Remarks
Row count	31717	31717	31717
Null count	0	0	0
Blank count	2	0	28071
Entirely uppercase count	9264	0	177
Entirely lowercase count	0	0	5
Total char count	383159	327908	85151
Max chars	43	18	45
Min chars	0	7	0
Avg chars	12	10	2
Max white spaces	3	2	7
Min white spaces	0	0	0
Avg white spaces	0	0	0
Uppercase chars	148743	32105	21786

Figure 4.2: String Analyser sample output

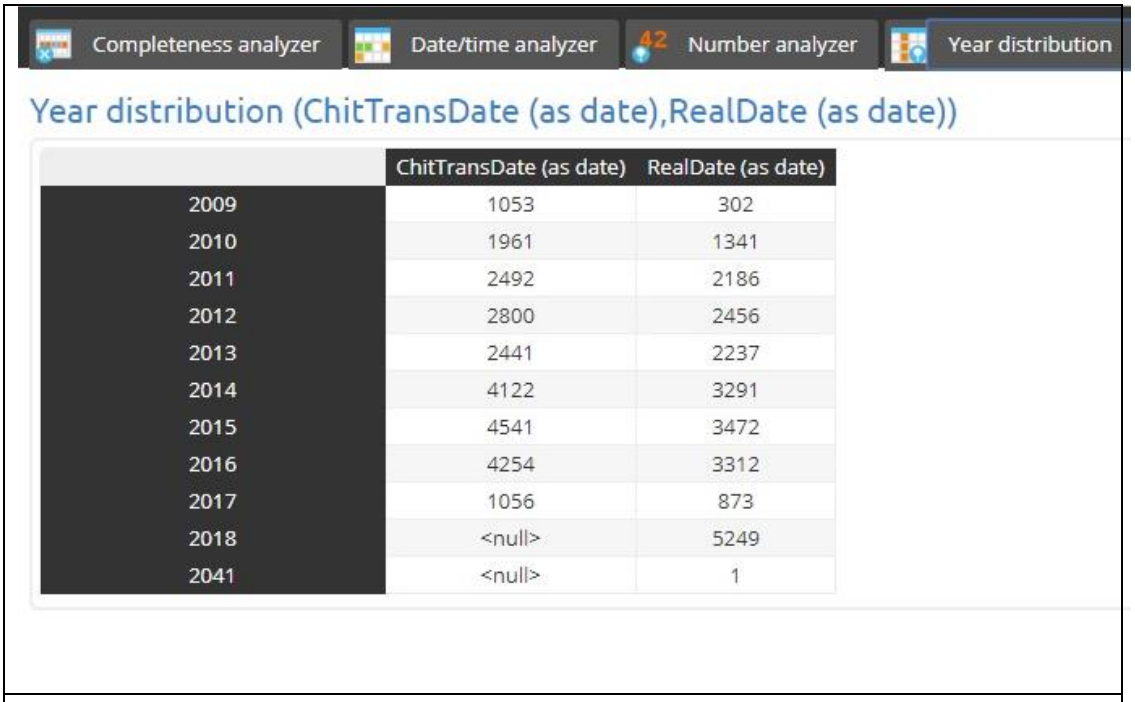
The third check was a number analyser that checked things like the number of instances of nulls, the highest value, the lowest value and the median value on nominated number features. See figure 4.3 for sample output that was run on 2500-111 to 2500-A.xlsx. This showed information such as there were no instances of nulls and the highest value for subamount was 94,725 and the lowest value was 0.



	SubscriberNo (as number)	BranchCode (as number)	SubAmount (as number)
Row count	31717	31717	31717
Null count	0	0	0
Highest value	59	14	94,725
Lowest value	1	1	0
Sum	742,782	201,817	76,391,603
Mean	23	6	2,408
Geometric mean	17	4	0
Standard deviation	14	4	1,511
Variance	203	18	2,284,347
Second moment	6,469,875	598,724	72,450,368,138
Sum of squares	23,865,124	1,882,897	256,442,456,127
Median	23	7	2,409
25th percentile	12	1	2,250
75th percentile	34	9	2,498

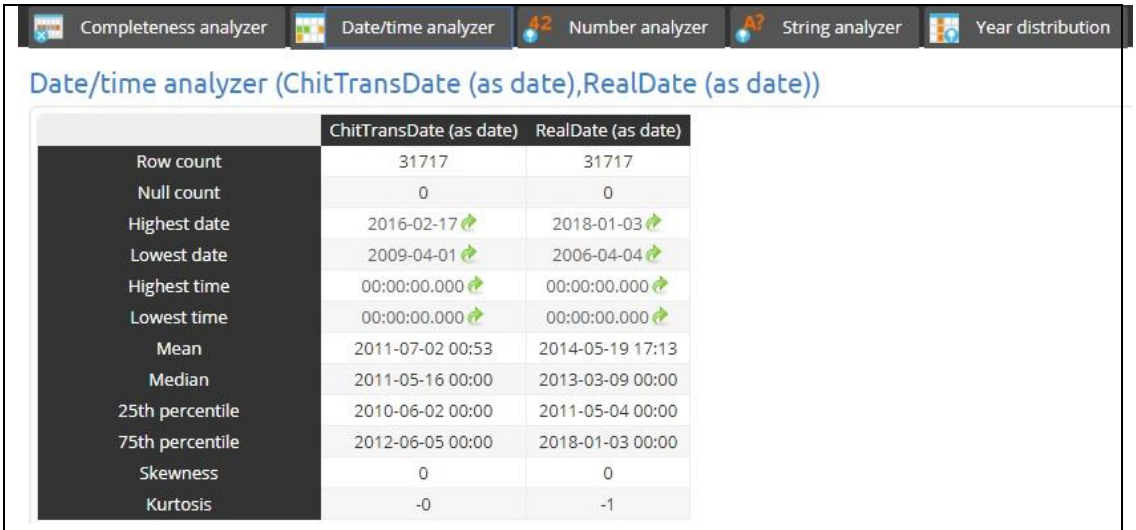
Figure 4.3: Sample output from number analyser

The fourth check was a Year distribution that was done on date features and this provided information on the amount of instances for each year and highlighted potential data quality issues with incorrectly input years. See figure 4.4 for an example of Year distribution output that was run on 3000-151 to 375-A.xlsx and showed that the earliest ChitTransDate was for 2009 and that there was one entry that had a RealDate of 2041. This was a data quality issue and the whole record was removed from the dataset. The last check was the date time analyser that was run on nominated date features and showed information like the highest and lowest date and the number of nulls. This shows that the lowest ChitTransDate was from 2009 and that there were no instances of nulls for ChitTransDate. See figure 4.5 for more information.



	ChitTransDate (as date)	RealDate (as date)
2009	1053	302
2010	1961	1341
2011	2492	2186
2012	2800	2456
2013	2441	2237
2014	4122	3291
2015	4541	3472
2016	4254	3312
2017	1056	873
2018	<null>	5249
2041	<null>	1

Figure 4.4: Year distribution sample output



	ChitTransDate (as date)	RealDate (as date)
Row count	31717	31717
Null count	0	0
Highest date	2016-02-17	2018-01-03
Lowest date	2009-04-01	2006-04-04
Highest time	00:00:00.000	00:00:00.000
Lowest time	00:00:00.000	00:00:00.000
Mean	2011-07-02 00:53	2014-05-19 17:13
Median	2011-05-16 00:00	2013-03-09 00:00
25th percentile	2010-06-02 00:00	2011-05-04 00:00
75th percentile	2012-06-05 00:00	2018-01-03 00:00
Skewness	0	0
Kurtosis	-0	-1

Figure 4.5: Date/time analyser sample output

4.3 Data Cleaning and Data Merge in Kettle

Kettle was used to merge the 13 files into one, factorise the categorical features, check for key dates, clean some data and create the default features. Kettle is a graphical tool with drag and drop icons for data cleaning and data integration. Figure 4.6 is a high level design of the data flow, files involved and activities at each step.

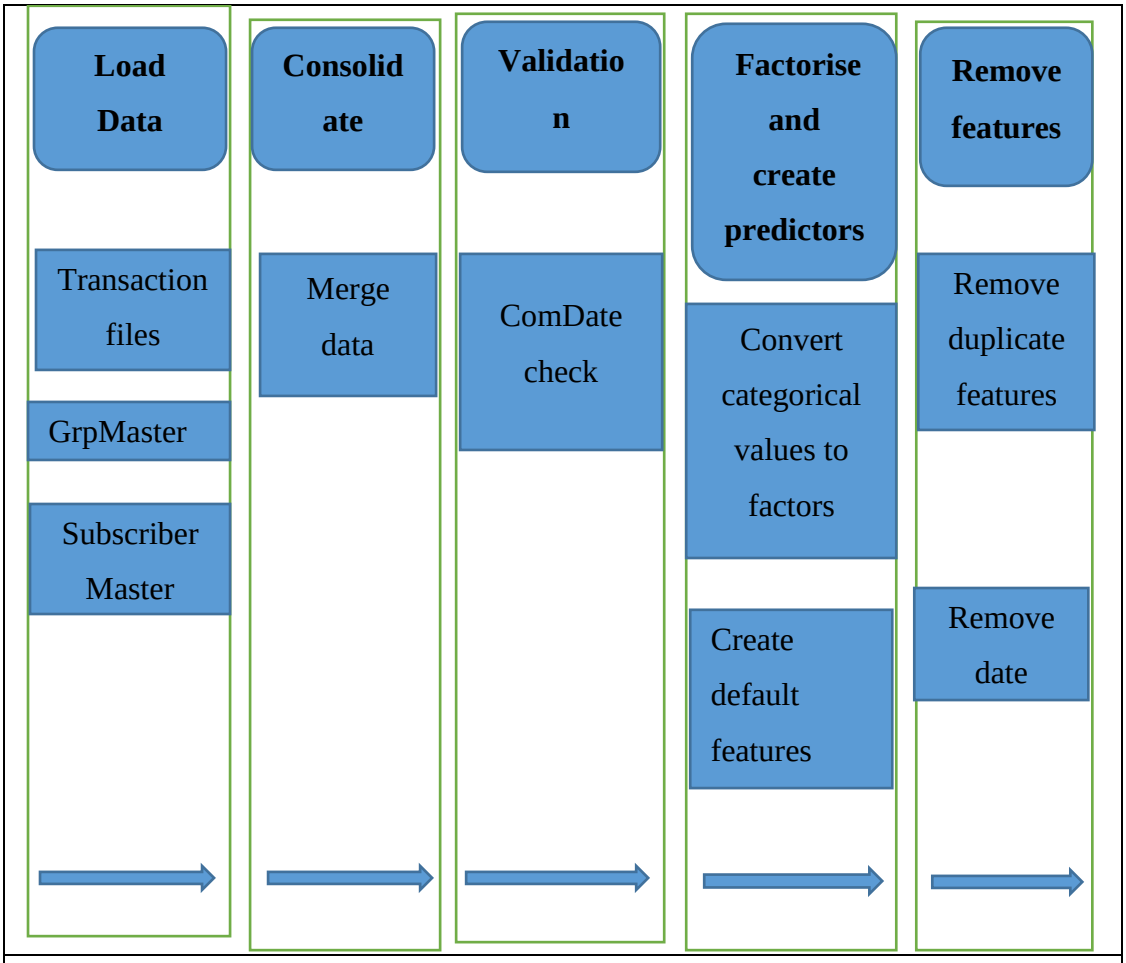


Figure 4.6: High level design for ETL process in Kettle

The 8 transaction files had the same format so instead of merging the files on a key field they were just added to the one output file. A field called ComDate which was the commencement date of the first transaction was then checked to see was it 2010 or greater. Transactions that were before 2010 were dropped and this resulted in 74,502 being dropped and 135,515 transactions were kept. The other files were added by merging on key features such as GroupID and InstNo. Company A were interested in the impact relatives had on default. They had a field called RelCode that was suitable

but its coverage was about 50% so a new ID field was created using the surname, age and relCode to identify the relative and this field replaced RelCode in the final output.

After all the features were merged into one file there were 128 features. Some features such as SecuityMode and CPin had to be cleaned of data quality issues. These had characters such as "/, -" that had to be removed and this was done using the Kettle Replace in a String feature. Many of the features such as Relation1 and Relation2 were categorical and had to be converted to numerical values and this was also done using the Replace in a String tool. An output file was then created with the 128 features and the file was checked in detail in excel by putting on the filter option and reviewing the filter values for each of the features. 56 of the 128 features were dropped at this first stage of pre-processing. The features were removed for reasons such as the field values only had one value besides null and so were not suitable for prediction and some features were duplicates created when merging the 13 files. See table A.2 in the appendix for further details on why these features were dropped.

After consultation with Company A it was decided to predict two types of default, with Default1 been a late payment of 30 days or more and Default2 been a late payment of 90 days or more. These two parameters were not readily available and so Kettle was used to deduce these parameters using the difference between the date of actual payment called ChitTransDate and date of expected payment called the AuctionDate. Once this useful information was gathered the 11 date features in table A.3 in the appendix were removed. The Kettle tools used for this data cleaning and integration task included Sort Rows, Merge Join, Data Validator, Filter Rows and Replace in a String. Figure 4.7 shows some of the Kettle tools used.

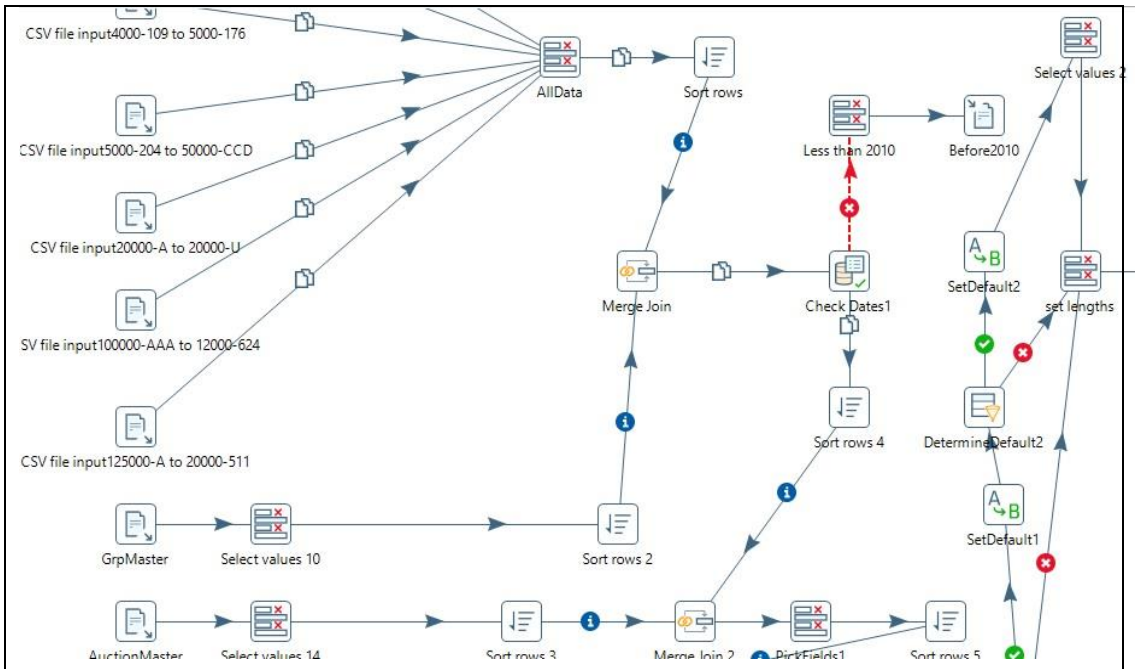


Figure 4.7: Kettle tools used in data integration process

4.4 Data pre-processing in R

The data was then loaded into R Studio and each of the features were checked for missing values. This showed that 6 features had a high percentage of missing values and these were:

Field Name	Number of Nas	As a %
Relation2	131994	97%
TenderID	121240	89%
SNo	121240	89%
PriceFlag	121240	89%
CPin	113534	84%
SecurityMode	96308	71%

The choice now was to remove all six features or see which ones the caret pre-process function had an issue with and the latter option was chosen. The caret pre-process function was ran for centring, scaling, knnImput and PCA (principle component analysis) and this returned an error with 4 of the features (TenderID, SNo, PriceFlag and SecurityMode) so these were removed. The check for features with nulls showed a total of 8 features had one or more null and they had to be dealt with. All of these features

were categorical features so it was decided to set the nulls to 0 as 0 wasn't used for any other categories. Now the data was ready for training and testing.

