A Project report
On
Credit Card Fraud Detection

Submitted in partial fulfillment for the award of the degree

Bachelor of Technology

By

PRATIK CHAUHAN (16CS001969)

S.V.VENU MADHAV (16CS0019680)

Submitted to

Department of Computer Science & Engineering
Sir Padampat Singhania University
Udaipur 313601 Rajasthan India

Under the supervision of
Prof. Arun Kumar
Department of Computer Science & Engineering
Sir Padampat Singhania University
Udaipur 313601 Rajasthan India

DECLARATION

We, Pratik Chauhan (16CS001969) and S.V. Venu Madhav (16CS001969), students of B.Tech.(CSE), hereby declare that the project titled ***"Credit Card Fraud Detection",*** which is submitted by us to the department of Computer Science & Engineering , School of Engineering, Sir Padampat Singhania University, Udaipur, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

Pratik Chauhan
(16CS001969)

S.V. Venu Madhav
(16CS001980)

Udaipur

Date: 15-May-2020

CERTIFICATE

This is to certify that the project entitled “Credit Card Fraud Detection” being submitted by Pratik Chauhan (16CS001969) and S.V. Venu Madhav (16CS001980), in partial fulfillment of the requirement for the award of Bachelor of Technology, has been carried out under my supervision and guidance.

The matter embodied in this report has not been submitted, in part or in full, to any other university or institute for the award of any degree, diploma or certificate.

Prof. Arun Kumar
Dean, School of Engineering,
Department of Computer Science & Engineering
Sir Padampat Singhania University
Udaipur 313601 Rajasthan India

Prof. Mukesh Kalla
Head of Department,
Department of Computer Science & Engineering
Sir Padampat Singhania University
Udaipur 313601 Rajasthan India

ACKNOWLEDGEMENT

We would like to express my sincere gratitude to my project guide Prof. Arun Kumar for giving me the opportunity to work on this topic.

It would never be possible for us to take this project to this level without his innovative ideas and his relentless support and encouragement.

Pratik Chauhan (16CS001969)
S.V. Venu Madhav (16CS001980)

ABSTRACT

Due to the rise and rapid growth of E-Commerce, use of credit cards for online purchases has dramatically increased and it caused an explosion in the credit card fraud. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of fraud associated with it are also rising.

In this project predicting the fraudulent transactions based on the given data is done using programming language R. By analyzing and Visualizing the Credit card frauds, banks can implement a way to stop the fraud. R is a statistical programming language and environment for data analysis and visualization. It is open source and supported by various libraries and packages.
In our minor project work, K-Nearest Neighbor supervised learning algorithms was used to solve this typical classification problem.

In this major project, the project is continued by applying Support Vector Machine and Random Forest Classification Algorithms. Along with it, the feature extraction processes like greedy search algorithm and random forest feature search process will be used for acquiring the best accuracy results from a set of selected features.

CONTENTS

LIST OF FIGURES

LIST OF ABBRIBIATIONS

ABBREVIATION	DESCRIPTION
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
RF	Random Forest

CHAPTER 1

INTRODUCTION

In minor project we have downloaded a sample of datasets which contains transactions made by credit cards in September 2013 by European cardholders. Data is downloaded from kaggle **[1].** This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

There are several techniques like Machine learning, Genetic Programming, fuzzy logic, sequence alignment, etc. are used for detecting credit card fraudulent transactions. Along with these techniques, KNN algorithm which was used in minor project to optimized solution for the fraud detection problem.
In major, a behavior based approach using support vector machines is applied for fraud detection. Support Vector Machine (SVM) is an active research area and successfully solves classification problems in noisy and complex domains. SVM played a major role in the area of machine learning due to its excellent generalization performance in a wide range of learning problems, such as hand written digit recognition, classification of web pages and face detection. The Problem of over fitting is very less in SVM applications. The problems of multi-local minima and curse of dimensionality rarely occurs in SVM. In this work, SVM is applied for fraud detection to classify and predict the data.

