NOVEMBER 17, 2019

MERLION FUNDING CO.

CREDIT RISK MODELLING

ACCT 337: Statistical Programming

Prepared for: Prof. Benjamin Lee

Prepared by: Group 4

1. Ho Ming Quan
2. Irving Yeo Jun Hui
3. Joelyn Chua Pei Xuan
4. Lee Jiieng Yie
5. Ryu SeungJe

Part 1: Management Report

Executive Summary

The objective of this project is to assist Merlion Funding Co. (“Merlion”) to develop a credit risk model for the company to improve its loan on-boarding procedures and policies. The project is stages out into the following phases:

- Phase 1: Identify outlier
- Phase 2: Predict loan amount and defaulters
- Phase 3: Suggest a suitable credit risk model

In phase 1, the team employed clustering, an unsupervised learning technique used to learn about the patterns of the data and group them into various clusters. As such, this technique will assist in identifying possible outliers from the dataset provided by Merlion and allows the company to be more wary about certain characteristics of its loan applicants in the future. Based on the clustering result, the team recommends Merlion to scrutinise the checks on clients with high accounts past due as it has been proven that people with high number of debt obligations have higher likelihood of default.

Next, in phase 2, the team employed linear regression, a supervised learning technique to predict the optimal loan amount of future applicants, based on their characteristics. The resulting model predicts correctly more than 90% of the optimal loan amount most of the time. Therefore, being able to predict an appropriate loan amount for Merlion’s clients ensures that the customer has the ability to pay back and allows the company to prevent the situation of late repayment. For instance, when too high amount of loan is given to a client, the monthly repayment will also be higher, which may possibly lead to late-repayment due to tight cash flow on the client’s end.

Following that, the team employed classification tree technique to assess the likelihood of customers with “Current” loan status defaulting on the loan. Instead of just relying on the guarantor-based system to assess whether a loan will default, the classification model will provide a more accurate measure of the probability of default of each loan applicant based on the certain data collected from them. Therefore, this allows Merlion to make more well-informed decisions when deciding whether to give out a loan.

Finally, in Phase 3, the team recommends Merlion to consider the 5’s Cs of Credit namely Character, Capacity, Capital, Collateral and Condition in order to better analyse the creditworthiness of potential borrowers. Moving forward, some of the data that Merlion can collect to better encompass the current dataset includes the client’s debts and liabilities such as mortgage and automobile loan as well as credit card spending. Merlion can also inquire the borrower’s class of occupation (e.g. professional, managerial, clerical, sales, blue collar, etc) to better access the applicant’s annual income. Moreover, with more data collected, the performance and accuracy of the predictive models increases overtime, which can further fuel Merlion’s loan on-boarding policies and procedures.

Table of Contents

1. Exploratory data analysis (EDA)

The team came up with findings after analysing several visualisations on Merlion's dataset:

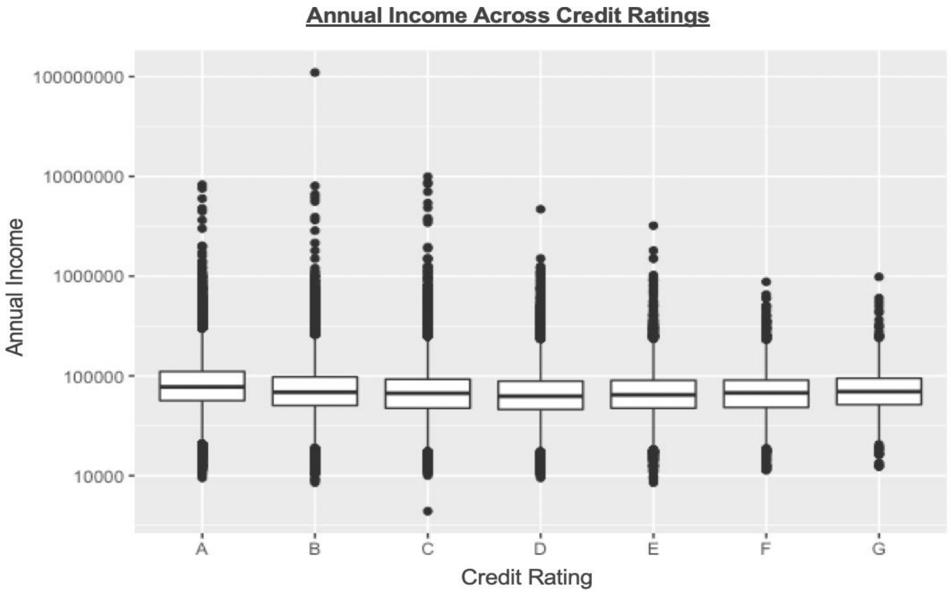


Figure 1: Annual Income across Credit Ratings

From Figure 1, we can see that the average income for all customers are about the same by looking at the boxes. Furthermore, there are several customers with abnormally high income.

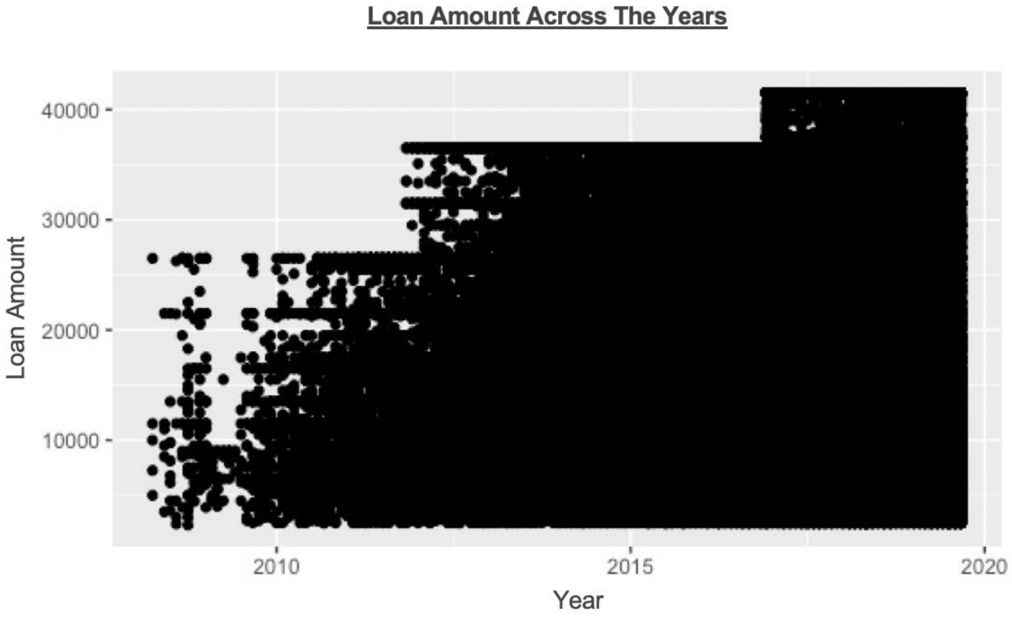


Figure 2: Loan Amount across Years

From Figure 2, we can see an increasing trend in the number of customers and loan amount. Before 2010, the number of loans is exponentially lesser. Thus, management should review if their business plan aligns with the increasing loan and loan amounts.

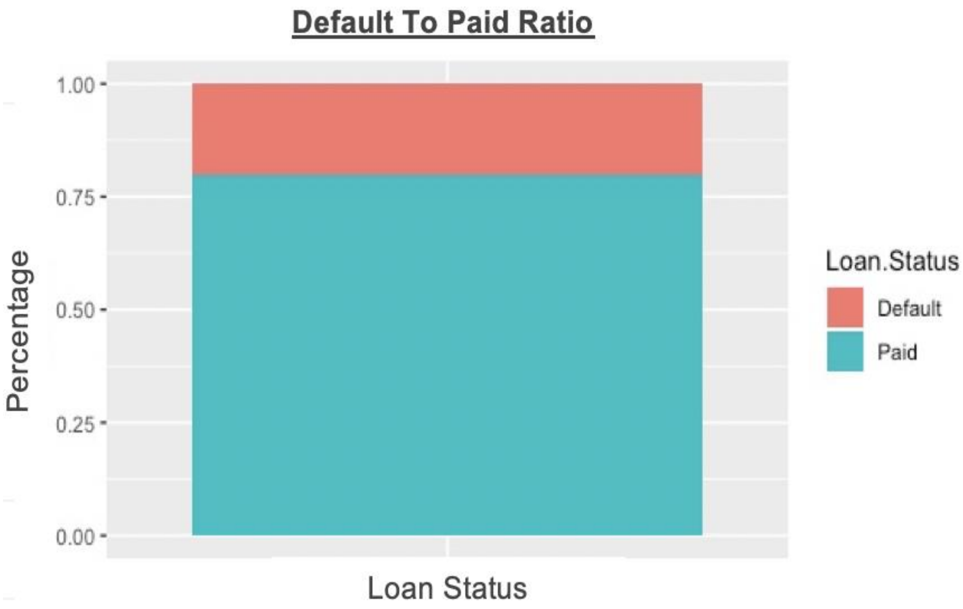


Figure 3: Default to Paid Ratio

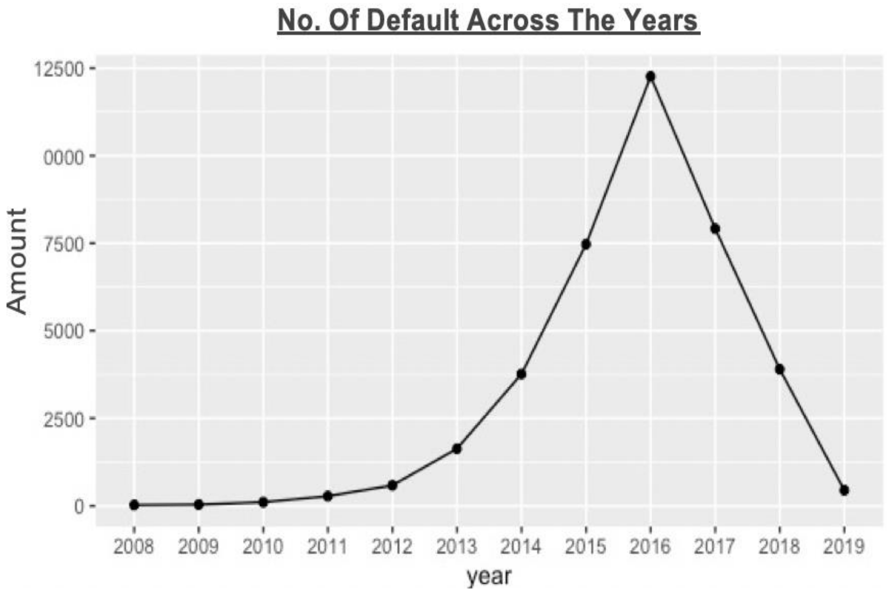


Figure 4: Number of Defaults across Years

From Figure 3, it can be seen that the ratio of default to paid is around 1:5. The management can compare this figure against their competitors.

Besides that, from Figure 4, the number of defaults has increased sharply from 2012 to 2016. The management can investigate whether this trend is attributed to the ineffective credit policy during the early days of Merlion.

We have also extracted the records that has discrepancies of beyond $251.51 between total payment received and sum of total interest and principal received for the management to review.

2. Clustering

To identify potential outliers, we employed clustering to group customers according to similar characteristics.

One of the findings of clustering was in tandem with the EDA. We noticed that the top outlier for various permutations of clustering always consisted of the particular individual that had an extremely high-income amount of $110,002,500. Additionally, the outliers consisted of similar people with very high-income amount ranging from 2 to 10 million. Credit ratings for these individuals were decent ranging from A to C.

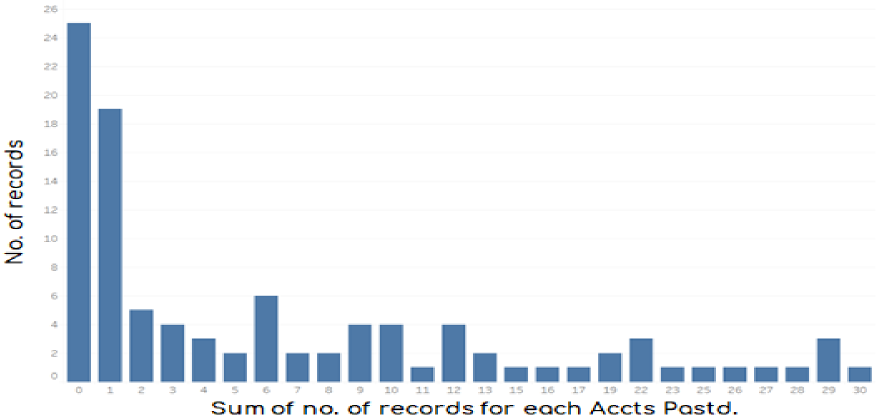


Figure 5: Accounts Past Due Records

In contrast, from Figure 5, when we isolated the data free of the income factor, the outlying individuals had high number of accounts past due. We recommend Merlion to scrutinise the checks on clients with accounts past due as research has shown that people with high number of debt obligations have higher likelihood of default (Kagan, 2019).