4.5 Model setup and Tuning for ANN

As previously stated it was decided to tune and train separate models for each of the two defaults. The first model created was an ANN model and the following is a description of the parameters used. Kuhn et al. (2018) explain that are a number of different ANN algorithms available in the caret package such as multi-layer perceptron and Neural Networks with Feature Extraction. Neural Networks with Feature Extraction first runs principle component analysis on the data and then uses it in the neural network. For these experiment it was decided to use multi-layer perceptron however there are also a number of varieties of the multi-layer perceptron such as Monotone Multi-Layer Perceptron Neural Network and the basic multi-layer perceptron (Bergmeir et al., 2017). For these experiments it was decided to use the multi-layer perceptron algorithm which is the most popular and is a feed-forward network because it met the needs of the experiments.

There are a number of different learning functions that can be used with it such as Std_Backpropagation which is the default and BackpropBatch (Bergmeir et al., 2017). Both of these work by taking two parameters which are the learning rate and the maximum output difference. The learning rate specifies the gradient descent step width and the maximum width specifies the difference between the output and the target value that is assumed to be zero error and is used to prevent overtraining. For this experiment Std_Backpropagation was used. According to Bergmeir et al. (2017) the defaults used by the learning functions usually don't have to be changed, however for these experiments it was decided to change these to determine did that lead to an improvement in performance.

Kuhn (2008) describes how the trainControl feature can be used to hold a set of control parameters for the caret train model. For these experiments the trainControl used is shown in figure 4.8. Method of cv means cross validation, number is the number of cross validations which was 3 for all of the experiments as recommended by Tsai et al. (2009).

ClassProbs of *TRUE* means that class probabilities should be computed for classification models along with their predicted values for each resample.

```
ctrl=trainControl(method = "cv",
                  number = cvnumber,
                  classProbs = TRUE,
                  preProcOptions = list(thresh = PCAThresh),
                  verboseIter = TRUE,
                  summaryFunction = twoClassSummary,
                  sampling = samplingMethod,
                  returnResamp='all')
```

Figure 4.8: trainControl used for experiments

VerboseIter of true means to print the training log, *summaryFunction* of twoClassSummary means the performance metrics for resamples are specific to measures suitable for two class summaries such as area under the ROC curve and sensitivity and specificity. S*ampling* is the sampling method used which for these experiments was SMOTE, down and none and *returnResamp* of all means return all the information on the resamples. Resamples are the data used for training taking into account the method used.

Kuhn (2008) explains how the train function in caret is used to take a number of inputs and train the data using these. The train function used in these experiments is shown in figure 4.9.

```
train(x=TrainNoPred,y=outcome,method = "mlp",preProcess = c("center",
"scale","knnImpute","pca"),trControl=ctrl,
 tuneGrid=tune_Grid1, learnFunc = "Std_Backpropagation",metric="ROC",
initFuncParams = initFuncParams1)
```

Figure 4.9: Train function parameters used

In this example x is the data to train excluding the predictors, *y* is the response feature, method is *mlp* (multi-layer perceptron) and *preProcess* is set to center and scale the data

and perform knnImpute and *pca* on it before it's trained. The pca threshold was set to 95% which means that enough principle components were chosen to explain 95% of the variance in the response. There was very little guidance found in the literature on a suitable amount of threshold but Scribilia[9] advised it depended on the application and to use at least 80% while Zekić-Sušac et al. (2016) used 99%. Here *trControl* is the trainControl measures mentioned previously, *tuneGrid* is the grid of tuning parameters that will be explained in detail in the next section, the *learnFunc* is set to Std_Backpropagation, the *metric* to pick the best fold for performance was *ROC* and *initFuncParams* are the parameters for the initialisation function that were also tuned which are the learning rate and the maximum output difference.

Parallel processing was used to speed up the time it took to train the models. Caret is setup to handle parallel processing if it's turned on. The modelling was run on a windows 10 laptop with quad core processor and 8 GB of memory. As it was a windows laptop many of the normal parallel processing options were not available but the doParralell package created by Weston & Calaway (2017) was suitable so was used. This was used to register all 4 cores: registerDoParallel (cores=4). To ensure results were reproducible the seed was set to the same value in all models and was set to 107 which was just the random number used.

For these experiments tuning was limited to the number of neurons in the hidden layer and the initialisation parameters. The number of hidden layers was set to one as recommended by Tsai et al., (2009). The configuration also used one output layer and tuned 1:30 neurons in the hidden layer as this was used by Zekić-Sušac et al. (2016). Tuning was also done on the initialisation parameters and Bardak et al., (2016) advised that it was best to find the optimum initialisation values using experimentation. However they didn't recommend values to start with, so it was decided to start with (-0.2, 0.2) and

[9] Scibilia, Bruno (2016, September 6). Creating Value from Your Data. Retrieved May 7, 2018, from http://blog.minitab.com/blog/applying-statistics-in-quality-projects/creating-value-from-your-data

move in increments of 0.1 up to (-0.5, 0.5) as the default for the algorithm was (-0.3, 0.3).

The last element that was tuned was the sampling method. Firstly the predictors were checked to see how imbalanced the classes were and this showed that the predictors were highly imbalanced. See figure 4.10 for imbalance details on one of the training datasets. As the datasets were broken up into training and testing using random sampling the percentage imbalance varied slightly between each experimental run.

Field	Value	%	Imbalance Ratio (IR)
Default1	Not Defaulted	98.96%	95
Default1	Defaulted	1.04%	
Default2	Not Defaulted	99.81%	532
Default2	Defaulted	0.19%	

Figure 4.10: Class imbalance of response features

To deal with the class imbalance it was decided to use sampling techniques. Initially experiments used no sampling techniques and returned very high accuracies but the class imbalance was so large for both response features that even if the classifier was not trained at all and always guessed not defaulted it would have an accuracy of 99.5%. The sampling techniques used were down and SMOTE as these were two popular methods in the studies reviewed. 6 more fields had to be dropped at this stage as when down sampling was used they reported the zero variance error.

4.6 Model Setup and Tuning for Logistic Regression

There are a number of different logistic regression algorithms available in the *caret* package such as boosted logistic regression (known as LogitBoost) and logistic model trees (Kuhn et al., 2018). Sun, Zhang, & Zhou (2014) explain that LogitBoost is a form of gradient descent that uses logistic loss and boosting type optimisation and has been used successfully in web page ranking. Logistic model trees use logistic regression and decision tree learning for classification. In this experiment it was decided to use *LogitBoost* as it performed well for a previous study (Sun et al., 2014). The same

trainControl parameters, sampling techniques, parallel processing and seed was used for LR as had previously been used for ANN. The call to the train model is in figure 4.11.

```
train(x=TrainNoPred,y=outcome, method = "LogitBoost",preProcess = c("center",
"scale","knnImpute","pca"),trControl=ctrl,metric = "ROC",tuneGrid=tune_Grid1)
```

Figure 4.11: Train parameters for logistic regression

The data was centred, scaled, and knnImpute and pca were used. The metric used to pick the best fold for performance was ROC and this time the tuning was called *nIter* which stands for number of iterations which Kuhn et al., (2018) explains is the number of boosting iterations.

4.7 Process to evaluate the models

It was decided to break the dataset into two parts, using a 70:30 split, with 70% used for training and 30% used for validation testing. Cross validation was used in the training dataset to help with tuning the algorithm parameters and the validation dataset was used for performance comparison. This method is recommended by Salzberg (1997) to avoid reporting incorrect statistical significance results as without it every single test run for tuning is a separate test and should be accounted for when determining the suitable statistical significance level. The main metrics used to compare the models were balanced accuracy and area under the curve (AUC). The class imbalance rate in table 2 shows how imbalanced the data was so the balanced accuracy was the most appropriate accuracy measurement to use. AUC is another important metric often used to compare models as it results in an easily comparable figure. ROC and F1 measures are also used to compare models and so are also presented and discussed. As the hold out validation sample was done using previously unseen data it is a method to mimic real world conditions and verify the ability of the models to generalise (Gschwind, 2007).

4.8 Correlation

Correlation analysis was performed in R studio using the cor function and snowfall package for parallel processing to determine how correlated the features were to both

default features. The results of the top 30 Pearson correlations to Default1 are shown in figure 4.12. All the features would be considered to have a very weak correlations with default 1. The results for the top 30 correlations to Default 2 are shown in figure 4.3 and table A.7 in the appendix and again all features have a weak correlation to default2. Figure 4.12 shows that some features are positively correlated (have columns above the line) and some are negatively correlated (have columns below the line). The strongest correlations were for unexpected items such as InstNo which is the instalment number for which the payment was made and TransID which is transaction ID. It also contains some interesting finds such as Age and PRZPMTFlag (flag to indicate prize money paid to subscriber) are negatively associated with default. AuctionID and UnDivBF are positively associated with default1. Each auction held by the company has a unique ID so this suggests that a particular id is associated with default. UnDivBF is left over dividend from the previous cycle and this could suggest that chit fund members that have backup in the form of left over dividend are less likely to default.

Figure 4.13 shows that the strongest correlation are InstNo and TransID. This is followed by SubAmount which is the subscription amount for a particular instalment. Again age and PRZPMTFlag are negatively correlated with Default2.

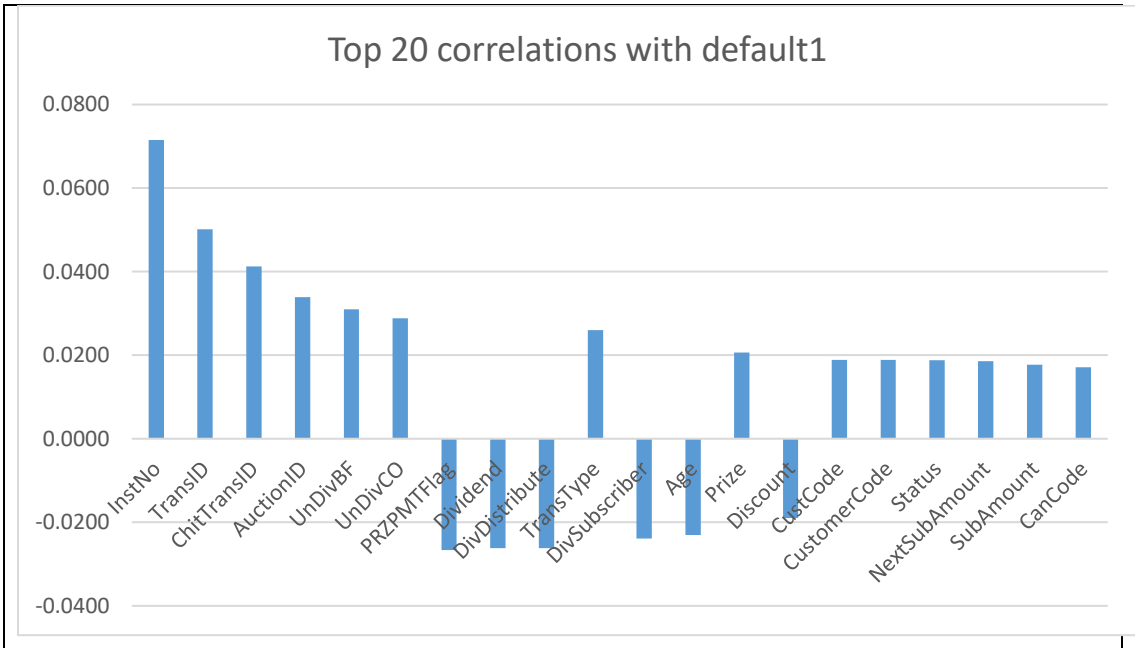


Figure 4.12: Top 20 correlations with Default1

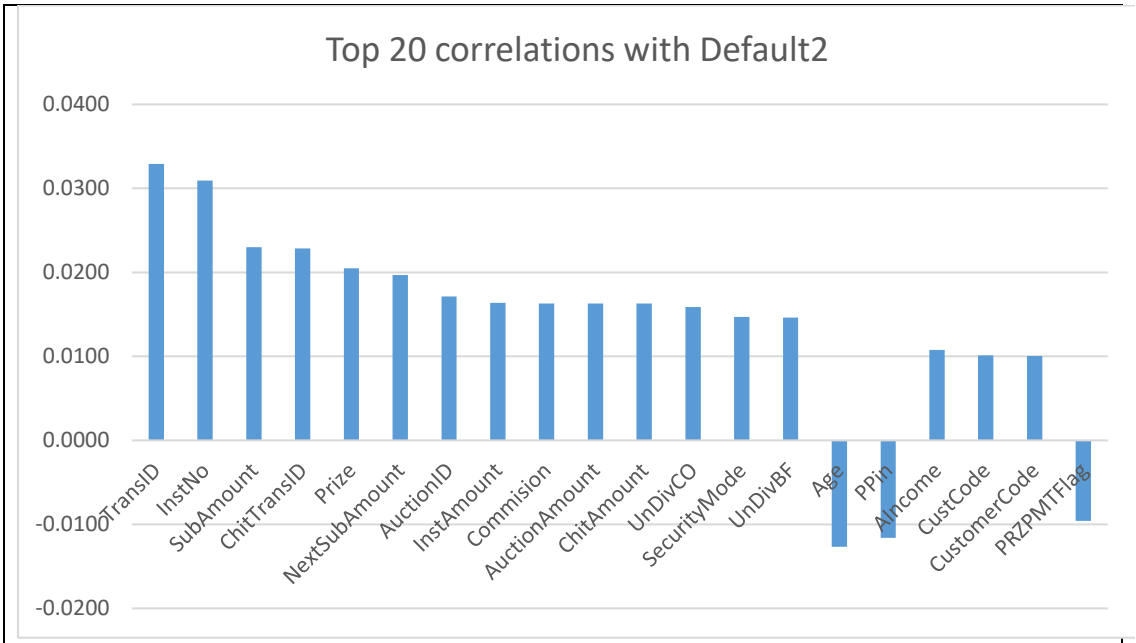


Figure 4.13: Top 20 correlations with Default2

4.9 PCA Results

Principle components analysis was used to reduce the number of features used as there were too many for modelling and the threshold was set at 95%. This meant that enough principle components were used to explain 95% of the variance in the data and the first 30 principle components were needed to do this. As you can see in figure 4.14 the first component explains the most variance and the amount explained by each principle component decreases as the principle component number increases. The cumulative scree plot in figure 4.15 shows the first 30 components explaining 95% of the data.

Prcomp is a built in R function and was used to do full principle component analysis as only limited analysis was available in caret. This function returns a number of values such as rotation, standard deviation and scale. The standard deviations, variances, % variances and cumulative variance % for the first 30 principle components are in table A.4 in the appendix.

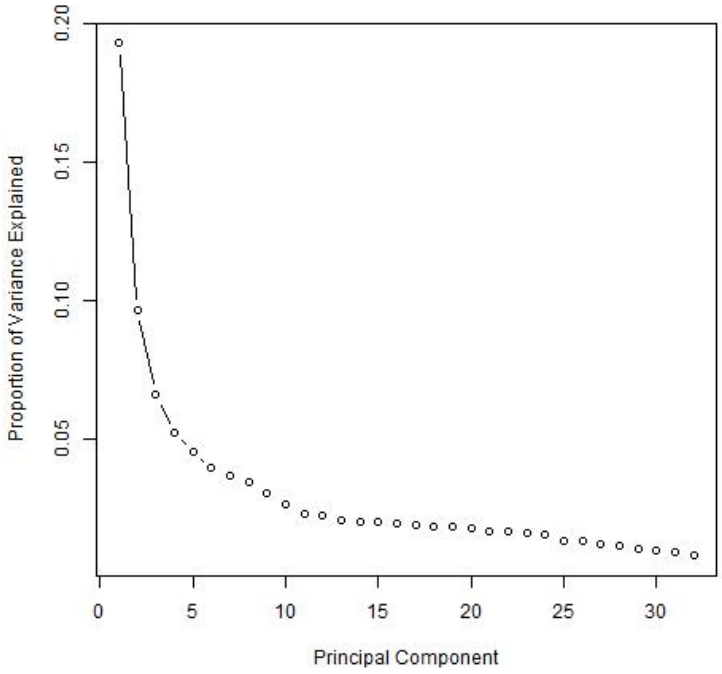


Figure 4.14: Scree plot of the first 30 principle components

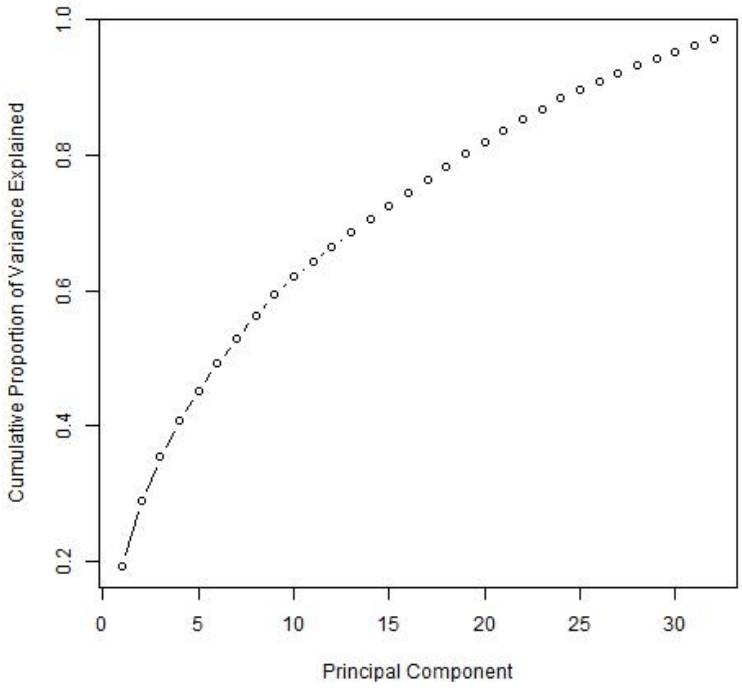


Figure 4.15: Scree Plot showing cumulative variance explained

The contrib function in the factoextra R package was used to determine the contribution of each feature to each of the principle components. The corrplot package was used to plot the contribution of the top 20 features to the principle components. Figure 4.16 shows the contribution of the top 20 features to the first 30 principle components and shows that there is very little difference in the contribution of the top 20 features to the first 30 principle components. Feature RelCode is the highest at 3.33% and Age1 made the smallest contribution of 2.47%.