Along with this, Random Forest which is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.
While applying these classification methods, Feature Selection, Feature Reduction and Data Visualization methods will also be studied and applied.

Keywords: - Credit Card, Imbalanced dataset, KNN Algorithm, Support Vector Machine (SVM), Random Forest, Fraud Detection, Feature Selection, Feature Reduction, Data Visualization.

1.1 Objectives:

The Project target mainly is to:

- To Study and analyze Credit Card Fraud Detection techniques.
- To develop a credit card fraud detection model which can effectively and accurately detect frauds.

CHAPTER 2

LITERATURE REVIEW

Nancy Demla and Alankrita Aggarwal [2] they examined the performance of advanced data mining techniques support vector machines, together with RBF kernel, for credit card fraud detection. Provide an indication of performance that may be expected when models are applied for fraud detection where the proportion of fraudulent transactions is typically low. Through SVM they predicted 94.3% customers correctly; only 6.7% true bad customers are predicted as good customers; and 13.3% true good customers are predicted as bad ones. They explained, ensemble methods have lower misclassification rates than the single tree method SVM where bagging shows the best predicting result that 94.3.7% customers in the test sample are predicted correctly.

Nana Kwame Gyamfi and Dr Jamal-Deen Abdulai [3] they studied two cases of fraud in banks: credit card fraud and money laundering. The performance of their proposed system was tested on the benchmarks General Ledger, Payables Data, created as similar to bank database. The precision obtained for the single class SVM method, was of about 80%, which represents a significant improvement in comparison to similar works reference. They concluded that results can be improved by studying the influence of various parameters used by the SVM-S architecture.

Navanshu Khare and Saad Yunus Sait [4] they concluded that Logistic regression has a accuracy of 97.7% while SVM shows accuracy of 97.5% and Decision tree shows accuracy of 95.5% but the best results are obtained by Random forest with a precise accuracy of 98.6%. The results obtained thus conclude that Random forest shows the most precise and high accuracy of 98.6% in problem of credit card fraud detection with dataset provided by ULB machine learning.

Devi Meenakshi et al [5] The Random forest algorithm will perform better with a larger number of training data, but speed during testing and application will suffer. Application of more pre-processing techniques would also help. The SVM algorithm still suffers from the imbalanced dataset problem and requires more preprocessing to give better results at the results shown by SVM is great but it could have been better if more preprocessing have been done on the data.

Vaishnave Jonnalagadda et al [6] they have acquired the result of an accurate value of credit card fraud detection i.e. 0.9994802867383512 (99.93%) using a random forest algorithm with new enhancements. In comparison to existing modules, this proposed module is applicable for the larger dataset and provides more accurate results.

Abhilasha Kulkarni et al [7] in this paper, two algorithms Random Forest Algorithm and Local Outlier Factor are compared for detecting fraudulent transactions from the given dataset. Random Forest Algorithm is better in detecting frauds than Local Outlier factor. Efficiency is 99% for Random Forest Algorithm and 94% for Local Outlier factor. Credit card fraud detection is efficient by both of these algorithms but every algorithm has its own specific advantages and disadvantages. Combining more than one algorithm will give higher efficiency.

B.Mohan Kumar et al [8] reviewed that random forest obtains good results on small dataset; there are still some problems such as imbalanced data. But data should be balanced by using smote technique, the accuracy level compared to other algorithm it gives more and also chooses best feature extraction and pre-processing technique to enhance the algorithm performance.

CHAPTER 3

FEASIBILITY STUDY

Three key considerations involved in the feasibility analysis are:

1. Economic Feasibility
2. Technical Feasibility
3. Social Feasibility

Economic Feasibility: This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget was achieved because most of the technologies used are freely available.

Technical Feasibility: This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical Data resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility: The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 4

FACILITIES REQUIRED

4.1 Required Software's

R Studio

R Studio is a free and open-source integrated development environment (IDE) for R, a programming language for statistical computing and graphics. R Studio was founded by JJ Allaire, creator of the programming language ColdFusion. Hadley Wickham is the Chief Scientist at R Studio.