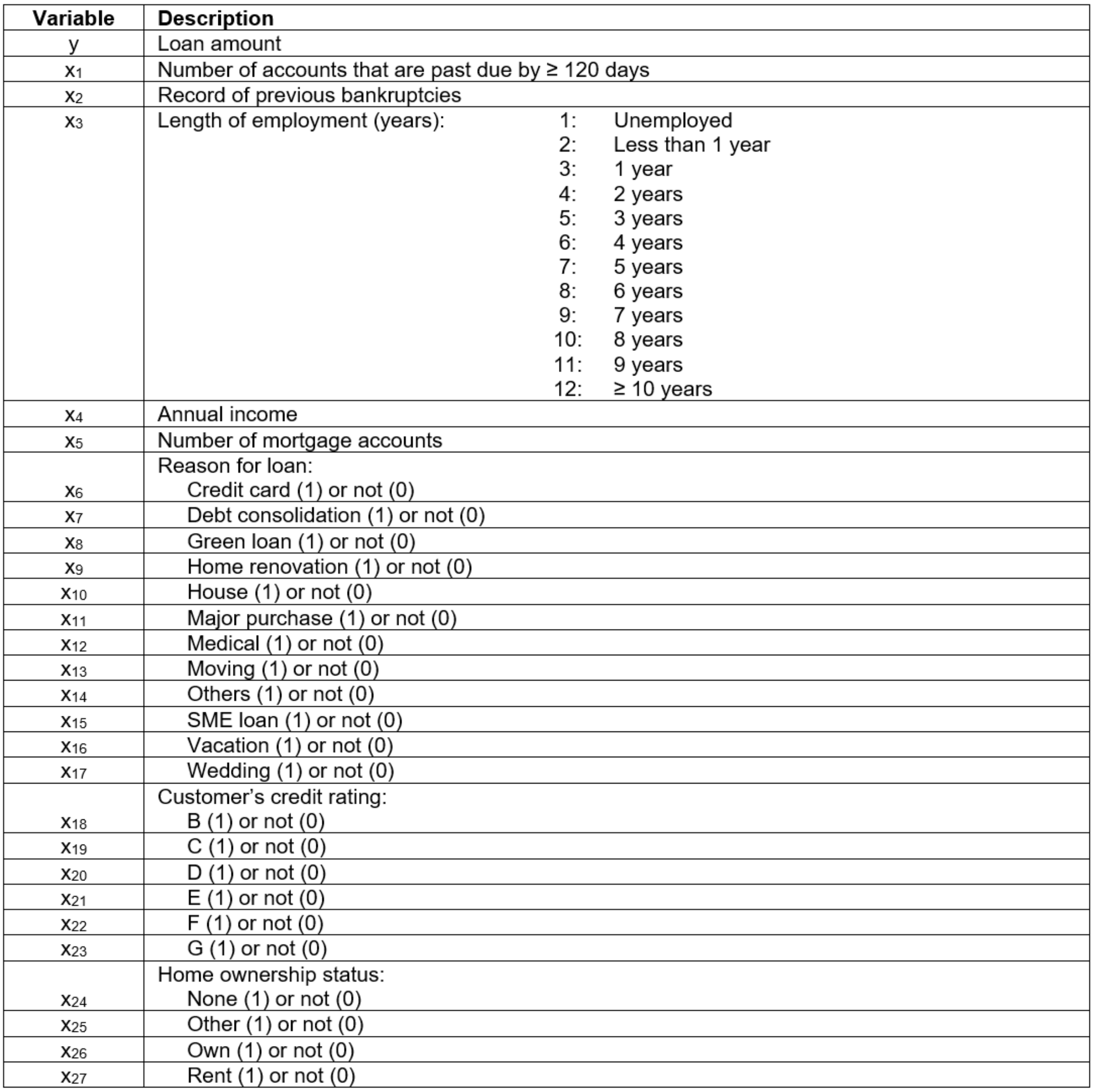
Furthermore, the team feels that the current data available are not able to substantially explain as to why high-income individuals were rejected by banks with their loan applications. Hence, this warrants for more data collection by Merlion in order to explain and investigate these individuals and potential clients thoroughly. The process should emulate how commercial banks’ compliance department carry out Know-Your-Client/ Anti-Money Laundering procedures to manage credit and counterparty risks.

3. Regression

To predict the loan amount of new customers, we employed linear regression, which is frequently used in forecasting and predictions.

The variables selected for regression analysis are indicative of the income, liabilities, legitimacy and background of the loan applicants, reflecting their loans eligibility. This has been verified through our model whereby these variables are statistically significant (Appendix F). They are also closely linked to the 5C's of Credit as elaborated in section 5 below.

Using our model, we have obtained the following regression equation:



Variable	Description
y	Loan amount
x_1	Number of accounts that are past due by ≥ 120 days
x_2	Record of previous bankruptcies
x_3	Length of employment (years): 1: Unemployed 2: Less than 1 year 3: 1 year 4: 2 years 5: 3 years 6: 4 years 7: 5 years 8: 6 years 9: 7 years 10: 8 years 11: 9 years 12: ≥ 10 years
x_4	Annual income
x_5	Number of mortgage accounts
x_6	Reason for loan: Credit card (1) or not (0)
x_7	Debt consolidation (1) or not (0)
x_8	Green loan (1) or not (0)
x_9	Home renovation (1) or not (0)
x_{10}	House (1) or not (0)
x_{11}	Major purchase (1) or not (0)
x_{12}	Medical (1) or not (0)
x_{13}	Moving (1) or not (0)
x_{14}	Others (1) or not (0)
x_{15}	SME loan (1) or not (0)
x_{16}	Vacation (1) or not (0)
x_{17}	Wedding (1) or not (0)
x_{18}	Customer's credit rating: B (1) or not (0)
x_{19}	C (1) or not (0)
x_{20}	D (1) or not (0)
x_{21}	E (1) or not (0)
x_{22}	F (1) or not (0)
x_{23}	G (1) or not (0)
x_{24}	Home ownership status: None (1) or not (0)
x_{25}	Other (1) or not (0)
x_{26}	Own (1) or not (0)
x_{27}	Rent (1) or not (0)

$\log_{10} L$oan Amount = 3.375 – 0.023 $\log_2 x_1$ – 0.071 $\log_2 x_2$ + 0.017 $\log_2 x_3$ + 0.111 $\log_2 x_4$ + 0.036 $\log_2 x_5$ + 0.190 x_6 + 0.190 x_7 + 0.008 x_8 + 0.109 x_9 + 0.161 x_{10} +0.076 x_{11} – 0.041 x_{12} – 0.059 x_{13} – 0.002 x_{14} + 0.158 x_{15} – 0.163 x_{16} + 0.048 x_{17} – 0.002 x_{18} + 0.030 x_{19} + 0.063 x_{20} + 0.111 x_{21} + 0.160 x_{22} + 0.194 x_{23} – 0.069 x_{24} – 0.019 x_{25} – 0.021 x_{26} - 0.042 x_{27}

To obtain loan amount, y = $10^{log_{10}\,\text{Loan Amount}}$.

The team has selected the model (Appendix G) with the highest accuracy of 95.42% when predicting future loan amounts. Therefore, loan amount can be predicted using the regression model formula shown above and this can be done before approving the loan to Merlion's customers. This is a good way forward as it allows Merlion to "optimise" the loan amount that is suitable for every customer.

However, from our initial data exploration, we realised that among the "paid" status loans, there often exists a $1,010 discrepancy between loan amount and principal repayment. Due to the nature of the dataset, we would thus recommend Merlion to give out loans ranging between y – $1,010 and y (with y being the maximum), avoiding the issue of uncollectible $1,010.

4. Classification

Classification is a tool that helps to predict the 1 of two outcomes, where the model will identify based on the rules, which the data tends to follow.

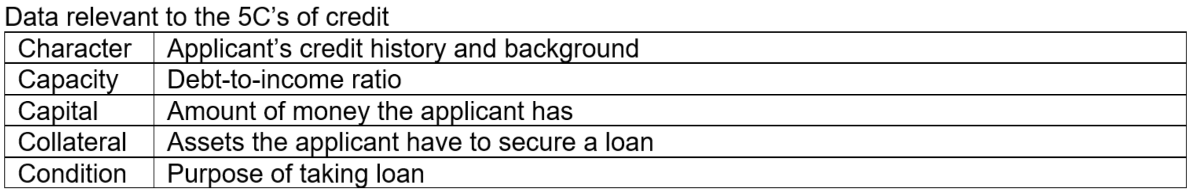
The model that we have created can predict defaults at a 69.45% accuracy. It provides us with the respective rules and visualisation as seen in Appendix M and N respectively. Using these rules on your Current Loans, it can predict their outcomes as seen in "**loan.current.predict.csv**". Moving forward, this model can be used as a credit risk tool at the point of approval of loans to help the Merlion identify if the loan is likely to default or not.

Furthermore, management should note that by applying our model to this .csv file, we have identified that approximately 43.51% of current loans will default. We would recommend management to act on these predicted to default loans such as to incentivize them to repay quickly before they default.

Should management seek to improve the accuracy of the model, they may attempt to execute the model in Appendix A below, where due to the time constraints of the project we were not able to build in time.

5. Recommendation and Conclusion

To better analyse the creditworthiness of potential borrowers, we recommend Merlion to consider the 5C's of Credit as seen in Figure 6, to better encompass Merlion's current dataset (Kagan & Segal, 2019).



Data relevant to the 5C's of credit

Character	Applicant's credit history and background
Capacity	Debt-to-income ratio
Capital	Amount of money the applicant has
Collateral	Assets the applicant have to secure a loan
Condition	Purpose of taking loan

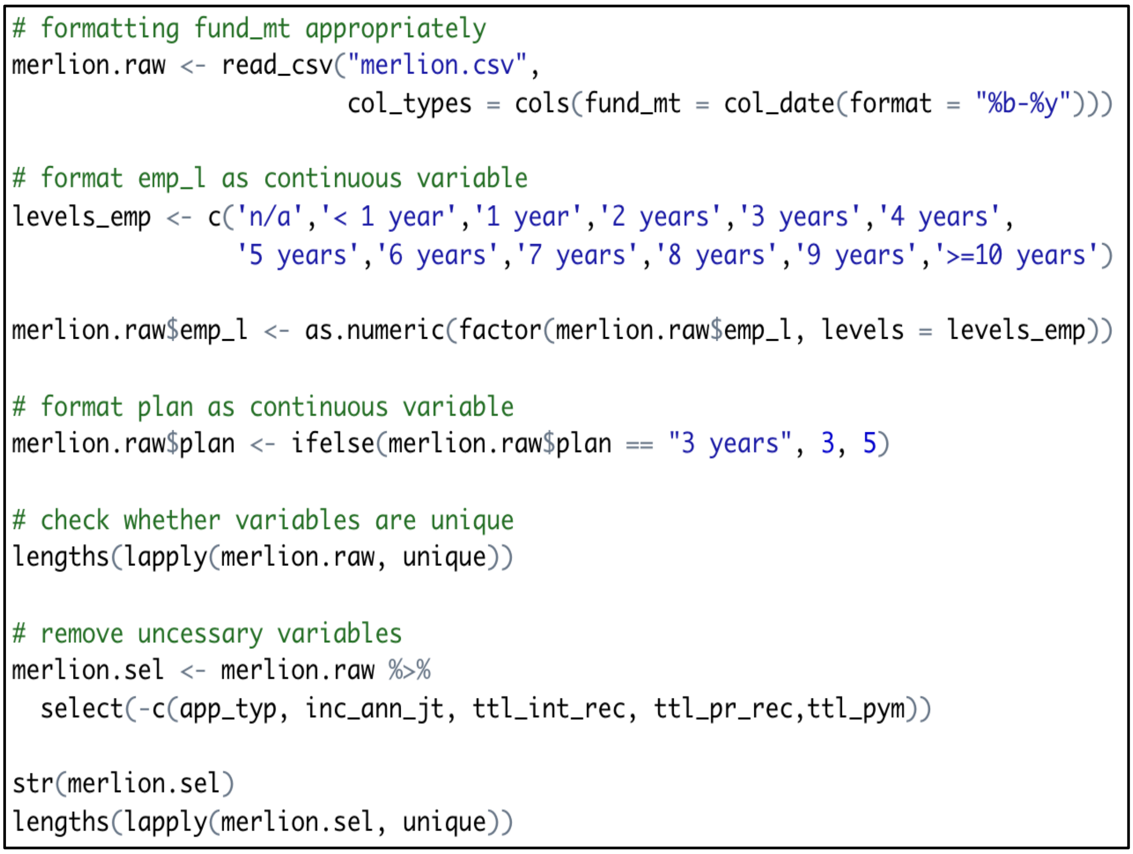
Figure 6: 5C's of Credit

As such, we recommend Merlion to collect the following quantitative and qualitative data. They include clients' liabilities and debts such as mortgage and automobile loan amount as well as credit card spending. With these data, Merlion will be able to calculate the debt-to-income ratio, which is a good indicator of the probability of default. Furthermore, qualitative data would be their class of occupation. Although Merlion already possessed their annual income, those income may come from sources other than employment income like dividend income or capital gains. To ensure that the clients have a constant flow of employment income, knowing the class of occupation would be beneficial. Furthermore, with more data, accuracy of the model improves overtime.