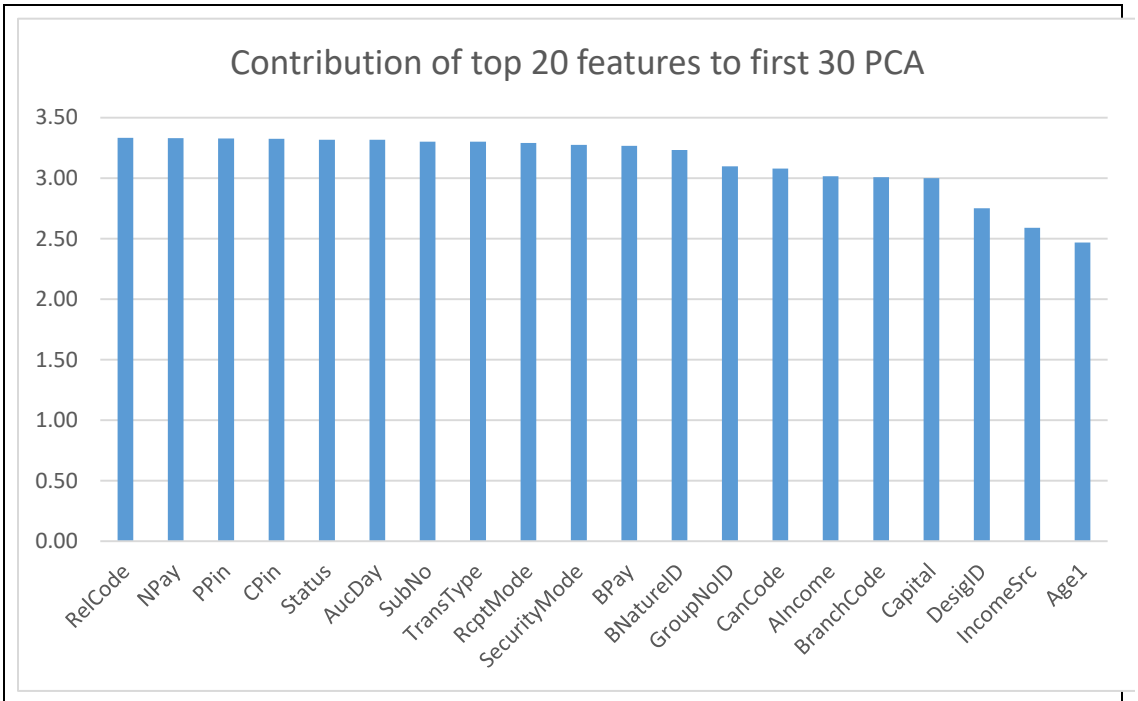


Figure 4.16: Top 20 features that contribute to first 30 PCA

Figure 4.17 shows the contribution of the bottom 20 features to the first 30 principal components. The lowest contribution was ChitAmount, AuctionAmount and commission. The contribution of each of the features will be discussed in detail in the next chapter.

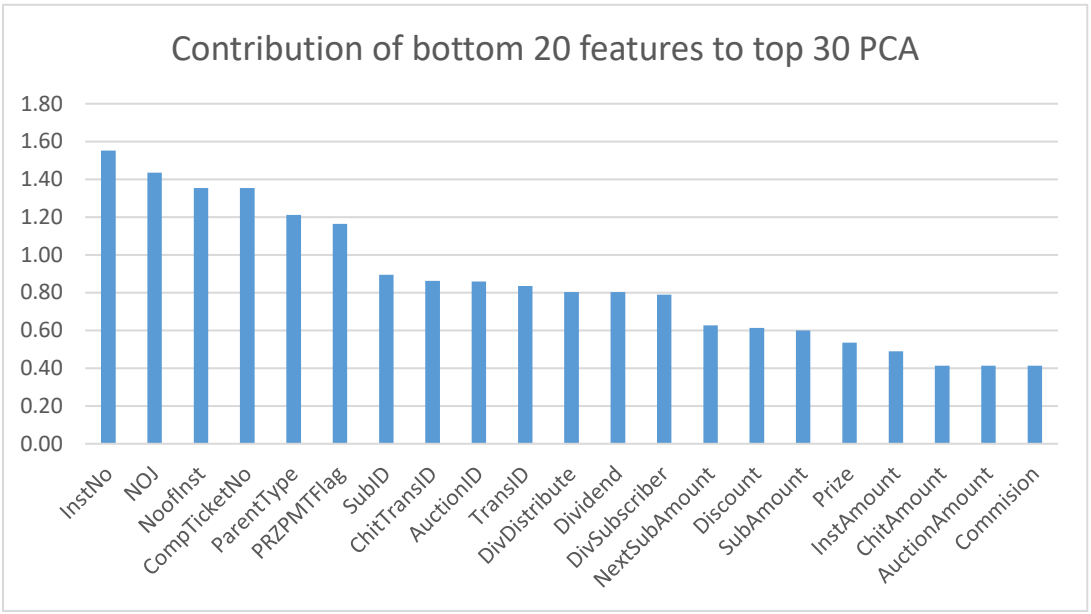


Figure 4.17: Contribution of bottom 20 features to first 30 PCA

4.10 Results

The initial tests were run using no sampling techniques, cross validation of 3 and the reported accuracies were very high but the balanced accuracies were low. See table 4.1 for details. As balanced accuracy was so low it was decided to use two sampling techniques to correct for this and the techniques used were SMOTE and down.

Sixteen tests were run, eight for default1 and eight for default2. The results for the 8 tests in default1 are in table 4.2. Each of the tests included a run of both the multi-layer perceptron (MLP) and the boosted logistic regression (LR) models. The same sampling technique was used for each test and the same cross validation of 3 and the same seed number so results could be compared. As stated previously the parameters that were tuned for MLP were the number of neurons in the hidden layer which was tuned on 1 to 30 neurons and the parameters for the initialisation function which were tuned from (-0.2, 0.2) to (-0.5, 0.5). The only parameter that was tuned for LR was the number of boosted iterations. As you can see the balanced accuracy results for the SMOTE sampling outperformed the down sampling results for all of the MLP train and test experiments except for initialisation parameters of (-0.5, 0.5).

Default #	Initial	Best Tune MLP	Best Tune LR	Accuracy on MLP Train	Accuracy on MLP Test	Accuracy on LR Train	Accuracy on LR Test	P value on Acc>NIR MLP	P value on Acc>NIR LR
1	(-0.4, 0.4)	size=24	20	0.9927	0.9900	0.9901	0.9901	0.0000	0.5087
2	(-0.3, 0.3)	size=29	65	0.9986	0.9984	0.9980	0.9980	0.0000	0.3760
Default #	**Initial**	**Best Tune MLP**	**Best Tune LR**	**Balanced Accuracy on MLP train**	**Balanced Accuracy on MLP test**	**Balanced Accuracy on LR train**	**Balanced Accuracy on LR test**	**NIR train**	**NIR test**
1	(-0.4, 0.4)	size=24	20	0.6491	0.5725	0.5000	0.5000	0.9901	0.9897
2	(-0.3, 0.3)	size=29	65	0.6631	0.6249	0.5263	0.5125	0.9980	0.9980

Table 4.1 : Initial Accuracy and balanced accuracy

Cells in blue indicate the best performing result for that specific category

The best MLP balanced accuracy result for 30 days past due was 0.9252 which was achieved using initialisation function parameters of (-0.4, 0.4) and SMOTE sampling and 29 hidden neurons. The best result on the hold out dataset was 0.8372 also with the same parameters. For the LR experiments down sampling often achieved better results than SMOTE. The best balanced accuracy for the train dataset was 0.7615 achieved using down sampling with a boosted iteration of 100. The best LR result on the test dataset was 0.7281 achieved again on down sampling using a boosted iteration of 60.

Sampling	Initial	Best Tune MLP	Best Tune LR	Balanced Accuracy on MLP train	Balanced Accuracy on MLP test	Balanced Accuracy on LR train	Balanced Accuracy on LR test
down	(-0.2, 0.2)	size=30	60	0.8860	0.7895	0.7608	**0.7281**
SMOTE	(-0.2, 0.2)	size=27	40	0.9225	0.8162	0.7248	0.6742
down	(-0.3, 0.3)	size=30	90	0.8723	0.7851	0.6589	0.5047
SMOTE	(-0.3, 0.3)	size=28	70	0.9197	0.7999	0.6995	0.6862
down	(-0.4, 0.4)	size=30	100	0.8861	0.8051	**0.7615**	0.6893
SMOTE	(-0.4, 0.4)	size=29	75	**0.9252**	**0.8372**	0.6491	0.6201
down	(-0.5, 0.5)	size=26	30	0.8840	0.8101	0.7490	0.6824
SMOTE	(-0.5, 0.5)	size=27	75	0.9202	0.8048	0.6491	0.6201

Table 4.2: Balanced Accuracy results for default1

Cells in blue indicate the best performing result for that specific category

The ROC results for default1 are shown in table 4.3. For MLP most of the SMOTE results were better than the down sampling results and the same was seen for LR. The best ROC result is the highest result and was 0.8945. This was achieved using SMOTE sampling, 27 hidden neurons and initialisation parameters of (-0.2, 0.2). An ROC figure wasn't determined using the validation dataset but an ROC curve was drawn using the validation dataset and is shown in figure 27. A related feature called area under the curve (AUC) was calculated for the test sample and will be discussed later.

The best ROC result for LR was 0.7492 that was achieved using SMOTE sampling and using 70 boosted iterations. A t test was done on the MLP and LR results to determine was the difference between these resample results statistically significant and a cut-off of 0.05 was used to test for significance with all values 0.05 or less considered statistically significant. As you can see all results are below 0.05 so all differences between the MLP and LR models were statistically significant. The ROC versus the number of hidden neurons for experiment 2 is shown in figure 4.18 and shows that as

the number of hidden neurons increases the ROC also increases up to 20 hidden neurons, after which it varies.

Run #	Sampling	Initial	Best Tune MLP	Best Tune LR	ROC on MLP train	ROC on LR train	p value T test on differences
1	down	(-0.2,0.2)	size=30	60	0.8626	0.7088	0.01
2	SMOTE	(-0.2,0.2)	size=27	40	**0.8945**	0.6887	0.00
3	down	(-0.3,0.3)	size=30	90	0.8529	0.7077	0.02
4	SMOTE	(-0.3,0.3)	size=28	70	0.8679	**0.7492**	0.01
5	down	(-0.4,0.4)	size=30	100	0.8509	0.6904	0.00
6	SMOTE	(-0.4,0.4)	size=29	75	0.8637	0.7258	0.01
7	down	(-0.5,0.5)	size=26	30	0.8801	0.7182	0.01
8	SMOTE	(-0.5,0.5)	size=28	75	0.8678	0.7258	0.02

Table 4.3: ROC results for default1

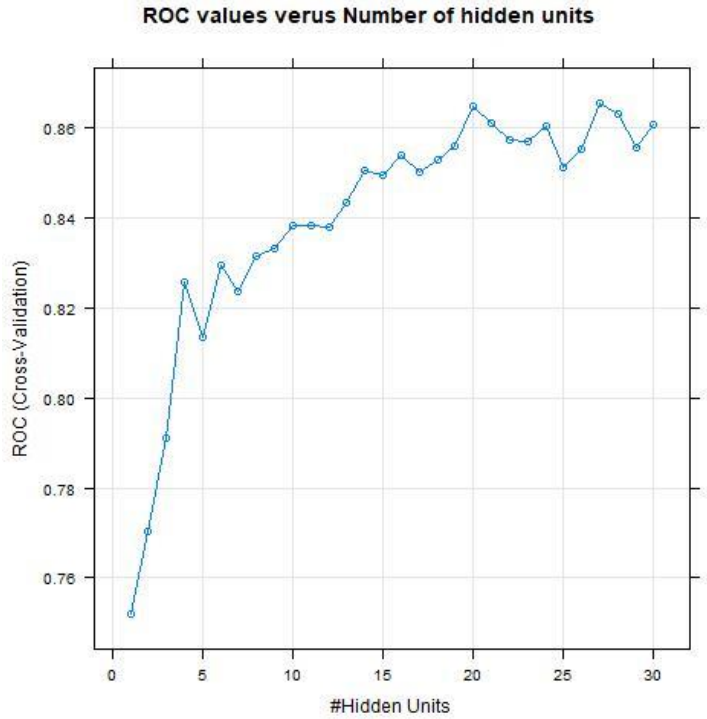


Figure 4.18: ROC versus number of hidden neurons for MLP

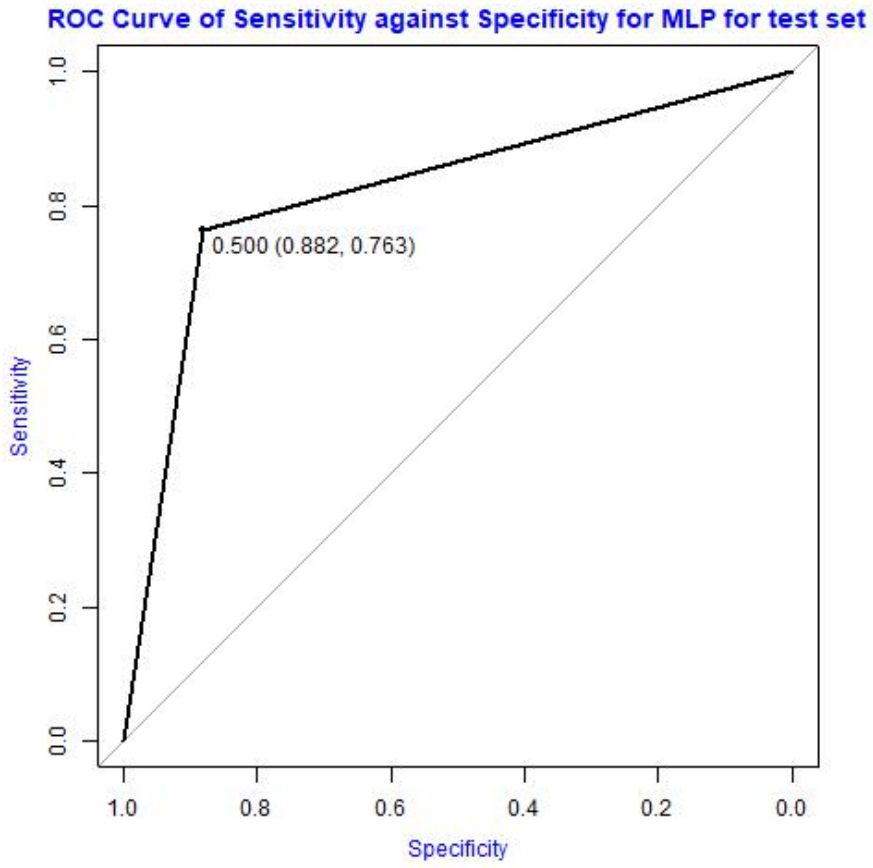


Figure 4.19: ROC curve for MLP validation for default1

Figure 4.19 shows the ROC curve for MLP validation dataset for default1. The ROC curve contains the specificity and sensitivity values at the ROC threshold of 0.50. A comparison of the MLP and LR results on the training dataset is shown in figure 20. The results show that on the training dataset MLP performed better for specificity, sensitivity and ROC.

