STAT2004 Analytics for Observational Data Semester 2, 2022 Final Project Report

Analytics for Observational Data Semester 2, 2022 Final Project Report

Detection of Credit Card Fraudulent Activity using Multivariate Data Analysis

Karmel Gandahusada – 19774158, Bachelor of Data Science, Honours
Harnoor Randhawa – 19633644, Bachelor of Actuarial Science

1. Declaration

The work presented in this report is my/our own work and all references are duly acknowledged.

This work has not been submitted, in whole or in part, in respect of any academic award at Curtin University or elsewhere.

JOANNES KARMEL GANDAHUSADA
31/10/22

HARNOOR RANDHAWA
31/10/22

2. Introduction

The rise of digital banking in the 21st century has made spending and borrowing money more convenient than ever. One can simply purchase products and services from anywhere in the world with just a click of a button and borrow money from the bank with a simple phone call. However, by increasing the accessibility to everyone's online bank, it is now possible to steal one's confidential credit card information and get away with a lot of money. These acts of digital crime cost many individuals and businesses millions of dollars per year and are required to be dealt with swiftly. As such, we will be looking at over 200,000 separate credit card transactions, both legitimate and fraudulent, and analyzing the correlations between the exploratory variables and the response class to understand which variables are most indicative of a fraudulent transaction.

The dataset we will be using is a table of 284,807 rows of European credit card transactions, 492 of them being fraudulent transactions. The dataset contains 31 unknown numerical variables (V1-V28) obtained using Principal Component Analysis (PCA), alongside seconds elapsed (time), numerical transaction amount (amount), and categorical class defining whether a class was fraudulent or not with 1s and 0s respectively. The original features of the principal components cannot be provided due to privacy issues. Alongside the multivariate data, capabilities of the Light GBM package will be investigated. The package contains the light gradient-boosting machine algorithms, which is based on decision trees and uses gradient-boosting methods to grow trees sequentially, meaning each subsequent tree will learn from the errors of the last tree. This package was chosen, as gradient-boosting methods work best on big data without overfitting. With the dataset and the packages chosen, the aim of this study will be to determine which variables in the dataset are most important in determining a fraudulent purchase. To do this, multivariate data analysis and inference techniques will be utilized to determine the significance of certain variables. Additionally, the classification techniques of logistic regression and boosted trees will be used to confirm that the significant variables seen in the previous analysis are important enough to use in the final and best models.

3. Methods

This section highlights the key methods used in this analysis and explain the reasoning behind their usage.

Exploratory data analysis (EDA) started by cleaning the data. This involved checking for missing or invalid values, zero-variance variates using the *nearZeroVar* function from the Caret package, or redundant columns not needed for analysis. After checking that the data was useable, two assumptions were made for this study to be able to use the logistic regression method.

1. The independence of errors – meaning the residuals do not have a relationship with the class variable. This was assumed as the data came from many different credit-card holders

2. There was a linear relationship between the logit of the class variable and the explanatory variables

The temporal context of transactions was of no interest in this paper due to the assumption of error independence, and as such, the Time column has been removed from the dataset. The assumption of multivariate normality was also essential to know, as having a normal dataset opens many doors to parametric testing and other methods with the assumption of multivariate normality. This was testing using the Royston, Mardia and Henze-Zirkler multivariate normality tests.

An essential visualisation to start with was the boxplots comparing the centers of variates. Conducting this comparison allowed for the observation of which variates from fraudulent transactions differed the most from variates from non-fraudulent transactions, and thus a good starting point for seeing which variates were the best indicator of fraudulent activity. Another informative plot was the scatter-paired plot between certain components. By differentiating the class of each data point by colour, the plot was able to point out hidden trends between variates from fraudulent data, despite each variable being orthogonal principal component variables.

Before continuing, the issue of data imbalance has to be dealt with, as this is a severe case of imbalanced classes, with almost 300,000 cases of non-fraudulent transactions and only 492 fraudulent transactions. Many issues can occur when proceeding with data analysis on imbalanced data, such as being unable to observe the true correlation between the explanatory variables and the response data, or overfitting to the majority class in the case of classification. This paper has decided to use Synthetic Minority Oversampling Technique (SMOTE) to deal with this issue, which is required to make any meaningful analysis of the fraudulent transactions. This sampling method works by creating synthetic samples for the minority class using pre-existing data, and under-sampling the majority class to match the number of minority classes. Synthetic samples are created by selecting n samples from the k-nearest neighbors of each minority data points, then using the following formula to generate a random sample that lies between point *x* and one of its neighbors.

$$x' = x + rand(0,1) * |x - x_n|$$

Figure 2.1: Formula used in SMOTE to create synthetic samples
rand(0,1) selects a random number from 0 to 1.

After data balance was ensured, K-means clustering was performed to mathematically differentiate the data points without the class variable. K-Means clustering is a linear method that aims to linearly group together data points by minimizing the distance between the cluster centroids and each data point in the same cluster. This is calculated using the within sum-of-squares (WSS) formula, which can be stated as below.

$$SS_W = \sum_{j=1}^{p} \sum_{i=1}^{n_j} (x - \underline{x})^2$$

Figure 2.2: Within sum of squares formula
p=number of groups, n = number of data points in group

By creating clusters for the data, more pairs of plots were then investigated to see whether two variates is enough to separate the data points into their respective clusters. However no combination of points seemed sufficient enough to separate the data points, and as such, t-distributed stochastic neighbour embedding (t-SNE) was used to better visualize the clusters through a non-linear dimension reduction technique.

Classification will be the last procedure in this report. The two methods chosen were logistic regression using coefficient penalisation and *Light Gradient Boosting Machine* (LightGBM), chosen as part of this report's package study. These two techniques were specifically chosen as they served as a form of feature selection and required few assumptions of the data distribution to perform well, aside from the assumptions stated earlier. During testing, logistic regression produced an easy to interpret equation, quick to train and predict, and was able to produce a sufficiently strong linear relationship between the logit of the response variable and the explanatory variables. Additionally by using coefficient penalization, variable importance can be gauged by identifying which variable were chosen in the final equation.

LightGBM is an ensemble model which uses decision trees as the learning method and enhances its performance using gradient boosting, meaning each tree is learnt sequentially from the previous tree. Under the hood, there are 2 principals to understand that form the backbone of the LightGBM algorithm

1. Gradients – Each observation of the data has a different contribution to the overall information gain criterion. This technique helps only retains observations that help increasing the information gain, which can be gauged by the observations with a higher gradient. LightGBM also randomly selects and drops observation with a lower gradient. This process helps to the algorithm to converge to a large accuracy. This can be expressed mathematically using the following equation.

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)} \right)$$

Figure 2.3: Equation to calculate information gain
gi denotes the negative gradients in the function.
Ai and Bi are the subsets in the data

2. Exclusive Feature Bundling – In a large dataset with multiple features, it is very important to reduce the number of features without loss of learning information. In a sparse feature matrix, it is rare to find non-zero variates. However for the variates that

are non-zero can be combined together to form an exclusive feature. The advantage of this technique is two-fold.

- Reduction of features without loss of accuracy.
- Increased speed of training.

While normal decision tree algorithms grow trees horizontal (known as level wise), LightGBM grows trees vertically (known as leaf wise). Due to this fundamental difference, the information loss in LightGBM is much lower compared to other decision tree algorithms.

4. Results

Examining the structure of the data, the principal components are centered around 0. No component was found to have any missing values or have near-zero variation either. The amount column represents the amount of money moved in the transaction, which we can assume to be in Euros. The biggest transaction in this dataset is 25,700, and the lowest is at 0. There are 1,826 transactions with 0 amount, and 2,713 transactions with a lower amount than 0.1. While this may seem like invalid data, this data may also be the result of companies charging very small amounts of money to customers to verify their bank accounts. Additionally, there are 32 fraudulent transactions having equal to or less than 0.1 amount, which goes against the common assumption of fraudulent transactions having big amounts of money moved.

When conducting multivariate normal testing using 2000 sampled rows (and the first 15 principal components due to memory restraints), it was found using the *Royston*, *Henze-Zirkler* and *Mardia* tests, that the data was not multivariate normal. All tests produced p-values of *0* respectively, which means we cannot accept the null hypothesis of multivariate normality at even a 1% level. As such, methods and tests that require the assumption of multi-variate normality are not able to be used. Additionally, when inspecting the adjusted quantile-quantile plot, 1011 out of 2000 data points are observed to be outliers, as they heavily deviate from the line of normality shown by the blue line.

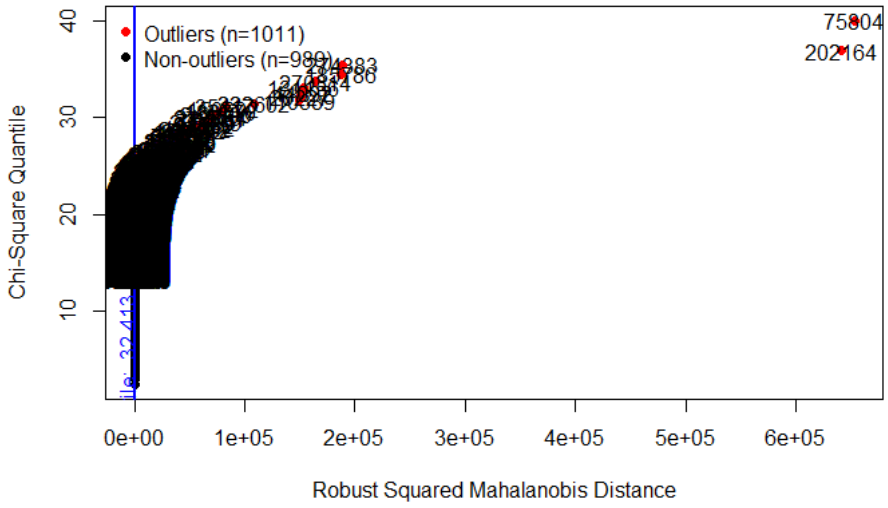


Figure 3.1: Adjusted Chi-Square Q-Q Plot

The skewed nature of the data can be further observed when comparing 6 variable distributions belonging to fraudulent and non-fraudulent transactions. These 7 variables were chosen for display due to the distinct differences in distribution between classes.