In conclusion, using the three models that we have developed, coupled with the improvised data collection, Merlion's loan on-boarding policies and procedures will be fortified, thus allowing them to make more well-informed decisions when approving loans.

Part 2: Technical Report

1. General data pre-processing & Dimension reduction



```
# formatting fund_mt appropriately
merlion.raw <- read_csv("merlion.csv",
                        col_types = cols(fund_mt = col_date(format = "%b-%y")))

# format emp_l as continuous variable
levels_emp <- c('n/a','< 1 year','1 year','2 years','3 years','4 years',
                '5 years','6 years','7 years','8 years','9 years','>=10 years')

merlion.raw$emp_l <- as.numeric(factor(merlion.raw$emp_l, levels = levels_emp))

# format plan as continuous variable
merlion.raw$plan <- ifelse(merlion.raw$plan == "3 years", 3, 5)

# check whether variables are unique
lengths(lapply(merlion.raw, unique))

# remove uncessary variables
merlion.sel <- merlion.raw %>%
  select(-c(app_typ, inc_ann_jt, ttl_int_rec, ttl_pr_rec,ttl_pym))

str(merlion.sel)
lengths(lapply(merlion.sel, unique))
```

Figure 1: General pre-processing steps

From Figure 1, we loaded the dataset and formatted month as “yyyy-mm-01”. Next, we transformed emp_l into numerical variable by assigning levels to the column input and changed plan column into numeric for performing mathematical calculation when necessary. Finally, we ran a code to determine whether all the columns are unique and dropped unnecessary variables before performing a final check that R has formatted all variables accordingly.

While working on dataset, it is inevitable that there are missing values. For Merlion’s data, there are no missing value. In event of missing values, we will perform the steps in Appendix B.

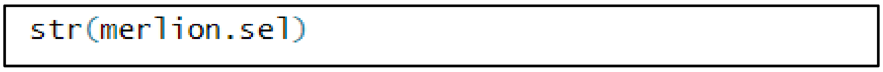
2. Clustering

2.1 Objective

Clustering is one of the unsupervised learning techniques used to identify records with homogenous characteristics as well as outliers in the dataset. We used k-means clustering as the algorithm.

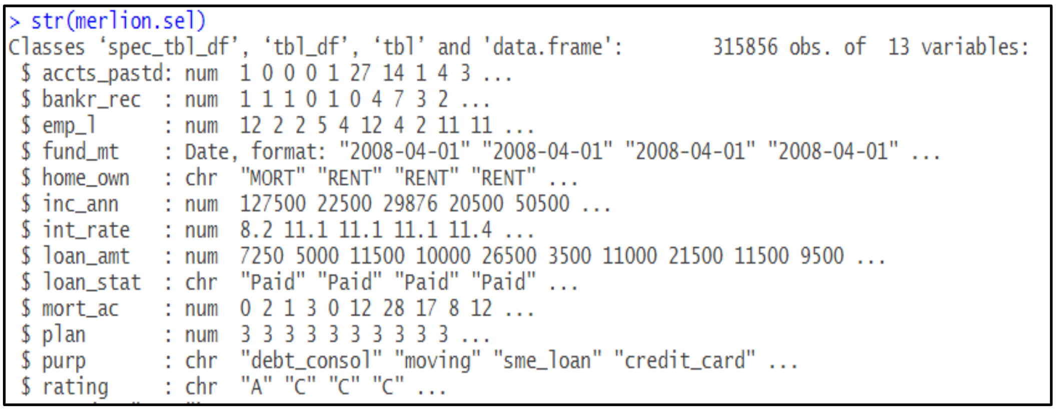
2.2 Pre-processing variables

After general pre-processing, 'str' function was run to review the variables.



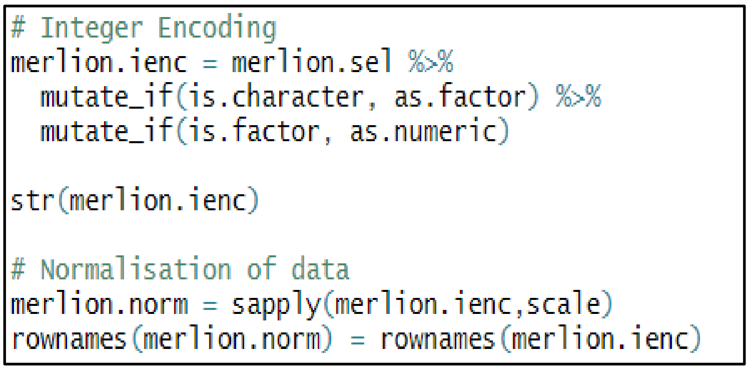
```
str(merlion.sel)
```

Figure 2: Pre-processing



```
> str(merlion.sel)
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':	315856 obs. of  13 variables:
 $ accts_pastd: num  1 0 0 0 1 27 14 1 4 3 ...
 $ bankr_rec  : num  1 1 1 0 1 0 4 7 3 2 ...
 $ emp_l      : num  12 2 2 5 4 12 4 2 11 11 ...
 $ fund_mt    : Date, format: "2008-04-01" "2008-04-01" "2008-04-01" "2008-04-01" ...
 $ home_own   : chr  "MORT" "RENT" "RENT" "RENT" ...
 $ inc_ann    : num  127500 22500 29876 20500 50500 ...
 $ int_rate   : num  8.2 11.1 11.1 11.1 11.4 ...
 $ loan_amt   : num  7250 5000 11500 10000 26500 3500 11000 21500 11500 9500 ...
 $ loan_stat  : chr  "Paid" "Paid" "Paid" "Paid" ...
 $ mort_ac    : num  0 2 1 3 0 12 28 17 8 12 ...
 $ plan       : num  3 3 3 3 3 3 3 3 3 3 ...
 $ purp       : chr  "debt_consol" "moving" "sme_loan" "credit_card" ...
 $ rating     : chr  "A" "C" "C" "C" ...
```

Figure 3: Pre-processing



```
# Integer Encoding
merlion.ienc = merlion.sel %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.factor, as.numeric)

str(merlion.ienc)

# Normalisation of data
merlion.norm = sapply(merlion.ienc,scale)
rownames(merlion.norm) = rownames(merlion.ienc)
```

Figure 4: Pre-processing

Clustering algorithm requires all input and output variables to be in numeric. So, through integer encoding (Figure 4), categorical data are converted into numerical data. Subsequently, data normalisation is carried out to standardise all the attributes of the dataset and give them equal weights to eliminate redundant or noisy objects.

2.3 Cluster Analysis

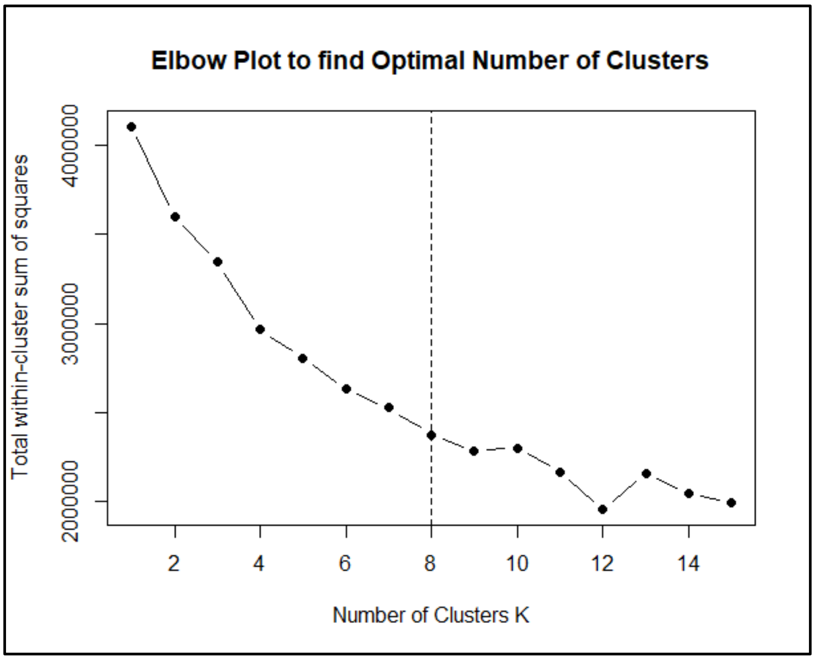


Figure 5: Elbow Plot

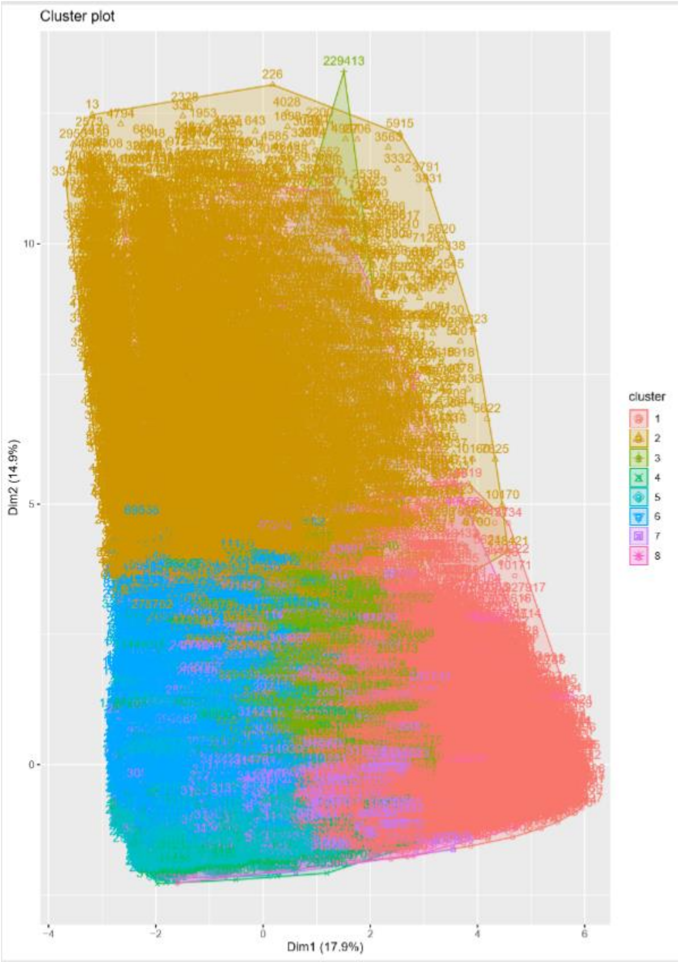
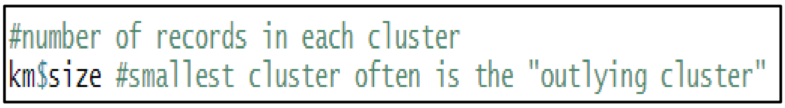


Figure 6: Cluster Plot

Thereafter, elbow plot was plotted to determine the optimal number of clusters i.e. 8 according to Figure 5. Cluster plot was then plotted as illustrated in Figure 6.

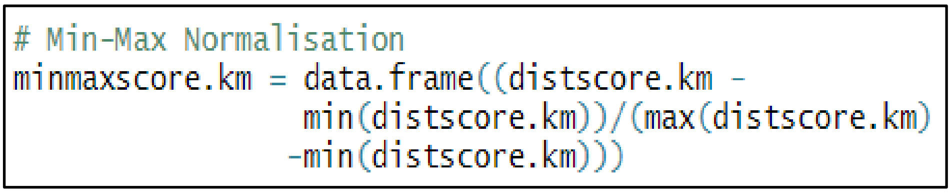
2.4 Outlier Detection



```
#number of records in each cluster
km$size #smallest cluster often is the "outlying cluster"
```

Figure 7: Outlier detections

From Figure 7, while the outcomes leads to a cluster with significantly lesser number of records, upon further investigation of the data, majority of the inputs in those clusters consisted of people who had non-alarming loan statuses (Decent credit ratings above D who are either Paid or Current).



```
# Min-Max Normalisation
minmaxscore.km = data.frame((distscore.km -
                  min(distscore.km))/(max(distscore.km)
                 -min(distscore.km)))
```

Figure 8: Min-max normalisation

To scale the distance of each input from cluster centroid, min-max normalisation was carried out (Figure 8). Lastly, top 100 outliers based on the distance from cluster centroid was extracted and analysed.

2.5 Dimension Reduction

The top outlier (labelled as 229413) was identified to be an individual with extremely high income of $110,002,500. The extremely high-income value skewed the dataset to produce the similar set of outliers when numerous permutations of clustering with different variable combinations was run.

Hence, we removed the single top outlier and repeated processes 2.2 to 2.4. Secondly, since income variable was deemed to be the ‘noisy’ factor to the dataset, we removed the entire income variable from the dataset and re-ran 2.2 to 2.4.

2.5 Summary of results

The result for the first-dimension reduction then produced the individual with second highest income to be the top outlier. The result for the second-dimension reduction resulted in the outliers with high number of accounts past due.