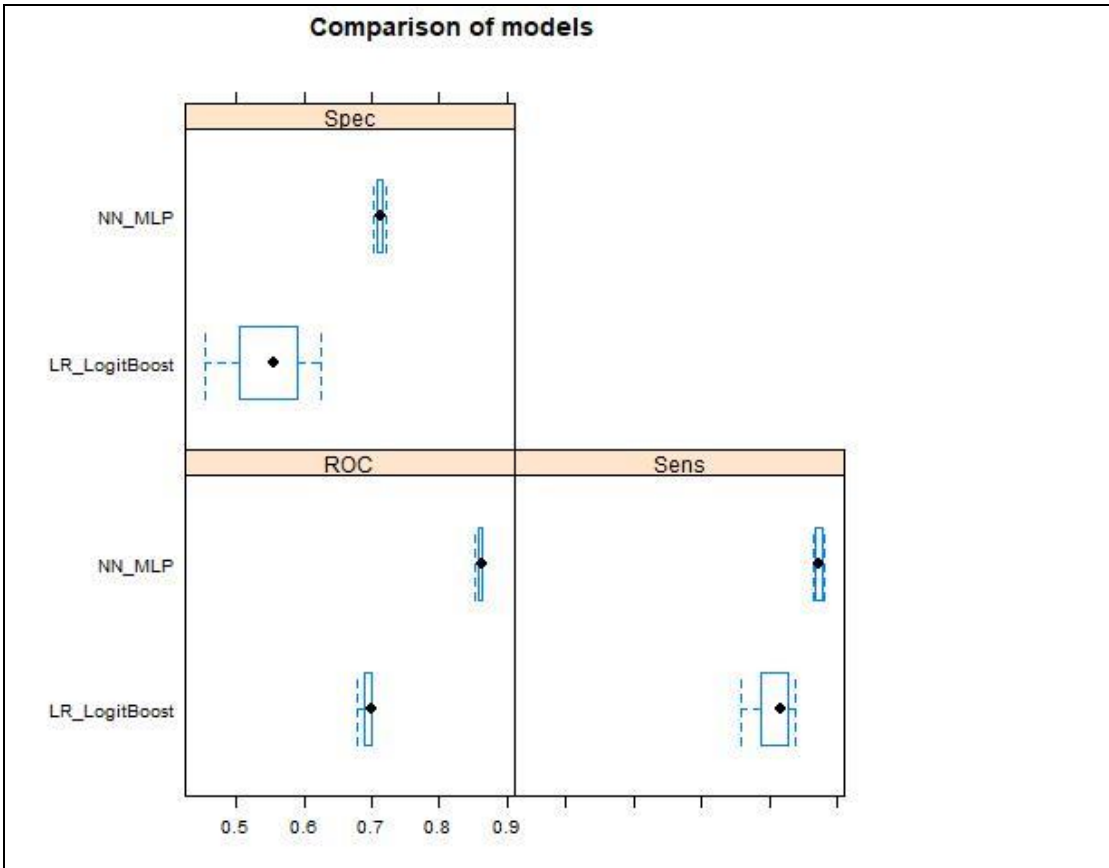


Figure 4.20: Comparison of MLP and LR models for default1

The F1 score is shown in table 4.4 and the highest result for MLP for train was 0.9426 and for validation was 0.9399, which was achieved with SMOTE sampling, initialisation parameters of (0.2, 0.2) and 27 hidden neurons. The highest F1 score for LR train was lower at 0.847 and 0.845 for validation, which was achieved with down sampling and 30 boosted iterations.

Run #	Sampling	Initial	Best Tune MLP	Best Tune LR	F1 on MLP train	F1 on MLP test	F1 on LR train	F1 on LR test
1	down	(-0.2,0.2)	size=30	60	0.8792	0.8761	0.8363	0.8336
2	SMOTE	(-0.2,0.2)	size=27	40	**0.9426**	**0.9399**	0.8127	0.8093
3	down	(-0.3,0.3)	size=30	90	0.8622	0.8600	0.5046	0.5093
4	SMOTE	(-0.3,0.3)	size=28	70	0.9077	0.9011	0.7609	0.7620
5	down	(-0.4,0.4)	size=30	100	0.8749	0.8733	0.7040	0.7055
6	SMOTE	(-0.4,0.4)	size=29	75	0.9365	0.9320	0.8173	0.8158
7	down	(-0.5,0.5)	size=26	30	0.7822	0.7828	**0.8472**	**0.8452**
8	SMOTE	(-0.5,0.5)	size=27	75	0.9420	0.9396	0.8173	0.8158

Table 4.4: Default 1 F1 measure

Table 4.5 shows AUC results and the best AUC on the train dataset was 0.925 and 0.837 on the validation dataset. For LR the best AUC on the train dataset was 0.762 and the best validation result was 0.728.

Run #	Sampling	Initial	Best Tune MLP	Best Tune LR	AUC on MLP train	AUC on MLP test	AUC on LR train	AUC on LR test
1	down	(-0.2,0.2)	size=30	60	0.8860	0.7895	0.7608	**0.7281**
2	SMOTE	(-0.2,0.2)	size=27	40	0.9225	0.8162	0.7248	0.6742
3	down	(-0.3,0.3)	size=30	90	0.8723	0.7851	0.6589	0.5047
4	SMOTE	(-0.3,0.3)	size=28	70	0.9197	0.7999	0.6995	0.6862
5	down	(-0.4,0.4)	size=30	100	0.8861	0.8051	**0.7615**	0.6893
6	SMOTE	(-0.4,0.4)	size=29	75	**0.9252**	**0.8372**	0.6491	0.6201
7	down	(-0.5,0.5)	size=26	30	0.8840	0.8101	0.7490	0.6824
8	SMOTE	(-0.5,0.5)	size=27	75	0.9202	0.8048	0.6491	0.6201

Table 4.5: Default 1 AUC values

Cells in blue indicate the best performing result for that specific category

The balanced accuracy results for default2 are shown in table 4.6. Again SMOTE sampling performed better for all MLP but not for LR. The highest balanced accuracy for MLP for the training dataset was 0.9564 and this was achieved with SMOTE sampling, 24 hidden neurons and initialisation parameters of (-0.3, 0.3). The highest balanced accuracy for MLP test was 0.8381 and this was achieved with SMOTE sampling, 21 hidden neurons and initialisation parameters of (-0.4, 0.4). Again LR didn't achieve the same levels of accuracy as MLP, with the highest LR on train recorded as 0.8762 using down sampling and 30 iterations. The highest balanced accuracy on LR using the validation dataset was 0.7734 and this was achieved on down sampling and 50 boosted iterations.

Run #	Sampling	Initial	Best Tune MLP	Best Tune LR	Balanced Accuracy on MLP train	Balanced Accuracy on MLP test	Balanced Accuracy on LR train	Balanced Accuracy on LR test
9	SMOTE	(-0.2,0.2)	size=30	50	0.9430	0.8379	0.8328	0.7369
10	down	(-0.2,0.2)	size=26	30	0.8638	0.8093	**0.8762**	0.7678
11	SMOTE	(-0.3,0.3)	size=24	25	**0.9564**	0.7780	0.8392	0.7077
12	down	(-0.3,0.3)	size=23	90	0.8829	0.8163	0.8699	0.7135
13	SMOTE	(-0.4,0.4)	size=21	35	0.9495	**0.8381**	0.8674	0.7537
14	down	(-0.4,0.4)	size=28	75	0.8954	0.8194	0.8537	0.7445
15	SMOTE	(-0.5,0.5)	size=25	100	0.9471	0.7870	0.8321	0.7096
16	down	(-0.5,0.5)	size=26	50	0.8895	0.8040	0.8337	**0.7734**

Table 4.6: Default2 Balanced Accuracy

Cells in blue indicate the best performing result for that specific category

The ROC results for default2 are in table 4.7 and again these were only calculated on the training dataset and the best results for both MLP and LR were achieved using SMOTE sampling. MLP achieved a better result than LR and the p value on the t test shows this difference was statistically significant. The best MLP result was on 24 neurons and initialisation parameter of (-0.2, 0.2) and was 0.9284, while the best LR result was 0.8499 and was achieved on 25 iterations. However ROC was plotted for the test dataset and the resulting graph is in figure 4.21. This contains the specificity and sensitivity at 0.50 threshold of (0.919, 0.662).

Run #	Sampling	Initial	Best Tune MLP	Best Tune LR	ROC on MLP train	ROC on LR train	p value T test on differences
9	SMOTE	(-0.1,0.1)	size=30	50	0.9047	0.7935	0.01
10	down	(-0.2,0.2)	size=27	75	0.8801	0.7182	0.01
11	SMOTE	(-0.2,0.2)	size=24	25	**0.9284**	**0.8499**	0.04
12	down	(-0.3,0.3)	size=23	90	0.8722	0.7488	0.15
13	SMOTE	(-0.3,0.3)	size=21	35	0.9085	0.7916	0.00
14	down	(-0.4,0.4)	size=28	75	0.8950	0.7883	0.01
15	SMOTE	(-0.4,0.4)	size=25	100	0.9015	0.7737	0.04
16	down	(-0.5,0.5)	size=26	50	0.8961	0.7552	0.02

Table 4.7: Default2 ROC results

Cells in blue indicate the best performing result for that specific category

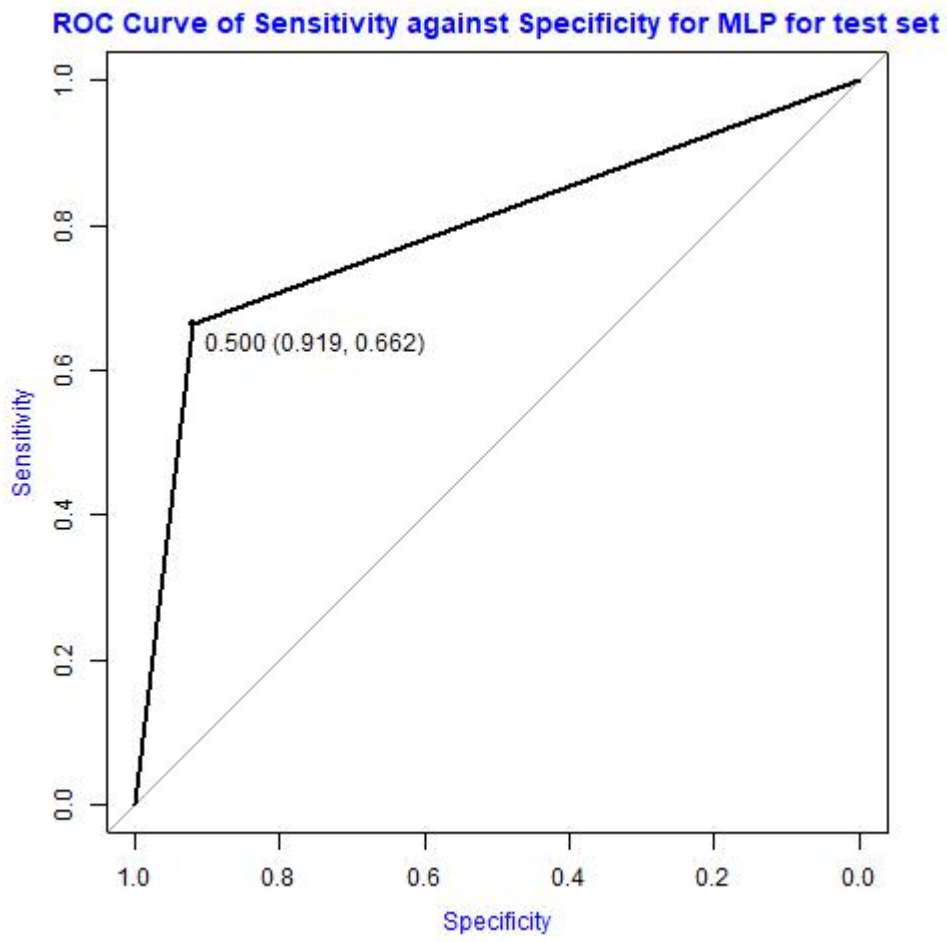


Figure 4.21: ROC curve for MLP for test data for default2

Figure 4.22 is a box plot comparing the train dataset results using MLP and LR. As can be seen MLP performed better than LR in all three measurements and these results were calculated on the resamples available only in the training dataset.

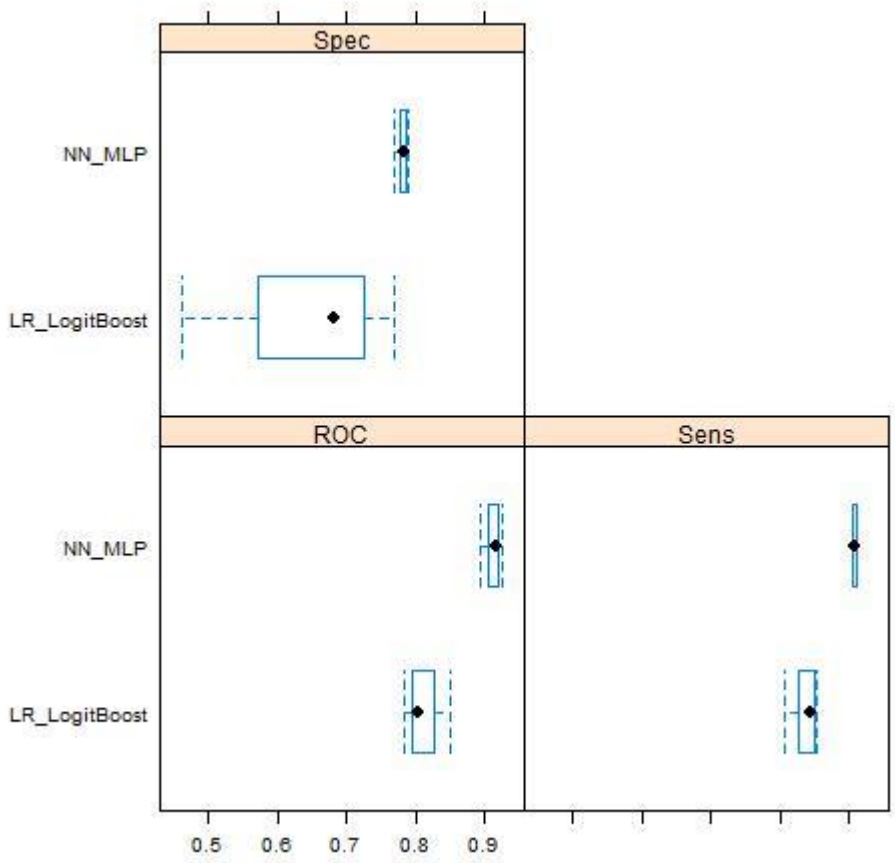


Figure 4.22: Comparison of MLR and LR for default2

F1 measures are shown in table 4.8 and interestingly these are very similar for MLP and LR for both the training and validation datasets. The F1 measure is often used to compare models and the highest value was for MLP for validation was 0.958 and for LR was 0.945. AUC values for default2 are shown in table 4.9 and these show a larger variation between the training and validation dataset with the best MLP for training at 0.956 and the largest for MLP test at 0.838. LR was lower and the largest AUC for LR train was 0.869 for 23 iterations and for LR test it was 0.773 achieved with 50 boosted iterations.