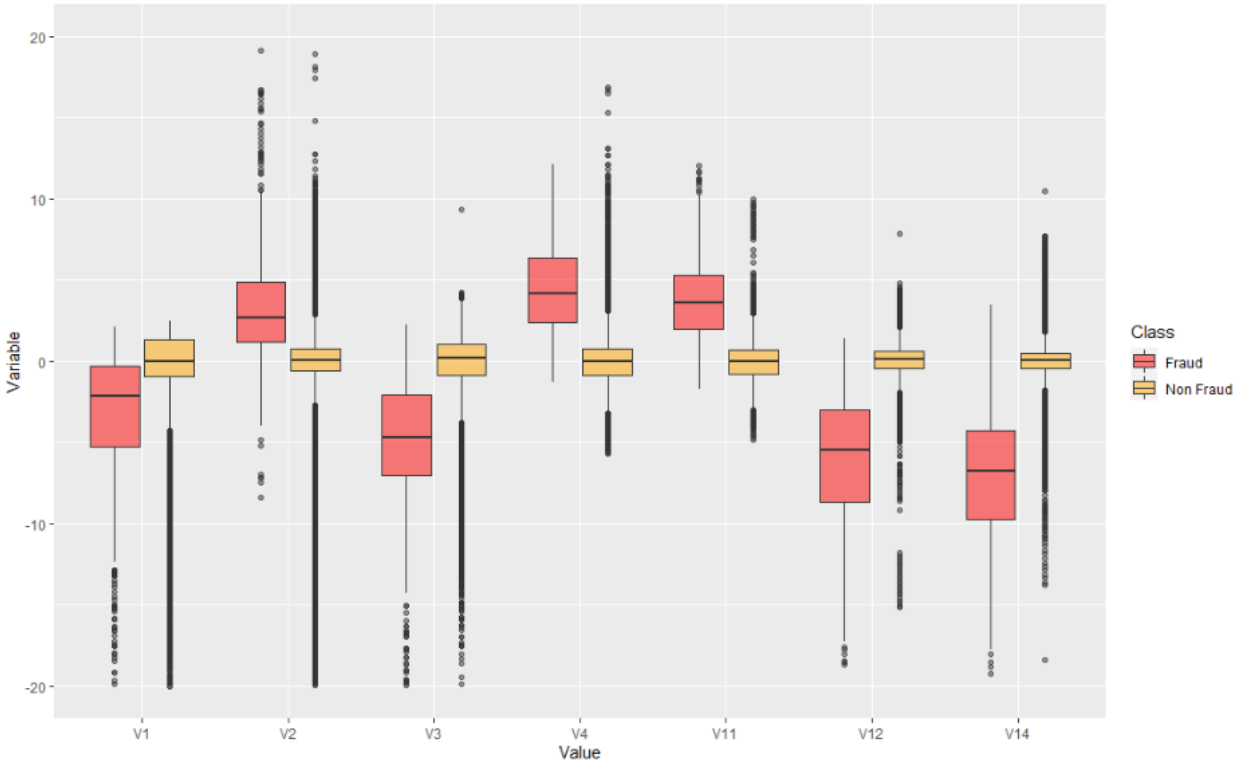


Figure 3.2: Boxplots of 7 variables separated by Class, with y-limits of (-20,20)

Using the boxplots to observe 7 components, we can note for each boxplot many outliers displayed as the black dots. For all variables except for V2, most of the outliers lie on one side of the boxplot, which explains the skewed nature of the data indicated by the previous quantile-quantile plot. When comparing the distributions of each class, the variates of non-fraudulent transactions are tightly centered around 0. This cannot be said for the variates of the fraudulent transactions. The fraudulent medians of V1 and V2 lie outside the interquartile ranges of their non-fraudulent counter parts, whereas V3, V4, V11, V12 and V14 have medians considered as outliers for non-fraudulent data. Additionally, the fraudulent variate distributions have a notably higher standard deviation than non-fraudulent variates, indicated by their bigger interquartile ranges. These observations indicate a significant different in the distribution of fraudulent and non-fraudulent transactions, and potentially show which variables

The standard deviations of each principal component were then calculated to observe the proportion of variance each component explains.

```
Importance of components:
                          Comp.1     Comp.2     Comp.3     Comp.4     Comp.5     Comp.6     Comp.7     Comp.8     Comp.9    Comp.10
Standard deviation     1.9586924 1.65130568 1.51625234 1.41586609 1.38024431 1.33226875 1.23709143 1.19435081 1.09863016 1.08884785
Proportion of Variance 0.1248376 0.08872945 0.07480934 0.06523148 0.06199045 0.05775592 0.04979852 0.04641696 0.03927497 0.03857867
Cumulative Proportion  0.1248376 0.21356702 0.28837636 0.35360784 0.41559829 0.47335421 0.52315273 0.56956969 0.60884466 0.64742333
                          Comp.11    Comp.12    Comp.13    Comp.14    Comp.15    Comp.16    Comp.17   Comp.18    Comp.19    Comp.20
Standard deviation     1.02071124 0.99919964 0.99527248 0.95859393 0.91531440 0.87625135 0.84933557 0.8381747 0.81403907 0.77092367
Proportion of Variance 0.03390148 0.03248758 0.03223271 0.02990076 0.02726173 0.02498448 0.02347316 0.0228603 0.02156271 0.01933907
Cumulative Proportion  0.68132481 0.71381239 0.74604510 0.77594586 0.80320759 0.82819207 0.85166522 0.8745255 0.89608823 0.91542731
                          Comp.21    Comp.22    Comp.23    Comp.24     Comp.25     Comp.26     Comp.27     Comp.28
Standard deviation     0.73452272 0.72570029 0.62445920 0.60564600 0.521277155 0.482226167 0.403631786 0.330082685
Proportion of Variance 0.01755591 0.01713671 0.01268881 0.01193577 0.008841994 0.007566837 0.005301314 0.003545344
Cumulative Proportion  0.93298321 0.95011992 0.96280874 0.97474451 0.983586505 0.991153342 0.996454656 1.000000000
```

Figure 3.3: Variance table for all principal components
Comp.1, Comp.2, Comp.3 correspond to V1, V2, V3, etc.

From looking at each components' proportion of variance, no component explains a significant proportion of the dataset's variance, with V1, the first principal component, only explaining 12.5% of the variance. As such, it would be unreasonable to choose the top 2 or 3 components and expect reasonable predictions. Using the rule of thumb of using only components with eigenvalues greater or equal to 1 (standard deviation >= 1), one may choose to use only the first 13 variables to sufficiently explain the variance in the dataset. The first 13 variables explain only 75% of the variability of the data, which may be low for others who may want at least 90% of the variability explained. These people would then have to select the top 20 principal components for prediction. It is impossible to tell which rule of thumb to use for feature selection just from glancing at these numbers and requires further testing to prove whether one rule of thumb works better than the other.

The distribution of the classes was heavily skewed towards non-fraudulent transactions, with only 0.017% of the data belonging to the minority class. To balance the number of fraudulent and non-fraudulent transactions, the SMOTE technique was applied to oversample the fraudulent transactions. After using the SMOTE function included in the *DWmR* R library, a resampled dataset with 1968 non-fraudulent transactions and 1476 fraudulent transactions was created. As the dataset started with 492 fraudulent transactions, this means the function created 2 synthetic samples for each fraudulent transaction. This proportion is much more sufficiently balanced for statistical analysis and allows for more meaningful observations of the data. To compare the statistical differences between the unbalanced and balanced dataset, a correlation heatmap was made using every single variable (except for time) from the unbalanced dataset

and the balanced dataset. The amount variable was also included to observe whether it would correlate with the class variable.

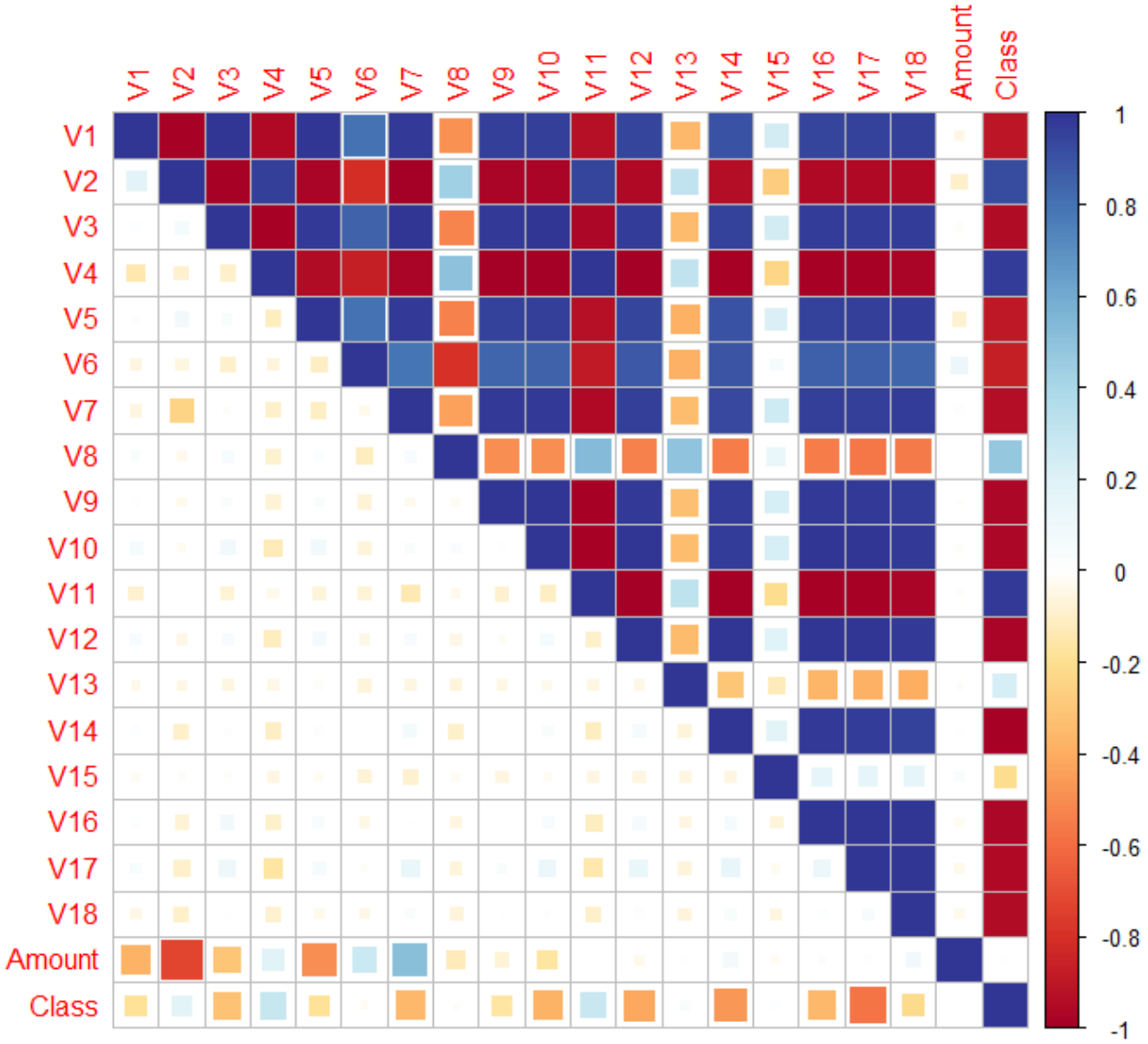


Figure 3.5: Correlation matrix of dataset before and after resampling
Lower half is correlation on unsampled data, upper half is correlation on resampled data
Time, V19-28 excluded for readability

From figure 3.5, several observations were made regarding the correlation of the explanatory variables and the response class in the resampled dataset.