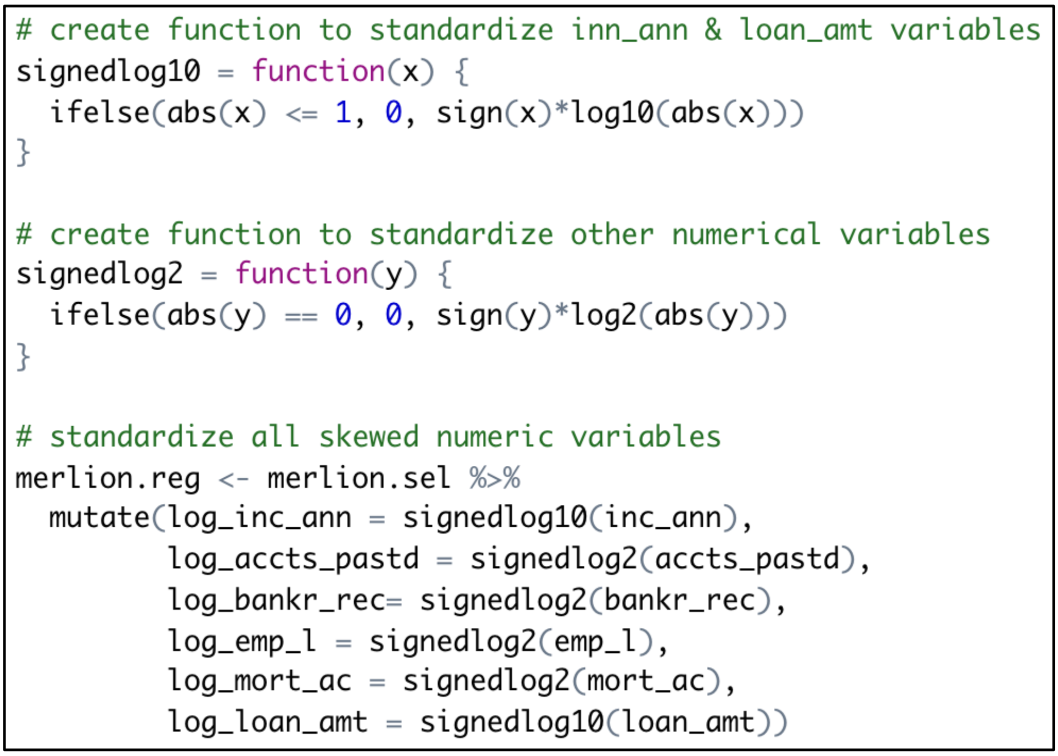
3. Regression

3.1 Objective

The objective is to predict the loan amount of new customers and regression model will be employed.

3.2 Pre-processing variables

Because dataset has varying scale (Appendix C), variable with the larger range may give a higher weight in the analysis (Galili, 2013). Hence, we performed logarithmic transformation on variables to prevent this problem.



```
# create function to standardize inn_ann & loan_amt variables
signedlog10 = function(x) {
  ifelse(abs(x) <= 1, 0, sign(x)*log10(abs(x)))
}

# create function to standardize other numerical variables
signedlog2 = function(y) {
  ifelse(abs(y) == 0, 0, sign(y)*log2(abs(y)))
}

# standardize all skewed numeric variables
merlion.reg <- merlion.sel %>%
  mutate(log_inc_ann = signedlog10(inc_ann),
         log_accts_pastd = signedlog2(accts_pastd),
         log_bankr_rec= signedlog2(bankr_rec),
         log_emp_l = signedlog2(emp_l),
         log_mort_ac = signedlog2(mort_ac),
         log_loan_amt = signedlog10(loan_amt))
```

Figure 9: Functions to standardize variables

From Figure 9, we created function of log-base10 and log-base2 to transform variables, allowing us to put the variables on a comparable scale, preventing distortion to our analysis seen in Appendix D and E.

3.3 Variables and records selection

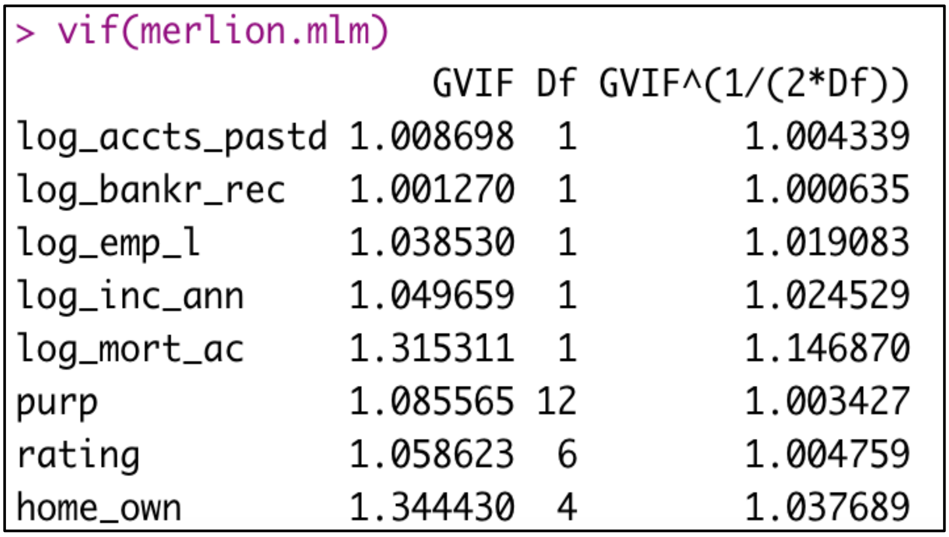
Variables are first selected using our domain knowledge followed by researching on some of the characteristics that banks typically look at before giving out personal loan. These includes credit score, current income, employment history and repayment history (Segal, 2017). Given that, we have selected a set of predictor variables to predict the loan amount of new customers (Appendix F).

Besides that, records between 2008 and 2012 were dropped, reason being significant lower loan amounts were given during this period. As regression predictive analysis assumes what happened in the past will continue to occur in the future, these records were removed to prevent the model from systematically underestimating future customers' loan amount.

3.4 Partition

In Appendix G, results from 70-30 split gives a better accuracy and correlation than the others.

3.5 Summary of results from R-script



```
> vif(merlion.mlm)
                    GVIF Df GVIF^(1/(2*Df))
log_accts_pastd 1.008698  1        1.004339
log_bankr_rec   1.001270  1        1.000635
log_emp_l       1.038530  1        1.019083
log_inc_ann     1.049659  1        1.024529
log_mort_ac     1.315311  1        1.146870
purp            1.085565 12        1.003427
rating          1.058623  6        1.004759
home_own        1.344430  4        1.037689
```

Figure 10: Test for multicollinearity

From Figure 10, there is little multicollinearity between predictor variables (Appendix H), satisfying the assumption that predictors are not highly correlated with each other in a multivariate model.

However, the adjusted R^2 is low, suggesting that predictor variables are only able to explain 18.16% of the variation in loan amount (Appendix F.1), albeit, most of the variables' p-value are statistically significant, indicating that there is a relationship between predictor variables and loan amount. This could be due to the nature of the dataset, which warrants the need for more data collection.

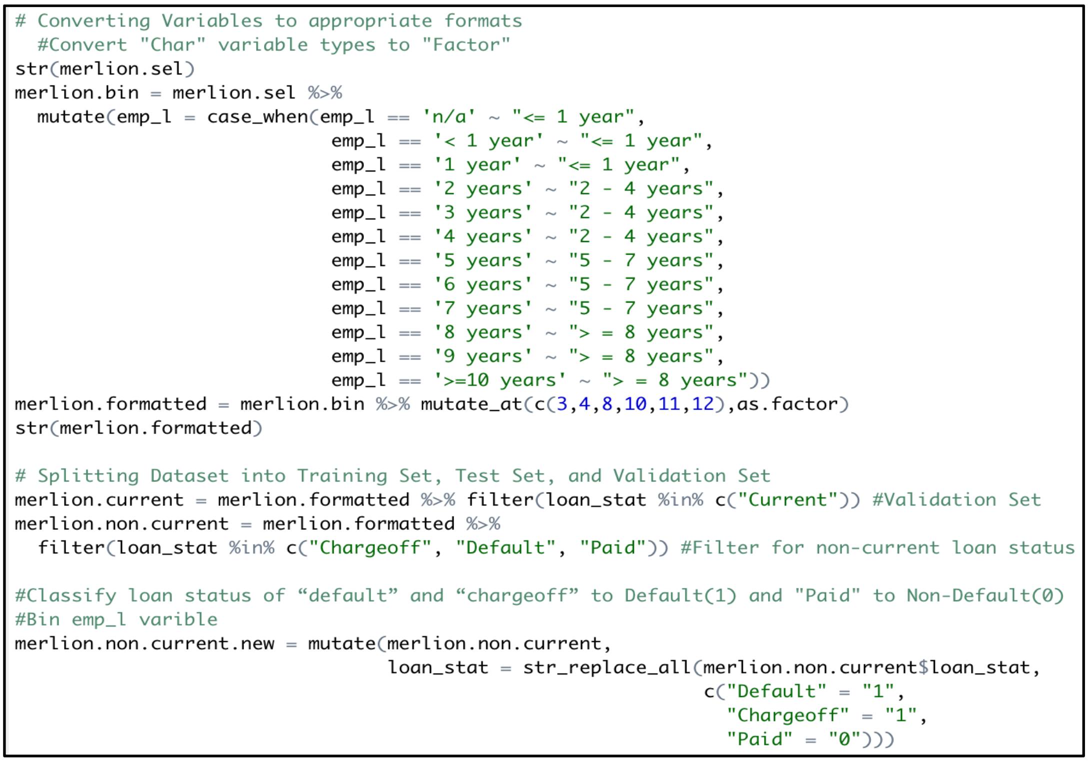
4. Classification

4.1 Objective

The objective is to obtain the classification rules to predict the likelihood of Merlion’s customer to default.

4.2 Pre-processing variables

Next, we split the dataset into “current” and “non-current” dataset according to loan status (Figure 11).



```
# Converting Variables to appropriate formats
  #Convert "Char" variable types to "Factor"
str(merlion.sel)
merlion.bin = merlion.sel %>%
  mutate(emp_l = case_when(emp_l == 'n/a' ~ "<= 1 year",
                           emp_l == '< 1 year' ~ "<= 1 year",
                           emp_l == '1 year' ~ "<= 1 year",
                           emp_l == '2 years' ~ "2 - 4 years",
                           emp_l == '3 years' ~ "2 - 4 years",
                           emp_l == '4 years' ~ "2 - 4 years",
                           emp_l == '5 years' ~ "5 - 7 years",
                           emp_l == '6 years' ~ "5 - 7 years",
                           emp_l == '7 years' ~ "5 - 7 years",
                           emp_l == '8 years' ~ "> = 8 years",
                           emp_l == '9 years' ~ "> = 8 years",
                           emp_l == '>=10 years' ~ "> = 8 years"))
merlion.formatted = merlion.bin %>% mutate_at(c(3,4,8,10,11,12),as.factor)
str(merlion.formatted)

# Splitting Dataset into Training Set, Test Set, and Validation Set
merlion.current = merlion.formatted %>% filter(loan_stat %in% c("Current")) #Validation Set
merlion.non.current = merlion.formatted %>%
  filter(loan_stat %in% c("Chargeoff", "Default", "Paid")) #Filter for non-current loan status

#Classify loan status of “default” and “chargeoff” to Default(1) and "Paid" to Non-Default(0)
#Bin emp_l varible
merlion.non.current.new = mutate(merlion.non.current,
                                 loan_stat = str_replace_all(merlion.non.current$loan_stat,
                                                             c("Default" = "1",
                                                               "Chargeoff" = "1",
                                                               "Paid" = "0")))
```

Figure 11: Processing Variables

The “non-current” dataset contains loan statuses “default” and “charge-off”, classified as default(1) and “Paid” as non-default(0). The “non-current” dataset is used for classification and the “current” dataset is where we will be predicting the likelihood of default of current loans using our model.

Next, we binned the variable emp_l in intervals to improve visualisation in the classification rules. Lastly, we converted the variable type of character variables to factors. We also decided to balance our dataset using oversampling to improve our model’s predictive power of the default class (Appendix I).

