Indian Vehicle loan Default Prediction & Profit Model

Practical Business Analytics (COMM053)

Group name: Hyperion

Group Members:

Ritin Jaiswal

Tavis Shore

Randall Kakaire

Harry Mahon

Jay Chandel

Beilul Baraki

Yazan Ibrahim

Introduction

Credit lending is one of the most important activities carried out by banks and financial institutions. It has a direct impact on their balance sheet, profitability and risk-bearing capacity. When conducted incorrectly it can result in the bank's insolvency and have even wider implications on the economy, resulting in financial crisis and expensive bailouts for large financial institutions.

The Basel Committee for Banking supervision has outlined that all banks should implement rigorous credit scoring systems in order to estimate the degree of credit risk, and consequently lower their risk exposure [1]. Credit scoring system can be conducted using *Judgemental* or *Statistical* approach. The judgemental approach is based primarily on the policy guidelines of the bank and the experience and intuitions of their credit analysts. This approach is very time consuming and results in inconsistent decision making due to the heavy involvement of the human analyst. Statistical approaches rely on statistical and machine learning techniques (e.g. LR, ANN, DT) to classify applicants mainly into two groups ("Good" or "Bad"). However, there is a strong argument for multi-classification systems, as considering more than just the default and non-default states would result in more accurate tailoring of credit terms and conditions. It would also enable early warning signal detection as most borrowers' transition through different credit profiles during the term of the loan [2].

Identifying the most important features or variables is also very important in building a good credit scoring model. Studies have shown that, by eliminating the ones that are irrelevant, redundant and noise-causing, one ends up with a more accurate, less complex and low dimensional model [2], as well as a better initial solution [3]. Furthermore, it is also understood that ensemble methods give a higher accuracy than that of single models.

Statistical and machine learning techniques are far superior to judgemental approaches as they reduce costs associated with credit analysis, allow for faster decisions and lower risk exposure. Furthermore, as banks become more sophisticated, they would be able to provide customised solution, offering personalised prices and products to each individual [4].

It is important to note that credit scoring models must be very reliable and accurate before being deployed by any financial institution, as getting it wrong could be very expensive and damaging for the business. However, increasing the accuracy of scoring systems even by as low as 1%, has been shown to reduce significant losses for the bank [3].

The credit system in India depends on a number of factors based on which the CIBIL score is generated. The factors on which the credit score depends are the length of the credit history, the past credit accounts, employee type [12].

Project definition

The aim of this project was to process a dataset that lists various details of individuals who have taken out vehicle loans, and to use machine learning to predict whether future loan applicants are likely to default on their loan. In addition to this, we have tried to introduce a profitability model that allows us to predict the extra profit that could be generated by granting loans to individuals that are predicted to default.

We expect that implemented models will predict some false positive values. In other words, it will predict that some applicants will default when given a loan, but in reality, they are likely to repay the loan. We aim to investigate whether granting loans to all applicants results in a gain or loss in profit, by analysing the resulting confusion matrix.

Based on the result of this, we determine whether it is worth implementing a probability measure, describing the likelihood of an applicant defaulting on their loan before increasing the interest rate so that a higher risk results in a higher profitability if the loan is repaid.

The dataset we used was taken from Kaggle, using the link:
https://www.kaggle.com/gauravdesurkar/lt-vehicle-loan-default-prediction.

It includes various details about vehicle loan applicants, as shown in the data dictionary in Table 1 in the appendix, along with a binary indicator of whether the applicant defaulted or not. We use some of these fields to predict whether applicants will default using untested data.

In essence, we aim to investigate the usefulness of implementing a classification of defaulting and non-defaulting loan applicants and explore potential options for increasing profit. We calculate an estimated profit or loss that would ensue from granting loans to all applicants, not just the ones predicted to repay the loan.

R Code

Design Patterns

At the top of each file within the project we have placed details about what the R file is and what its purpose is. This comment indented section includes details about the file including its title, date of creation along with its author/s. The projects code has been separated into three key files. ‘main.R’ is where the program executes from, completing all the necessary functions. ‘nick_code.R’ contains code created by Professor Ryman-Tubb, this includes many functions such as his simple decision tree function and ‘NdetermineThreshold’ which finds the optimum threshold for classification of ‘good’ outcomes. The final file is ‘functions.R’, this contains the functions we have created along with references of code we have taken from an external source. This is where the majority of process of execution is, containing code for pre-processing, modelling and evaluation. Any function with a capital N at the front was created by Professor Ryman-Tubb.

For consistency between variables and constants, we have placed all constant values at the top of the main file, all of which are declared before execution of the main program begins, e.g. ‘DATASET_FILENAME’. The majority variables and field names within data files have also been created with similar syntax, usually with lowercase letters and words separated by an underscore, e.g. ‘loan_default’.

All code, where is standard, has been indented to make the project simpler to interpret for everyone. This includes code written within functions being indented by two spaces and statements within for loops being indented by 2 spaces. The indentation level is the same across the project files. Above all functions is a definition of what the functions purpose is. This also displays the inputs and outputs of the code, where suitable.

Statements written within functions created by the team contain many individual lines of commented information. The purpose of which is to improve readability for other programmers who may wish to follow the functions order of operations. This is also used above the calling of external functions from a library, to give a brief description of why a function may have been called.

For ease of use, all libraries included and used within the project are concatenated into a variable which is then executed, importing the necessary libraries. This takes place before the main() function is executed, ensuring that the required libraries are installed and included before they’re called. This method of including libraries also decreases the effort required to import additional libraries as they may just be added to the end of the list of libraries. The library, ‘pacman’ is loaded separately so that it may be used to import all of the libraries within the list mentioned previously.

The methods of design remaining consistent throughout the entirety of files ensures simplicity, providing the simplest manner for other individuals attempting to follow the direction and execution of code.

Within the code itself, there are a few methods used when attempting to unify the overall format and presentation of the code. This includes techniques such as where specific columns have been identified by name using the ‘table$column’ technique in all cases, not once using the method of table[,’column’]. There are other similar examples of this design tactic. By deciding to use a technique consistently like here, we have helped reduced the codes overall complexity for all potential readers.

Choices and Assumptions

Various assumptions have been made during the creation of the projects code. Firstly, where data has been present within a field and row and within the expected range of data, we have made the assumption that this data has been correctly gathered and is completely accurate and correct.

Another assumption we have made is that future vehicle loan datasets which may be entered into the model will be written in the same format including column names and datatypes within the columns. If true, this will ensure that the pre-processing section will be successfully executed, outputting a cleaned dataset containing only relevant data. For example, for the employment type column, we have assumed that for all future data that may be used as the input, the two options of ‘Salaried’ and ‘Self-employed’ will remain unchanged.

We assumed that whoever may wish to execute this project will be using version 3.6.1 of R. Otherwise some functions and imported libraries may not be wholly compatible.