Run #	Sampling	Initial	Best Tune MLP	Best Tune LR	F1 on MLP train	F1 on MLP test	F1 on LR train	F1 on LR test
9	SMOTE	(-0.1,0.1)	size=30	50	0.9517	0.9502	**0.9455**	**0.9446**
10	down	(-0.2,0.2)	size=26	30	0.7822	0.7828	0.8472	0.8452
11	SMOTE	(-0.2,0.2)	size=24	25	**0.9599**	**0.9584**	0.9017	0.9013
12	down	(-0.3,0.3)	size=23	90	0.8707	0.8711	0.8542	0.8536
13	SMOTE	(-0.3,0.3)	size=21	35	0.9528	0.9519	0.8407	0.8407
14	down	(-0.4,0.4)	size=28	75	0.8923	0.8906	0.8287	0.8272
15	SMOTE	(-0.4,0.4)	size=25	100	0.9441	0.9430	0.9113	0.9107
16	down	(-0.5,0.5)	size=26	50	0.8789	0.8743	0.7147	0.7104

Table 4.8: F1 measure for default2

Run #	Sampling	Initial	Best Tune MLP	Best Tune LR	AUC on MLP train	AUC on MLP test	AUC on LR train	AUC on LR test
9	SMOTE	(-0.2,0.2)	size=30	50	0.9430	0.8379	0.8328	0.7369
10	down	(-0.2,0.2)	size=26	30	0.8840	0.8101	0.7490	0.6824
11	SMOTE	(-0.3,0.3)	size=24	25	**0.9564**	0.7780	0.8392	0.7077
12	down	(-0.3,0.3)	size=23	90	0.8829	0.8163	**0.8699**	0.7135
13	SMOTE	(-0.4,0.4)	size=21	35	0.9495	**0.8381**	0.8674	0.7537
14	down	(-0.4,0.4)	size=28	75	0.8954	0.8194	0.8537	0.7445
15	SMOTE	(-0.5,0.5)	size=25	100	0.9471	0.7870	0.8321	0.7096
16	down	(-0.5,0.5)	size=26	50	0.8895	0.8040	0.8337	**0.7734**

Table 4.9: AUC results for default2

4.11 Comparison of Model Results

A paired t test was performed on the differences in validation results for balanced accuracy, AUC and F1 score to determine were the differences between the ANN and LR models statistically significant. The statistical significance was initially set at 0.05 but after making the Bonferroni adjustment to take into account the 16 experiments it was reduced to 0.003. The t test results in table 4.10 show a p value less than 0.003 for the one tailed test that checked were the ANN results greater than LR so the null hypothesis can be rejected which means that for AUC the mean of ANN results are statistically significantly greater than the mean of LR results. This test was performed on all 16 AUC results, 8 of which were for default1 and 8 for default2.

Type of test	**Test details**	**p value**	**t statistic**	**df**	**95% CI**	**Alternative Hypothesis**
two tailed test on AUC	Mean of ANN AUC results are significantly different to mean of LR AUC results	0.000001627	7.5935	15	(0.08600206, 0.15312294)	true difference in means is not equal to 0
one tailed test on AUC	Mean of ANN results are significantly less than mean of LR results	1.0000000000	7.5935	15	-Inf 0.1471649	true difference in means is less than 0
one tailed test on AUC	Mean of ANN results are significantly greater than mean of LR AUC results	**0.0000008134**	7.5935	15	0.09196009 Inf	true difference in means is greater than 0

Table 4.10: t Test results of differences for AUC

Cells in blue indicate the best performing result for that specific category

The t test results for balanced accuracy are shown in table 4.11 and show a p value less than 0.003 for the one tailed test checking if the ANN results were greater than the LR results so again the null hypothesis can be rejected, which means that for balanced accuracy the mean of ANN results are statistically significantly greater than the mean of LR results. Again this test was performed on all 16 balanced accuracy results.

Type of test	Test details	p value	t statistic	df	95% CI	Alternative Hypothesis
two tailed test	Mean of ANN BA results are significantly different to mean of LR results	0.0000048780	6.9219	15	(0.07896988, 0.14924262)	true difference in means is not equal to 0
one tailed test	Mean of ANN BA results are significantly less than mean of LR results	1.0000000000	6.9219	15	-Inf 0.1430048	true difference in means is less than 0
one tailed test	Mean of ANN BA results are significantly greater than mean of LR results	**0.0000024390**	6.9219	15	0.08520769 Inf	true difference in means is greater than 0

Table 4.11: t Test results of differences for balanced accuracy

Cells in blue indicate the best performing result for that specific category

Type of test	Test details	p value	t statistic	df	95% CI	Alternative Hypothesis
two tailed test	Mean of ANN F1 results are significantly different to mean of LR results	0.0034260000	3.4703	15	(0.03368243, 0.14093007)	true difference in means is not equal to 0
one tailed test	Mean of ANN F1 results are significantly less than mean of LR results	0.9983000000	3.4703	15	-Inf 0.1314102	true difference in means is less than 0
one tailed test	Mean of ANN F1 results are significantly greater than mean of LR results	**0.0017130000**	3.4703	15	0.04320234 Inf	true difference in means is greater than 0

Table 4.12: t Test results of differences for F1 score

Cells in blue indicate the best performing result for that specific category

The t test results for F1 score are shown in table 4.12 and again show a p value less than 0.003 so the null hypothesis can be rejected, which in this case means that for F1 score the mean of ANN results are statistically significantly greater than the mean of LR results. Again this test was performed on all 16 results.

This chapter described the data and explained how it was profiled using data cleaner that flushed out some data quality issues. It then shows how the data was cleaned, merged and transformed using an integration tool called Kettle. The main part of pre-processing such as dealing with null values, centring and scaling the data was done in R studio. The configuration and implementation of the modelling was also done in R studio. The results achieved were shown in both tabular and graphical form and key performance metrics such as balanced accuracy, ROC, F1 measure and AUC were presented for both the MLP and LR models. It ended with the presentation of the t test results on the differences in results for balanced accuracy, AUC and F1 score. The next chapter analyses these results and compares them to other studies.

5. ANALYSIS, EVALUATION AND DISCUSSION

This chapter compares the results seen in the last chapter to other published studies and then discusses the features that contributed to the first 30 principle components.

5.1 Comparing modelling results to other studies

ANN scored well in accuracy in previous experiments and the aim is to have the highest accuracy possible. Tsai et al. (2009) achieved a training accuracy of 97.8% and a validation accuracy rate of 98.5% when predicting consumer loan default with data from a financial institution in Taiwan. Another study achieved a classification accuracy rate of 71% for validation testing when predicting company growth using Croatian company data (Zekić-Sušac et al. 2016). Yu, Wang, & Lai (2008) achieved an accuracy of 80% when predicting credit risk using a multistage neural network and ensemble learning approach. None of these studies mentioned data imbalance or the use of sampling techniques to deal with imbalanced data.

This study presented balanced accuracy which tends to be lower than regular accuracy. No previous results on balanced accuracy for ANN models were found but Gorzałczany & Rudziński (2016) achieved a balanced accuracy of 89% for a fuzzy rule based model for predicting credit scores using financial data. For this experiment the highest balanced accuracy for default1 for the ANN model was 92.5% for training and 83.7% for validation and the average was 90.2% for training and 80.6% for validation. The highest balanced accuracy was achieved using SMOTE sampling, 29 hidden neurons and initialisation parameters of (-0.4, 0.4) which give a learning rate of -0.4 and maximum output difference of 0.4. The best balanced accuracy for default2 which measures 90 day default achieved in this study was 95.6% for training and 83.8% for validation. Again the best result for validation was achieved using SMOTE sampling and initialisation parameters of (-0.4, 0.4) and for default2 the optimum number of hidden neurons was 21. Although the validation results are in the range of results seen in other studies they would not be considered in the top of the range. This may be because this is balanced accuracy which is usually less than regular accuracy. To get results at the top of the range tuning on a wider range of parameters may be required such as tuning on 1:60 hidden neurons and using initialisation parameters of (-0.1,0.1) to (-0.9, 0.9).

Tsai et al. (2009) study ran tests using logistic regression models for loan default using Taiwanese data and achieved an accuracy of 94% for training and 92% for validation while Peng, Wang, Kou, & Shi (2011) achieved 86% accuracy for financial risk prediction using data from six different countries. Another study reported lower accuracy on the validation dataset without sampling of 73% and even lower again with oversampling of 63% (Bapna et al. 2011). However no studies were found reporting balanced accuracy using LR.

The LR results for the experiments in this study showed the best balanced accuracy for default1 was 76.1% for training and 72.8% for validation and the average was 70.7% for train and 65.1% for validation. For default2 the best result achieved was 87.6% for training and 77.3% for validation and the average was 85.1% for train and 73.8% for validation. It should be noted that the highest validation results for both default1 and default2 were achieved using down sampling and 60 boosted iterations for default1 and 50 iterations for default2. Again the key results are the validation results which were in the lower range of reported results. This may because balanced accuracy is reported and to achieve an improvement in balanced accuracy a wider range of tuning parameters may be required or a different logistic regression algorithm may be needed.

Another popular metric used to measure performance is AUC and the aim is to have the highest AUC value as possible. Zekić-Sušac et al. (2016) measured this in their study on company growth and reported 0.684 for a model without factors and 0.675 for a model with factors. In Ząbkowski & Szczesny's (2012) study on insolvency in a telecommunications firm, they reported a AUC of 0.886 for ANN train and 0.883 for ANN test. Wray et al. (2010) used AUC for medical diagnosis when predicting different types of diseases using a classifier with a genetic predictor and achieved a wide range of AUC values, the highest was 0.95 and the lowest was 0.15.

For this particular study, the highest AUC for ANN default1 was 0.925 for train and 0.837 for validation and for default2 was 0.956 for train and 0.838 for the validation dataset. The average seen for ANN default1 for train was 0.902 and 0.806 for test and the average seen for default2 for train was 0.918 and 0.811 for test. Again the results for validation are the results to compare with other models and these compare favourably to

previous studies. In terms of optimum tuning parameters the best results were achieved using the same tuning parameters as for balanced accuracy which were SMOTE, initialisation parameters of (-0.4,0.4) and 29 hidden neurons for default1 and 21 for default2. The results from this particular study had a much greater difference between training and validation than Ząbkowski & Szczesny (2012). Differences in results between train and validation are normally seen as validation is performed on unseen data and it's unusual to see such similar results for train and validation as in Ząbkowski & Szczesny's (2012) study.

For tests on an LR model Zekić-Sušac et al. (2016) achieved an AUC of 0.735 for a model with no factors and 0.573 for a model with factors. For this particular experiment, the maximum AUC achieved for LR for default 1 was 0.762 for train and 0.728 for validation and for default2 was 0.870 for train and 0.773 for validation. For default1 the average seen was 0.701 for train and 0.651 for validation and for default2 the average seen was 0.835 for train and 0.728 for validation. It's worth noting that again the highest validation results for both default1 and default2 were achieved using down sampling with 60 boost iterations for default1 and 50 for default2. Again when comparing the validation results in this experiment against what other studies have reported the results are about average and a wider range of tuning on boosted iterations or different algorithms may have been required to achieve better results.

Another popular measure to compare classifier performance is the F1 score and again the aim is to have the highest F1 score possible. Page et al. (2015) used this to assess the performance of a number of different classifiers such as k nearest neighbour, logistic regression and support vector machines. They found that logistic regression had the highest average F1 score of 0.912 when predicting medical seizures in patients although their sample size of 10 was very small. Another study reported F1 scores in the range of 0.207 to 0.850 when predicting text classification with support vector machines and Naïve Bayes (Zhang et al., 2015).

This experiment achieved a max F1 score for ANN default1 of 0.943 for training and 0.940 for validation and an average of 0.891 for training and 0.888 for validation. The results for LR were lower at 0.847 for training and 0.845 for validation and an average

of 0.763 for train and 0.762 for validation. For default2 the results were similar to default1 at 0.960 for ANN training and 0.958 for ANN validation and an average of 0.904 for training and 0.903 for validation. In terms of tuning, SMOTE sampling produced the best F1 score but this time the initialisation parameters were (-0.2, 0.2) with 27 hidden neurons for default1 and 24 for default2. This time the results for LR were still lower but much closer to the ANN results at 0.945 for train and 0.945 for validation with an average of 0.855 for train and 0.854 for test. Again down sampling gave the best result for default1 with 30 boosted iterations but for default2 the best result was achieved using SMOTE sampling and 50 boosted iterations. The F1 score performance results for this experiment compare favourably to the results for other studies.

There are a number of studies which compared the performance of the LR and ANN models. Zekić-Sušac et al. (2016) ran a statistical test of difference on the best ANN to the best LR when predicting company growth and their study found that the results were not statistically different. They also ran another test called McNemar's test which is suitable for testing differences in nonparametric data and found that there was no statistically significant difference in how the ANN and LR models classified company growth. Gschwind (2007) compared a number of models including the ANN and LR models to determine which model performed best when predicting late payment of tenants renting properties. A performance measurement called lift was used to compare the models which in this case was the proportion of tenants in the top X deciles predicted to have a late payment, divided by the proportion in the general population. A good classifier should have a higher proportion in the top X deciles than the general population and in this study the ANN model gave the highest lift. However this study did not report any statistical tests of differences between the ANN and LR results so it's not known if the differences reported were statistically significant. They also did not report using cross validation but they did run their performance measurements on the validation dataset using the best configuration found during training. Hsiao & Whang (2009) configured a number of prediction models including ANN and LR to predict financial insolvency for life insurers and reported that ANN achieved a higher accuracy than LR at 95%. However again this study did not report performing any statistical tests on the

differences in results between the models to determine were the differences statistically significant.

For this particular study three metrics underwent a paired t test on the differences in validation results to determine where they statistically significantly and the results for all three tests show that the mean of the ANN results were statistically significantly different to the mean for LR the metrics tested. This is different to what Zekić-Sušac et al. (2016) reported although a review of their study showed they only ran the difference on the best accuracy for ANN and LR while this study ran the difference on all 16 experiments for 3 metrics including accuracy. Overall these results show that for this data ANN outperformed logistic regression in the 3 key metrics that were measured. These findings are context specific and in this case the context is chit funds from a digital chit fund operator in India.

Lastly it's worth noting the factor analysis played a key role in this analysis to reduce the number of features to an amount that was suitable for modelling. Zekić-Sušac et al. (2016) study used factor analysis as well to explain 99% of the variance in their data but they found that results for all variables were better when not using factors because the t test on difference in proportions was used and was significantly more accurate. This particular study didn't run tests without factors so no specific comparisons to their study can be made for this.

5.2 Main Contributors to Principle Components

In the last chapter it was seen that there is very little difference in the contribution of the top 20 features to the first 30 principal components that are used to predict default. The top ten contributors are RelCode, NPay, PPin, CPin, Status, AucDay, SubNo, TransType, RcptMode and SecurityMode. It's interesting that RelCode is the highest contributor as Company A were very interested in the contribution of this feature. They wanted to know was this an important feature in predicting default and the results show it is the highest contributor to the principle components that are used to predict default. This feature is an id for each family name and it originally had 50% coverage. The remaining 50% was derived using the parent name and age to make them unique,

bringing the coverage up to 100%. The contribution of this feature shows that family name is a key contributor to default.

NPay is the next highest contributor and this represents monthly net income. It makes sense that monthly net income is a key contributor to both types of default and this shows that it is. PPin and CPin are zip codes of the permanent address provided by the parent and the customer and came in third and fourth overall which shows that the address of the parent and customer is a key determinant of whether a customer will default. RcptMode is the mode of payment such as cheque or cash and SecurityMode is the form of collateral taken from the subscriber at the time of disbursing the prize money. These results show that mode of payment and collateral taken are important contributors to the principle components that are used to predict default. AucDay is the day of the month that the auction is held and while this may lead to a few days default if a customer is paid at the end of the month and the auction day is before this, this doesn't explain why a customer would be 90 days in default.

There are a few surprising results such as SubNo and TransType are main contributors to the principal components for default. SubNo is the unique id of the customer in the chit group who wins the auction and TransType is the type of transaction. Neither of these feature seem like good predictors of default but they have scored well in their contributions. Company A indicated that TransType should always have a value of sub for subscription but in fact a little over 10% of cases have a value of SBR and transtype is a high contributor to the principle components that are used to predict default.

There are some interesting results in contributors 11 to 21, many of these are related to income. For example DesigID is job type and this ranks 18 out of 52 in terms of importance for predicting default. Another related feature is BNatureID which explains where customers get their income such as husband Income and rental income or from their own job such as gold jewel manufacturer. This scores 12 out of 52 so it's more important than job type. This could be because it contains a bit more useful information such as if they have other sources of income such as rental income. IncomeSrc is another feature and that specifies whether it's the customers own income or guardian income and that came 19th so one places less then job type. BPay is monthly basic income and

came 11th and AIncome is annual income and also 15th overall. It makes sense that income is an important contributor to the principle components used to predict default and this confirms that it is.

There are other interesting results such as the actual chit group is an important contributor and comes 13th. This suggests that certain chit groups are more likely to default than others. This may be because certain chit groups are in more economically disadvantages areas or they put pressure on potential defaulters to pay on time or there is someone who will pay for the chit group members when they can't. BranchCode is another related feature that indicates which branch the chit group is in and that came 16th. This shows that like the chit group the actual branch is important and this may be because there are different policies or staff in place in the branches that spot potential defaults or coach the chit fund members in how to avoid default.