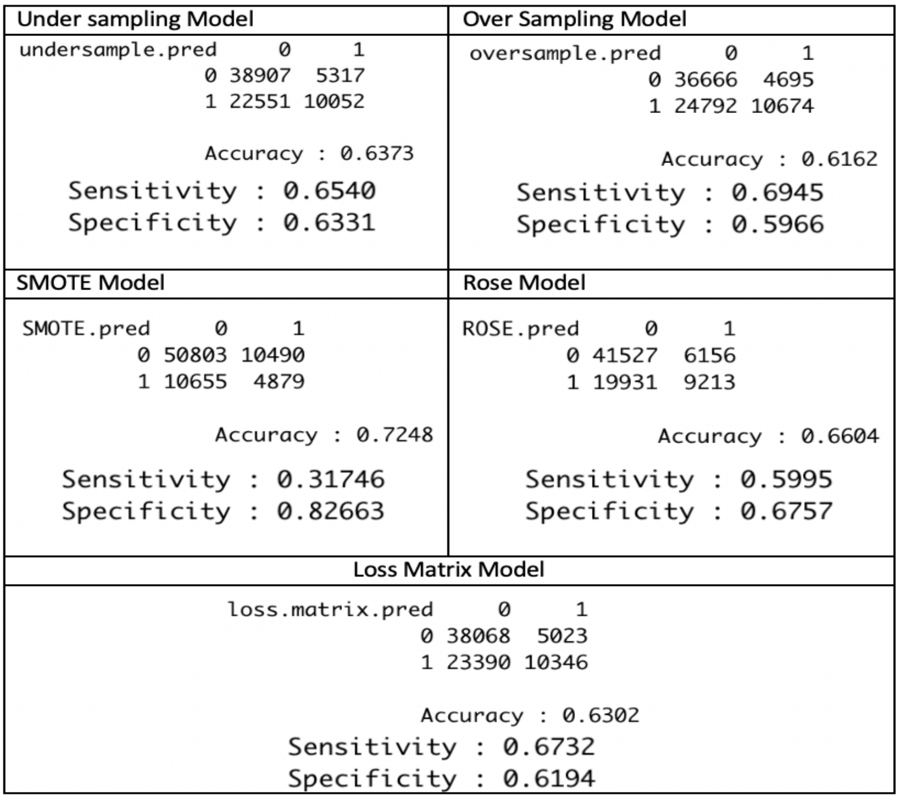
4.3 Variables and records selection

Fund_mt variable was removed because it doesn't provide significant analytical value to the model. The remaining variables were selected based on domain knowledge that they can be used to classify customers.

4.4 Partition

We partitioned our data into 60-40 using the “createDataPartition” function to apply the same proportion of “default” and “Non-default” records for both train and test dataset (Appendix J).

5.5 Summary of results from R-script



Under sampling Model

```
undersample.pred     0     1
               0 38907  5317
               1 22551 10052

               Accuracy : 0.6373
    Sensitivity : 0.6540
    Specificity : 0.6331
```

Over Sampling Model

```
oversample.pred     0     1
              0 36666  4695
              1 24792 10674

               Accuracy : 0.6162
    Sensitivity : 0.6945
    Specificity : 0.5966
```

SMOTE Model

```
SMOTE.pred     0     1
         0 50803 10490
         1 10655  4879

               Accuracy : 0.7248
    Sensitivity : 0.31746
    Specificity : 0.82663
```

Rose Model

```
ROSE.pred     0     1
        0 41527  6156
        1 19931  9213

               Accuracy : 0.6604
    Sensitivity : 0.5995
    Specificity : 0.6757
```

Loss Matrix Model

```
loss.matrix.pred     0     1
               0 38068  5023
               1 23390 10346

               Accuracy : 0.6302
    Sensitivity : 0.6732
    Specificity : 0.6194
```

Figure 12: Confusion Matrix Comparison

Sensitivity was used as the selection criteria because it indicates the model’s ability to predict the likelihood of customers to default. Hence, the Over-Sampling Model was selected (Figure 12). The model’s accuracy, sensitivity and specificity are 61.62%, 69.45% and 59.66% respectively. The sensitivity score shows that the model was able to correctly predict 10674 defaults out of the 15369 predicted defaults. Other results can be found in Appendix K and L. Overall, our model’s ability to predict customer defaults is higher than its ability to predict non-defaults.

Appendix N and Appendix M shows how customers are classified, based on their credit rating, mortgage account, interest rate, loan plan, loan amount and length of employment.

Main Text Word Count: 2,000 words

References:

Galili, T. (2013). Log Transformations for Skewed and Wide Distributions. Retrieved from: https://www.r-statistics.com/2013/05/log-transformations-for-skewed-and-wide-distributions-from-practical-data-science-with-r/

Kagan, J., & Segal, T. (2019). Five Cs of Credit. Retrieved from: https://www.investopedia.com/terms/f/five-c-credit.asp

Mekala, H. (2018). Dealing with Missing Data using R. Retrieved from: https://medium.com/coinmonks/dealing-with-missing-data-using-r-3ae428da2d17

Segal, B. (2017). How Do Banks Decide My Personal Loan Eligibility?. Retrieved from: https://www.gobankingrates.com/loans/personal/banks-decide-personal-loan-eligibility/

Appendices

Appendix A: More complex Classification Model

A more complex Classification Model is possible given the current dataset, however, creating said model, would take a significantly longer time. As such, the following paragraphs will attempt to explain how the model would work, and that it should help to improve the accuracy of the model as compared to the ROSE model.

This new model will be more accurate as it will use information obtained from the Clustering Model above to better predict the defaulters.

We will first perform the clustering similar to above, where now it will be crucial to add to the dataset which cluster each loan will belong to.

After which, we will segment the dataset into each cluster, where we will perform a round of classification to identify the best model and the respective rules for each unique cluster.

From this point onwards, for any new customer loans, we will run a separate classification to identify which cluster the new loan will fall into. After which, we will use the specific rules of the cluster to predict the default or non-default of the loan.

Appendix B: How to handle missing values

In the event of missing values, we will perform the following steps:

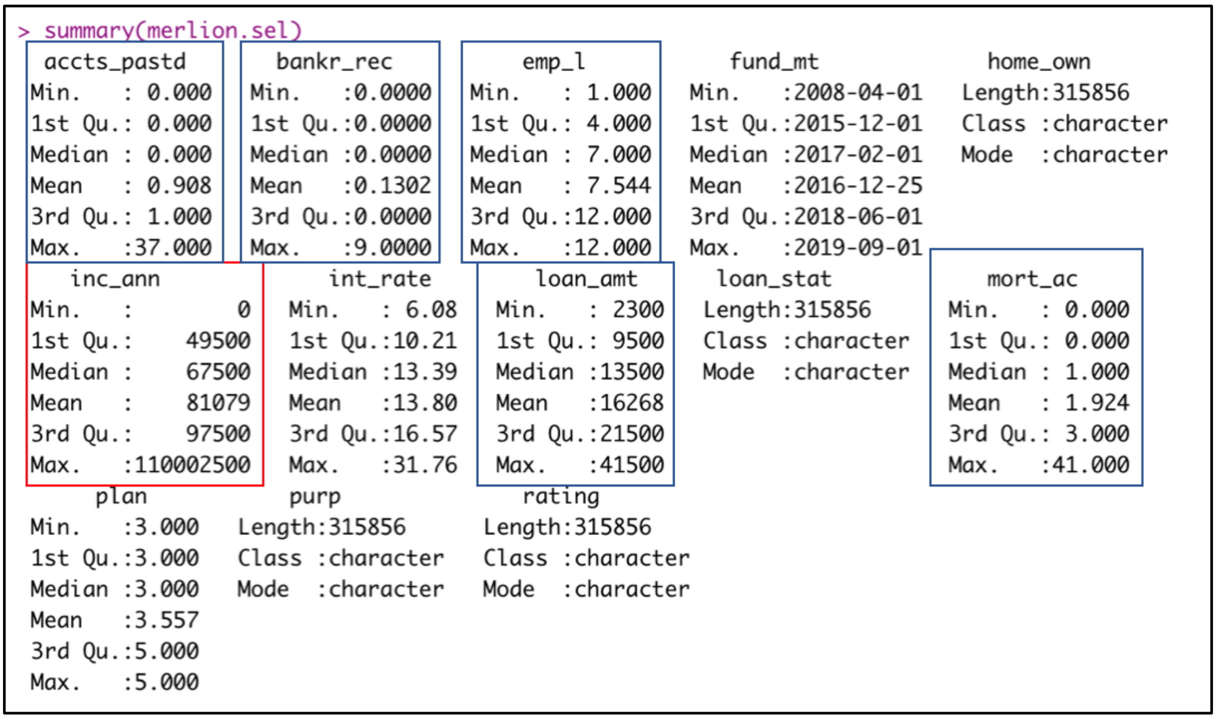
I. If the proportion of missing value is less than 5% of the entire dataset, they can be disregarded and analysis can be performed on the rest of the data.

II. If the missing values are too large, we will impute these values using mean, median or mode of the variable. For instance, numerical data can be filled with mean, and categorical data can be imputed with the mode so that the overall mean of the variable remains constant and the dataset is strong at predicting outcomes that occur more frequently.

III. Otherwise, there are also various packages available in R – Hmisc, MICE and Amelia – for handling missing values which could assist us in imputing these values (Mekala, 2018).

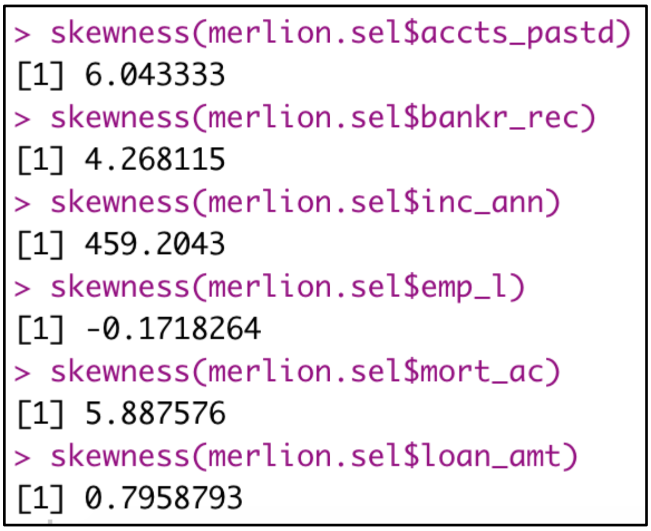
Appendix C: Check the skewness of variables

We ran a summary code and checked on the skewness of the variables to standardize them before carrying out regression analysis.



```
> summary(merlion.sel)
  accts_pastd       bankr_rec           emp_l             fund_mt              home_own        
 Min.   : 0.000   Min.   :0.0000   Min.   : 1.000   Min.   :2008-04-01   Length:315856     
 1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 4.000   1st Qu.:2015-12-01   Class :character  
 Median : 0.000   Median :0.0000   Median : 7.000   Median :2017-02-01   Mode  :character  
 Mean   : 0.908   Mean   :0.1302   Mean   : 7.544   Mean   :2016-12-25                     
 3rd Qu.: 1.000   3rd Qu.:0.0000   3rd Qu.:12.000   3rd Qu.:2018-06-01                     
 Max.   :37.000   Max.   :9.0000   Max.   :12.000   Max.   :2019-09-01                     
    inc_ann             int_rate        loan_amt       loan_stat             mort_ac      
 Min.   :        0   Min.   : 6.08   Min.   : 2300   Length:315856      Min.   : 0.000  
 1st Qu.:    49500   1st Qu.:10.21   1st Qu.: 9500   Class :character   1st Qu.: 0.000  
 Median :    67500   Median :13.39   Median :13500   Mode  :character   Median : 1.000  
 Mean   :    81079   Mean   :13.80   Mean   :16268                      Mean   : 1.924  
 3rd Qu.:    97500   3rd Qu.:16.57   3rd Qu.:21500                      3rd Qu.: 3.000  
 Max.   :110002500   Max.   :31.76   Max.   :41500                      Max.   :41.000  
      plan            purp              rating         
 Min.   :3.000   Length:315856      Length:315856     
 1st Qu.:3.000   Class :character   Class :character  
 Median :3.000   Mode  :character   Mode  :character  
 Mean   :3.557                                        
 3rd Qu.:5.000                                        
 Max.   :5.000                                        
```

The purpose of running this code is to give us an overview of the statistical measures of various numerical variables that are important for our analysis. From this summary, we can see that inc_ann (highlighted in red) has a wide distribution. As for the other numerical variables (highlighted in blue), it is not obvious whether they have a large distribution. Hence, we used skewness() function from e1071 package to check on their skewness as shown below.