Main Functions

Execution of the code is placed into four main files, ‘main’ is where overall control of the program is and then three other files which hold functions, one holding Professor Ryman-Tubbs functions provided during previous laboratory sessions, a third containing our teams functions for pre-processing, logistic regression and our decision tree algorithms. The final file holds the teams code for creating a neural network.

Within ‘main.R’, there are two functions, ‘main()’ and ‘process()’. The main function executes the pre-processing algorithm before then executing the modelling algorithm. The function ‘process’ is the first item to execute in the program. Its purpose it to read the dataset into the environment from file before running the pre-processing functions held in ‘functions.R’, using a group of irrelevant fields and fields that need to be normalised for later use. We have selected these fields by using our prior knowledge of the field. E.g. all of our currency fields require normalisation.

Within the ‘function.R’ file, there are five functions which we have created, preprocessing, collate_gov, model_learning, classify_model and forest. The purpose of ‘preprocessing()’ is to remove any fields which won’t affect our model. ‘collate_gov()’ collates all forms of government ID within the dataset into one column and removes the original columns. This makes interpreting whether or not someone has provided their government ID simpler. ‘model_learning()’ runs the initial modeling techniques, logistic regression and classifies the data, before creating a decision tree, before executing the final function which uses the random forest technique to improve the performance of the tree model.

‘myEvaluateClassifier’ and ‘myPerformancePlot’ Functions

Function, ‘myEvaluateClassifier’, takes as parameters the probability of data being ‘class 1’, the data frame upon which the model is to be tested and the threshold for when the data is classed as ‘class 1’ or not. This function returns a confusion matrix which may be used to evaluate the performance of the model. We have used this function once through calling the function ‘myPerformancePlot’. The function ‘myPerformancePlot’ takes two inputs. The probability of a data item being ‘class 1’ and the dataset of which this probability is to be tested against. It returns a list containing evaluation measures of the predicted class probability.

Data Understanding & Visualisation

Our original dataset is a 40.8MB file and contains 233155 rows and 41 columns, and therefore 40potential features excluding the outcome column. This data included both numeric and symbolic type features.

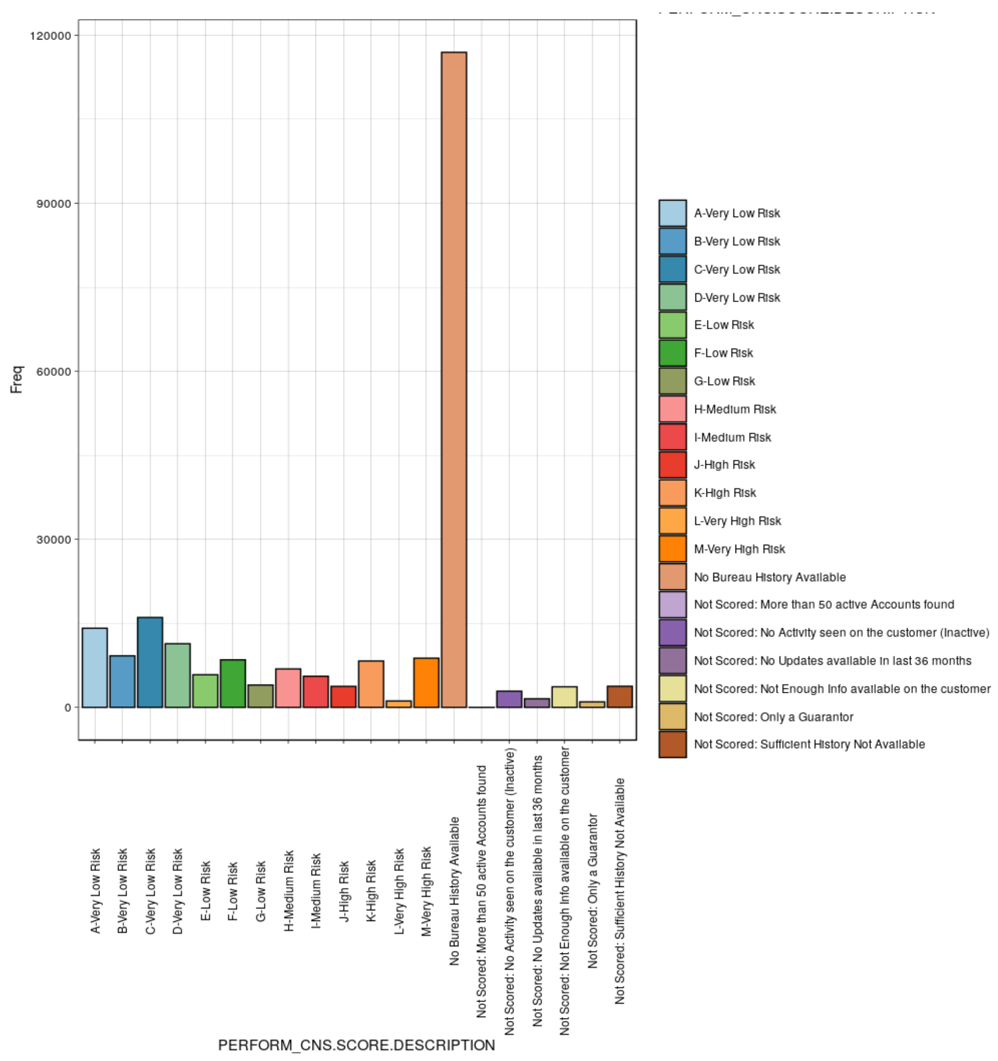


Figure 1. Frequency Plot for PERFORM_CNS.SCORE.DESCRIPTION

By plotting the frequency for the PERFORM_CNS.SCORE.DESCRIPTION column which is the Bureau Score it can be seen that a large proportion of the rows in our dataset contain no bureau history available. However, under visual inspection of our dataset those with no bureau history can still be granted loans suggesting that field with bureau score field with no history should not be taken as a null value.

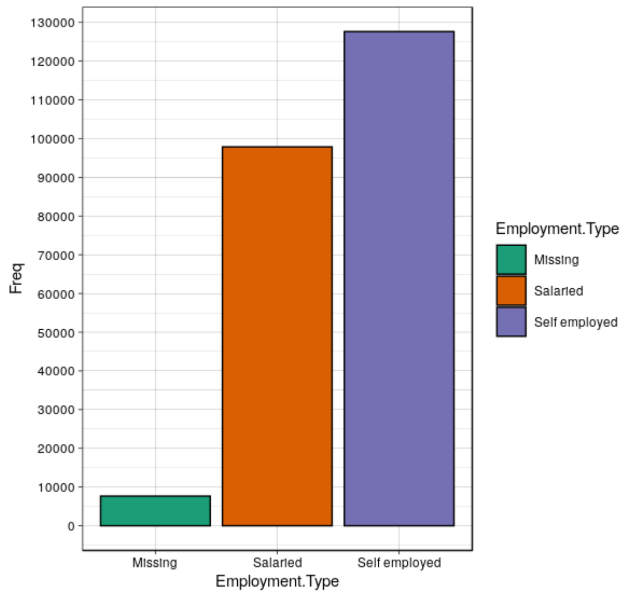


Figure 2. Frequency Plot for Employment.Type

Through further frequency plotting of our dataset it was notable that Employment.Type column contained almost ten thousand missing fields as displayed in Figure 2. These fields required subsequent cleaning prior to model implementation.