Age of the customer and age of the first nominee and age of the second nominee are important contributors coming 20th, 21st and 25th respectively. This is an interesting finding and suggests that age is a reasonably important contributor for default. This may be because they have access to more income at certain ages such as middle age than other ages such as old age. Surprisingly the actual customer is not an important contributor to predicting default, coming 29th overall and the ID of a customer in a group which is measured by subscriberNo positioned better at 28th. This is unexpected and it shows that there are more important predictors than the actual customer such as income, relations and age.

Features that had little or no contribution to the principle components include features such as Commission, AuctionAmount, ChitAmount, Prize and PRZPMTFlag which came 52th, 51st, 50th, 48th and 37th respectively. Commission is the amount of commission the chit fund organiser gets paid per auction, prize is the prize paid to the winner at each auction, PRZPMTFlag is a flag to indicate if a customer won the chit prize, AuctionAmount is the total chit value available at each auction and it's the same as the chit value and ChitAmount. It will be useful to Company A to know that these features which are all an integral part of the chit business process make no contribution to the principle components that are used to predict default. There are other interesting

findings such as ParentType has little or no impact on the principle components as it came 36th overall. ParentType has values such as son of, wife of or daughter of and from the results it sounds like this is more like a type of relation that is sponsoring the customer being in the chit group.

This chapter compared the results seen in the last chapter to other published studies and found the results were favourable for some metrics and average for others. The top ten contributors to the first 30 principle were presented along with other key contributors such as job type and income source. The next chapter is the concluding chapter and will discuss the research overview as well as the contribution and impact of the research.

6. CONCLUSIONS

This chapter presents an overview of the research and explains how the research solves the research problem. The experimental design and results are then presented. The next section discusses the contributions and impact of the research and the last section is suggested future work.

6.1 Research Overview

The four main objectives of this research were 1)Perform a detailed literature review of chit funds, default, prediction models and suitable configurations for prediction models; 2)Statistically analyse the factors that can predict default in chit funds; 3)Evaluate the performance of an ANN and LR model for predicting default using chit fund data from an Indian digital chit fund operator; 4)Provide empirical evidence to accept or reject the null hypothesis based on the statistically significance difference in results for the two models using a 95% confidence level.

The literature review revealed that chit funds are unique to India and are a type of ROSCA, formed when a group of people come together and agree to pay a specified amount into a fund on a monthly basis for an agreed number of months and this fund is won by one chit fund member each month. Default in chit funds occurs when a chit fund member is late with their monthly payment and the two defaults measured in this study were 30 days late payment and 90 days late payment.

The data was provided by a chit fund operator and was spread over thirteen spreadsheet files that included transaction files covering the previous 9 years. Initially the data was profiled using a tool called DataCleaner which showed that some features in the files were mostly blank and highlighted a few data quality issues. Then a data integration graphical tool called Kettle was used to do a number of tasks including merging and cleaning the data. The values in the final file were then reviewed and features that only had one value, or were duplicates of other features created when merging files or had no useful information, were discarded at this stage.

R studio was used to pre-process and model the data. Pre-processing included replacing the missing values with a specified value, centring and scaling the data which normalised it for the next step. The next step was principle component analysis which was used to reduce the feature set down to enough principle components to explain 95% of the data. It was found through scree plot analysis and analysis of the cumulative variance that the first 30 principle components were required for this.

The literature review showed that two of the most popular and accurate methods for predicting default are ANN and LR. Experiments were ran for each of the on both the ANN and LR modes. Results were captured and graphs were drawn for each experiment and these results will be summarised in section three in this chapter. This research focused on tuning the number of hidden neurons and the initialisation weights in the ANN model and the number of boosted iterations in the LR model.

6.2 Problem Definition

Company A use domain knowledge of chit fund employees to determine the risk of a loan default or late payment of a chit fund subscriber. The problem is that this domain knowledge can take years to develop and is lost with employee turnover. To overcome this issue Company A needed an automated solution that determines the factors that contribute most to default so they can use this knowledge to screen potential chit fund customers more accurately. This research helped solve that problem by statistically analysing the feature set provided to determine key contributors to default.

6.3 Design/Experimentation, Evaluation & Results

In this study the performance of an ANN multi-layer perceptron model and a LR (logitboost) model was evaluated to determine which had the best performance for predicting default in chit funds. Two types of default were used in this study, the first was late payment of 30 days and the second was late payment of 90 days. The target classes were highly imbalanced and only 1% of instances were defaulted for at least 30 days and 0.2% were defaulted for at least 90 days. Two sampling methods were used to overcome the class imbalance and these were SMOTE sampling and down sampling.

Random sampling was used to break the dataset up into 70% for training and the remaining 30% for validation. K fold cross validation was used on the training dataset to assess performance of the tuning parameters. The validation dataset was used to assess performance of both algorithms. PCA was used to reduce the feature set and still explain 95% of the variance in the data. In this study 16 experiments were ran, 8 for each of the two defaults. The three key metrics that were measured for these experiments were balanced accuracy, AUC and F1 score. After adjusting the p value for Bonferroni's adjustment statistical significance was set to 0.003 when comparing results from multiple experiments.

The results of these experiments revealed that the best balanced accuracy was better for ANN than LR (83% vs 77%), the best AUC was also higher and better for ANN than LR (0.84 vs 0.77) as was the best F1 score (0.958 vs 0.945). The averages for each of the results were also higher for ANN than LR. Statistical analysis using a paired t test between results of ANN and LR for AUC, balanced accuracy and F1 score showed that there was a statistically significant difference in the results and the one tailed test results showed that the mean of ANN results was greater than the mean of the LR results for all 3 metrics.

These results will be useful to Company A as previously they were relying on employee experience to know the important contributors to default and now they will have quantifiable statistical results. Previously when employees left the business their valuable knowledge was lost so this will help overcome that. The first thing these results show is that there is little difference in the top twenty contributors to default. Expected features such as family id, net monthly income and permanent address are important features when determining default. So too are the mode of payment and the collateral taken. There are unexpected results such as unique id of the customer in the chit group and the type of transaction are both important contributors.

As expected, income and income sources are important contributors and a number of these related features score well for contribution to principle components. Certain chit groups and certain chit branches are more likely to default than others. Company A could use this information to determine are there differences in how chit groups or branches

are run or are the only differences due to experience of staff. This could be used to drive an update of policies or increase staff training in areas of poor performance. Key features in the business such as chit amount, auction amount and prize make very little contribution to predicting default while age did make a good contribution. The correlation results show a weak positive association of certain features with both types of default. Age and the prize money flag are negatively associated with both types of default. This suggests that as a chit fund member gets older they are less likely to default and when a chit fund member wins the prize they are less likely to default.

6.4 Contributions and impact

The main contribution this study has are the one tailed t test experimental results indicating that there was a statistical significant difference in means between ANN and LR for balanced accuracy, AUC and F1 score and that the true difference in means is greater than 0. This means that ANN was shown to perform better than LR for all three metrics. Previous published studies compared the performance of these two machine learning algorithms and either found the difference in performance not to be statistically significant or did not run a statistical test on the difference. To the best of my knowledge, this is the first study to find a difference in the results for ANN and LR that was statistically significant.

Statistical significance was initially set at 0.05 but after Bonferroni's adjustment for the 16 experiments this was reduced to 0.003 and all the p values for the two tailed test and one tailed test (testing mean of ANN was greater than LR), were less than 0.003. These results show that for this study the null hypothesis can be rejected and the alternative hypothesis can be accepted. The alternative hypothesis is:

H_a: considering demographic, transactional, referral history and family connection factors, an artificial neural network model will statistically enhance (using 95% CI), the prediction of risk of default and late payments as compared to considering the same factors with a logistic regression model using area under the curve and balanced accuracy.

The next contribution this study has is in analysing chit fund data. Of the journal articles reviewed for this study, there were no studies found that analysed reasons for late payment or default for chit funds. However there were some studies on reasons for late payment in group lending which are similar to chit funds. The results of these experiments showed that there was very little difference in the contribution of the top 20 features to the first 30 principal components, which were used to predict default.

The actual experimental results achieved only average results for balanced accuracy and AUC when compared to previous studies and this suggests that more tuning of parameters or different algorithms are required. Interestingly the F1 score results were higher than two previous studies, although these studies were not measuring default or late payment.

6.5 Future Work & recommendations

There are a number of items that could be considered for future work. Company A have indicated they would like a risk score for each individual customer. This risk score will be similar to a credit score used to make decisions about whether to grant a loan or not, but it will vary over time even when all other attributes remain constant. An automated method of computing the risk score will enable chit funds to scale up faster and ensure that domain knowledge stays within the company. For each feature, specific data ranges and categories will be provided and for each specific category a score will be provided. For each potential chit fund member, the scores can be added over all the features and categories and based on the total risk score a decision can be made on whether or not to approve a new chit fund member.

Future work could also focus on correcting the limitations of this study which are its focus on 2 types of default only and not investigating any interactions between the features. Interesting interactions to study include a) how does the impact of having a number of family members in the same group affect default rates; b) how does screening affecting rates of default (could be measured by if there is a nominee); c) how does screening by a relative affects default. It would be interesting to determine the features

that contribute to other types of late payment such as late payment of 1 day and late payment of 10 days and the features that contribute to early payment.

BIBLIOGRAPHY

Al-Azzam, M., Carter Hill, R., & Sarangi, S. (2012). Repayment performance in group lending: Evidence from Jordan. *Journal of Development Economics*, *97*(2), 404–414. https://doi.org/10.1016/j.jdeveco.2011.06.006

Allaby, M. (2010). Eigen vector. In *A Dictionary of Ecology*. Oxford University Press. Retrieved from http://0-www.oxfordreference.com.ditlib.dit.ie/view/10.1093/acref/9780199567669.001.0001/acref-9780199567669-e-1815

Ardener, S., & Burman, S. (1995). Women making money go round : ROSCAs revisited | Titel. Retrieved April 25, 2018, from http://library.wur.nl/WebQuery/titel/878607

Bapna, R., Goes, P., Kwok Kee Wei, & Zhongju Zhang. (2011). A Finite Mixture Logit Model to Segment and Predict Electronic Payments System Adoption. *Information Systems Research*, *22*(1), 118–133. https://doi.org/10.1287/isre.1090.0277

Bardak, S., Tiryaki, S., Bardak, T., & Aydin, A. (2016). Predictive Performance of Artificial Neural Network and Multiple Linear Regression Models in Predicting Adhesive Bonding Strength of Wood. *Strength of Materials*, *48*(6), 811–824. https://doi.org/10.1007/s11223-017-9828-x

Bellotti, T., & Crook, J. (2009). Support vector machines for credit scoring and discovery of significant features. *Expert Systems with Applications*, *36*(2, Part 2), 3302–3308. https://doi.org/10.1016/j.eswa.2008.01.005

Bergmeir, C., Benítez, J. M., team), A. Z. (Part of original S. development, team), N. M. (Part of original S. development, team), G. M. (Part of original S. development, team), M. V. (Part of original S. development, … Version 4.3), A. E.-A.

(Contributors to S. (2017). *RSNNS: Neural Networks using the Stuttgart Neural Network Simulator (SNNS)*. Retrieved from https://CRAN.R-project.org/package=RSNNS

Blagus, R., & Lusa, L. (2012). Evaluation of SMOTE for High-Dimensional Class-Imbalanced Microarray Data. In *2012 11th International Conference on Machine Learning and Applications* (Vol. 2, pp. 89–94). https://doi.org/10.1109/ICMLA.2012.183

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The Balanced Accuracy and Its Posterior Distribution. In *2010 20th International Conference on Pattern Recognition* (pp. 3121–3124). https://doi.org/10.1109/ICPR.2010.764

Deng, X., Liu, Q., Deng, Y., & Mahadevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, *340–341*, 250–261. https://doi.org/10.1016/j.ins.2016.01.033

Fernandez, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, *61*, 863–905.

García, S., & Herrera, F. (2009). Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy. *Evol. Comput.*, *17*(3), 275–306. https://doi.org/10.1162/evco.2009.17.3.275

Geertz, C. (1962). The Rotating Credit Association: A "Middle Rung" in Development. *Economic Development and Cultural Change*, *10*(3), 241–263. https://doi.org/10.1086/449960

Goldblatt, R., You, W., Hanson, G., & K Khandelwal, A. (2016). Detecting the Boundaries of Urban Areas in India: A Dataset for Pixel-Based Image Classification in Google Earth Engine. *Remote Sensing*, *8*, 634. https://doi.org/10.3390/rs8080634

Gorzałczany, M. B., & Rudziński, F. (2016). A multi-objective genetic optimization for fast, fuzzy rule-based credit classification with balanced accuracy and interpretability. *Applied Soft Computing*, *40*, 206–220. https://doi.org/10.1016/j.asoc.2015.11.037

Gschwind, M. (2007). Predicting Late Payments: A Study in Tenant Behavior Using Data Mining Techniques. *Journal of Real Estate Portfolio Management*, *13*(3), 269–288.

Hooper, D. (2012). Exploratory Factor Analysis. *Books/Book Chapters*. Retrieved from https://arrow.dit.ie/buschmanbk/8

Hsiao, S.-H., & Whang, T.-J. (2009). A study of financial insolvency prediction model for life insurers. *Expert Systems with Applications*, *36*(3, Part 2), 6100–6107. https://doi.org/10.1016/j.eswa.2008.07.024

Kelly, R., & McCann, F. (2016). Some defaults are deeper than others: Understanding long-term mortgage arrears. *Journal of Banking & Finance*, *72*, 15–27. https://doi.org/10.1016/j.jbankfin.2016.07.006

Kolodinsky, J. M., & Roche, E. (2009). Objective measures as a predictor of late payments by high-risk borrowers. *International Journal of Consumer Studies*, *33*(5), 591–595. https://doi.org/10.1111/j.1470-6431.2009.00802.x

Krogh, A. (2008). What are artificial neural networks? *Nature Biotechnology*, *26*(2), 195–197. https://doi.org/10.1038/nbt1386

Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, *28*(5). Retrieved from https://doaj.org

Kuhn, M., Cooper, T., Mayer, Z., Ziem, A., Scrucca, L., Tang, Y., … Wing, J. (2018). Package 'caret.'

Li, J., & Ng, C. (2013). The Normalization of Deviant Organizational Practices: The Non-performing Loans Problem in China. *Journal of Business Ethics*, *114*(4), 643–653. https://doi.org/10.1007/s10551-013-1710-6

Louridas, P., & Ebert, C. (2016). Machine Learning. *IEEE Software*, *33*(5), 110–115. https://doi.org/10.1109/MS.2016.114

Mallapragada, P. K., Jin, R., Jain, A. K., & Liu, Y. (2009). SemiBoost: Boosting for Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(11), 2000–2014. https://doi.org/10.1109/TPAMI.2008.235

Matuschek, H., Kliegl, R., Vasishth, S., Baayen, H., & Bates, D. (2017). Balancing Type I error and power in linear mixed models. *Journal of Memory and Language*, *94*, 305–315. https://doi.org/10.1016/j.jml.2017.01.001

Nicholson, J. (2014). Type I error. In *The Concise Oxford Dictionary of Mathematics*. Oxford University Press. Retrieved from http://0-www.oxfordreference.com.ditlib.dit.ie/view/10.1093/acref/9780199679591.001.0001/acref-9780199679591-e-2907

Page, A., Sagedy, C., Smith, E., Attaran, N., Oates, T., & Mohsenin, T. (2015). A Flexible Multichannel EEG Feature Extractor and Classifier for Seizure Detection. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *62*(2), 109–113. https://doi.org/10.1109/TCSII.2014.2385211

Pandey, D. S., Das, S., Pan, I., Leahy, J. J., & Kwapinskia, W. (2016, January 12). Artificial neural network based modelling approach for municipal solid waste gasification in a fluidized bed reactor - ScienceDirect. Retrieved May 5, 2018, from https://0-www.sciencedirect.com.ditlib.dit.ie/science/article/pii/S0956053X1630486X?_rdoc=1&_fmt=high&_origin=gateway&_docanchor=&md5=b8429449ccfc9c30159a5f9aeaa92ffb

Park, C., & Allaby, M. (2017). Correlation. In *A Dictionary of Environment and Conservation*. Oxford University Press. Retrieved from http://0-www.oxfordreference.com/view/10.1093/acref/9780191826320.001.0001/acref-9780191826320-e-1734

Peng, Y., Wang, G., Kou, G., & Shi, Y. (2011). An empirical study of classification algorithm evaluation for financial risk prediction. *Applied Soft Computing*, *11*(2), 2906–2915. https://doi.org/10.1016/j.asoc.2010.11.028

Pereira, S. M. C., & Leslie, G. (2009). Hypothesis testing. *Australian Critical Care*, *22*(4), 187–191. https://doi.org/10.1016/j.aucc.2009.08.003

Polyak, B., & Khlebnikov, M. (2017). Principle component analysis: Robust versions. *Automation & Remote Control*, *78*(3), 490–506. https://doi.org/10.1134/S0005117917030092

Salamon, H., Kaplan, S., & Goldberg, H. (2009). What goes around, comes around: rotating credit associations among Ethiopian women in Israel. *African Identities*, *7*(3), 399–415. https://doi.org/10.1080/14725840903031882

Salzberg, S. L. (1997). On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. *Data Mining and Knowledge Discovery*, *1*(3), 317–328. https://doi.org/10.1023/A:1009752403260

Sun, P., Zhang, T., & Zhou, J. (2014). A Convergence Rate Analysis for LogitBoost, MART and Their Variant. In *ICML Proceedings of the 31st International Conference on Machine Learning* (pp. 1251–1259).