- V2, V4, and V11 have the largest positive correlations with the Class variable. This means that more transactions with bigger values in the V2, V4 and V11 columns tended to be fraudulent than not

- Many variables between V1 and V18 (excluding V13 and V15) have a moderate to strong negative correlation with the Class variable. This means that more transactions with smaller values in these columns tended to be fraudulent than not

- 15 out of the 18 principal components have a strong correlation with the Class variable. This is likely not due to all 15 components being strong predictors for fraudulent transactions, but due to the strong correlations these variables have with each other in the resampled dataset

- Amount has almost no correlation to the Class variable

- Principal components V19 to V28 still had very weak correlations between the class variables and other principal components in both datasets. As such, they were not included in the matrix

There is a significant difference between the principal component correlations of the two datasets. Correlation is essentially non-existent in the original dataset, as principal components are calculated to be orthogonal to one another. However after more fraudulent transaction samples were produced, there are very strong correlations between each principal component shown above except for V8, V13 and V15 with moderate to little correlation. To investigate whether it was an issue with SMOTE, another correlation matrix was constructed with an under sampled dataset, meaning that only 492 non-fraudulent transactions were selected to match the 492 fraudulent transactions in the dataset. This produced a similar correlation matrix to the one above.

This can be explained by breaking down the covariance formula and using pair plots of some variates.

$$Cov_{x,y} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$

Figure 3.6: Covariance formula
X = variate 1, Y = variate 2
N = number of data points

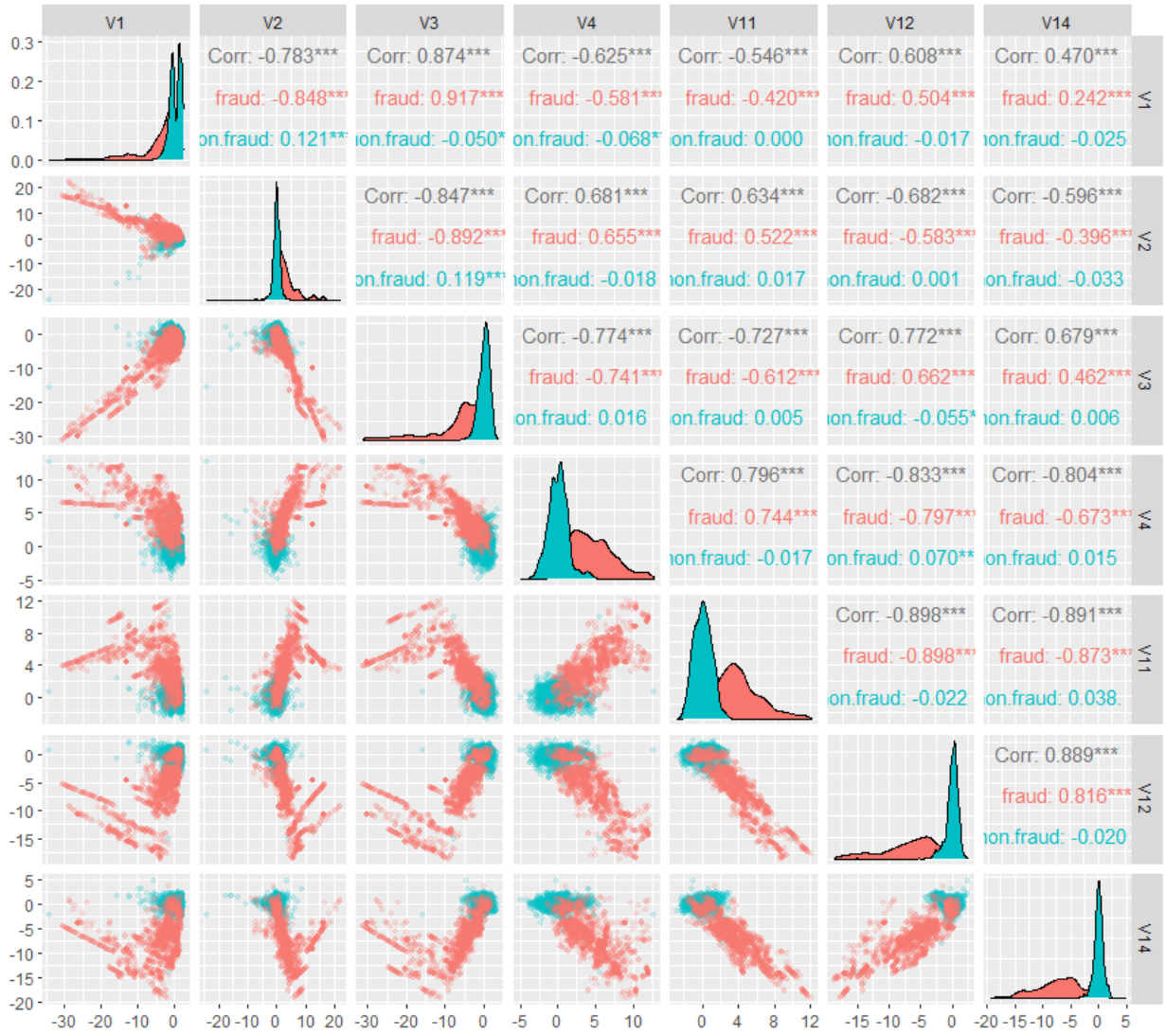


Figure 3.7: Pair plots of 7 variables

It can be observed from figure 3.7 that all variates of non-fraudulent transactions were tightly centered around 0, and with non-significant correlation coefficients. Fraudulent transactions on the other hand can be observed with highly correlated variates, such as the six variates in figure 3.7, and stronger linear trends when plotting 2 variables against each other. These trends did not show up in the original dataset due to the overwhelming amount of non-fraudulent transactions. Using the equation from figure 3.6, this meant that $\bar{x}$ and $\bar{y}$ would be 0, and $\Sigma(x_i - \bar{x})(y_i - \bar{y})$ also approaching 0 due to how close the other variates were to 0. It can be derived from these calculations that the covariance and correlation between any variable would always approach 0 in the original dataset. In contrast, the resampled dataset with clearer trends in the variates will produce more notable correlation coefficients.

From the observation of the figure 3.7, there are 2 different trends of data that correspond to the response classes. As mentioned above, non-fraudulent transactions are still tightly clustered around 0 for each principal component, while a moderate to strong linear trend can be observed

when plotting fraudulent transactions against many variates. However, between the pairs of covariates above, there is a large overlap between non-fraudulent data and a sizable portion of the fraudulent data. This means that if only two variables were used to separate fraudulent and non-fraudulent data, the probability of misclassification is high.

To deal with the data separation issue, clusters were constructed using all variates in the resampled dataset. In figure 3.8, 3 clusters constructed using the K-means method and plotted onto a two-dimension space generated by two new principal components after PCA was re-applied to all the principal components and the amount. 3 clusters were chosen, as any more clusters yielded diminishing decreases of the WSS. PCA also had to be applied to the resampled variables to plot the clusters on two new principal components that can explain more variability and possibly visualise the separation of classes better.

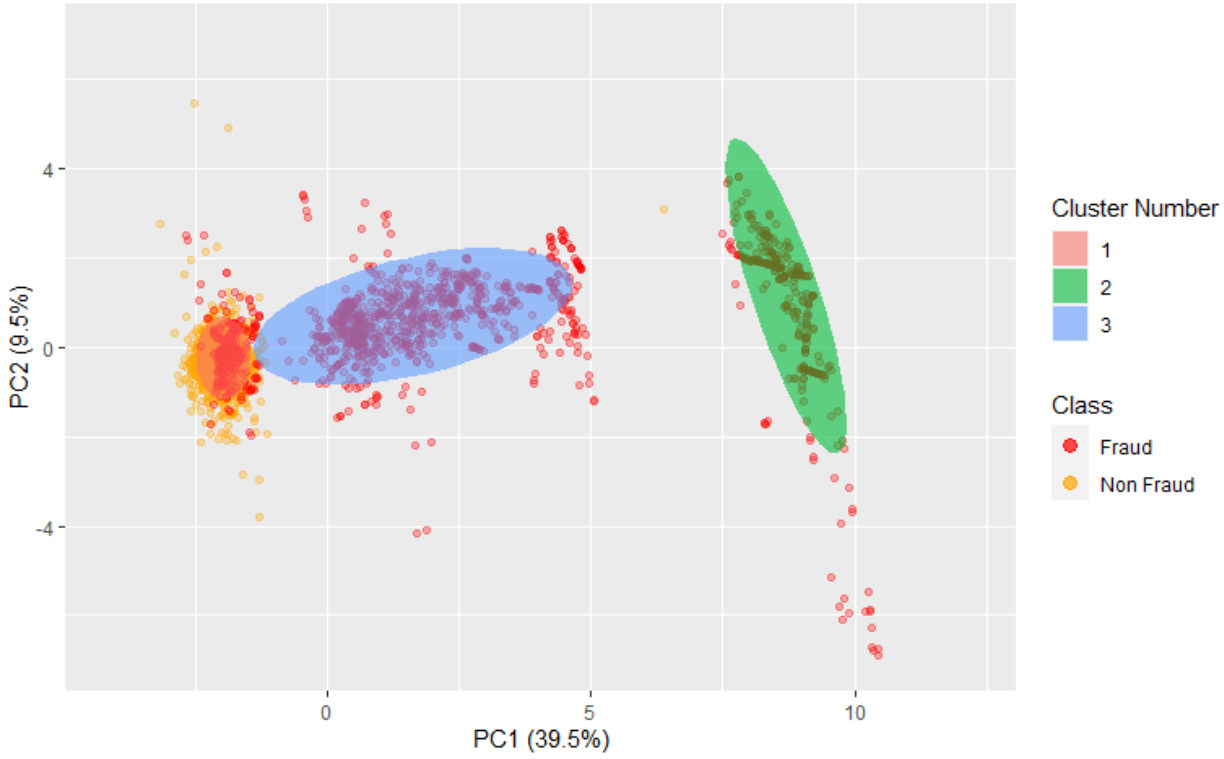


Figure 3.8: K-means clusters visualised

K-means clustering was able to separate two whole groups of exclusively fraudulent transactions into clusters 2 and 3. However, Cluster 1 was contained all the non-fraudulent transactions as well as a considerable group of fraudulent data. Despite plotting against new principal components which can explain more variability, the different classes of transactions were not much better separated than in the pair plots of figure 3.7. However, using an unsupervised machine learning algorithm, the t-SNE algorithm can produce a clean separation between fraudulent and non-fraudulent transactions. While there are still fraudulent transactions grouped with the non-fraudulent transactions, the majority of fraudulent datapoints are grouped far away from the non-fraudulent datapoints.