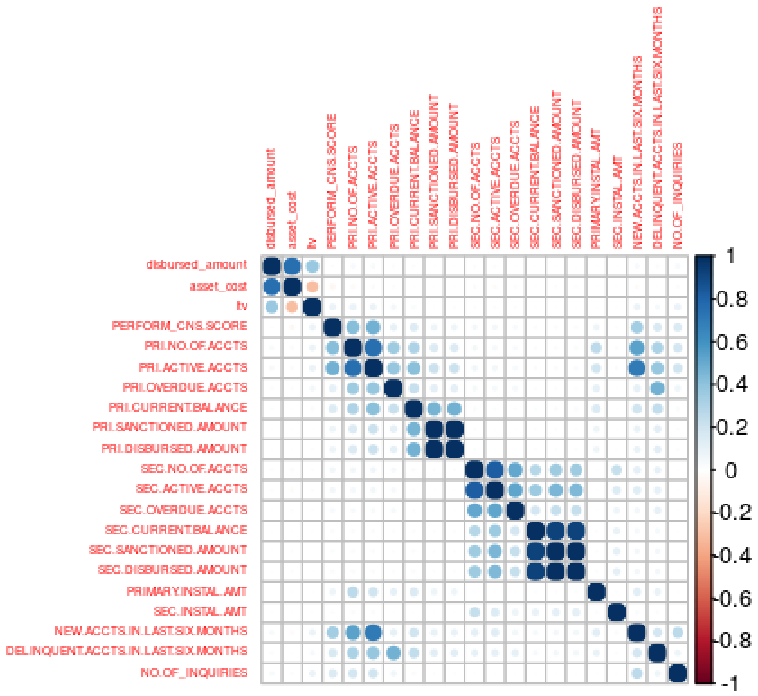


Figure 3. Correlation Matrix for numeric variables

The correlation matrix portrayed in Figure 3 displays a strong correlation between fields of disbursed and sanctioned amounts. This makes logical sense since the loan will only be disbursed if it is first sanctioned. Strong correlations between features are taken into consideration when choosing variables for machine learning models.

Data preparation

In order to have a suitable data set for machine learning algorithms to take place effectively, we used various data cleaning and data processing techniques. In this section the various techniques are discussed in detail, explaining the reason behind their implementation in the context of the business task.

Field manipulation

As part of the data preparation, we created an entire function to perform pre-processing of the data set. The function includes the following processing techniques and algorithms.

Firstly, we remove the fields irrelevant to the project goals (these fields are determined by the user by specifying a list called irrelevant_fields in the main code) and then using this as an argument of the preprocess_dataset function. We opted to remove the fields “manufacturer_id”, “supplier_id”, “PERFORM_CNS.SCORE.DESCRIPTION”, "MobileNo_Avl_Flag" and "branch_id", as we felt these were irrelevant when predicting whether an individual would default on their vehicle loan.

We also remove tuples where the value is null or zero in the “Employment” field, ensuring that all applications we are using to train from have a value of salaried or self-employed.

We changed some fields in the dataset so that they used age in years or months, rather than having a field with a timestamp, with help using code by Zafer Cesur on Stack Overflow [5]. We did this to make these fields more compatible for data analysis. We changed the fields “Date.of.Birth” to age in years, “DisbursalDate”, “AverageAcctAge” and “Credit.History.Length” to months.

As part of the data exploration process, we noticed that there are a number of binary fields that act as flags as to whether an applicant has shared a certain ID with the loan company. We decided to collate this data into a collated field called “Gov_ID” so that if any of the six fields (Aadhar_Flag, PAN_Flag, VoterID_Flag, Driving_Flag, Passport_Flag) are 1, then the collated field “Gov_ID” can be set as 1. If all flags are 0, then “Gov_ID” is set to 0. By doing this however, it turned out that all applicants have shared at least one of these IDs with the loan company, so all applicants had a value of 1 for our collated column. This data was therefore negligible in our data analysis and we removed it.

We determined the types of the fields of the dataset; symbolic, ordinal/continuous numeric and discreet numeric. In order to determine discrete and continuous values, we use a histogram equalisation method. This categorises each numeric field into discrete bins so that the number of values of the field within the bin is determined. We used a cutoff value of 5 (meaning that five bins must be empty for the field to be determined as discrete, otherwise it is continuous). This was done in order to determine outliers in the fields determined as ordinal.

One-hot encoding

For the “employment” field, we employed the one-hot encoding technique to create two new binary fields classifying whether an applicant is self-employed or under salaried employment.

For this we used the caret package dummyVars to separate the column into two separate columns and assign 1 or 0 to the columns.

Removing outliers

It was important to remove outliers in the ordinal fields of the dataset and replace them with the mean value for the field. In order to detect the outliers in our program, we set a confidence value of 95% so that a p-value of less than 0.05 indicates that the record in the field is an outlier. More specifically, a chi-squared value is calculated based on the distance between each record and a function that the distribution is assumed to approximately take [6]. This is used to determine the p-value of the record. The record is replaced with the mean of the field if determined to be an outlier using this method.