4.2 Languages Required

R Language

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis.

CHAPTER 5

METHODOLOGY

5.1 Data Collection and Data Analysis

The dataset which has been selected and used holds the records of European cardholders who made transactions using their credit cards in the month of September 2013. This dataset holds the record of transactions that were made within two days and total transactions made within two days are 284,807 transactions from which 492 transactions were found as fraudulent which makes the dataset highly imbalanced, more oriented as the positive class i.e., fraud transactions are 0.172% out of total transactions. And the dataset is in CSV format i.e., in a format where the data values are separated by commas. As PCA transformation of input values has been done in the dataset which makes the dataset contain only numerical input values. Unfortunately, the source of dataset did not provide the more background information, original features and information due to confidentiality issues. The numeric values under attributes V1 to V28 and the only feature that were not transformed using the transformation by Principal Component Analysis are the attributes or features 'TIME' and 'AMOUNT'. 'TIME' attributes or feature holds the data that denotes the elapsed time in between the first transaction of the dataset and each transaction. And the attribute or feature 'AMOUNT' hold the data which represents nothing but the amount of the transaction and this feature can also find its use for cost-sensitive and example-dependent machine learning. And finally the attribute or feature 'CLASS' is the response variable and it takes values '1' in the case of fraudulent transaction i.e., positive result and value '0' in the case of the genuine transaction.

Source: https://www.kaggle.com/dalpozz/creditcardfraud

Let's first install all the packages and load the libraries that we are going to need for execution of the code. R packages are a collection of R functions, complied code and sample data. They are stored under a directory called **"library"** in the R environment. By default, R installs a set of packages during installation. More packages are added later, when they are needed for some specific purpose. When we start the R console, only the default packages are available by default. Other packages which are already installed have to be loaded explicitly to be used by the R program that is going to use them.