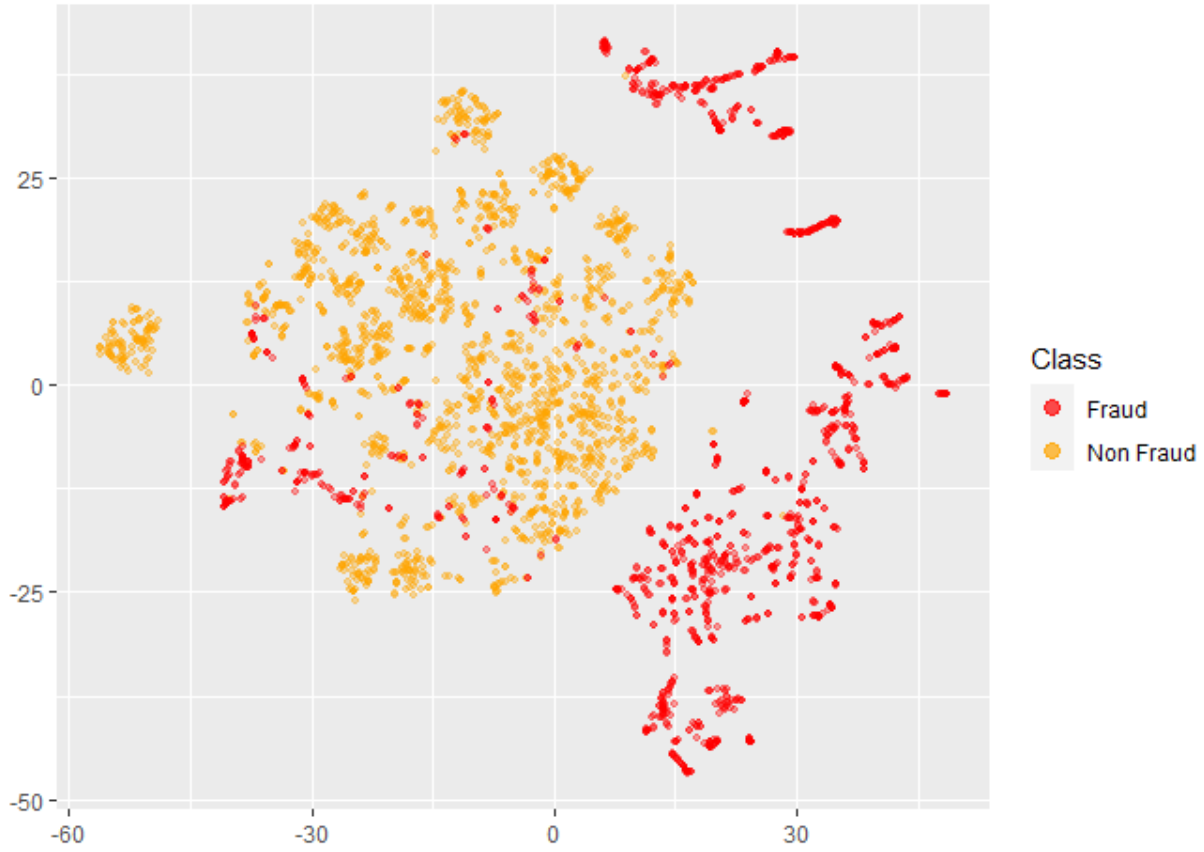


Figure 3.9: t-SNE clusters visualised

These two clustering results imply that linear techniques are not suited well for differentiating fraudulent and non-fraudulent transactions in this dataset and is done much better with machine learning algorithms that work well with non-linear data. This is a finding that stays true during classification.

During classification, it was decided that the recall and specificity rate would be the focus on the specificity rather than overall accuracy. Fraudulent transaction a very rare in the dataset, and as such is harder to detect than a non-fraudulent transaction. Hence detecting them correctly is of utmost important to mitigate any fraudulent transactions from being misclassified. Optimizing for recall rate helped with this issue, as it minimised the number of false negatives.

Training was done on the logistic models using a split of 90% training data, and 10% testing data. To optimize the parameters of the coefficient penalization equation, a 10-fold cross-validation process was used on the training data, each fold having 9 parts resampled with SMOTE, and 1 part as the testing data. After the optimal parameters had been found, a final model would be retrained on all the training data with the optimal parameters. It was found that using an *alpha* of 0.5 (Elastic net regularization) and a *lambda* of 0.0001 yielded Maximum recall. This essentially means that the model trained with very little coefficient penalization produced the largest recall. The coefficients to the model generated are as below.

V1	-0.386	V7	0.240	V13	0.306	V19	-0.091	V25	0.063
V2	-0.328	V8	0.578	V14	1.212	V20	1.005	V26	0.522
V3	-0.128	V9	0.277	V15	0.139	V21	0.254	V27	0.043
V4	-0.831	V10	0.663	V16	0.429	V22	-0.399	V28	-2.249
V5	-0.345	V11	-0.541	V17	0.393	V23	-0.121	Amount	-1.398
V6	0.243	V12	0.988	V18	0.376	V24	-0.162		

Figure 3.10: coefficients of the most optimal logistic regression model

While it is great that the logistic regression model has a good recall rate, the inclusion of all principal components, including components that did not have a high correlation with the class variable, does not give any indication of which variable is the most influential.

Next, two LightGBM models were constructed to predict the fraudulent behavior. The differences of each model are that the first model has been trained on the unbalanced dataset, while the second model was trained on the SMOTE balanced dataset. This has been done to assess whether resampling was required to improve the algorithm.

Much like for the logistic regression model, only the training data was resampled by SMOTE, whereas the validation and test data remained the same to simulate real world proportions. Since it is a binary classification problem, the binary log loss was used as the cost function. The results for all the models above are summarized below.

Approach	**Overall Accuracy**	**Recall**	**Specificity**	**Balanced Accuracy**
LightGBM 1	0.9996	1.0000	0.7674	0.8837
LightGBM 2	0.9894	0.9895	0.9302	0.9598
Logistic Regression	0.9715	0.9715	0.9293	0.9504

Figure 3.11: scores of each model created

We see that with LightGBM 1, it appears that the low number of fraudulent transactions resulted in the algorithm not fully learning its features. This can be observed through the low specificity. While it may be an acceptable number in another context, it is unacceptable for a bank to let 23% of fraudulent transactions go undetected.

However, using SMOTE drastically improved the specificity of the model. With balanced instances of both the classes, the algorithm was able to learn the features more comprehensively and improve the classification of the fraudulent transaction. The results for the SMOTE trained LightGBM model even surpassed the logistic regression model.

Another important output is the variable importance plots for the two approaches given below.

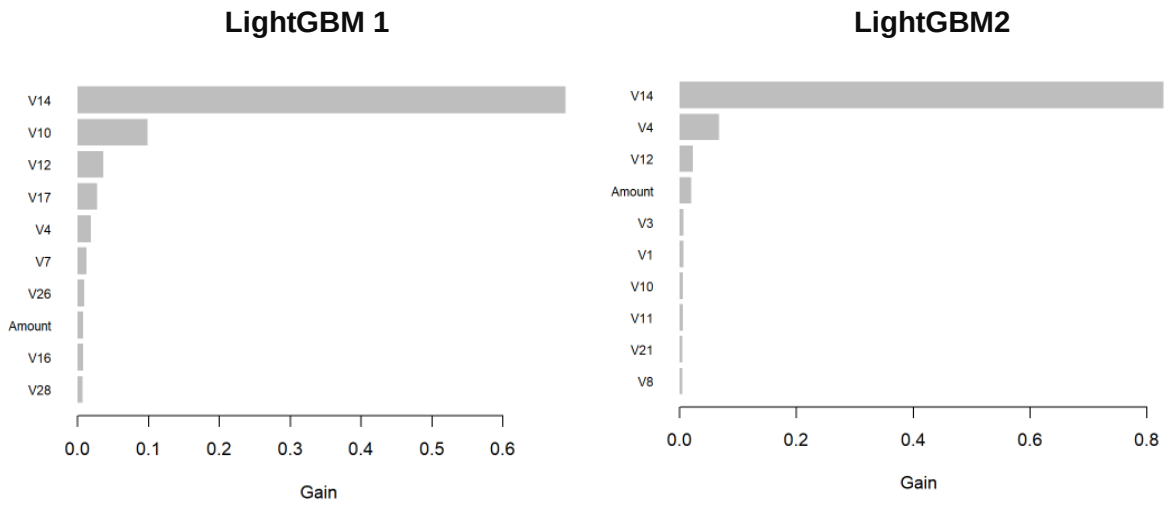


Figure 3.12: importance scores of each LightGBM model

The important variables in this context refer to the nodes used in splitting the dataset to reduce the entropy or uncertainty in the decision making. The gain variable refers to the improvement in accuracy when using a certain mode to split the dataset. Unlike results for logistic regression, is a clear answer for the most significant variable, which was V14, as it as the significantly highest information gain for both models. The other variables are mixed in order of importance.

5. Discussion (4 marks multivariate, 2 marks package)

When using linear techniques and statistical analysis, while there is an idea of which variables seem important, there is no concrete answer as to which variables contribute the most to a fraudulent transaction. This inconclusive evidence is most likely due to no principal components explaining a significant amount of variance in the data. Additionally, using the resampled dataset with multi-collinearity violated the no-multicollinearity assumption, and while this does not affect the performances of the models, it skews the importance of the correlated variables, making it even harder to judge which variables were important.

Using the comparison of boxplots in figure 3.2, it could be said that all 7 variables were the influential in detecting fraudulent detections due to how the means differed between the fraudulent and non-fraudulent transactions. It could also be said that the first 18 principal components with the exceptions of V8, V13 and V15 were highly influential on the response class. However, it was difficult to separate the two groups of transactions using only linear visualisations and calculations. Using the same variables to create pair plots in figure 3.7, while

there were pairs such as V12 and V14 where a notable difference could be seen between the two groups, there was always an overlap between the non-fraudulent transactions and fraudulent transactions. This continued during K-means clustering, with 1 cluster containing both non-fraudulent and fraudulent transactions, and logistic regression, where the best model misclassified a notable amount of non-fraudulent transactions as a tradeoff for the best recall of fraudulent transactions. When delving into machine learning algorithms, clusters were more defined using t-SNE clustering, and the best model boosted trees was able to perform slightly better than logistic regression, with a high recall and specificity. This indicates that the data of this table may not be suited to be divided in a linear space, and that machine learning algorithms such as t-SNE and LightGBM are better suited to learn the non-linear features of each transaction.