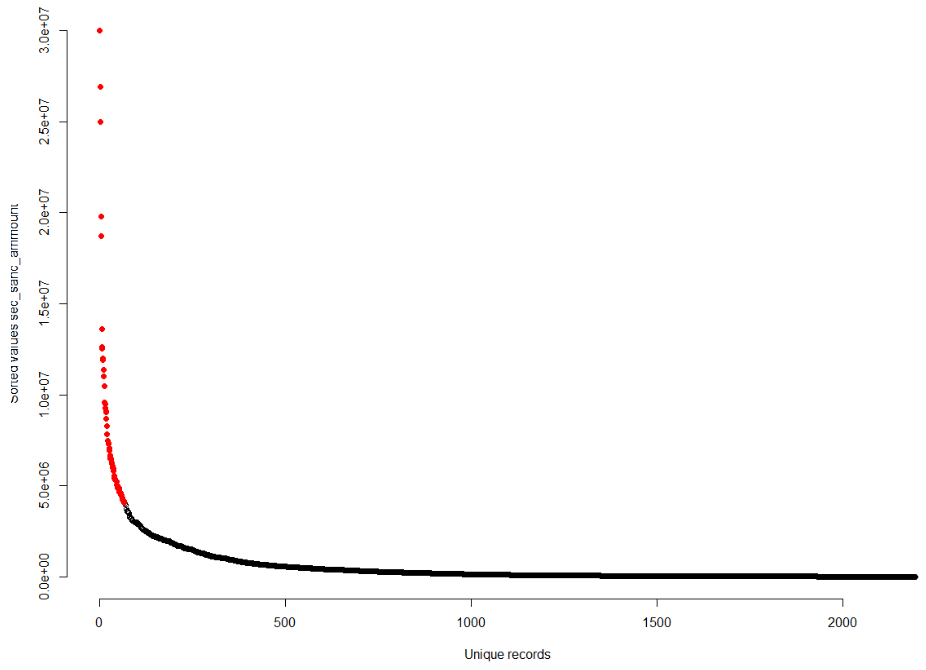
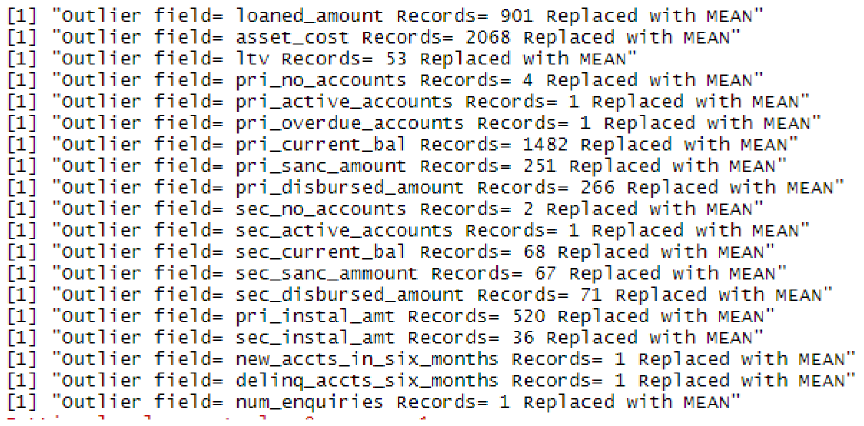


Figure 4. Sample plot showing outliers (red points) that are removed from the dataset (for sec_sanc_amount)

The reason for removing outliers is to ensure that our pre-processed dataset is suitable for the machine learning algorithms we use. Given the nature of the car loans dataset, there were likely to be significant outliers in a number of fields. For example, two entries in the dataset showed loans of almost Rs100,000 being granted. Such extreme values cause a dataset to be unbalanced, which can hinder the performance of many machine learning algorithms. The number of outliers replaced with mean values is shown in the code screenshot in Figure 5.



```
[1] "Outlier field= loaned_amount Records= 901 Replaced with MEAN"
[1] "Outlier field= asset_cost Records= 2068 Replaced with MEAN"
[1] "Outlier field= ltv Records= 53 Replaced with MEAN"
[1] "Outlier field= pri_no_accounts Records= 4 Replaced with MEAN"
[1] "Outlier field= pri_active_accounts Records= 1 Replaced with MEAN"
[1] "Outlier field= pri_overdue_accounts Records= 1 Replaced with MEAN"
[1] "Outlier field= pri_current_bal Records= 1482 Replaced with MEAN"
[1] "Outlier field= pri_sanc_amount Records= 251 Replaced with MEAN"
[1] "Outlier field= pri_disbursed_amount Records= 266 Replaced with MEAN"
[1] "Outlier field= sec_no_accounts Records= 2 Replaced with MEAN"
[1] "Outlier field= sec_active_accounts Records= 1 Replaced with MEAN"
[1] "Outlier field= sec_current_bal Records= 68 Replaced with MEAN"
[1] "Outlier field= sec_sanc_ammount Records= 67 Replaced with MEAN"
[1] "Outlier field= sec_disbursed_amount Records= 71 Replaced with MEAN"
[1] "Outlier field= pri_instal_amt Records= 520 Replaced with MEAN"
[1] "Outlier field= sec_instal_amt Records= 36 Replaced with MEAN"
[1] "Outlier field= new_accts_in_six_months Records= 1 Replaced with MEAN"
[1] "Outlier field= delinq_accts_six_months Records= 1 Replaced with MEAN"
[1] "Outlier field= num_enquiries Records= 1 Replaced with MEAN"
```

Figure 5. Outliers replaced with mean value for each field

Machine learning Modelling

Decision tree

Decision trees are used in the banking industry extensively due to their accuracy and transparency, in contrast to “black box” algorithms such as neural networks. It allows decision makers to understand the logic applied in reaching a conclusion. This is very attractive to financial institutions since lending practices are carefully monitored in many regions of the world. It helps executives stay in line with banking regulations and also helps applicants understand the reasons for their rejections, allowing them to build their credit rating more accurately. Moreover, decision trees are very useful for data exploration and do not require any pre-processing of the dataset. It is important to note however, that decision trees have some limitations. They are very sensitive to small perturbation in the data, therefore a minor change in the training set can affect the structure of the decision tree drastically. Furthermore, they are prone to overfitting although this can be improved to some extent by pruning and using ensemble methods such as random forest. Additionally, they can also be computationally expensive to train [7].

In technical terms, decision trees create training models that can predict class (y) of an object using it’s observed features (x) by learning decision rules (f(x)) inferred from prior examples (training data) [8]. It does this by first identifying the attribute that is the best predictor of the class and placing it at the top of the tree (known as the root node). The order in which attributes are used is determined by statistical techniques such as information gain. Information gain provides a measure of the expected reduction in uncertainty that results from splitting the dataset on a given attribute. In the next step it generates a condition for separating the data based on the selected attribute and the relative frequency with which each class occurs in the dataset, expressed formally as the prior probability distribution. It then forms branches that split the datasets into subsets known as internal nodes. The Gini Index is used to provide a score of how good a split is based on the resulting subsets. A perfect split results in a score of 0 whilst a poor split result in a 50:50 score. Finally, it repeats the first two steps until it reaches the data subset that cannot be divided further known as a leaf node [8].

For this project we will be developing a credit approval model using the Ctree decision trees [9]. We will also be exploring how the results of the model can be adjusted to reflect the objective of the financial institution, namely the reduction of financial loss and profit maximisation.

Our decision tree used a number of fields to determine the paths and nodes. The entire tree has many nodes and is quite complex, so is included in a separate image file called DecisionTree.png. The fields used for the tree are shown below:

```
tree                                                                                    <-
ctree(loan_default~CNS_score+sec_no_accounts+sec_overdue_accounts+sec_current_bal
+delinq_accts_six_months+
sec_active_accounts+new_accts_in_six_months+loaned_amount+pri_instal_amt+asset_co
st+pri_current_bal+
new_accts_in_six_months+ltv+emp.empSalaried+emp.empSelf.employed,train,controls=c
tree_control(mincriterion=0.9, minsplit=50))
```

Each data point is fed through the decision tree to determine a probability that the datapoint is a defaulter. A threshold was set so that a probability of defaulting above the threshold would result in the data point being categorised as a defaulter [10].