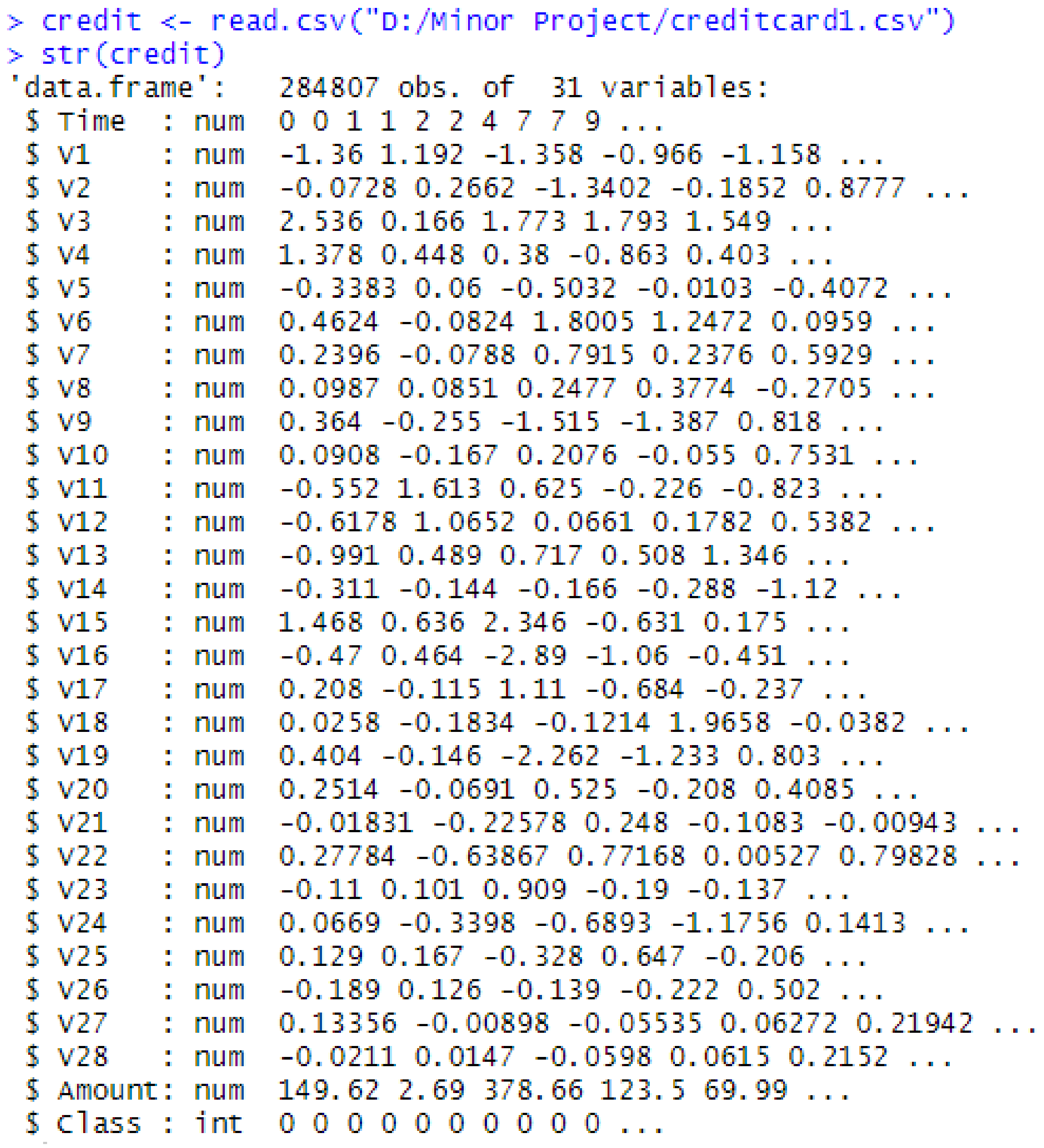
Before a package can be used in the code, it must be loaded to the current R environment. You also need to load a package that is already installed previously but not available in the current environment.

All packages which are going to be used are loaded using the following command:

- library(class) -https://cran.r-project.org/web/packages/class/index.html
- library(caret) -https://cran.r-project.org/web/packages/caret/index.html
- library(e1071) -https://cran.r-project.org/web/packages/e1071/index.html
- library(ggplot2) -https://cran.r-project.org/web/packages/ggplot2/index.html
- library(Boruta) -https://cran.r-project.org/web/packages/Boruta/index.html
- library(mlbench) -https://cran.r-project.org/web/packages/mlbench/index.html
- library(randomForest) -https://cran.r-project.org/web/packages/randomForest/index.html
- library(plotly) -https://cran.r-project.org/web/packages/plotly/index.html
- library(stats) -https://cran.r-project.org/web/packages/stats/index.html
- library(cluster) -https://cran.r-project.org/web/packages/cluster/index.html
- library(caTools) -https://cran.r-project.org/web/packages/caTools/index.html
- library(smotefamily) -https://cran.r-project.org/web/packages/smoteFamily/index.html
- library(pROC) -https://cran.r-project.org/web/packages/pROC/index.html
- library(descr) -https://cran.r-project.org/web/packages/descr/index.html
- library(party) -https://cran.r-project.org/web/packages/party/index.html
- library(Hmisc) -https://cran.r-project.org/web/packages/Hmisc/index.html
- library(magrittr) -https://cran.r-project.org/web/packages/magrittr/index.html
- library(unbalanced) -https://cran.r-project.org/web/packages/unbalanced/index.html
- library(rattle) -https://cran.r-project.org/web/packages/rattle/index.html
- library(nnet) -https://cran.r-project.org/web/packages/nnet/index.html
- library(stringi) -https://cran.r-project.org/web/packages/string/index.html)
- library(readr) -https://cran.r-project.org/web/packages/readr/index.html
- library(dplyr) -https://cran.r-project.org/web/packages/dplyr/index.html