In the current problem, the cost and impact to business of misclassifying a fraudulent transaction as non - fraudulent is greater compared to misclassifying a non - fraudulent transaction as fraudulent. However in the real world, only a low percentage of transactions are fraudulent, and training models on an unbalanced dataset will lead to biased results. SMOTE turned out to be a great approach to balancing the data, improving the recall and specificity rates of both the logistic regression and the LightGBM model.

The LightGBM package is an excellent and easy package to train a LightGBM model. Provided a research team has a computer capable of running the algorithm over several iterations and a large dataset to avoid overfitting, LightGBM is a capable model that excels in classification, as proven in this dataset. Even without the usage of SMOTE, it was able to handle the extreme imbalance of data fairly well, with a surprising recall score of 1 and specificity of 0.77, which may be passable in another context unrelated to banking. Additionally, it was also able to calculate that V14 was the most important variable in the detection of fraudulent transactions, something that was only slightly noticeable during the boxplot comparison where the medians of both classes were significantly far apart, and not noticeable in any other linear analysis or visualisation. This paper was able to show that a machine learning algorithm like LightGBM can perform much better than a linear model like logistic regression.

There are some limitations to this paper, and clear step to be taken to further analyse the dataset. This paper ignored the no outlier assumption and the no multicollinearity assumption required of the logistic regression model. While the logit function deals with outlier well due to the sigmoid function tapering the outliers between 0 and 1, it is better for extreme outliers to be removed from the model for better performance. Additionally due to the multicollinearity assumption being violated by 15 of the first 18 principal components, it was not possible to gauge which variable was really important out of the correlated variables, as most had strong correlations to the class variable, and all were included in the final model. Should one want to continue this study, outlier analysis and removal should be considered, or a non-linear model should be used in place of logistic regression to determine variable importance. Additionally, while it would be hard to ask, it would be useful to be able to study the original variables of the dataset instead of the principal components, as it would allow researchers to translate findings of variable importance into practice with many different banks and improve security across all banks. For example, some components may have been comprised of transaction location, time of day, and frequency of attempts. Only being able to work with principal components was very limiting, as even though V14 was found to be a significant variable, only the company behind the dataset will know what V14 comprises of.

6. Conclusion:

Through the goal of finding the most important variables to identify fraudulent transactions, it was learnt that linear techniques were not best suited for such a task and did not differentiate fraudulent and non-fraudulent transactions as well as the machine learning algorithms used in this paper. With the help of multivariate analysis and visualisations, the skewed distribution of the data and the trickiness of class separation was made clear and required necessary machine learning models for the best results. One of the machine learning models is discovered to be LightGBM, which was able to perform very well with the SMOTE method, having over a 99% recall rate and 93% specificity rate, which is more important than the overall accuracy of classification in this paper's context. In addition, it was discovered that V14 was the most important variable by far across the two LightGBM models trained. It is unfortunate that the large amount of time taken to train the LightGBM model did not allow for much tuning to be done. It would be interesting to cooperate with the bank, as one could understand the impact of implementing the models trained in this paper and tweak the model to suit the bank's requirements and goals of preventing fraudulent transactions.

7. References

Basic walkthrough. (n.d.). Welcome to LightGBM's documentation! — LightGBM 3.3.2 documentation.
https://lightgbm.readthedocs.io/en/latest/R/articles/basic_walkthrough.html

Clustering in R programming. (2021, December 3). GeeksforGeeks.
https://www.geeksforgeeks.org/clustering-in-r-programming

How to perform logistic regression in R (step-by-step). (2021, September 29). Statology.
https://www.statology.org/logistic-regression-in-r/

Lee, M. (2022, October 19). *What is tSNE and when should I use it?* Sonrai Analytics.
https://sonraianalytics.com/what-is-tsne/

Likebupt. (n.d.). *Smote*. Microsoft Learn: Build skills that open doors in your career.
https://learn.microsoft.com/en-us/azure/machine-learning/component-reference/smote

8. Appendix

Harnoor

Library Needed

```
library(tidyverse)## Warning: package 'tidyverse' was built under R version 4.1.3## --
Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.1.4      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.2## Warning: package 'ggplot2' was built under R
version 4.1.3## Warning: package 'tibble' was built under R version 4.1.3## Warning:
package 'readr' was built under R version 4.1.3## Warning: package 'dplyr' was built
under R version 4.1.3## Warning: package 'forcats' was built under R version 4.1.3## --
Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()# library(corrplot)
library(grid)
library(gridExtra)##
## Attaching package: 'gridExtra'
##
```

```
## The following object is masked from 'package:dplyr':
##
##     combinelibrary(ggpubr)

library(dplyr)


#install.packages("lightgbm")
library(lightgbm)## Loading required package: R6
##
## Attaching package: 'lightgbm'
##
## The following object is masked from 'package:dplyr':
##
##     slice#library(pROC)
#install.packages("smotefamily")
library(smotefamily) ## Warning: package 'smotefamily' was built under R version
4.1.3#library(RColorBrewer)
#library(scales)
library(Matrix)##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack#install.packages('caret')
#install.packages("parallelly")
library(caret)## Warning: package 'caret' was built under R version 4.1.3## Loading
required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     liftlibrary(ROCR)## Warning: package 'ROCR' was built under R version 4.1.2
```

Importing the data

```
cc_data <- read.csv("creditcard.csv", header = T, stringsAsFactors = F)
head(cc_data)##   Time         V1          V2        V3         V4          V5
V6
## 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##            V7          V8         V9         V10        V11         V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531
```

```
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##          V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
##           V19         V20          V21          V22         V23         V24
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658
##          V25        V26          V27         V28 Amount Class
## 1  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
## 5 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
## 6 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0
```

Dividing the data into train(0.8), valid(0.1), test(0.1)

```
# Define the partition (e.g. 75% of the data for training)
trainIndex <- createDataPartition(cc_data$Class, p = .80,
                                  list = FALSE,
                                  times = 1)

# Split the dataset using the defined partition
train_df_o <- cc_data[trainIndex,-1 ,drop=FALSE]
test_plus_val_data <- cc_data[-trainIndex, ,drop=FALSE]

# Define a new partition to split the remaining 25%
test_plus_val_index <- createDataPartition(test_plus_val_data$Class,
                                           p = .5,
                                           list = FALSE,
                                           times = 1)

# Split the remaining ~20% of the data: 50% (tune) and 50% (val)
valid_df_o <- test_plus_val_data[-test_plus_val_index,-1 ,drop=FALSE]
test_df_o <- test_plus_val_data[test_plus_val_index, -1,drop=FALSE]
```

Creating matrix for running light gbm

```
train_df <- train_df_o
valid_df <- valid_df_o
test_df <- test_df_o

trainm = sparse.model.matrix(Class ~., data = train_df)
train_label = train_df[,"Class"]

validm = sparse.model.matrix(Class ~., data = valid_df)
valid_label = valid_df[,"Class"]

testm = sparse.model.matrix(Class~., data= test_df)
test_label = test_df[,"Class"]

train_matrix = lgb.Dataset(data = as.matrix(trainm), label = train_label)
valid_matrix = lgb.Dataset(data = as.matrix(validm), label = valid_label)
test_matrix = lgb.Dataset(data = as.matrix(testm), label = test_label)
```

LightGBM paramaters

```
valid = list(test = valid_matrix)

# model parameters
params = list( learning_rate = 0.001,
               objective = "binary",
               metric = 'binary_logloss')

#model training
bst = lightgbm(params = params, train_matrix, valid, nrounds = 1000)## [LightGBM] [Info]
Number of positive: 402, number of negative: 227444
## [LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing
was 0.027199 seconds.
## You can set `force_col_wise=true` to remove the overhead.
## [LightGBM] [Info] Total Bins 7395
## [LightGBM] [Info] Number of data points in the train set: 227846, number of used
features: 29
## [LightGBM] [Info] [binary:BoostFromScore]: pavg=0.001764 -> initscore=-6.338207
## [LightGBM] [Info] Start training from score -6.338207
```

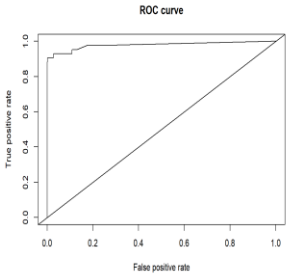
Confusion matrix

```
p = predict(bst, testm)
test_df$predicted = ifelse(p > 0.5,1,0)
confusionMatrix(factor(test_df$predicted), factor(test_df$Class))## Confusion Matrix and
Statistics
##
```

```
##           Reference
## Prediction     0     1
##          0 28438    10
##          1     0    33
##
##                Accuracy : 0.9996
##                  95% CI : (0.9994, 0.9998)
##     No Information Rate : 0.9985
##     P-Value [Acc > NIR] : 1.596e-09
##
##                   Kappa : 0.8682
##
##  Mcnemar's Test P-Value : 0.004427
##
##             Sensitivity : 1.0000
##             Specificity : 0.7674
##          Pos Pred Value : 0.9996
##          Neg Pred Value : 1.0000
##              Prevalence : 0.9985
##          Detection Rate : 0.9985
##    Detection Prevalence : 0.9988
##       Balanced Accuracy : 0.8837
##
##        'Positive' Class : 0
##
```

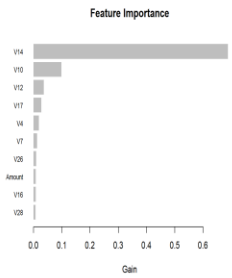
AUC

```
pred=prediction(p,test_df$Class)
roc=performance(pred,"tpr","fpr")
plot(roc,main="ROC curve")
abline(a=0,b=1)
```



Important variable

```
tree_imp <- lgb.importance(bst, percentage = TRUE)lgb.plot.importance(
  tree_imp,
  top_n = 10L,
  measure = "Gain",
  left_margin = 10L,
  cex = NULL
)
```



Using SMOTE

```
train_df1 <- train_df_o
valid_df1 <- valid_df_o
test_df1 <- test_df_o


prop.table(table(train_df1$Class))##
##          0          1
## 0.99823565 0.00176435
```