The advantage of using a decision tree, especially for our dataset, is that missing values don't affect the outcome as much as other machine learning techniques. It can give a comprehensive outcome for each value that enters the tree. However, it is hard to analyse whether changing data in the dataset (as we have done in the field manipulation) has a discernible impact on the results.

Logistic regression model

The initial problem to predict the loan defaulters is a classification problem and hence a logistic regression model was built and trained on the pre-processed data. The "loan_default" column is used as the predicted value, which uses 1 to represent that the loan recipient defaulted and 0 indicating that the loan was repaid. Logistic regression is an efficient and simpler compared to other complex machine learning models, hence it is not as prone to overfitting to the training data as other machine learning algorithms. On the other hand, the primary disadvantage of this chosen algorithm is that it requires independent observations for training, if these variables depend on each other the resultant model will exaggerate the importance of these variables [11].

Logistic regression is simply a modified version of linear regression. However, unlike linear regression logistic outputs a value between 0 and 1, a threshold is established to determine whether the output belongs to the 0 or 1 class, enabling binary classification. Moreover, a non-linear log function is applied for linear regression therefore the same assumptions are not made for logistic regression, such as a linear relationship between the dependent and independent variables [13].

Initially, the threshold is set 0.7 so if the model predicts an output with a probability of 0.7 or higher it will be classified as a default label, below 0.7 is defined as a non-default label. The threshold is iteratively tuned for the most ideal metric results, for the purpose of a profit model granting a loan to a defaulter is extremely detrimental, this is considered when determining the ideal threshold. The 'quasibinomial' parameter is passed in the making of this model to account for the large variance in the data. This means that a quasibinomial function is used to separate defaulters and non-defaulters for each value in the fields we are interested in.

Neural Networks

Following on from our logistic regression model, we decided to tackle the same problem using neural network. Neural network can dig deeper and find finer details regarding the relationship between our feature variables and the response variable, which may conclude in higher accuracy in terms of classification. The dataset was partitioned into a training, validation and test set of proportion 70/20/10. The main limitation for the neural network is that you can only produce a model as the data that pass into. Considering our dataset, we have a distribution of 79% vs 21% for our response variable which is binary variable of No Loan Default (0) Vs. Loan Default (1). With this kind of distribution our dataset is unbalanced and passing this through our network would cause it to learn to become bias towards only predicting No Loan Default (0) values. There are few techniques employed to combat this, main technique used is sampling to make up for the lack of data point for default loans. I have used under, over and ROSE sampling. ROSE is a library package for r which can conduct the sampling, by producing synthetically generated data for the minority class by enlarging the feature space.

In order to use the neural network technique on our dataset, the fields we were investigating had to be normalised to values between 0 and 1.

For the neural network architecture, I employed a model, with 4 layers, two of these which were hidden layers had 8 neurons each then followed by the output layer with a single neuron applied with a sigmoid activation function. Each of the hidden layers are using a relu activation function. The model was trained for 10 epochs. Removing all variables which had correlation of 0.75 or more, caused the model to achieve extremely low loss (Figure 13). According to the confusion matrix the model was achieving 100 percent accuracy in term of classification, hence this our dataset after processing is now linearly separable.

Random forest

As mentioned in the section above, Decision Trees have some limitations. As a result, we can now introduce Random Forest which is an ensemble method that gives a much higher accuracy, as it doesn’t rely on a single model.

In Random Forest each individual tree outputs a class prediction and the final model prediction is built by taking the class with the most votes. However, rather than simply taking the average prediction of all trees, the model also relies on the random sampling of training data points (with replacement) when building trees and the selection of random subsets of features when splitting nodes (results in lower error rates) [14]. Furthermore, low correlation between models is key, as the lower the correlation the less significant the error from each tree prediction, resulting in a lower forest error rate.

If conducted properly, random forest should not overfit and their predictive performance competes with the best supervised learning algorithms. However, the increased performance of this ensemble method comes at a cost as random forest are uninterpretable and often qualify as “black-boxes” due to the complexity of their operations. Additionally, the large number of trees in the model results in high computational costs, significantly more memory space and a much slower time for predictions [15].

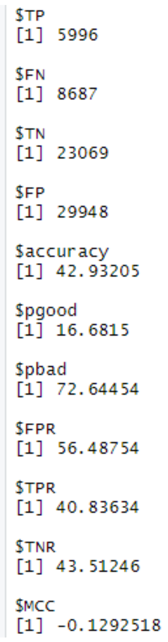
Random forest has only 2 free parameters and is user friendly. The forest grows decision trees based on which Random model to build the forest. Random forest has the first parameter as number of trees denoted by ntree (default value for ntree is 500), second parameters is variables that are randomly selected as candidates at each split. These variables are represented by mtry.

Random Forest starts by drawing ntree number of bootstrap samples and then for each bootstrap sample it grows as an un-pruned tree by choosing the best split based on random samples of mtry predictors at each node, it then predicts new data using majority votes for classification and average for regression based on ntrees.

We have used an arbitrary default value of 500 for ntree.

Results & Evaluation

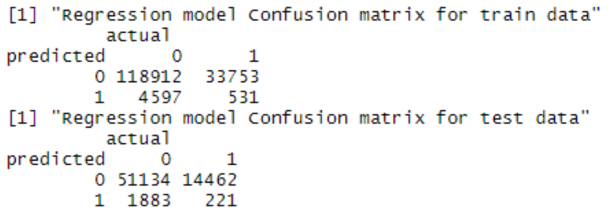
We will mainly incur a loss by disbursing a loan that cannot be repaid, however, we will also miss profit by denying a customer that can payback their loan. By summing the gain from the true positive and the true negative, and subtracting the loss from the false positive and false negative values (an example of these is shown for our linear regression model in Figure 6), we can derive a simple metric that will indicate profit or loss.



```
$TP
[1] 5996

$FN
[1] 8687

$TN
[1] 23069

$FP
[1] 29948

$accuracy
[1] 42.93205

$pgood
[1] 16.6815

$pbad
[1] 72.64454

$FPR
[1] 56.48754

$TPR
[1] 40.83634

$TNR
[1] 43.51246

$MCC
[1] -0.1292518
```

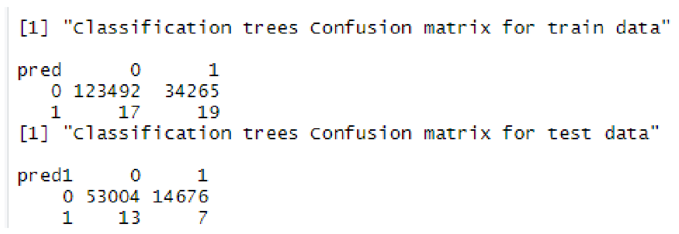
Figure 6. Results of the linear regression model, showing the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN)

We analyse the confusion matrices from the different models in Figures 7, 8, 9 and 10 below.