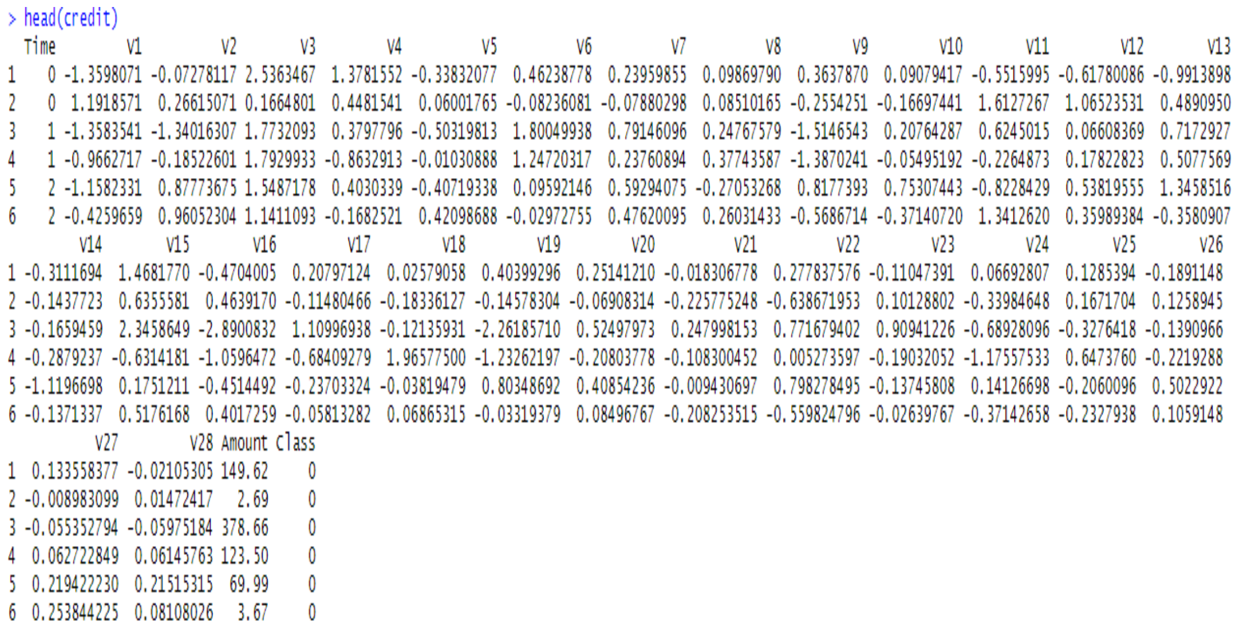
- library(data.table) -https://cran.r-project.org/web/packages/data.table/index.html
- library(DMwR) -https://cran.r-project.org/web/packages/DMwR/index.html
- library(ROSE) -https://cran.r-project.org/web/packages/ROSE/index.html
- library(MLmetrics) -https://cran.r-project.org/web/packages/MLmetrics/index.html
- library(rpart) -https://cran.r-project.org/web/packages/rpart/index.html
- library(rpart.plot) -https://cran.r-project.org/web/packages/rpart.plot/index.html
- library(ROCR) -https://cran.r-project.org/web/packages/ROCR/index.html
- library(GGally) -https://cran.r-project.org/web/packages/GGslly/index.html
- library(Metrics) -https://cran.r-project.org/web/packages/Metrics/index.html
- library(forcats) -https://cran.r-project.org/web/packages/forcats/index.html
- library(gridExtra) -https://cran.r-project.org/web/packages/gridExtra/index.html
- library(grid) -https://cran.r-project.org/web/packages/grid/index.html
- library(lattice) -https://cran.r-project.org/web/packages/lattice/index.html
- library(corrplot) -https://cran.r-project.org/web/packages/corrplot/index.html
- library(knitr) -https://cran.r-project.org/web/packages/knitr/index.html
- library(kableExtra) -https://cran.r-project.org/web/packages/kableExtra/index.html
- library(formattable) -https://cran.r-project.org/web/packages/formattable/index.html