Creating synthetic data using only training data set.

```
smote_df <- SMOTE(train_df_o,train_df1$Class,K = 5)
train_df_s <- smote_df$data[,-30]

prop.table(table(train_df_s$Class))## numeric(0)dim(train_df_s)## [1] 454574
30colnames(train_df_s)##  [1] "V1"     "V2"     "V3"     "V4"     "V5"     "V6"     "V7"
"V8"
##  [9] "V9"     "V10"    "V11"    "V12"    "V13"    "V14"    "V15"    "V16"
## [17] "V17"    "V18"    "V19"    "V20"    "V21"    "V22"    "V23"    "V24"
## [25] "V25"    "V26"    "V27"    "V28"    "Amount" "class"head(smote_df$data)##
V1         V2         V3         V4         V5         V6         V7
## 1 -13.89720587  6.344280 -14.281666  5.5810090 -12.887133 -3.1461764 -15.45047
## 2  -6.49808617  4.750515  -8.966558  7.0988543  -6.958376 -2.8221262 -10.33341
## 3  -0.08298347 -3.935919  -2.616709  0.1633101  -1.400952 -0.8094194   1.50158
## 4  -4.86174666 -2.722660  -4.656248  2.5020047  -2.008346  0.6154218  -3.48568
## 5 -15.19206401 10.432528 -19.629515  8.0460750 -12.838167 -1.8758587 -21.35974
## 6  -5.26805322  9.067613 -15.960728 10.2966028  -4.708241 -3.3953749 -11.16106
##           V8         V9         V10         V11         V12         V13         V14
## 1  9.0602815 -5.486121 -14.676470  5.1483517 -10.7495920  1.5854890  -9.7980121
## 2  4.0319073 -6.648778 -11.634414  6.8775709 -13.6976856  0.4630404 -13.0441824
## 3 -0.4709996  1.519743  -1.134454  0.7138776   0.9796748 -1.3390313   0.9849932
## 4  1.8788555 -1.116268  -5.112971  2.9456209  -4.1021274 -0.4772030  -4.3504345
## 5 -3.7178495 -5.969782 -17.141514  5.9023998 -13.5801473 -0.4514069  -8.3347626
## 6  5.4999626 -5.667376 -11.627194 11.0270591 -16.3880542  0.3639215 -17.2302022
##             V15         V16          V17         V18         V19         V20
## 1  0.7332769991 -12.6752689 -20.74066372 -8.1536680 5.2283418 -1.0252279
```

```
## 2 -0.3092292657 -12.3175797 -24.01909855 -9.3351931 1.9518905  0.5683380
## 3 -0.3824709683  -0.9346158  -0.05241943  0.5111618 0.7319987  1.8786116
## 4  0.0007163435  -5.0374494  -8.07801050 -3.4579812 0.5966964  0.2855588
## 5 -2.0251446865 -10.1963337 -17.27098517 -7.0790958 1.5176953  1.6574762
## 6 -0.4374875592 -10.1223919 -13.63920899 -4.9864573 1.1267844  1.4558782
##         V21        V22        V23        V24         V25         V26        V27
## 1  3.058082  0.9411803 -0.2327103  0.7635080  0.07545624 -0.45383957 -1.5089683
## 2  2.158143  0.1115104  0.2164138  0.5846613  0.76035951  0.08197234  1.4150675
## 3  0.702672 -0.1823048 -0.9210170  0.1116355 -0.07162221 -1.12588110 -0.1709474
## 4  1.138876  1.0336643 -0.8061989 -1.5110464 -0.19173140  0.08099883  1.2151516
## 5 -3.474097  1.7654455  1.7012568  0.3815868 -1.41341749 -1.02307789 -2.6347613
## 6  2.004110  0.1910584  0.6229277 -1.2092636 -0.37479886  0.64879804  1.5846973
##           V28  Amount Class class
## 1 -0.68683574    9.99     1     1
## 2  0.03512414   83.38     1     1
## 3  0.12622138 1096.99     1     1
## 4 -0.92314161  592.90     1     1
## 5 -0.46393055    1.00     1     1
## 6  0.72005587    1.00     1     1
```

Train dataset is created using SMOTE, Validation and Test data are retain from the original data partition

```
valid_df_s <- valid_df_o
colnames(valid_df_s)[30]= "class"

test_df_s <- test_df_o
colnames(test_df_s)[30]= "class"trainm_s = sparse.model.matrix(class ~., data =
train_df_s)
train_label_s = train_df_s[,"class"]

validm_s = sparse.model.matrix(class ~., data = valid_df_s)
valid_label_s = valid_df_s[,"class"]


testm_s = sparse.model.matrix(class~., data= test_df_s)
test_label_s = test_df_s[,"class"]

train_matrix_s = lgb.Dataset(data = as.matrix(trainm_s), label = train_label_s)
valid_matrix_s = lgb.Dataset(data = as.matrix(validm_s), label = valid_label_s)
test_matrix_s = lgb.Dataset(data = as.matrix(testm_s), label = test_label_s)
```

training the model with SMOTE data

```
valid = list(test = valid_matrix_s)

# model parameters
params = list( learning_rate = 0.001,
```

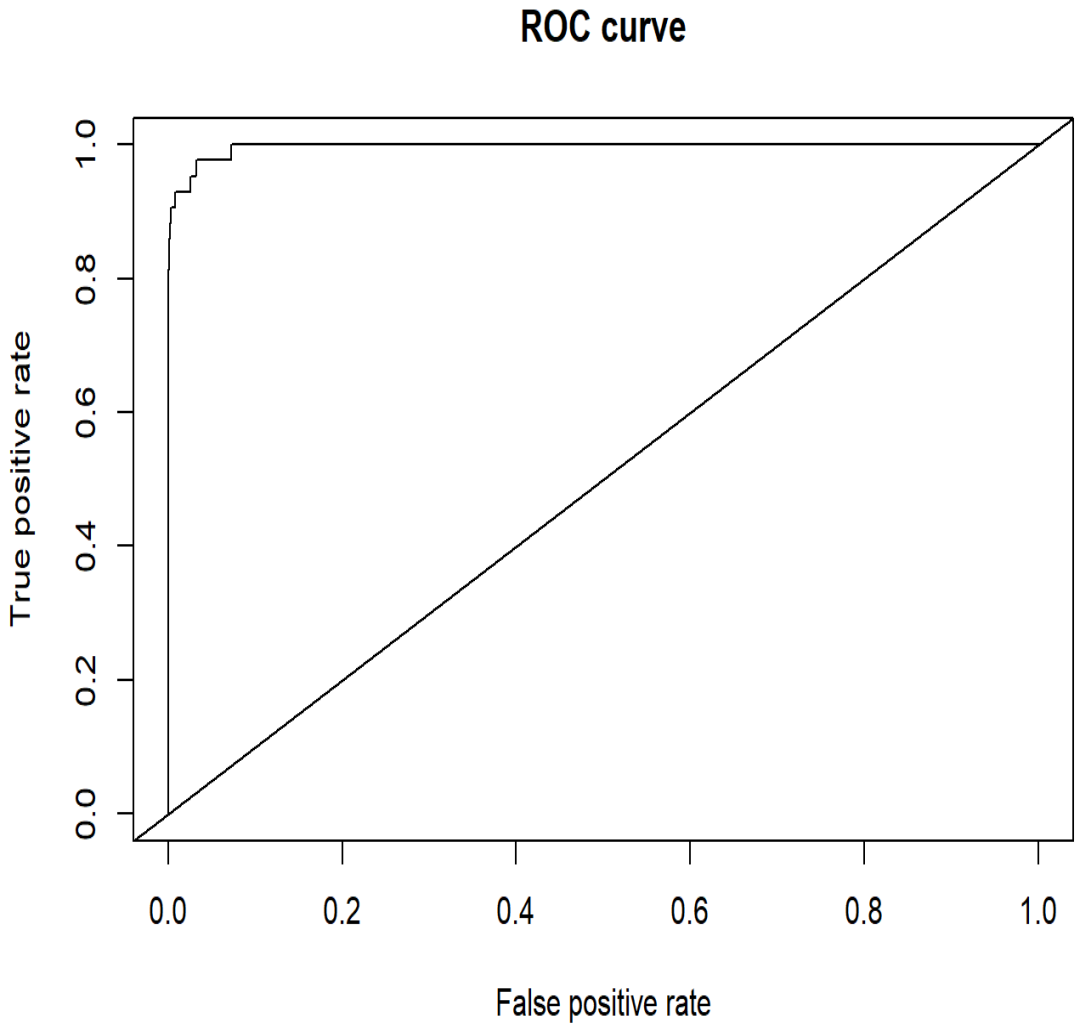
```
              objective = "binary",
              metric = 'binary_logloss')

#model training
bst1 = lightgbm(params = params, train_matrix_s, valid, nrounds = 1000)## [LightGBM]
[Info] Number of positive: 227130, number of negative: 227444
## [LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.011215 seconds.
## You can set `force_row_wise=true` to remove the overhead.
## And if memory is not enough, you can set `force_col_wise=true`.
## [LightGBM] [Info] Total Bins 7395
## [LightGBM] [Info] Number of data points in the train set: 454574, number of used
features: 29
## [LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499655 -> initscore=-0.001382
## [LightGBM] [Info] Start training from score -0.001382
```

Confusion matrix

```
p = predict(bst1, testm_s)
test_df_s$predicted = ifelse(p > 0.5,1,0)
confusionMatrix(factor(test_df_s$predicted), factor(test_df_s$class))## Confusion Matrix
and Statistics
##
##           Reference
## Prediction     0     1
##          0 28138     3
##          1   300    40
##
##                Accuracy : 0.9894
##                  95% CI : (0.9881, 0.9905)
##     No Information Rate : 0.9985
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2068
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9895
##             Specificity : 0.9302
##          Pos Pred Value : 0.9999
##          Neg Pred Value : 0.1176
##              Prevalence : 0.9985
##          Detection Rate : 0.9880
##    Detection Prevalence : 0.9881
##       Balanced Accuracy : 0.9598
##
##        'Positive' Class : 0
##
```