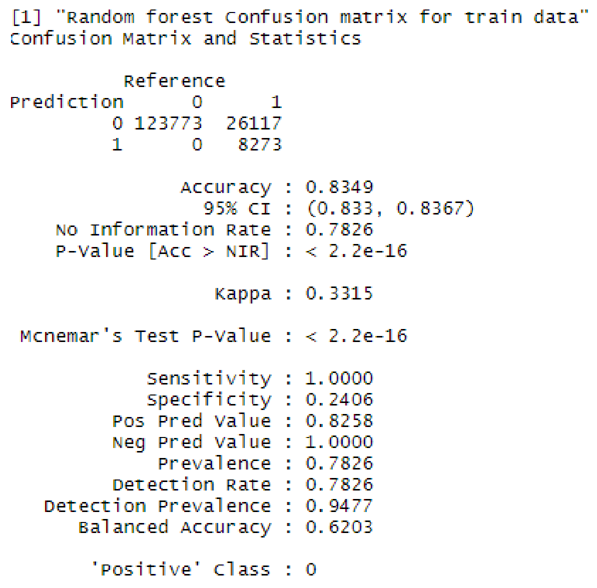
```
[1] "Regression model Confusion matrix for train data"
         actual
predicted      0      1
        0 118912  33753
        1   4597    531
[1] "Regression model Confusion matrix for test data"
         actual
predicted     0     1
        0 51134 14462
        1  1883   221
```

Figure 7: Confusion matrix for the linear regression model

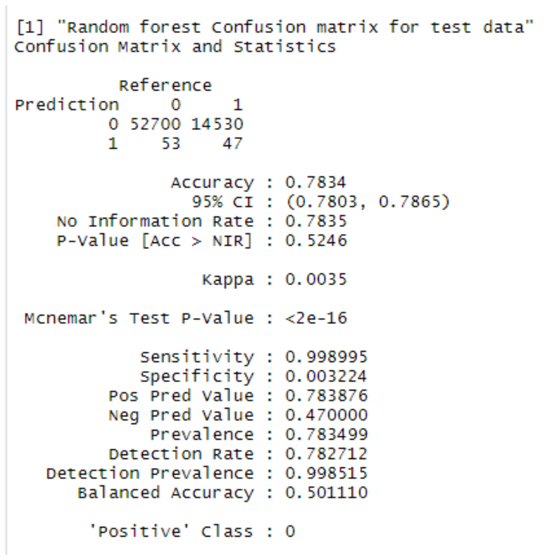


```
[1] "Classification trees Confusion matrix for train data"

pred      0      1
   0 123492  34265
   1     17     19
[1] "Classification trees Confusion matrix for test data"

pred1     0     1
    0 53004 14676
    1    13     7
```

Figure 8: Confusion matrix for the decision tree model

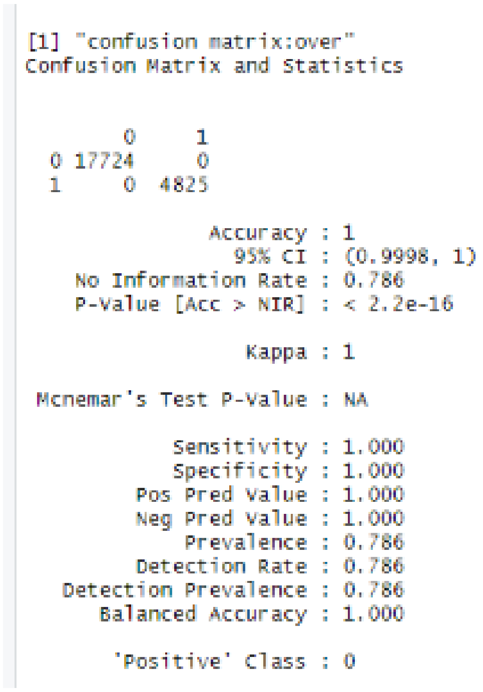


```
[1] "Random forest Confusion matrix for train data"
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0 123773  26117
         1      0   8273

               Accuracy : 0.8349
                 95% CI : (0.833, 0.8367)
    No Information Rate : 0.7826
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3315

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.2406
         Pos Pred Value : 0.8258
         Neg Pred Value : 1.0000
             Prevalence : 0.7826
         Detection Rate : 0.7826
   Detection Prevalence : 0.9477
      Balanced Accuracy : 0.6203

       'Positive' Class : 0
```



```
[1] "Random forest Confusion matrix for test data"
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 52700 14530
         1    53    47

               Accuracy : 0.7834
                 95% CI : (0.7803, 0.7865)
    No Information Rate : 0.7835
    P-Value [Acc > NIR] : 0.5246

                  Kappa : 0.0035

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.998995
            Specificity : 0.003224
         Pos Pred Value : 0.783876
         Neg Pred Value : 0.470000
             Prevalence : 0.783499
         Detection Rate : 0.782712
   Detection Prevalence : 0.998515
      Balanced Accuracy : 0.501110

       'Positive' Class : 0
```

Figure 9: Confusion matrix for the random forest model (applied on train and test datasets)



```
[1] "confusion matrix:over"
Confusion Matrix and Statistics


       0     1
 0 17724     0
 1     0  4825

               Accuracy : 1
                 95% CI : (0.9998, 1)
    No Information Rate : 0.786
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.786
         Detection Rate : 0.786
   Detection Prevalence : 0.786
      Balanced Accuracy : 1.000

       'Positive' Class : 0
```

Figure 10: Confusion matrix for the neural network model

The confusion matrix gives the estimate of true negatives, false negatives, false positives and true positives in the respective order. On comparing the confusion matrices, we found that the best model in terms of accuracy is random forest.

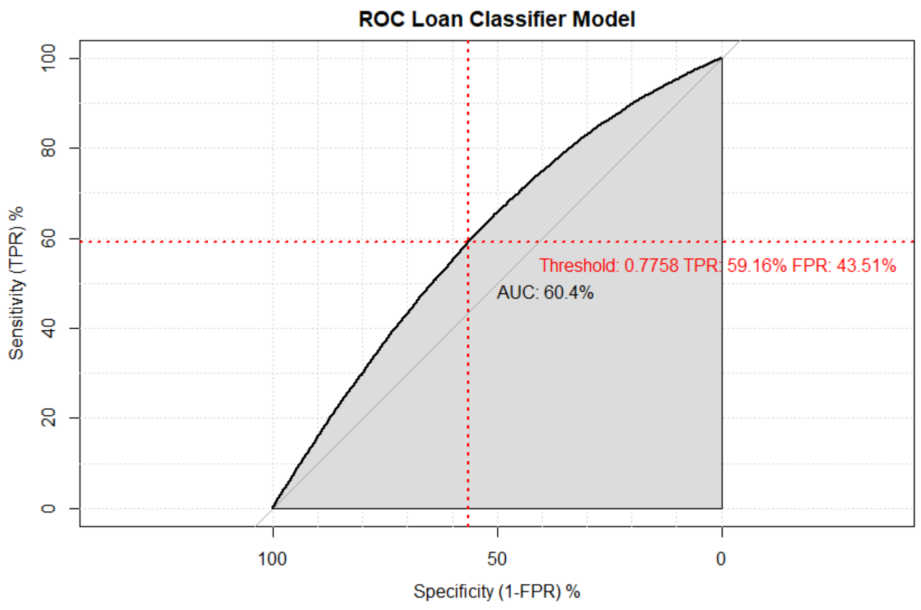


Figure 11: AUC – ROC Curve for the logistic regression model, showing the optimum threshold calculated as being 0.7758