Let’s read the dataset and look at the structure of the dataset:



```
> credit <- read.csv("D:/Minor Project/creditcard1.csv")
> str(credit)
'data.frame':   284807 obs. of  31 variables:
 $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
 $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
 $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
 $ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
 $ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
 $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
 $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.0959 ...
 $ V7    : num  0.2396 -0.0788 0.7915 0.2376 0.5929 ...
 $ V8    : num  0.0987 0.0851 0.2477 0.3774 -0.2705 ...
 $ V9    : num  0.364 -0.255 -1.515 -1.387 0.818 ...
 $ V10   : num  0.0908 -0.167 0.2076 -0.055 0.7531 ...
 $ V11   : num  -0.552 1.613 0.625 -0.226 -0.823 ...
 $ V12   : num  -0.6178 1.0652 0.0661 0.1782 0.5382 ...
 $ V13   : num  -0.991 0.489 0.717 0.508 1.346 ...
 $ V14   : num  -0.311 -0.144 -0.166 -0.288 -1.12 ...
 $ V15   : num  1.468 0.636 2.346 -0.631 0.175 ...
 $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451 ...
 $ V17   : num  0.208 -0.115 1.11 -0.684 -0.237 ...
 $ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
 $ V19   : num  0.404 -0.146 -2.262 -1.233 0.803 ...
 $ V20   : num  0.2514 -0.0691 0.525 -0.208 0.4085 ...
 $ V21   : num  -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
 $ V22   : num  0.27784 -0.63867 0.77168 0.00527 0.79828 ...
 $ V23   : num  -0.11 0.101 0.909 -0.19 -0.137 ...
 $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
 $ V25   : num  0.129 0.167 -0.328 0.647 -0.206 ...
 $ V26   : num  -0.189 0.126 -0.139 -0.222 0.502 ...
 $ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
 $ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
 $ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
 $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

Fig 5.1: Structure of Dataset