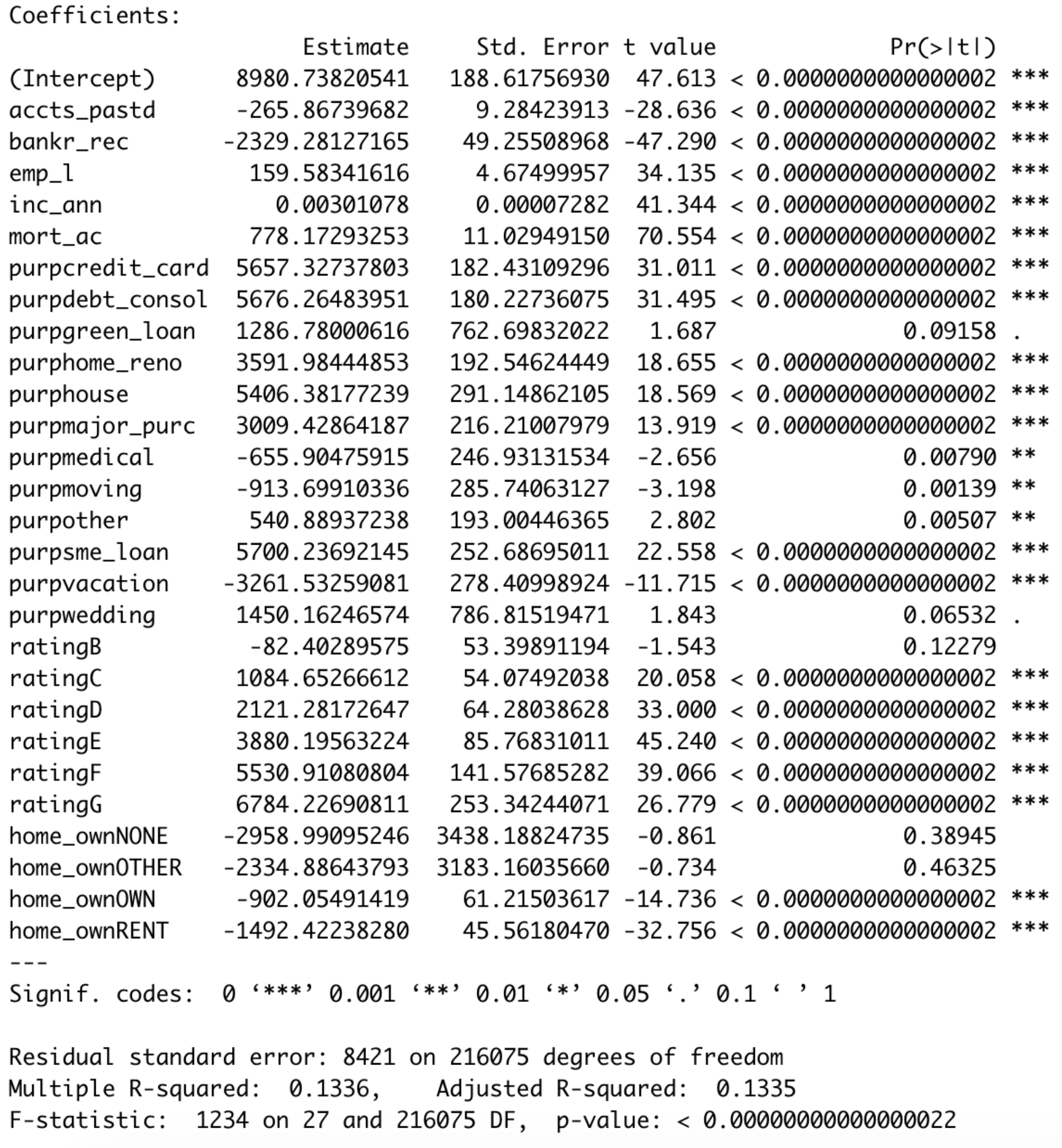


```
> skewness(merlion.sel$accts_pastd)
[1] 6.043333
> skewness(merlion.sel$bankr_rec)
[1] 4.268115
> skewness(merlion.sel$inc_ann)
[1] 459.2043
> skewness(merlion.sel$emp_l)
[1] -0.1718264
> skewness(merlion.sel$mort_ac)
[1] 5.887576
> skewness(merlion.sel$loan_amt)
[1] 0.7958793
```

From the image above, we can see that accts_pastd, bankr_rec and mort_ac are highly skewed to the right and loan_amt is moderately skewed. Although emp_l is rather symmetric, we will still perform standardization to the variable to prevent the problem of larger range variable giving higher weight in the regression analysis.

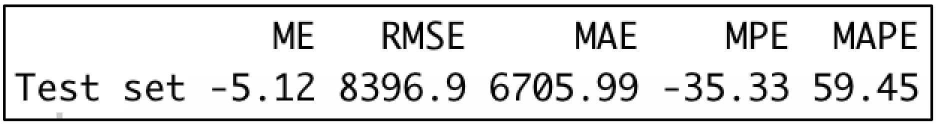
Appendix D: Results attained without standardising variables

The regression results shown below are attained without standardising the variables.



```
Coefficients:
                    Estimate     Std. Error t value             Pr(>|t|)    
(Intercept)      8980.73820541   188.61756930  47.613 < 0.0000000000000002 ***
accts_pastd      -265.86739682     9.28423913 -28.636 < 0.0000000000000002 ***
bankr_rec       -2329.28127165    49.25508968 -47.290 < 0.0000000000000002 ***
emp_l             159.58341616     4.67499957  34.135 < 0.0000000000000002 ***
inc_ann             0.00301078     0.00007282  41.344 < 0.0000000000000002 ***
mort_ac           778.17293253    11.02949150  70.554 < 0.0000000000000002 ***
purpcredit_card  5657.32737803   182.43109296  31.011 < 0.0000000000000002 ***
purpdebt_consol  5676.26483951   180.22736075  31.495 < 0.0000000000000002 ***
purpgreen_loan   1286.78000616   762.69832022   1.687              0.09158 .  
purphome_reno    3591.98444853   192.54624449  18.655 < 0.0000000000000002 ***
purphouse        5406.38177239   291.14862105  18.569 < 0.0000000000000002 ***
purpmajor_purc   3009.42864187   216.21007979  13.919 < 0.0000000000000002 ***
purpmedical      -655.90475915   246.93131534  -2.656              0.00790 ** 
purpmoving       -913.69910336   285.74063127  -3.198              0.00139 ** 
purpother         540.88937238   193.00446365   2.802              0.00507 ** 
purpsme_loan     5700.23692145   252.68695011  22.558 < 0.0000000000000002 ***
purpvacation    -3261.53259081   278.40998924 -11.715 < 0.0000000000000002 ***
purpwedding      1450.16246574   786.81519471   1.843              0.06532 .  
ratingB           -82.40289575    53.39891194  -1.543              0.12279    
ratingC          1084.65266612    54.07492038  20.058 < 0.0000000000000002 ***
ratingD          2121.28172647    64.28038628  33.000 < 0.0000000000000002 ***
ratingE          3880.19563224    85.76831011  45.240 < 0.0000000000000002 ***
ratingF          5530.91080804   141.57685282  39.066 < 0.0000000000000002 ***
ratingG          6784.22690811   253.34244071  26.779 < 0.0000000000000002 ***
home_ownNONE    -2958.99095246  3438.18824735  -0.861              0.38945    
home_ownOTHER   -2334.88643793  3183.16035660  -0.734              0.46325    
home_ownOWN      -902.05491419    61.21503617 -14.736 < 0.0000000000000002 ***
home_ownRENT    -1492.42238280    45.56180470 -32.756 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8421 on 216075 degrees of freedom
Multiple R-squared:  0.1336,	Adjusted R-squared:  0.1335 
F-statistic:  1234 on 27 and 216075 DF,  p-value: < 0.00000000000000022
```

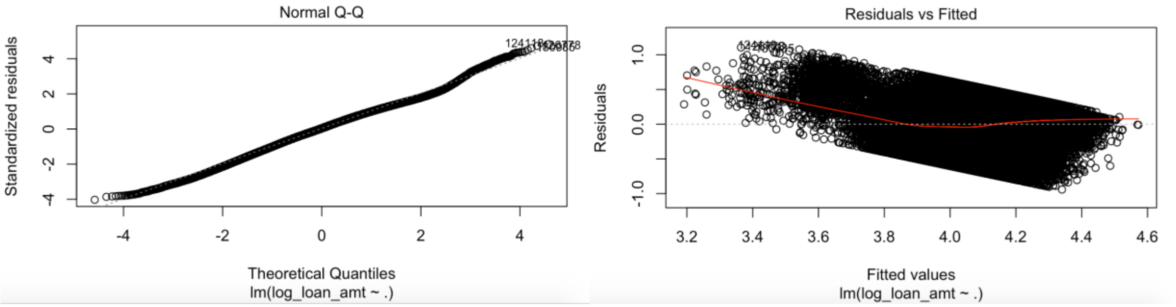


```
            ME    RMSE     MAE    MPE  MAPE
Test set -5.12 8396.9 6705.99 -35.33 59.45
```

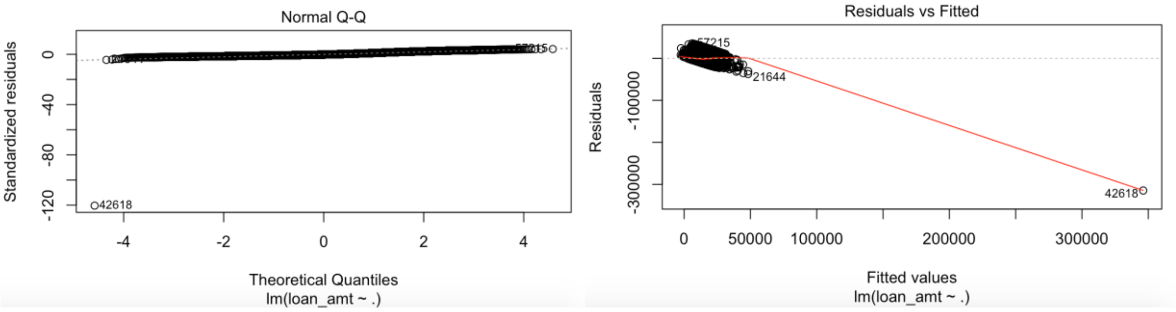
Although the p-value for most of the predictor variables are statistically significant, looking at the accuracy measures above, the MAPE of the regression model without standardising the variables is 59.45%. This suggests that the model is only 40.55% accurate in predicting the loan amount. Therefore, it is important to transform variables to comparable scales and prevent larger range variables from distorting our analysis.

Appendix E: Compare spread of residual with and without log transformation

Spread of residual around the logarithmic transformed regression model:



Spread of residual around the regression model **without** logarithmic transformation:



The log-base 10 function is used to transform monetary amounts in orders of 10 and the log-base 2 is used to transform other numerical variables in orders of 2.

Referring to the plots above, it can be seen that the spread of residual around the logarithmic transformed regression model is better than the regression model without logarithmic transformation. The dotted line on the “Residual vs Fitted” plot represents the hyperplane of the linear model and the red solid line is the fitted line of the residuals’ distances around the model. As for the “Normal Q-Q” plot, the dotted line represents the regression model.

Based on the “Normal Q-Q” plot from the logarithmic transformed regression model, the residuals fit nicely around the model. As such, the fitted line on the “Residual vs Fitted” plot is close to the hyperplane. However, the “Normal Q-Q” plot from the regression model without logarithmic transformed, has an extreme outlier, resulting in the fitted line on the “Residual vs Fitted” plot to be pulled towards the outlier.

Therefore, based on these plots, we can conclude that without logarithmic transformation, large range variable is likely to distort the model, resulting in high inaccuracy. Hence, it is important to standardise the variables before carrying out the regression analysis.

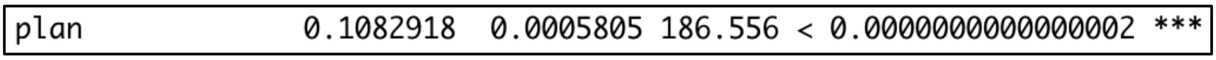
Appendix F: Rationale for selected variables and justification

Predictor Variables	Rationale	Justification (Appendix E.1)
accts_pastd	This variable tells us the number client's accounts that are not paid on time. As such, the larger the number, the ability of the client to pay back the loan is lower. Thus, Merlion should give out lesser loan to the customer.	This is also consistent with the output of our model, as the coefficient for log_accts_pastd is negative. This means that for every one unit increase in accts_pastd, the loan amount should decrease by 0.023.
bankr_rec	This variable tells us the number of previous bankruptcies record of the customer. The higher the number, the ability of the client to pay back the loan is lower, and Merlion should give out lesser loan to the customer.	This is also consistent with the output of our model, as the coefficient for log_bankr_rec is negative. This means that for every one unit increase in bankr_rec, the loan amount should decrease by 0.071.
emp_l	This variable tells us the client's length of employment. The greater the number, the higher the ability of the client to pay back the loan and thus Merlion can give out larger loan amount to such customer.	This is also consistent with the output of our model, as the coefficient for log_emp_l suggests that for every one unit increase in emp_l, the loan amount will increase by 0.016.
inc_ann	This variable tells us the client's annual income. The higher the income, the greater ability the client has to pay back and thus Merlion can give out larger loan amount.	This is also consistent with the output of our model, as the coefficient for log_inc_ann is positive. This means that for every one unit increase in inc_ann, the loan amount will increase by 0.111.
mort_ac	This variable indicates the number of mortgage savings accounts the client has. The higher the number, the greater the ability of the client in paying back the loan. Thus, Melion can give out larger loan amount to such customer.	This is also consistent with the output of our model, as the coefficient for log_mort_acc is positive. This means that for every one unit increase in mort_acc, the loan amount will increase by 0.036.

purp	This variable tells us the objective of the client acquiring the loan and it is important for our analysis. For instance, if the client take personal loan from Merlion in order to satisfy his other debt, then the chances of the client to default on the loan is high.	Depending on the purpose, the loan amount may increase or decrease.
rating	This variable suggests the credit worthiness of the client. If the client has lower ratings, it is more likely that they are not in an ideal financial situation and therefore will want to take up a higher amount of loan from Merlion, and vice versa.	The negative coefficient for rating B shows that a customer having a higher credit rating of B would undertake a lower loan amount.
home_own	This variable suggests the other liability of the client. If the client is still paying rental for the house or do not have a proper lodging, Merlion should not give out huge amount of loan to such customer..	Depending on the status of the home_own, the coefficient impact on loan amount varies.