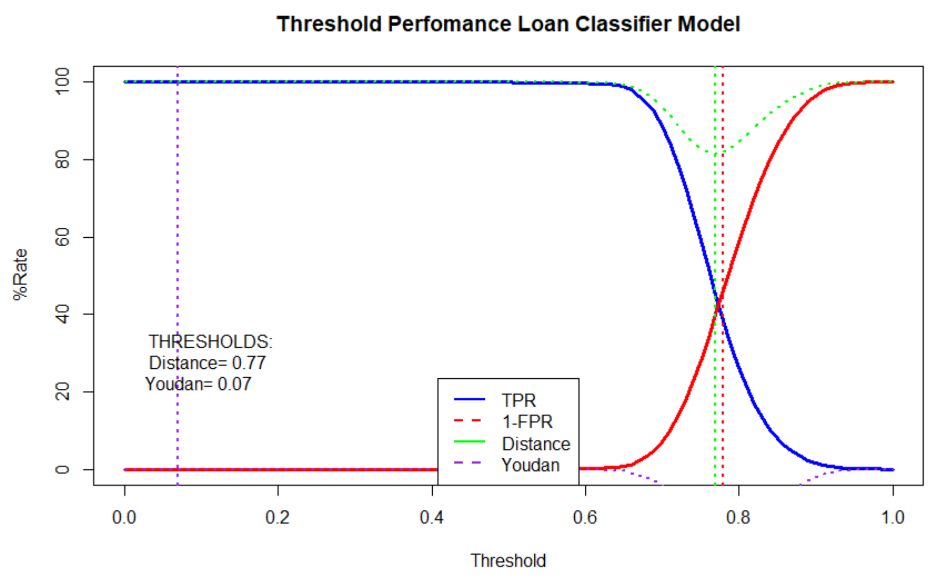


Figure 12: Threshold performance

To evaluate the performance of our logistic regression, we decided to plot an ROC curve. As shown in figure 11, the model concluded with an AUC (Area Under Curve) as 60.4%. This model was therefore used as the baseline performance to compare all following models against. The graph displayed in graph 12 shows the values of each variable used within the creation of the ROC curve, showing the drop in distance and Youdan distance as the tested threshold got closer to the optimum value of 0.7758.

The most useful metrics in terms of deriving a profitare the True Positive and True Negative rate, also referred to as Sensitivity and Specificity. Using Figure 8 for the decision tree the specificity and sensitivity are 0.35, and 0.783. Therefore, just using the specificity and sensitivity the best performing model for profit is the decision tree.

Further analysis

Following on from our profit prediction, we had the idea of potentially implementing a scale that determines how likely it is that an individual would default on their loan, a decimal number between 0 & 1 to show likelihood. Using this, we could calculate a different interest rate for each applicant based on their likelihood of defaulting. Thus, someone more likely to default would have a higher interest rate on their loan, so that if they did in fact pay their loan back, the loan company would receive a higher profit after granting a riskier loan.

This could in fact be done using the probabilities given in the decision tree (see separate image file named DecisionTree.png and the section of this document on decision tree). Using the probability that a loan applicant will default; we can raise the interest rate accordingly.

Another idea that could be suggested to the organisation involved would be to collect more data from customers using either incentives or by passively increasing the number of fields they expect customers to give data about. This method could give the company more relevant fields, especially if they are more detailed about an individual's annual income for example. This would lead to the creation of models which would be more accurate and better at predicting the probability that a customer would default on their loan, leading to the potential to dramatically increase profits.

Acknowledgements

We would like to express our appreciation for the amount and quality of code provided for our use by Professor Nick Ryman-Tubb. Through use of his functions, the overall ability of our project was drastically increased over the short period of time. From the pre-processing stage of our dataset to creating the neural network, his contribution has been indispensable.

Appendix – data dictionary

Table 1 gives the data dictionary for our chosen dataset. The data dictionary is taken directly from the Kaggle dataset page: https://www.kaggle.com/gauravdesurkar/lt-vehicle-loan-default-prediction.

Field name	**Description**
UniqueID	Identifier for customers
loan_default	Payment default in the first EMI on due date
disbursed_amount	Amount of Loan disbursed
asset_cost	Cost of the Asset
ltv	Loan to Value of the asset
branch_id	Branch where the loan was disbursed
supplier_id	Vehicle Dealer where the loan was disbursed
manufacturer_id	Vehicle manufacturer (Hero, Honda, TVS etc.)
Current_pincode	Current pincode of the customer
Date.of.Birth	Date of birth of the customer
Employment.Type	Employment Type of the customer (Salaried/Self Employed)
DisbursalDate	Date of disbursement
State_ID	State of disbursement
Employee_code_ID	Employee of the organization who logged the disbursement
MobileNo_Avl_Flag	if Mobile no. was shared by the customer then flagged as 1
Aadhar_flag	if aadhar was shared by the customer then flagged as 1 (identity number for Indian residents)
PAN_flag	if pan was shared by the customer then flagged as 1
VoterID_flag	if voter was shared by the customer then flagged as 1
Driving_flag	if DL was shared by the customer then flagged as 1
Passport_flag	if passport was shared by the customer then flagged as 1
PERFORM_CNS.SCORE	Bureau Score
PERFORM_CNS.SCORE.DESCRIPTION	Bureau score description
PRI.NO.OF.ACCTS	count of total loans taken by the customer at the time of disbursement
PRI.ACTIVE.ACCTS	count of active loans taken by the customer at the time of disbursement
PRI.OVERDUE.ACCTS	count of default accounts at the time of disbursement

PRI.CURRENT.BALANCE	total Principal outstanding amount of the active loans at the time of disbursement
PRI.SANCTIONED.AMOUNT	total amount that was sanctioned for all the loans at the time of disbursement
PRI.DISBURSED.AMOUNT	total amount that was disbursed for all the loans at the time of disbursement
SEC.NO.OF.ACCTS	count of total loans taken by the customer at the time of disbursement
SEC.ACTIVE.ACCTS	count of active loans taken by the customer at the time of disbursement
SEC.OVERDUE.ACCTS	count of default accounts at the time of disbursement
SEC.CURRENT.BALANCE	total Principal outstanding amount of the active loans at the time of disbursement
SEC.SANCTIONED.AMOUNT	total amount that was sanctioned for all the loans at the time of disbursement
SEC.DISBURSED.AMOUNT	total amount that was disbursed for all the loans at the time of disbursement
PRIMARY.INSTAL.AMT	EMI Amount of the primary loan
SEC.INSTAL.AMT	EMI Amount of the secondary loan
NEW.ACCTS.IN.LAST.SIX.MONTHS	New loans taken by the customer in last 6 months before the disbursment
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	Loans defaulted in the last 6 months
AVERAGE.ACCT.AGE	Average loan tenure
CREDIT.HISTORY.LENGTH	Time since first loan
NO.OF_INQUIRIES	Enquiries done by the customer for loans