Tsai, M.-C., Lin, S.-P., Cheng, C.-C., & Lin, Y.-P. (2009). The consumer loan default predicting model – An application of DEA–DA and neural network. *Expert Systems with Applications*, *36*(9), 11682–11690. https://doi.org/10.1016/j.eswa.2009.03.009

Upton, G., & Cook, I. (2014). Principal components analysis. In *A Dictionary of Statistics*. Oxford University Press. Retrieved from http://0-www.oxfordreference.com.ditlib.dit.ie/view/10.1093/acref/9780199679188.001.0001/acref-9780199679188-e-1289

Van der Aalst, W. M. P., Rubin, V., Verbeek, H. M. W., Van Dongen, B. F., Kindler, E., & Günther, C. W. (2010). Process mining: a two-step approach to balance between underfitting and overfitting. *Software & Systems Modeling*, *9*(1), 87–111. https://doi.org/10.1007/s10270-008-0106-z

Were, K., Bui, D. T., Dick, Ø. B., & Singh, B. R. (2015). A comparative assessment of support vector regression, artificial neural networks, and random forests for predicting and mapping soil organic carbon stocks across an Afromontane landscape. *Ecological Indicators*, *52*(Supplement C), 394–403. https://doi.org/10.1016/j.ecolind.2014.12.028

Weston, S., & Calaway, R. (2017). Getting Started with doParallel and foreach. *Date of Access*, *30*(03).

Williams, A. (2016, March 27). Everything you did and didn't know about PCA · Its Neuronal. Retrieved April 29, 2018, from http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/

Wong, T. T., & Yang, N. Y. (2017). Dependency Analysis of Accuracy Estimates in k-Fold Cross Validation. *IEEE Transactions on Knowledge and Data Engineering*, *29*(11), 2417–2427. https://doi.org/10.1109/TKDE.2017.2740926

Wray, N. R., Yang, J., Goddard, M. E., & Visscher, P. M. (2010). The genetic interpretation of area under the ROC curve in genomic profiling. *PLoS Genetics*, *6*(2), e1000864.

Yadav, S., & Shukla, S. (2016, August 18). Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification - IEEE Conference Publication. Retrieved April 25, 2018, from https://ieeexplore.ieee.org/document/7544814/

Yu, L., Wang, S., & Lai, K. K. (2008). Credit risk assessment with a multistage neural network ensemble learning approach. *Expert Systems with Applications*, *34*(2), 1434–1444. https://doi.org/10.1016/j.eswa.2007.01.009

Ząbkowski, T. S., & Szczesny, W. (2012). Insolvency modeling in the cellular telecommunication industry. *Expert Systems with Applications*, *39*(8), 6879–6886. https://doi.org/10.1016/j.eswa.2012.01.008

Zekić-Sušac, M., Šarlija, N., Has, A., & Bilandžić, A. (2016). Predicting company growth using logistic regression and neural networks. *Croatian Operational Research Review*, *7*(2), 229–248.

Zhang, D., Wang, J., Zhao, X., & Wang, X. (2015). A Bayesian Hierarchical Model for Comparing Average F1 Scores. In *2015 IEEE International Conference on Data Mining* (pp. 589–598). https://doi.org/10.1109/ICDM.2015.44

APPENDIX

File #	Name of File	Num of rows	Brief description of data in file
1	2500-111 to 2500-A.xlsx	31719	Transaction file containing details like date of transaction, customer number, customer name and branch name.
2	3000-151 to 375-A.xlsx	24721	same as 2500-111 to 2500-A.xlsx
3	4000-109 to 5000-176.xlsx	24699	same as 2500-111 to 2500-A.xlsx
4	5000-204 to 50000-CCD.xlsx	43324	same as 2500-111 to 2500-A.xlsx
5	6000-301 to 7500-317.xlsx	39218	same as 2500-111 to 2500-A.xlsx
6	20000-A to 20000-U.xlsx	36621	same as 2500-111 to 2500-A.xlsx
7	100000-AAA to 12000-624.xlsx	40809	same as 2500-111 to 2500-A.xlsx
8	125000-A to 20000-511.xlsx	14733	same as 2500-111 to 2500-A.xlsx
9	Auction Master.xlsx	6215	Contains auction related data and one entry per auction date per group. Example of data is auction date, tender price, commission and instalment number
10	Customer Master.xlsx	6112	Contains customer specific info and entry per customer. Example of data is customer name, customer address, customer city and customer mobile.
11	GroupMast.xlsx	537	Contains info on each chit fund and one line per group. Example of data is group number, chit start date, first auction date and the number of instalments in the chit group.
12	Subscriber master.xlsx	12652	Similar to customer master but as a a subscriber is a chit fund member more fund specific such as date when subscriber entered into chit fund, nominee and relation to the nominee.
13	TenderReceived.xlsx	24139	Chit tender specific info such as the tender id, auction id, subscriber id and group id.

Table A.1: Details of the 13 input files

Field Name	Why dropped
age3	only 38 out of 140,000 values populated
Age_1	duplicate field so not needed
AuctionID_1	duplicate field so not needed
BNature	used id field for these instead
BNatureName	used id field for these instead
BranchCode_1	duplicate field so not needed
BranchName	used id field for these instead
CCity	used Cpin instead
Cdist	usinc Cpin instead
CMobile1	unique identifiers not needed
CMobile2	unique identifiers not needed
CPhone	unique identifiers not needed
CanCode2	duplicate field so not needed
CompComm	same value for all instances
CustomerName	unique identifiers not needed
CustomerName_1	unique identifiers not needed
Dept	used id field for these instead
Desig	used id field for these instead
DesigName	used id field for these instead
Discount_1	duplicate field so not needed
EMailID	unique identifiers not needed
Expr1	same value for all instances
FirmCode	same value for all instances
FirmName	unique identifiers not needed
GroupID	used GroupNoID instead
GroupNo	used GroupNoID instead
GroupNo_1	used GroupNoID instead
GroupNo_1_1	used GroupNoID instead
GroupNo_2	used GroupNoID instead
GroupNo_3	used GroupNoID instead
IDCard	same value for all instances
InstNo_1	duplicate field so not needed
JVNo	Not a useful id
JVNo_1	duplicate field so not needed
LandMark	all nulls
MCustCode	not needed as custid instead which is better

Nominee1	unique identifiers not needed
Nominee2	unique identifiers not needed
Nominee3	unique identifiers not needed
OpenBal	same value for all instances
PANNo	same value for all instances
PCity	used Ppin instead
Pdist	used Ppin instead
PMobile1	unique identifiers not needed
PMobile2	unique identifiers not needed
PPhone	unique identifiers not needed
ParentName	unique identifiers not needed
ParentName_1	unique identifiers not needed
RelCode_1	duplicate field so not needed
relation3	only 38 out of 140,000 values populated
Remarks	no useful info
RepBy	all instances null
RptAmt	same value for all instances
Status_1	duplicate field so not needed
SubNo_1	duplicate field so not needed
SubscriberNo_1	duplicate field so not needed

Table A.2 List of features dropped in the early stages of pre-processing

Field Name	**Reason Removed**
AuctionDate	Date field not needed
ChitTransDate	Date field not needed
ComDate	Date field not needed
EnrlDate	Date field not needed
FADate	Date field not needed
NextAucDate	Date field not needed
PRZDate	Date field not needed
PayDate	Date field not needed
RealDate	Date field not needed
RtrmDate	Date field not needed
TermDate	Date field not needed

Table A.3 List of date features removed

	st Dev	eigenvalue	variance in %	cumulative Variance as a %
PCA 1	3.158482	9.9760	19.1846	19.1846
PCA 2	2.240128	5.0182	9.6503	28.8350
PCA 3	1.854082	3.4376	6.6108	35.4458
PCA 4	1.652303	2.7301	5.2502	40.6960
PCA 5	1.532695	2.3492	4.5176	45.2136
PCA 6	1.432581	2.0523	3.9467	49.1603
PCA 7	1.379011	1.9017	3.6571	52.8173
PCA 8	1.3362	1.7854	3.4335	56.2509
PCA 9	1.258606	1.5841	3.0463	59.2972
PCA 10	1.168924	1.3664	2.6277	61.9248
PCA 11	1.093395	1.1955	2.2991	64.2239
PCA 12	1.073395	1.1522	2.2157	66.4396
PCA 13	1.034119	1.0694	2.0565	68.4962
PCA 14	1.023649	1.0479	2.0151	70.5113
PCA 15	1.02151	1.0435	2.0067	72.5180
PCA 16	1.007098	1.0142	1.9505	74.4685
PCA 17	0.991571	0.9832	1.8908	76.3593
PCA 18	0.981963	0.9643	1.8543	78.2136
PCA 19	0.976621	0.9538	1.8342	80.0478
PCA 20	0.954983	0.9120	1.7538	81.8016
PCA 21	0.932808	0.8701	1.6733	83.4750
PCA 22	0.924514	0.8547	1.6437	85.1187
PCA 23	0.905202	0.8194	1.5758	86.6944
PCA 24	0.89897	0.8081	1.5541	88.2485
PCA 25	0.829149	0.6875	1.3221	89.5706
PCA 26	0.824162	0.6792	1.3062	90.8769
PCA 27	0.78351	0.6139	1.1806	92.0574
PCA 28	0.772123	0.5962	1.1465	93.2039
PCA 29	0.731276	0.5348	1.0284	94.2323
PCA 30	0.717846	0.5153	0.9910	95.2233

Table A.4 Statistics for first 30 PCAs

	Feature	% Cont
1	RelCode	3.3317
2	NPay	3.3304
3	PPin	3.3264
4	CPin	3.3241
5	Status	3.3180
6	AucDay	3.3163
7	SubNo	3.3019
8	TransType	3.3004
9	RcptMode	3.2900
10	SecurityMode	3.2751
11	BPay	3.2664
12	BNatureID	3.2333
13	GroupNoID	3.0985
14	CanCode	3.0801
15	AIncome	3.0165
16	BranchCode	3.0076
17	Capital	2.9996
18	DesigID	2.7509
19	IncomeSrc	2.5908
20	Age1	2.4674
21	Age	2.2515
22	ACClosed	2.1869
23	UnDivCO	1.6844
24	Relation2	1.6724
25	Age2	1.6661
26	CustCode	1.6622
27	ChitNo	1.6615
28	SubscriberNo	1.6615
29	CustomerCode	1.6549
30	UnDivBF	1.6336
31	Relation1	1.6240
32	InstNo	1.5530
33	NOJ	1.4358
34	NoofInst	1.3535
35	CompTicketNo	1.3535
36	ParentType	1.2109
37	PRZPMTFlag	1.1638
38	SubID	0.8940
39	ChitTransID	0.8621
40	AuctionID	0.8584
41	TransID	0.8354
42	DivDistribute	0.8026
43	Dividend	0.8026

44	DivSubscriber	0.7896	
45	NextSubAmount	0.6264	
46	Discount	0.6126	
47	SubAmount	0.5991	
48	Prize	0.5358	
49	InstAmount	0.4892	
50	ChitAmount	0.4125	
51	AuctionAmount	0.4125	
52	Commision	0.4125	

Table A.5: Contribution of each feature to top 30 PCA

Main variable	Other feature	Correlation	Pvalue
Default1	InstNo	0.0716	0.0000
Default1	TransID	0.0501	0.0000
Default1	ChitTransID	0.0412	0.0000
Default1	AuctionID	0.0339	0.0000
Default1	UnDivBF	0.0310	0.0000
Default1	UnDivCO	0.0289	0.0000
Default1	PRZPMTFlag	-0.0267	0.0000
Default1	Dividend	-0.0262	0.0000
Default1	DivDistribute	-0.0262	0.0000
Default1	TransType	0.0260	0.0000
Default1	DivSubscriber	-0.0239	0.0000
Default1	Age	-0.0231	0.0000
Default1	Prize	0.0206	0.0000
Default1	Discount	-0.0190	0.0000
Default1	CustCode	0.0189	0.0000
Default1	CustomerCode	0.0189	0.0000
Default1	Status	0.0188	0.0000
Default1	NextSubAmount	0.0186	0.0000
Default1	SubAmount	0.0177	0.0000
Default1	CanCode	0.0171	0.0000
Default1	ParentType	-0.0152	0.0000
Default1	NOJ	0.0138	0.0000
Default1	Age1	-0.0133	0.0000
Default1	SecurityMode	0.0127	0.0000
Default1	Commision	0.0126	0.0000
Default1	AuctionAmount	0.0126	0.0000
Default1	ChitAmount	0.0126	0.0000
Default1	InstAmount	0.0121	0.0000
Default1	RcptMode	-0.0097	0.0004

Table A.6: Top 30 correlated features to Default1

name1	name2	estimate	p.value
Default2	TransID	0.0329	0.0000
Default2	InstNo	0.0309	0.0000
Default2	SubAmount	0.0230	0.0000
Default2	ChitTransID	0.0228	0.0000
Default2	Prize	0.0205	0.0000
Default2	NextSubAmount	0.0197	0.0000
Default2	AuctionID	0.0171	0.0000
Default2	InstAmount	0.0164	0.0000
Default2	Commision	0.0163	0.0000
Default2	AuctionAmount	0.0163	0.0000
Default2	ChitAmount	0.0163	0.0000
Default2	UnDivCO	0.0159	0.0000
Default2	SecurityMode	0.0147	0.0000
Default2	UnDivBF	0.0146	0.0000
Default2	Age	-0.0127	0.0000
Default2	PPin	-0.0116	0.0000
Default2	AIncome	0.0108	0.0001
Default2	CustCode	0.0101	0.0002
Default2	CustomerCode	0.0100	0.0002
Default2	PRZPMTFlag	-0.0096	0.0004
Default2	SubscriberNo	0.0091	0.0008
Default2	ChitNo	0.0091	0.0008
Default2	CanCode	0.0080	0.0031
Default2	Age1	-0.0078	0.0039
Default2	IncomeSrc	-0.0078	0.0041
Default2	Dividend	-0.0073	0.0073
Default2	DivDistribute	-0.0073	0.0073
Default2	GroupNoID	0.0072	0.0082
Default2	BranchCode	0.0065	0.0159
Default2	ParentType	-0.0064	0.0193