AUC

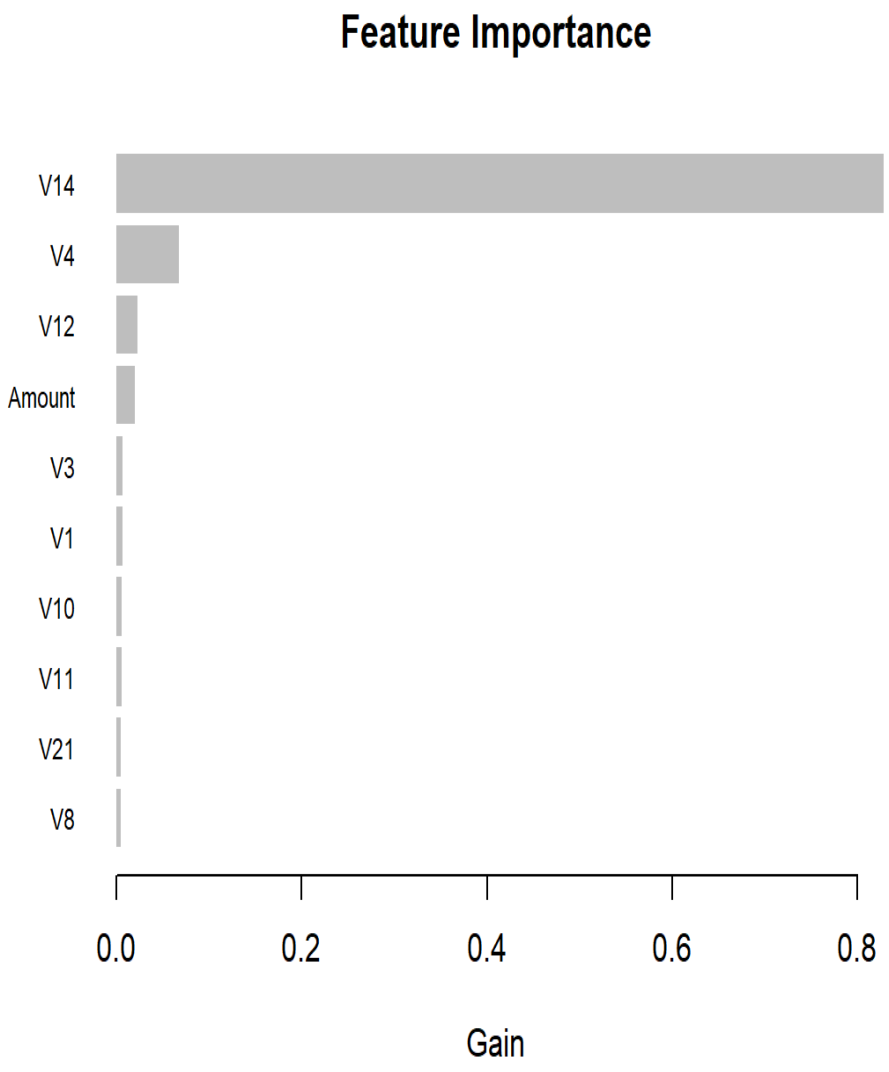
```
pred=prediction(p,test_df_s$class)
roc=performance(pred,"tpr","fpr")
plot(roc,main="ROC curve")
abline(a=0,b=1)
```



####

Important variable

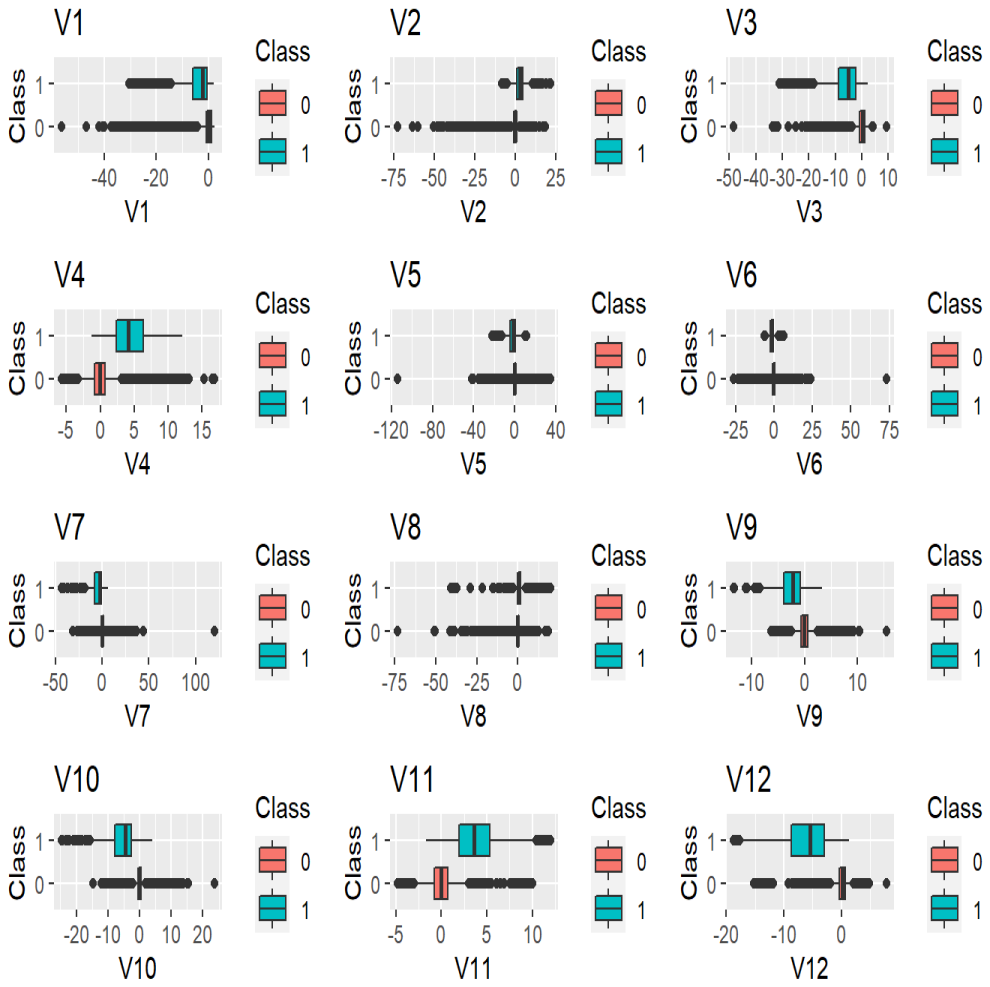
```
tree_imp1 <- lgb.importance(bst1, percentage = TRUE)lgb.plot.importance(
  tree_imp1,
  top_n = 10L,
  measure = "Gain",
  left_margin = 10L,
  cex = NULL
)
```



Boxplot updated

```
cc_data$Class<- as.factor(cc_data$Class)

desc_plot <- function(x){
  df_sub = cc_data[,c(x, "Class", "Amount")]
  plt2 <- ggplot(df_sub, aes(x = eval(parse(text = x)), y = Class, fill = Class) )+
geom_boxplot() + xlab(x) + labs(title =x)
  return(plt2)
}


var_names<- colnames(cc_data)
output_plot <- list()
```

```
for(i in 1:21){
  output_plot[[i]]<- desc_plot(var_names[i])
}

print(ggarrange(output_plot[[2]],
                output_plot[[3]],
                output_plot[[4]],
                output_plot[[5]],
                output_plot[[6]],
                output_plot[[7]],
                output_plot[[8]],
                output_plot[[9]],
                output_plot[[10]],
                output_plot[[11]],
                output_plot[[12]],
                output_plot[[13]],ncol = 3,nrow = 4))
```



KARMEL

````
---
title: "CC"
author: "Karmel Noor"
date: "11/09/2022"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r}
library('caret')
library('caTools')
library('pROC')
library('mclust')
library('tidyverse')
library('DMwR')
library('GGally')
library('glmnet')
library('factoextra')
library('Rtsne')
library('DescTools')
library("PerformanceAnalytics")
library('MLmetrics')
library('MVN')
library('reshape')
library('lmtest')
library('corrplot')
library('lightgbm')
```


```{r}
cc <- read.csv('creditcard.csv')
head(cc,100)
```
```{r}
pca <- princomp(cc[,2:29])
summary(pca)
pca$loadings
#we can see pca is already in order

```
````

```{r, fig.height=6,fig.width=8}
royston <- mvn(data=cc[sample(nrow(cc), 2000),2:15],  mvnTest =c("royston"), univariateTest = "AD", univariatePlot = "histogram", multivariatePlot = "qq", multivariateOutlierMethod = "adj", showOutliers = TRUE, showNewData = TRUE)
royston$multivariateNormality

hz <-
  mvn(data=cc[sample(nrow(cc), 2000),2:15],  mvnTest =c("hz"), univariateTest = "AD", univariatePlot = "histogram", multivariatePlot = "qq", multivariateOutlierMethod = "adj", showOutliers = TRUE, showNewData = TRUE)
hz$multivariateNormality

mardia <- mvn(data=cc[sample(nrow(cc), 2000),2:15],  mvnTest =c("hz"), univariateTest = "AD", univariatePlot = "histogram", multivariatePlot = "qq", multivariateOutlierMethod = "adj", showOutliers = TRUE, showNewData = TRUE)
mardia$multivariateNormality
```

```{r}
set.seed(1234)
cc$Class <- factor(ifelse(cc$Class==0,'non.fraud','fraud'))
melt_data <- melt(cc[,c('V1','V2','V3','V4','V11','V12','V14','Class')], id = c("Class"))

melt_data
```

```{r, fig.height=7}
cols <- c("fraud" = "red", "non.fraud" = "orange")
col_guide <- guide_legend(override.aes=c(alpha=0.5))
melt_data %>% ggplot(aes(x=variable,y=value,fill=Class)) + geom_boxplot(alpha=0.5) + theme_gray() + labs(x='Value',y='Variable')  + scale_fill_manual(values=cols,labels=c('Fraud','Non Fraud')) +
  guides(fill=col_guide) +
  ylim(-20, 20)
```

```{r}
head(cc)
set.seed(1234)
# sample_idx <- createDataPartition(cc$Class, p = .5,
#                                   list = FALSE,
#                                   times = 1)
# sample_data <- cc[sample_idx,]
# table(sample_data$Class)
# USING SMOTE
# levels(cc$Class) <- c('fraud','non.fraud')
cc$Amount <- (cc$Amount - mean(cc$Amount))/sd(cc$Amount)
```

```
sample_data  <- SMOTE(Class ~ ., data=cc[,-1])


# sample_data <- downSample(x = cc[, -ncol(cc)],
#                           y = cc$Class)
table(sample_data$Class)
sample_data
```

```{r}
nearZeroVar(sample_data[,1:30])
```


```{r}
downsample_data <- downSample(x = cc[, -1],
                         y = cc$Class)
table(downsample_data$Class)
```


```{r,fig.width=7,fig.height=7}
set.seed(1234)


#takes too long to do ggpairs with all, so sample 10,000 rows
# chart.Correlation(sample_data[,-c(1,30)], histogram=TRUE, pch=19)
temp_data <- sample_data
temp_data$Class <- ifelse(temp_data$Class=='non.fraud',0,1)
temp_data
samp_cor <- cor(temp_data[,-c(19:28)])
upper <- cor(samp_cor)

sample_idx2 <- createDataPartition(cc$Class, p = .1,
                                   list = FALSE,
                                   times = 1)
sample_data2 <- cc[sample_idx2,-c(1,20:29)]
sample_data2$Class <- ifelse(sample_data2$Class=='non.fraud',0,1)
samp_cor <- cor(sample_data2)
lower <- cor(samp_cor)

upper[lower.tri(upper)] <- lower[lower.tri(lower)]
corrplot(upper,method='color',col = COL2('RdYlBu'))
```


```{r, fig.height = 8, fig.width = 9}
```

```
ggpairs(sample_data, columns=c('V1','V2','V3','V4','V11','V12','V14'), aes(color=Class),lower =
list(continuous = wrap("points", alpha = 0.15, size=1))) + theme_gray()
```

hotelling