Table 1: Data dictionary for vehicle loans original dataset

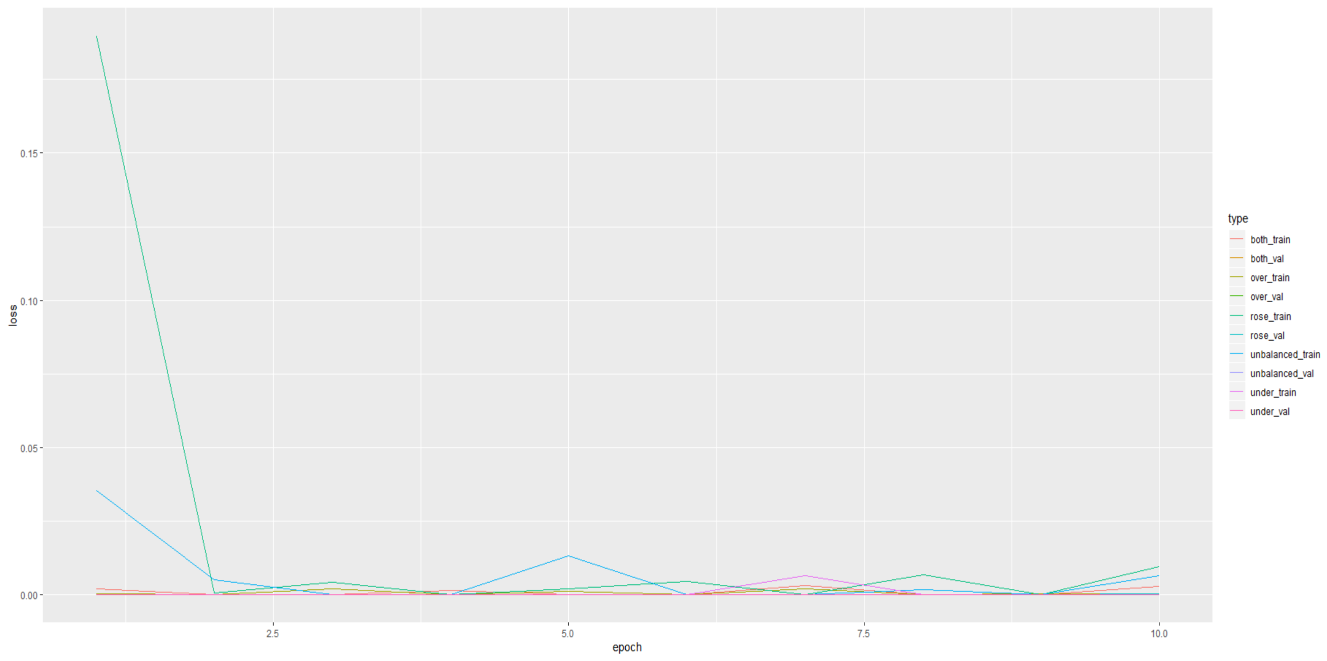


Figure 13: Loss and Val_Loss

References

[1] Basel committee on banking supervision, *Credit Ratings and Complementary Sources of Credit Quality Information*, Bank for International Settlements, Basel 2000, Working Paper 3. Available online from: https://www.bis.org/publ/bcbs72a.pdf

[2] Alaraj, M., Abbod, M., Hunaiti, Z. 2014, *Evaluating Consumer Loans Using Neural Networks Ensembles,* International Conference on Machine Learning, Electrical and Mechanical Engineering, viewed 30 Nov 2019, https://www.researchgate.net/publication/303022114_Evaluating_Consumer_Loans_Using_Neural_Networks_Ensembles

[3] Chen, Y., Guo, R., Huang, R., *Two stages credit evaluation in bank loan appraisal*, Econ. Model., vol. **26**, pp. 63-70, 2009. http://dx.doi.org/10.1016/j.econmod.2008.05.008

[4] Härle, P., Havas, A., Kremer, A., Rona, D., Samandari, H. (n.d.). *The future of bank risk management,* McKinsey Working Papers on Risk. Available online from: https://www.mckinsey.com/~/media/mckinsey/dotcom/client_service/risk/pdfs/the_future_of_bank_risk_management.ashx

[5] Zafer Cesur 2016, *Converting dates before January 1, 1970 in R,* viewed 22 November 2019, https://stackoverflow.com/questions/38508886/converting-dates-before-january-1-1970-in-r

[6] Anderson, A., Semmelroth, D. *Hypothesis Tests for Data Outliers,* viewed 29 November 2019, https://www.dummies.com/programming/big-data/data-science/hypothesis-tests-for-data-outliers/

[7] Nduwayo, N., 2018. *An introduction to machine learning with decision trees*, viewed 2 December 2019, https://www.station10.co.uk/blogs/2018/machine-learning-decision-trees

[8] Ray, S., 2015. *Decision Tree | Predictive Analytics*, viewed 2 December 2019, https://www.analyticsvidhya.com/blog/2015/01/decision-tree-simplified/2/

[9] Hothorn, T., Hornik, K., Zeileis, A. *Conditional Inference Trees, viewed 2 December 2019,* https://cran.r-project.org/web/packages/partykit/vignettes/ctree.pdf

[10] Dr. Bharatendra Rai, B., (youtube video(s) used for logistic regression and random forest)

[11] Robinson, N., 2018. *The Disadvantages of Logistic Regression,* viewed 2 December 2019, https://www.theclassroom.com/disadvantages-logistic-regression-8574447.html

[12] Bankbazaar.com. *Factors that Affect Your CIBIL Score & Tips to Improve it,* viewed 2 December 2019, https://www.bankbazaar.com/cibil/major-factors-that-affect-your-cibil-score.html

[13] Statisticssolutions.com, 2013. *Assumptions of Logistic Regression,* viewed 2 Dec 2019, https://www.statisticssolutions.com/wp-content/uploads/wp-post-to-pdf-enhanced-cache/1/assumptions-of-logistic-regression.pdf

[14] Koehrsen, W., 2018. *An Implementation and Explanation of the Random Forest in Python,* viewed 2 December 2019, https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

[15] Jansen, S., 2018. *Pros and cons of random forests,* viewed 3 December 2019, https://www.oreilly.com/library/view/hands-on-machine-learning/9781789346411/e17de38e-421e-4577-afc3-efdd4e02a468.xhtml