Table A.7: Top 30 correlated features to Default2

#Code file to deal with default1

```
wd="C:/CK/College/Masters_DataAnalytics/Masters_Semester6/Dissertation/Analysis
/FactorReduction"
setwd(wd)
library(caret)
rm(list=ls())
chitLoans=read.csv("FinalOutput.csv",header=T,sep=",")
attach(chitLoans)
data1 <- data.frame(chitLoans)
#throws a warning message as some variables have large number of NAs in them
TestA=preProcess(data1,method= c("center", "scale","knnImpute"))
result1A=sapply(data1, function(x) sum(is.na(x)))
result1A

#removing some variables as huge number of NAs in them
#warning Std. deviations could not be computed for: SecurityMode, TenderID, SNo,
PriceFlag
#removed as zero variance when
sampling=PRZFlag,OpenInstNo,ITPayee,DPD,ClosedGroup,Intimation
data2 <- subset(data1, select = -
c(TenderID,SNo,PriceFlag,PRZFlag,OpenInstNo,ITPayee,DPD,ClosedGroup,Intimati
on))
#checks for NAs only
result1A=sapply(data2, function(x) sum(is.na(x)))
result1A
#round all the data to 4 decimal places
data2=round(data2,4)
#now set nas to 0
data2$RcptMode[is.na(data2$RcptMode)] <- 0
data2$Status[is.na(data2$Status)] <- 0
data2$Relation1[is.na(data2$Relation1)] <- 0
data2$Relation2[is.na(data2$Relation2)] <- 0
data2$RelCode[is.na(data2$RelCode)] <- 0
```

```
data2$PPin[is.na(data2$PPin)] <- 0
data2$CPin[is.na(data2$CPin)] <- 0
data2$SecurityMode[is.na(data2$SecurityMode)] <- 0

#setting up the date time for all the output files
dateTime=as.character(Sys.time())
dateTime=gsub(pattern=":",replacement="_",x=dateTime)
dateTime=gsub(pattern=" ",replacement="_",x=dateTime)
dateTime=gsub(pattern="-",replacement="_",x=dateTime)
ModelRunDetailsFile=paste(dateTime,"ModelRunDetails",sep="_")
ModelRunDetailsFile=paste(ModelRunDetailsFile,".csv",sep="")
TraningFile=paste(dateTime,"trainingData",sep="_")
TraningFile=paste(TraningFile,".csv",sep="")

inTrain=createDataPartition(y=data2$Default1,times=1,p=7/10,list=FALSE)
training=data2[inTrain,]
write.csv(training,TraningFile)
testing=data2[-inTrain,]
nrow(training)
nrow(testing)
TrainNoPred= subset(training, select = -c(Default1,Default2))
TestNoPred=subset(testing, select = -c(Default1, Default2))

#setting some useful parameters
samplingMethod="smote"
cvnumber=3
PCAThresh = 0.95

TrainANN=function(x)
{

  StartDetails=paste("start of MLP process",Sys.time())
  write.csv(StartDetails,ModelRunDetailsFile)
```

```
 SamplingDetails=paste("Sampling used is ",samplingMethod)
 write.table(SamplingDetails,ModelRunDetailsFile,append=TRUE)
 SamplingDetails=paste("Cross validation used is",samplingMethod)
 write.table(cvnumber,ModelRunDetailsFile,append=TRUE)
 ctrl <- trainControl(method = "cv",
                      number = cvnumber,
                      classProbs = TRUE,
                      preProcOptions = list(thresh = PCAThresh), #or list(pcaComp = 30)
                      verboseIter = TRUE, #keeps track of progress
                      summaryFunction = twoClassSummary,
                      sampling = samplingMethod)
 set.seed(107)
 tune_Grid1 = expand.grid(size = 1:30)
 Tune_Details=paste("Tune Grid is",tune_Grid1)
 write.table(Tune_Details,ModelRunDetailsFile,append=TRUE,sep=",")
 initFuncParams1 = c(-0.2, 0.2)
 initFuncParamsDetails=paste("Initialisation Function",initFuncParams1)
 write.table(initFuncParamsDetails,ModelRunDetailsFile,append=TRUE,sep=",")
 registerDoParallel(cores=4)        # Registrer a parallel backend for train
 getDoParWorkers()
 TrainANN=train(x=TrainNoPred,y=outcome,method = "mlp",preProcess =
c("center", "scale","knnImpute","pca"),trControl=ctrl,
 tuneGrid=tune_Grid1, learnFunc = "Std_Backpropagation",metric="ROC",
initFuncParams = initFuncParams1)
}



TrainLR=function(x)
{
 StartDetails=paste("start of Log Regression process",Sys.time())
 write.table(StartDetails,ModelRunDetailsFile,append=TRUE)
 SamplingDetails=paste("Sampling used is ",samplingMethod)
```

```
  write.table(SamplingDetails,ModelRunDetailsFile,append=TRUE)
  ctrl <- trainControl(method = "cv", #cross validation
                       number = cvnumber, #number of folds in the k folds cross validation
                       classProbs = TRUE, #determines whether class probabilities should be computed for held-out samples during resample.
                       preProcOptions = list(thresh = PCAThresh), #list(thresh = 0.85), #setting the max num of principle components to use to 30
                       summaryFunction = twoClassSummary,
                      sampling = samplingMethod) #will compute the sensitivity, specificity and area under the ROC curve:
  set.seed(107)
  tune_Grid1 = expand.grid(nIter = c(10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100))
  Tune_Details=tune_Grid1
  write.table(Tune_Details,ModelRunDetailsFile,append=TRUE,sep=",")
  registerDoParallel(cores=4)
  getDoParWorkers()
  TrainLR=train(x=TrainNoPred,y=outcome,method = "LogitBoost",preProcess = c("center", "scale","knnImpute","pca"),trControl=ctrl,metric = "ROC",tuneGrid=tune_Grid1)
 }

 #install.packages("doParallel")
library(doParallel)
outcome=as.factor(training$Default1)
levels(outcome) <- c("NotDefaulted", "Defaulted")

#Call MLP Model
modelMLP=TrainANN(1)
EndDetails=paste("End of MLP process",Sys.time())
write.table(EndDetails,ModelRunDetailsFile,append=TRUE,sep=",")
out=prop.table(table(outcome))
```

```
write.table("Ratio in training data for Default1",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(out,ModelRunDetailsFile,append=TRUE,sep=",")
trainDetils=paste("Number of rows in train is",nrow(training))
write.table(trainDetils,ModelRunDetailsFile,append=TRUE,sep=",")
PCAresults=modelMLP$preProcess$rotation
PCAresults
write.table("PCA Results for MLP",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(PCAresults,ModelRunDetailsFile,append=TRUE,sep=",")
summary(modelMLP)
modelMLP
names(modelMLP)

SizeFold1=NROW(modelMLP$control$index$Fold1)
FoldDetails=paste("Size of Fold1 is =",SizeFold1,sep="")
write.table(FoldDetails,ModelRunDetailsFile,append=TRUE,sep=",")
SizeFold2=NROW(modelMLP$control$index$Fold1)
FoldDetails=paste("Size of Fold2 is =",SizeFold2,sep="")
write.table(FoldDetails,ModelRunDetailsFile,append=TRUE,sep=",")
SizeFold3=NROW(modelMLP$control$index$Fold3)
FoldDetails=paste("Size of Fold3 is =",SizeFold3,sep="")
write.table(FoldDetails,ModelRunDetailsFile,append=TRUE,sep=",")
bestTune=modelMLP$bestTune
write.table("Best tune is ",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(bestTune,ModelRunDetailsFile,append=TRUE,sep=",")

result1=modelMLP$results
result1
write.table("Results are",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(result1,ModelRunDetailsFile,append=TRUE,sep=",")

result1=modelMLP$resample
result1
```

```
write.table("Resample Results are",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(result1,ModelRunDetailsFile,append=TRUE,sep=",")

AnnMLP_File1=paste(dateTime,"ANN_MLP_HiddenUnitsModel",sep="_")
AnnMLP_File1=paste(AnnMLP_File1,".jpg",sep="")
jpeg(AnnMLP_File1)
plot(modelMLP,main="ROC values verus Number of hidden units",col.main="blue",
col.lab="blue")
dev.off()

#predict using trained data
testMLP1=predict(modelMLP)
out2=prop.table(table(testMLP1))
out3=prop.table(table(testing$Default1))
write.table("Ratio of Default1 in test",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(out3,ModelRunDetailsFile,append=TRUE,sep=",")
MLP1_outcome=paste(dateTime,"MLP1_PredictResults",sep="_")
MLP1_outcome=paste(MLP1_outcome,".csv",sep="")
write.csv(testMLP1,MLP1_outcome)

#convert actuals and predicted back to 1 and 0 for results
predicted=ifelse(testMLP1=="Defaulted",1,0)
actuals=training$Default1

library(pROC)
curve1 = roc(response = actuals,
        predictor = predicted)
AnnMLP_ROC=paste(dateTime,"ANN_MLP_ROC_Curve",sep="_")
AnnMLP_ROC=paste(AnnMLP_ROC,".jpg",sep="")
jpeg(AnnMLP_ROC)
plot(curve1,main="ROC Curve of Sensitivity verus Specificity for MLP for training
set",col.main="blue",                col.lab="blue",print.thres="best",
print.thres.best.method="closest.topleft")
```

```
dev.off()
write.table("AUC MLP Training Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(curve1$auc,file=ModelRunDetailsFile,append=TRUE,sep=",")

#output confusion matrix results to file
cv=confusionMatrix(predicted,actuals,mode="everything")
cv
write.table("Confusion Matrix table",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(cv$table,file=ModelRunDetailsFile,append=TRUE,sep=",")
tocsv <- data.frame(cbind(t(cv$overall),t(cv$byClass)))
write.table("Confusion Matrix Results",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tocsv,file=ModelRunDetailsFile,append=TRUE,sep=",")

#predict using hold out test sample
testMLP2=predict(modelMLP,TestNoPred)
out4=prop.table(table(testMLP2))
MLP1_outcomeTest=paste(dateTime,"MLP1_PredictResultsTest",sep="_")
MLP1_outcomeTest=paste(MLP1_outcomeTest,".csv",sep="")
write.csv(testMLP2,MLP1_outcomeTest)

#convert actuals and predicted back to 1 and 0 for results
predictedMlpTest=ifelse(testMLP2=="Defaulted",1,0)
actualsMlpTest=testing$Default1

#library(pROC)
curve1 = roc(response = actualsMlpTest,
        predictor = predictedMlpTest)
AnnMLP_ROC=paste(dateTime,"ANN_MLP_ROC_CurveTestSample",sep="_")
AnnMLP_ROC=paste(AnnMLP_ROC,".jpg",sep="")
jpeg(AnnMLP_ROC)
```

```
plot(curve1,main="ROC Curve of Sensitivity against Specificity for MLP for test set",col.main="blue", col.lab="blue",print.thres="best", print.thres.best.method="closest.topleft")
dev.off()

write.table("AUC MLP Test Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(curve1$auc,file=ModelRunDetailsFile,append=TRUE,sep=",")

#output confusion matrix results to file
cv2=confusionMatrix(predictedMlpTest,actualsMlpTest,mode="everything")
cv2
write.table("Confusion Matrix table Test Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(cv$table,file=ModelRunDetailsFile,append=TRUE,sep=",")
tocsv2 <- data.frame(cbind(t(cv2$overall),t(cv2$byClass)))
write.table("Confusion Matrix Results Test Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tocsv2,file=ModelRunDetailsFile,append=TRUE,sep=",")

#########################################################################
###
#Logistic Regression
#########################################################################
###
modelLR=TrainLR(1)
summary(modelLR)
modelLR

EndDetails=paste("End of LR process",Sys.time())
write.table(EndDetails,ModelRunDetailsFile,append=TRUE,sep=",")
bestTuneLR=modelLR$bestTune
write.table("Best tune is ",ModelRunDetailsFile,append=TRUE,sep=",")
```

```
write.table(bestTuneLR,ModelRunDetailsFile,append=TRUE,sep=",")

resultLR=modelLR$results
write.table("results LR",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(resultLR,ModelRunDetailsFile,append=TRUE,sep=",")

resampleLR=modelLR$resample
resampleLR
write.table("resample Restults LR",ModelRunDetailsFile,append=TRUE,sep=",")
write.table(resampleLR,ModelRunDetailsFile,append=TRUE,sep=",")

testLR1=predict(modelLR)
prop.table(table(testLR1))
LR1_outcome=paste(dateTime,"LR1_PredictResults",sep="_")
LR1_outcome=paste(LR1_outcome,".csv",sep="")
write.csv(testLR1,LR1_outcome)

#convert actuals and predicted back to 1 and 0 for results
predictedLR=ifelse(testLR1=="Defaulted",1,0)
actuals=training$Default1

curveLR1= roc(response = actuals,
        predictor = predictedLR)
ROC_CurveLR=paste(dateTime,"ROC_Curve_LR",sep="_")
ROC_CurveLR=paste(ROC_CurveLR,".jpg",sep="")
jpeg(ROC_CurveLR)
plot(curveLR1,main="ROC Curve of Sensitivity against Specificity for LR training
data",col.main="blue", col.lab="blue",print.thres="best",
print.thres.best.method="closest.topleft")
dev.off()

write.table("AUC LR Training
Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
```

```
write.table(curveLR1$auc,file=ModelRunDetailsFile,append=TRUE,sep=",")

cvLR=confusionMatrix(predictedLR,actuals,mode="everything")
#cv
write.table("Confusion Matrix table",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(cvLR$table,file=ModelRunDetailsFile,append=TRUE,sep=",")
tocsvLR <- data.frame(cbind(t(cvLR$overall),t(cvLR$byClass)))
write.table("Confusion Matrix results",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tocsvLR,file=ModelRunDetailsFile,append=TRUE,sep=",")

#Check predictions using test sample
testLR1_test=predict(modelLR,TestNoPred)
prop.table(table(testLR1_test))
table(testLR1_test)

LR1_outcome=paste(dateTime,"LR1_PredictResults",sep="_")
LR1_outcome=paste(LR1_outcome,".csv",sep="")
write.csv(testLR1_test,LR1_outcome)
#convert actuals and predicted back to 1 and 0 for results
predictedLR=ifelse(testLR1_test=="Defaulted",1,0)
#actuals=testing$Default2

curveLR1= roc(response = actualsMlpTest,
          predictor = predictedLR)
ROC_CurveLR=paste(dateTime,"ROC_Curve_LR_Test",sep="_")
ROC_CurveLR=paste(ROC_CurveLR,".jpg",sep="")
jpeg(ROC_CurveLR)
plot(curveLR1,main="ROC Curve of Sensitivity against Specificity for LR test data",col.main="blue", col.lab="blue",print.thres="best", print.thres.best.method="closest.topleft")
dev.off()
```

```
write.table("AUC LR Test Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(curveLR1$auc,file=ModelRunDetailsFile,append=TRUE,sep=",")

cvLR=confusionMatrix(predictedLR,actualsMlpTest,mode="everything")
write.table("Confusion Matrix table Test Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(cvLR$table,file=ModelRunDetailsFile,append=TRUE,sep=",")
tocsvLR <- data.frame(cbind(t(cvLR$overall),t(cvLR$byClass)))
write.table("Confusion Matrix results Test Sample",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tocsvLR,file=ModelRunDetailsFile,append=TRUE,sep=",")

############################################################
#Compare Models
############################################################

Comparison=resamples(list(NN_MLP=modelMLP,LR_LogitBoost=modelLR))
summary(Comparison)

write.table("Comparison of model run values",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(Comparison$values,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table("Comparison of model run timings",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(Comparison$timings,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table("Comparison of model run methods",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(Comparison$methods,file=ModelRunDetailsFile,append=TRUE,sep=",")

Comparison1=paste(dateTime,"Comparison1",sep="_")
Comparison1=paste(Comparison1,".jpg",sep="")
jpeg(Comparison1)
bwplot(Comparison,main="Comparison of models",col.main="blue", col.lab="blue")
```

```
dev.off()

####################################
#Compare residuals between train and predict
####################################

tTest=t.test(predictedMlpTest - actualsMlpTest, predictedLR - actualsMlpTest, paired =
TRUE)
tTest1=as.data.frame(tTest$p.value)
tTest2=as.data.frame(tTest$estimate)
tTest3=as.data.frame(tTest$conf.int)
tTest4=as.data.frame(tTest$statistic)
tTest5=as.data.frame(tTest$parameter)
tTest6=as.data.frame(tTest$method)

write.table("T test of differences between the models using predicted and actual
outcomes",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tTest1,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tTest2,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tTest3,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tTest4,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tTest5,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(tTest6,file=ModelRunDetailsFile,append=TRUE,sep=",")

#################################
#t test on differences
#################################
modelDifferences=diff(Comparison)
modelDifferences1=as.data.frame(modelDifferences$statistics$ROC$NN_MLP.diff.L
R_LogitBoost$p.value)
modelDifferences2=as.data.frame(modelDifferences$statistics$ROC$NN_MLP.diff.L
R_LogitBoost$estimate)
```

```
modelDifferences3=as.data.frame(modelDifferences$statistics$ROC$NN_MLP.diff.LR_LogitBoost$conf.int)
modelDifferences4=as.data.frame(modelDifferences$statistics$ROC$NN_MLP.diff.LR_LogitBoost$statistic)
modelDifferences5=as.data.frame(modelDifferences$difs)
modelDifferences6=as.data.frame(modelDifferences$confLevel)

write.table("T test of differences using resamples",file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(modelDifferences1,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(modelDifferences2,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(modelDifferences3,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(modelDifferences4,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(modelDifferences5,file=ModelRunDetailsFile,append=TRUE,sep=",")
write.table(modelDifferences6,file=ModelRunDetailsFile,append=TRUE,sep=",")

summary(modelDifferences)
Comparison1=paste(dateTime,"bwPlotOfTtstDifferences",sep="_")
Comparison1=paste(Comparison1,".jpg",sep="")
jpeg(Comparison1)
bwplot(modelDifferences, layout = c(2, 1),
      scales = list(x = list(relation="free")),col.main="blue", col.lab="blue")
dev.off()
```