```{r}
mean_of_no_fraud <- colMeans(sample_data[sample_data$Class == 'non.fraud',-30])
mean_of_no_fraud
t2test <- HotellingsT2Test(sample_data[sample_data$Class == 'fraud',-
30],sample_data[sample_data$Class == 'non.fraud',-30])
#need to check if multivar normal (mvntest), independent (assume), random
#fraud and non.fraud are from same population
t2test
t2test$statistic[1]
crit_t2 <- function(p,n,alpha)  ##
{
  critval= sqrt((n-1)*p/(n-p)*qf(alpha,p,n-p, lower.tail = FALSE))
  return(critval)
}
crit_t2(2,dim(sample_data[sample_data$Class == 'non.fraud',-30])[1],0.05)
```

```{r}

fviz_nbclust(sample_data[,1:29], kmeans, method = "wss")
```

```{r,fig.width=8}

k <- kmeans(sample_data[,1:29], centers = 3, nstart = 50)
pca_for_cluster <- prcomp(sample_data[,1:29], center=TRUE, scale = TRUE)

ggdata <- data.frame(pca_for_cluster$x)
ggdata$Cluster <- k$cluster
ggdata$Class <- sample_data$Class
# sample_of_sample <- createDataPartition(ggdata$Class, p = .2,
#                                         list = FALSE,
#                                         times = 1)
# fviz_cluster(k, data = sample_data[,1:29],geom = NULL, ellipse.type='norm')+
#   geom_point(aes(color = sample_data$Class), alpha = 0.1) +
#
#   theme_grey()+
#   scale_fill_discrete(labels=c("a","b","c",'d','e'))
col_guide <- guide_legend(override.aes = c(alpha=0.7,size=3))
ggdata %>% ggplot() +
  geom_point(aes(x = PC1, y = PC2, color = factor(Class)), size = 1.5,alpha=0.3) +
  stat_ellipse(aes(x = PC1, y = PC2, fill = factor(Cluster)),


```

```
            geom = "polygon", level = 0.95, alpha = 0.6) +
  labs(fill = "Cluster Number",
       color = "Class",
       x='PC1 (39.5%)',
       y='PC2 (9.5%)')+
  scale_color_manual(values=cols,labels=c('Fraud','Non Fraud'))+
  theme_gray() +
  guides(color=col_guide)+
  ylim(-7,7)
```
```{r}
s <- summary(pca_for_cluster)
s$importance
```

```{r}
tsne_model = Rtsne(sample_data[,1:29], check_duplicates=FALSE, pca=TRUE, perplexity=35, theta=0.5,
dims=2)
data_tsne = as.data.frame(tsne_model$Y)

```
```{r,fig.width = 7,fig.height=5}
data_tsne %>% ggplot(aes(x=V1, y=V2, color=factor(sample_data$Class))) +
geom_point(size=1, alpha=0.35) +
xlab("") + ylab("") +
theme(axis.text.x=element_blank(),
      axis.text.y=element_blank(),
      legend.position = "right") +
  scale_color_manual(values=cols,labels=c('Fraud','Non Fraud'),name='Class') +
  theme_gray() +
  guides(color=col_guide)
```

#getting ready for training:

```{r}
set.seed(1234)
#split data into training and testing
testing_idx <- createDataPartition(cc$Class, p = .1,
                                   list = FALSE,
                                   times = 1)
test_data <- cc[testing_idx,-1]
table(test_data$Class)
train_data <- cc[-testing_idx,-1]
table(train_data$Class)
```

```{r}
folds <- createFolds(train_data$Class, k = 10, list = FALSE,returnTrain=TRUE)
ctrl <- trainControl(method = "cv",
                     classProbs = TRUE,
                     summaryFunction = prSummary,
                     sampling = 'smote',
                      savePredictions = TRUE,
                     search = 'grid')
log_grid <- expand.grid(
  alpha =c(0,0.5,1),
  lambda = seq(0, 0.001, length = 21)
)
bayes_grid <- expand.grid(
  adjust =c(0,0.5,1),
  fL = seq(0.0001, 0.2, length = 20),
  usekernel=c(TRUE,FALSE)
)

```



```{r}
set.seed(1234)
glmnet.train.log <- train(Class ~ .,
                     data=train_data,
                       method = 'glmnet',
                      trControl = ctrl,
                      family = 'binomial',
                     metric='Recall',
                     tuneGrid=log_grid,
                    model = FALSE,
                    returnData = FALSE)
glmnet.train.log
plot(glmnet.train.log)
```



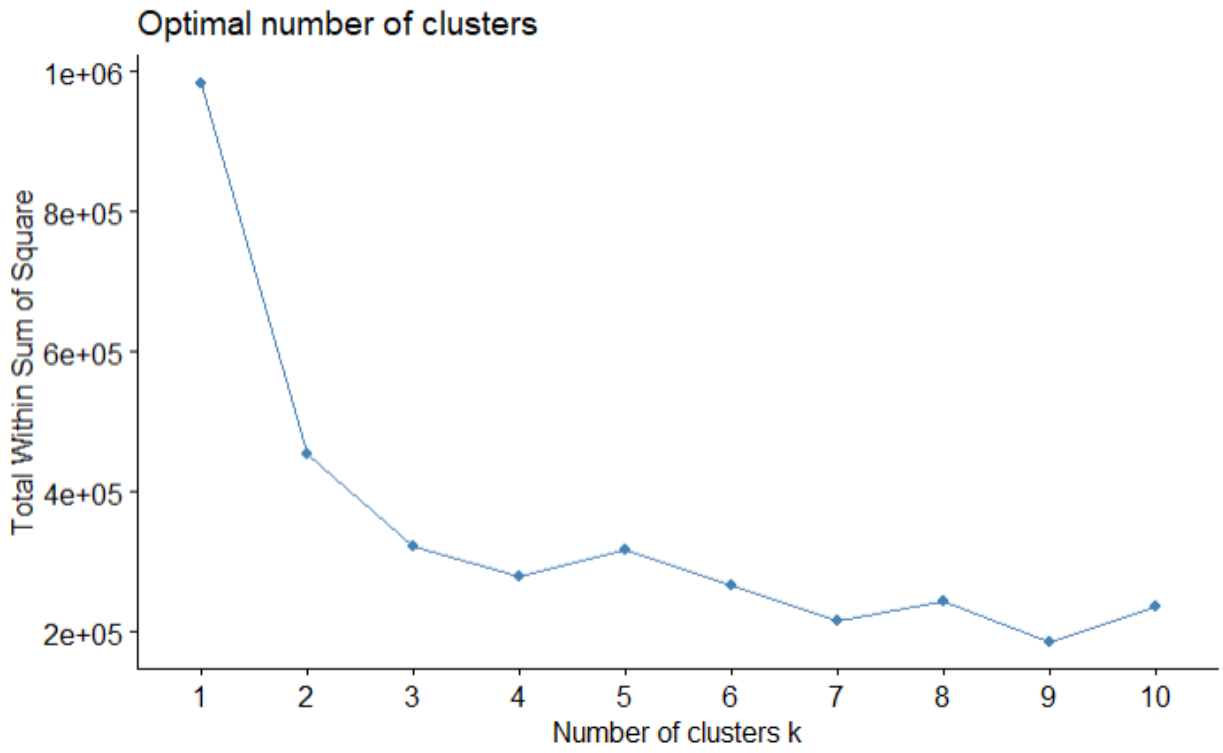
```{r}
smote_train <- SMOTE(Class ~ ., data=train_data)
levels(smote_train[,30]) <- c('non.fraud','fraud')
levels(test_data[,30]) <- c('non.fraud','fraud')
final_log_model <- glmnet(x=smote_train[,-30],y=smote_train[,30], family='binomial', lambda=0.0001,
                          alpha=0.5,data=smote_train)

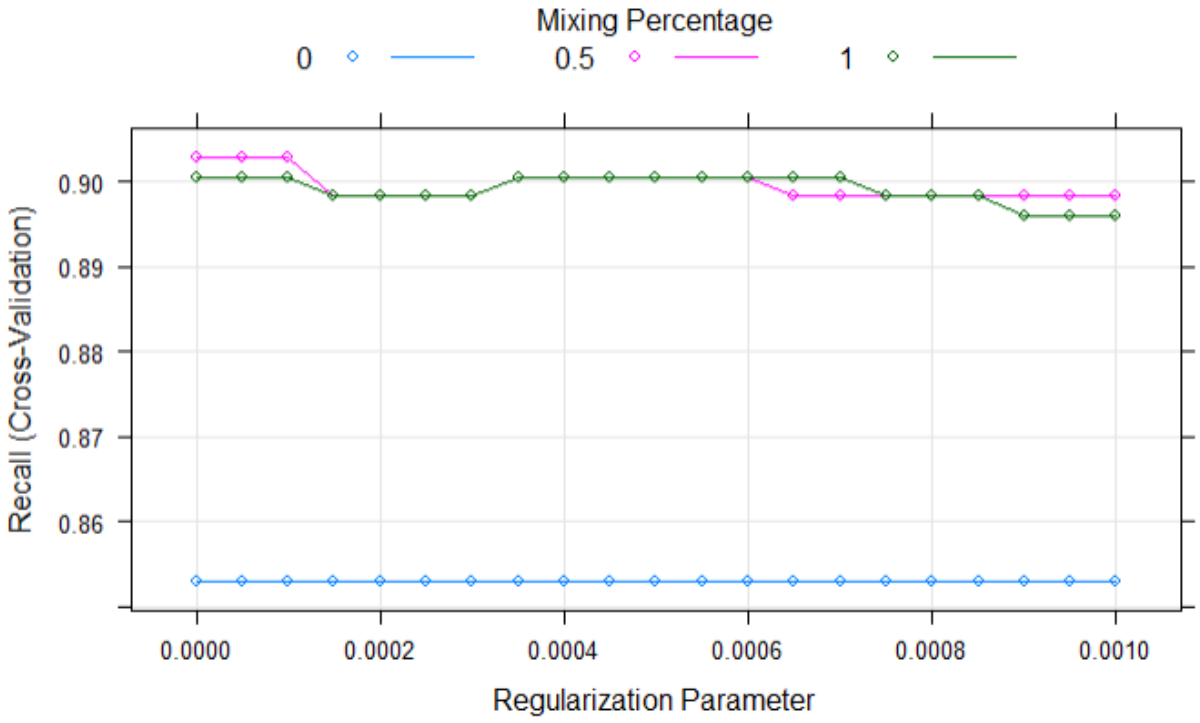
results <- predict(final_log_model, newx=as.matrix(test_data[,-30]))
results_factor <- ifelse(results > 0.50, "fraud", "non.fraud")
```

confusionMatrix(factor(results_factor[,1]),test_data$Class,positive='fraud',mode="sens_spec")
```

```{r}
final_log_model$beta
auc <- colAUC(results,  test_data$Class, plotROC = FALSE)
auc
```

EXTRA PLOTS


```


Mixing Percentage
0
0.5
1
Recall (Cross-Validation)
0.90
0.89
0.88
0.87
0.86
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
Regularization Parameter