** The repayment history is excluded since we lack repayment details of new customer. Nevertheless, this information is substituted with debt and liability of the customer using accts_pastd and home_own to determine his/her ability to pay back.

Besides that, the plan variable is also not included in our regression analysis.

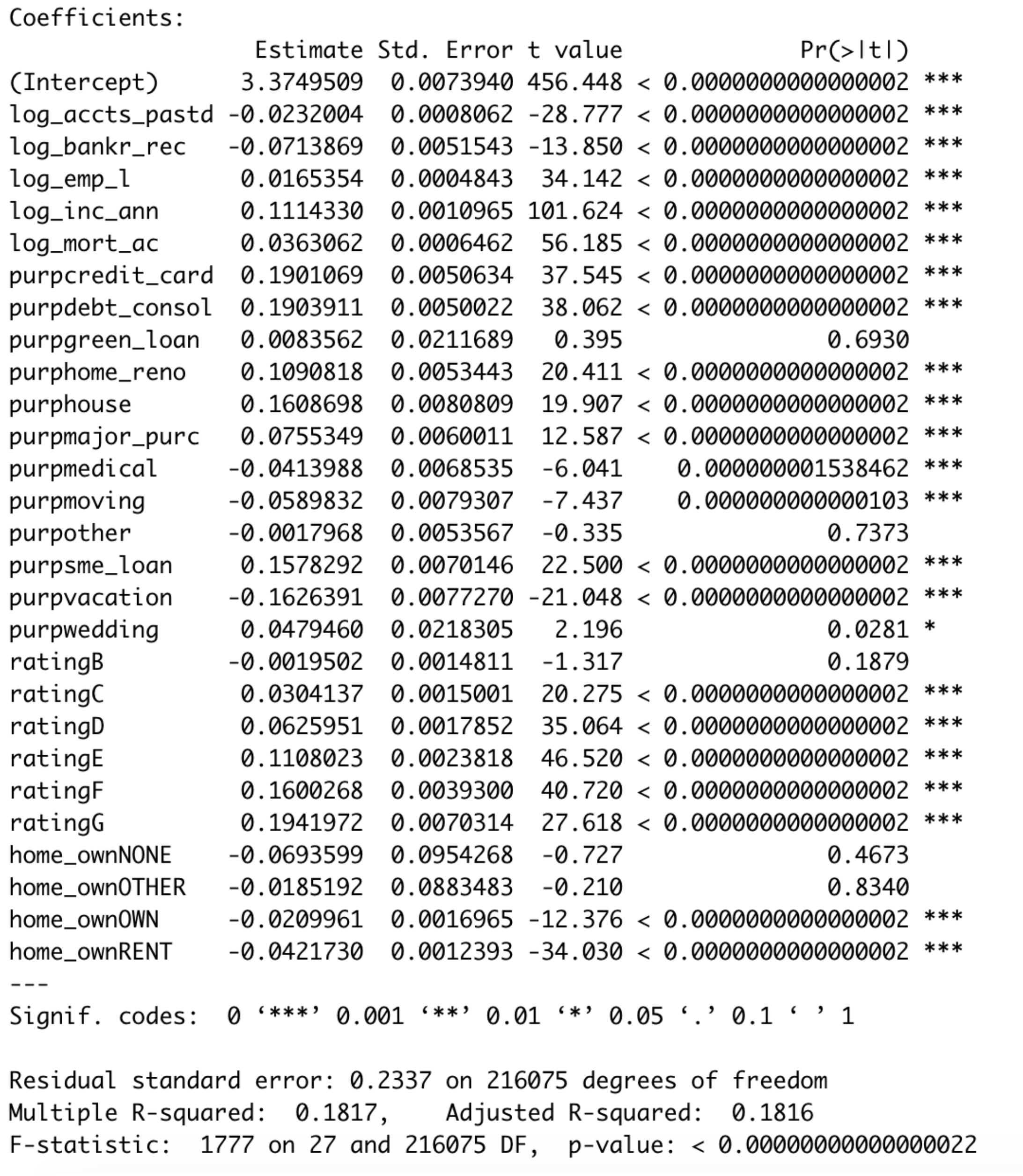


```
plan             0.1082918  0.0005805 186.556 < 0.0000000000000002 ***
```

Referring to the image above, when the plan variable is included in the regression analysis, the coefficient impact is positive. This means that the longer the plan, the higher the loan amount. However, the team feels that regardless of the length of the plan, Merlion should not give out higher loan amount for longer plan. This is because a borrower with a bad credit rating may request for a longer personal loan plan and the possibility of defaulting the loan is also higher. Therefore, by including plan as the predictor variable, it may distort the regression analysis and misrepresent the predicted loan amount.

In contrast, the plan variable can act as a rough gauge for Merlion to forecast how much the company is expected to receive from the borrower each month. Merlion can compute the Equated Monthly Installment (EMI) to form an expectation of the monthly payment to be received from the borrower after accepting the loan application. This can further help Merlion to assess whether the customer has the ability to pay back before deciding to give out the loan.

Appendix F.1: Coefficients of selected variables for the linear model



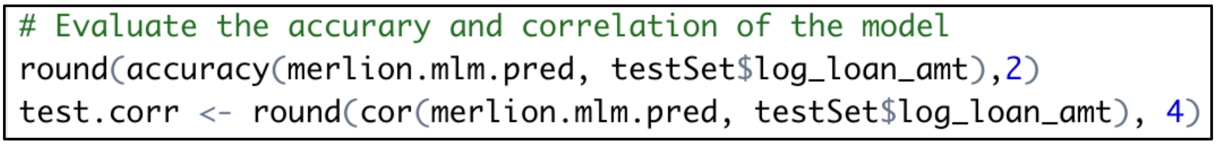
```
Coefficients:
                  Estimate Std. Error t value             Pr(>|t|)    
(Intercept)      3.3749509  0.0073940 456.448 < 0.0000000000000002 ***
log_accts_pastd -0.0232004  0.0008062 -28.777 < 0.0000000000000002 ***
log_bankr_rec   -0.0713869  0.0051543 -13.850 < 0.0000000000000002 ***
log_emp_l        0.0165354  0.0004843  34.142 < 0.0000000000000002 ***
log_inc_ann      0.1114330  0.0010965 101.624 < 0.0000000000000002 ***
log_mort_ac      0.0363062  0.0006462  56.185 < 0.0000000000000002 ***
purpcredit_card  0.1901069  0.0050634  37.545 < 0.0000000000000002 ***
purpdebt_consol  0.1903911  0.0050022  38.062 < 0.0000000000000002 ***
purpgreen_loan   0.0083562  0.0211689   0.395               0.6930    
purphome_reno    0.1090818  0.0053443  20.411 < 0.0000000000000002 ***
purphouse        0.1608698  0.0080809  19.907 < 0.0000000000000002 ***
purpmajor_purc   0.0755349  0.0060011  12.587 < 0.0000000000000002 ***
purpmedical     -0.0413988  0.0068535  -6.041    0.000000001538462 ***
purpmoving      -0.0589832  0.0079307  -7.437    0.000000000000103 ***
purpother       -0.0017968  0.0053567  -0.335               0.7373    
purpsme_loan     0.1578292  0.0070146  22.500 < 0.0000000000000002 ***
purpvacation    -0.1626391  0.0077270 -21.048 < 0.0000000000000002 ***
purpwedding      0.0479460  0.0218305   2.196               0.0281 *  
ratingB         -0.0019502  0.0014811  -1.317               0.1879    
ratingC          0.0304137  0.0015001  20.275 < 0.0000000000000002 ***
ratingD          0.0625951  0.0017852  35.064 < 0.0000000000000002 ***
ratingE          0.1108023  0.0023818  46.520 < 0.0000000000000002 ***
ratingF          0.1600268  0.0039300  40.720 < 0.0000000000000002 ***
ratingG          0.1941972  0.0070314  27.618 < 0.0000000000000002 ***
home_ownNONE    -0.0693599  0.0954268  -0.727               0.4673    
home_ownOTHER   -0.0185192  0.0883483  -0.210               0.8340    
home_ownOWN     -0.0209961  0.0016965 -12.376 < 0.0000000000000002 ***
home_ownRENT    -0.0421730  0.0012393 -34.030 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2337 on 216075 degrees of freedom
Multiple R-squared:  0.1817,	Adjusted R-squared:  0.1816 
F-statistic:  1777 on 27 and 216075 DF,  p-value: < 0.00000000000000022
```

Appendix G: Data partition justification

Data partitioning for linear regression is done using trial-and-error.

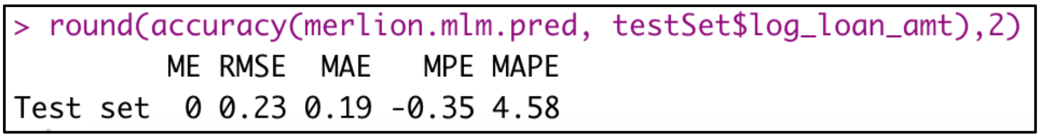
The accuracy and correlation of the model are tested through these codes:



```
# Evaluate the accurary and correlation of the model
round(accuracy(merlion.mlm.pred, testSet$log_loan_amt),2)
test.corr <- round(cor(merlion.mlm.pred, testSet$log_loan_amt), 4)
```

60-40 split

Accuracy from 60-40 split

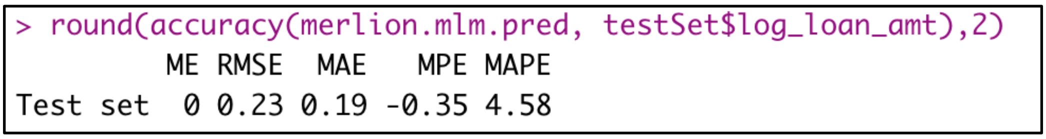


```
> round(accuracy(merlion.mlm.pred, testSet$log_loan_amt),2)
         ME RMSE  MAE   MPE MAPE
Test set  0 0.23 0.19 -0.35 4.58
```

Correlation from 60-40 split = 0.4313

70-30 split

Accuracy from 70-30 split (same as 60-40 split)

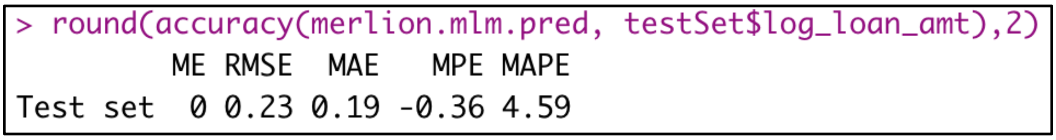


```
> round(accuracy(merlion.mlm.pred, testSet$log_loan_amt),2)
         ME RMSE  MAE   MPE MAPE
Test set  0 0.23 0.19 -0.35 4.58
```

Correlation from 70-30 split = 0.4332

80-20 split

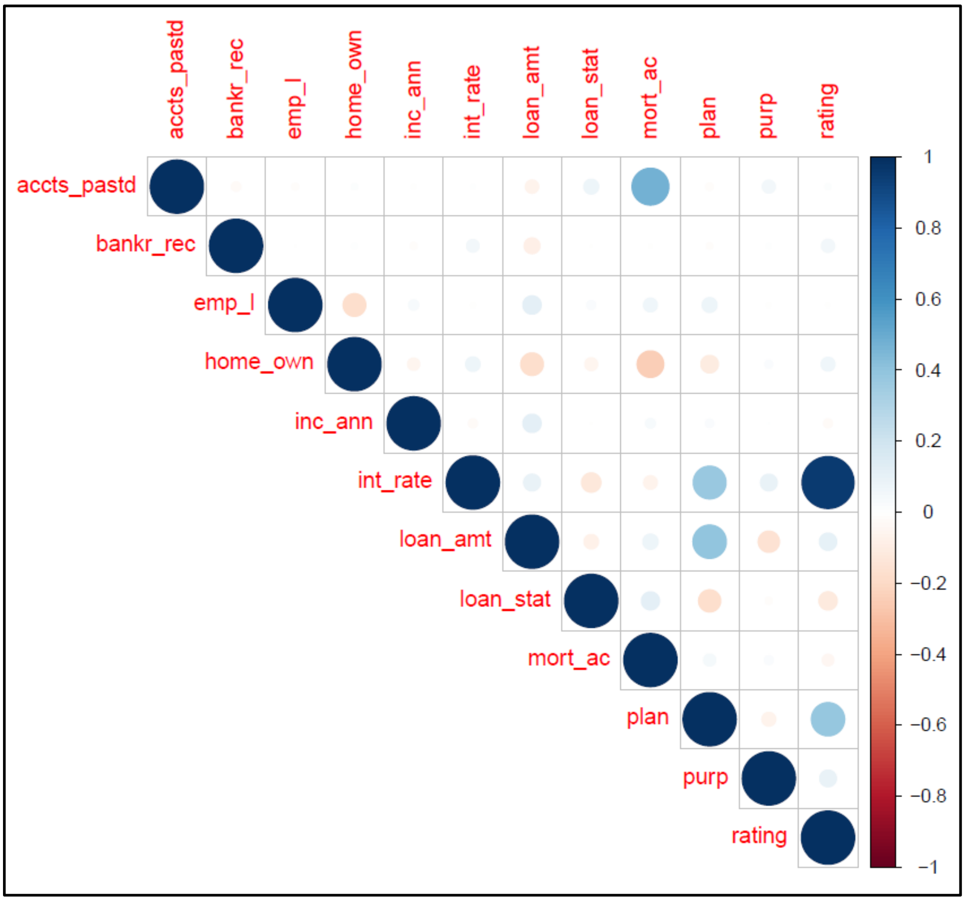
Accuracy from 80-20 split



```
> round(accuracy(merlion.mlm.pred, testSet$log_loan_amt),2)
         ME RMSE  MAE   MPE MAPE
Test set  0 0.23 0.19 -0.36 4.59
```

Correlation from 80-20 split = 0.431

Appendix H: Correlation matrix

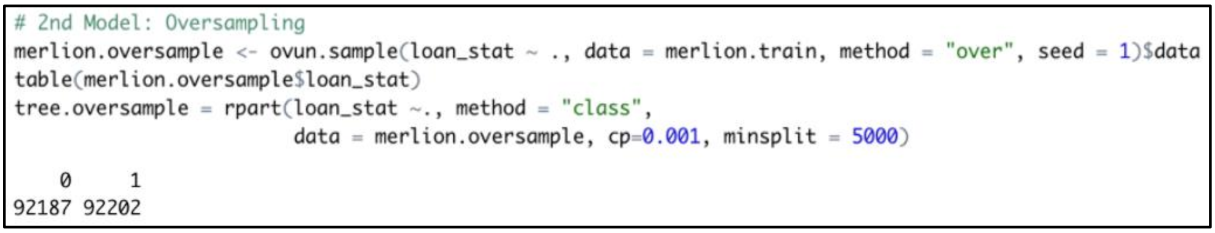


The correlation matrix is a quick guide to visualise relationships between variables. Positive correlation is displayed by the colour blue, while negative correlation is shown in red. The extent of the correlation is defined by the intensity of the colour and size of the circle. This means that the higher the correlation, the circle will be larger and darker shade.

Referring to the image above, rating and int_rate are correlated with one another while there is little multicollinearity between other variables. As such, in order to prevent multicollinearity from distorting the regression analysis, we will not include int_rate in the model. The reason is that int_rate is usually determined based on other characteristics of the borrower and is usually charged after the loan has been made. Hence, int_rate variable is not relevant for our regression analysis. On the other hand, the rating variable tells us the creditworthiness of the borrower and is often the lender's interest before he/she approves the loan.

Therefore, by removing int_rate from our analysis, it satisfies one of the assumptions that predictors are not highly correlated with each other in a multivariate model.

Appendix I: Data Balancing Using Oversampling



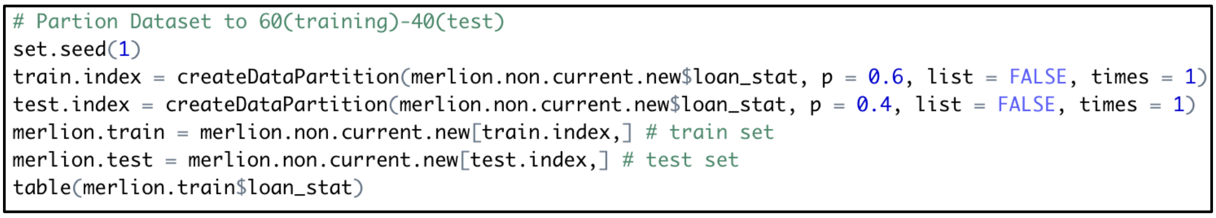
```
# 2nd Model: Oversampling
merlion.oversample <- ovun.sample(loan_stat ~ ., data = merlion.train, method = "over", seed = 1)$data
table(merlion.oversample$loan_stat)
tree.oversample = rpart(loan_stat ~., method = "class",
                        data = merlion.oversample, cp=0.001, minsplit = 5000)

    0     1
92187 92202
```

The original dataset comprises of a low amount of “Default” status records compared to “Non-defaults”. This allows our classification model to better predict customer who is unlikely to default but the predictive power of customers who are likely to default will be low. However, in the case of Merlion, the importance of predicting customers that will default is higher because they do not generate profits and will incurred heavy losses for Merlion.

After oversampling, the dataset is relatively balanced with 92187 records of “Non-default” and 92209 records of “Non-default”.

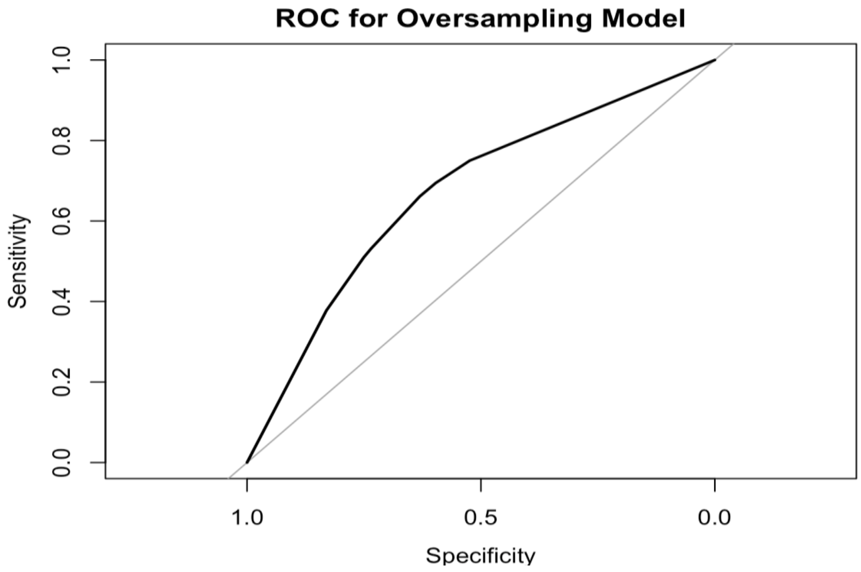
Appendix J: Data Partitioning Using “createDataPartition” Function



```
# Partion Dataset to 60(training)-40(test)
set.seed(1)
train.index = createDataPartition(merlion.non.current.new$loan_stat, p = 0.6, list = FALSE, times = 1)
test.index = createDataPartition(merlion.non.current.new$loan_stat, p = 0.4, list = FALSE, times = 1)
merlion.train = merlion.non.current.new[train.index,] # train set
merlion.test = merlion.non.current.new[test.index,] # test set
table(merlion.train$loan_stat)
```

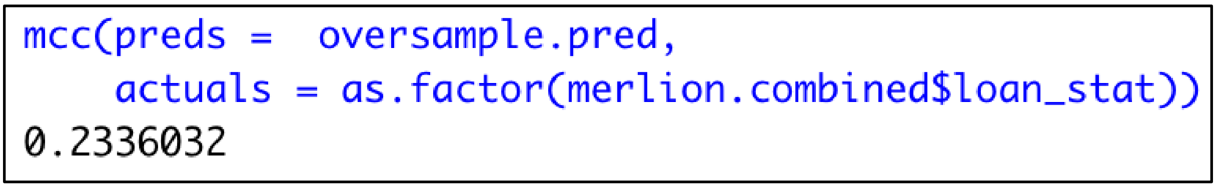
Using the “createDataPartition” would improve the accuracy of the classification model

Appendix K: ROC for Oversampling Model



The area under the curve: 0.6717. This shows the ability of our model to distinguish between whether the customer will default is slightly above average. (throw to appendix)

Appendix L: MCC (Measure of Performance)



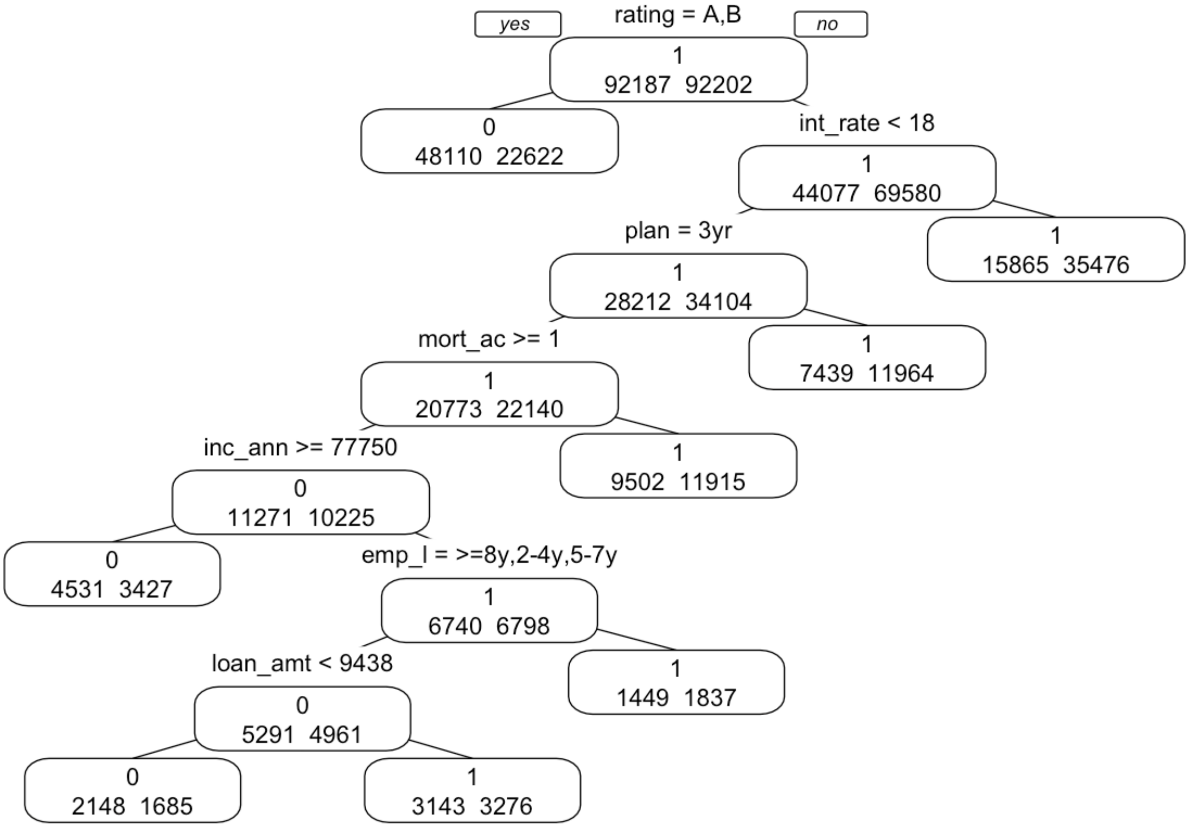
```
mcc(preds =  oversample.pred,
    actuals = as.factor(merlion.combined$loan_stat))
0.2336032
```

Our model's MCC of 0.2336032 shows the model is able to predict the likelihood of the customer's default to a certain extent.

Appendix M: Loss Matrix Model - Rules (Output) Table

Over Sampling Model - ClassificationRules (Output)
when rating is A or B -> **Non-Default**
when rating is C or D or E or F or G & int_rate < 18 & plan is 3 years & mort_ac >= 1 & inc_ann >= 77750 -> **Non-Default**
when rating is C or D or E or F or G & int_rate < 18 & plan is 3 years & mort_ac >= 1 & inc_ann < 77750 & emp_l is > = 8 years or 2 - 4 years or 5 - 7 years & loan_amt < 9438 -> **Non-Default**
when rating is C or D or E or F or G & int_rate < 18 & plan is 3 years & mort_ac >= 1 & inc_ann < 77750 & emp_l is > = 8 years or 2 - 4 years or 5 - 7 years & loan_amt >= 9438 -> **Default**
when rating is C or D or E or F or G & int_rate < 18 & plan is 3 years & mort_ac < 1 -> **Default**
when rating is C or D or E or F or G & int_rate < 18 & plan is 3 years & mort_ac >= 1 & inc_ann < 77750 & emp_l is <= 1 year -> **Default**
when rating is C or D or E or F or G & int_rate < 18 & plan is 5 years -> **Default**
when rating is C or D or E or F or G & int_rate >= 18 -> **Default**

Appendix N: Classification Tree Plot


yes
rating = A,B
no
1
92187 92202
0
48110 22622
int_rate < 18
1
44077 69580
plan = 3yr
1
15865 35476
1
28212 34104
mort_ac >= 1
1
7439 11964
1
20773 22140
inc_ann >= 77750
1
9502 11915
0
11271 10225
0
4531 3427
emp_l = >=8y,2-4y,5-7y
1
6740 6798
loan_amt < 9438
1
1449 1837
0
5291 4961
0
2148 1685
1
3143 3276