```
> head(credit)
  Time         V1          V2        V3         V4          V5          V6          V7          V8         V9         V10        V11         V12        V13
1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086 -0.9913898
2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531  0.4890950
3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369  0.7172927
4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823  0.5077569
5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555  1.3458516
6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384 -0.3580907
         V14        V15        V16         V17         V18         V19         V20          V21          V22         V23         V24        V25        V26
1 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807  0.1285394 -0.1891148
2 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648  0.1671704  0.1258945
3 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096 -0.3276418 -0.1390966
4 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533  0.6473760 -0.2219288
5 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698 -0.2060096  0.5022922
6 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658 -0.2327938  0.1059148
           V27         V28 Amount Class
1  0.133558377 -0.02105305 149.62     0
2 -0.008983099  0.01472417   2.69     0
3 -0.055352794 -0.05975184 378.66     0
4  0.062722849  0.06145763 123.50     0
5  0.219422230  0.21515315  69.99     0
6  0.253844225  0.08108026   3.67     0
```

Fig 5.2: Head of Dataset

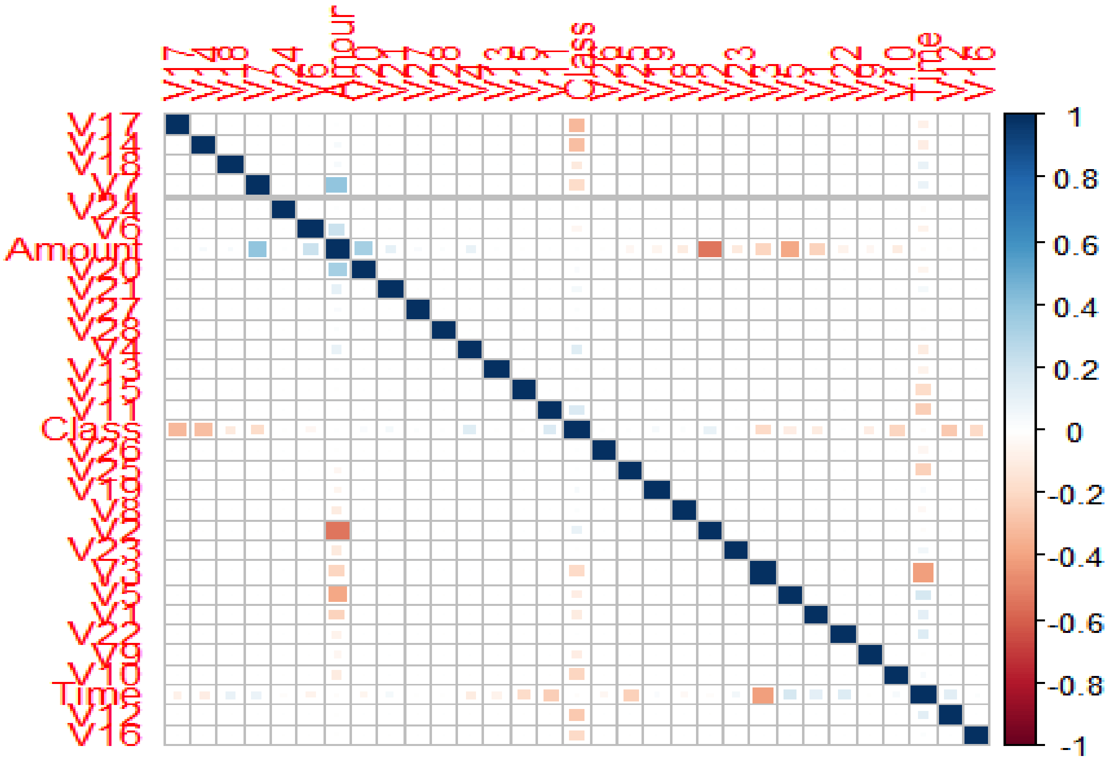


Fig 5.3: Correlation Plot of Dataset

Let’s plot the data and observe the variation of classes according to fraud and not fraud.

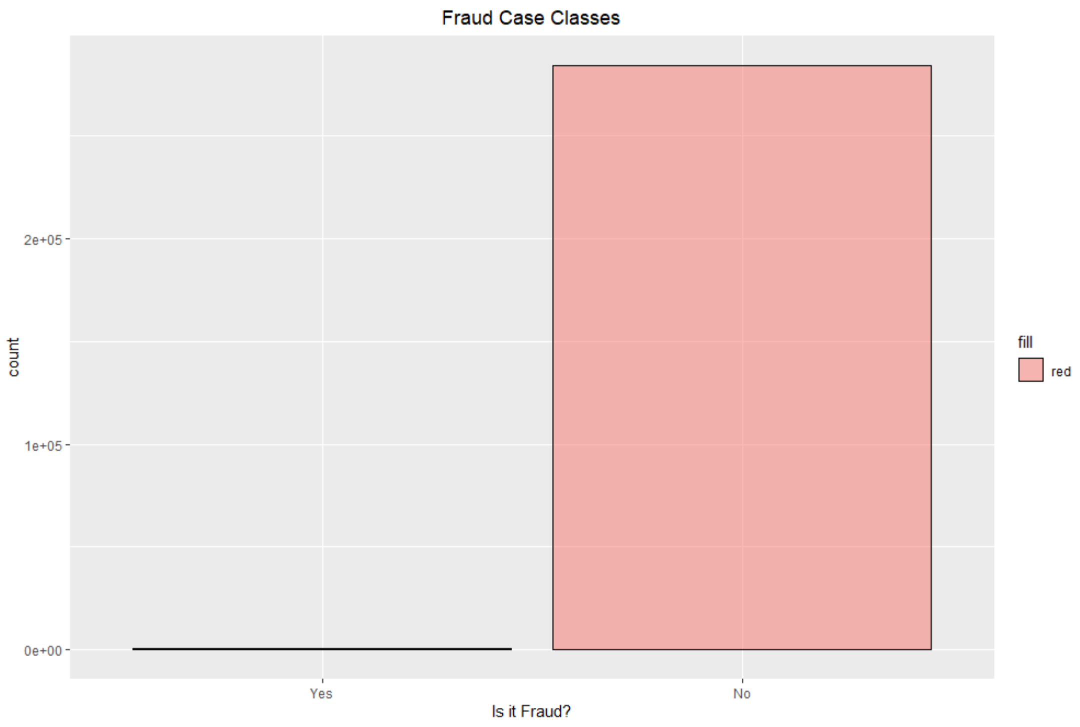


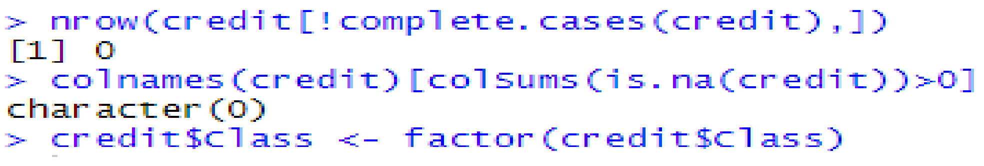
Fig 5.4: Plot of Dataset according to fraud and not fraud

5.2 Data Cleaning

Data cleaning is also called and known as Data cleansing because in this process the inaccurate and corrupted records from the dataset or a record-set or a table are identified and corrected i.e., removed and also this process focuses on identification of incorrect, irrelevant, inaccurate or incomplete parts of the data and then modification of that particular part by replacing it with some different value or completely deleting the dirty data**.**

In this dataset, the very first row wasn’t required for the actual implementation of the algorithms which are used though it helped to analyze the dataset easily because that row is nothing but the attributes of the present data in the dataset and this is the reason why we are performing this step of data cleaning after understanding the dataset in the first step, so in this process of data cleaning or data cleansing, that particular row will be deleted from the dataset.

Because the variable "Class" is of a class "integer", the factor transformation was performed:



```
> nrow(credit[!complete.cases(credit),])
[1] 0
> colnames(credit)[colSums(is.na(credit))>0]
character(0)
> credit$Class <- factor(credit$Class)
```

Fig 5.5: No Empty Rows and Columns

5.3 Feature Selection

- Why to do feature selection?
    a. Consumption of time
    b. Reduces accuracy
    c. Consumes resources
- Due to confidentiality issues, Kaggle doesn't provide the background information about the 28 features out of 30. The only Features defined are 'Time' and 'Amount'. 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the Transaction Amount. 'Class' is the target variable and it is 1 in case of fraud and 0 otherwise.

- Some feature selection methods are
    - ➢ Pearson Correlation method
    - ➢ Recursive Feature Elimination
    - ➢ Chi Squared
    - ➢ Boruta, etc.
- We have used boruta algorithm for feature selection process.
- **Working of Boruta algorithm:-**
    - ➢ Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).
    - ➢ Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.
    - ➢ Then based on importance, it divides the attributes into important and unimportant and tentative attributes.

- If original attribute is doing better than shadow attribute,it will state that as important attribute. If original attribute is not doing better than shadow attribute,it will state that as unimportant attribute.
- If the original attribute has equal importance as shadow attribute ,then this attribute will be stated as tentative attribute.
- Then using the TentativeRoughFix method of boruta package, we will decide whether these tentative attributes are important or unimportant.

- **Results after using Boruta:-**
  - 20 attributes confirmed important: Amount, V1, V10, V11, V12 and 15 more;
  - 8 attributes confirmed unimportant: Time, V13, V19, V22, V24 and 3 more;
  - 2 tentative attributes left: V20, V23;
- **Final results after tentative fixation:-**
  - 21 attributes confirmed important: Amount, V1, V10, V11, V12 and 16 more;
  - 9 attributes confirmed unimportant: Time, V13, V19, V20, V22 and 4 more;

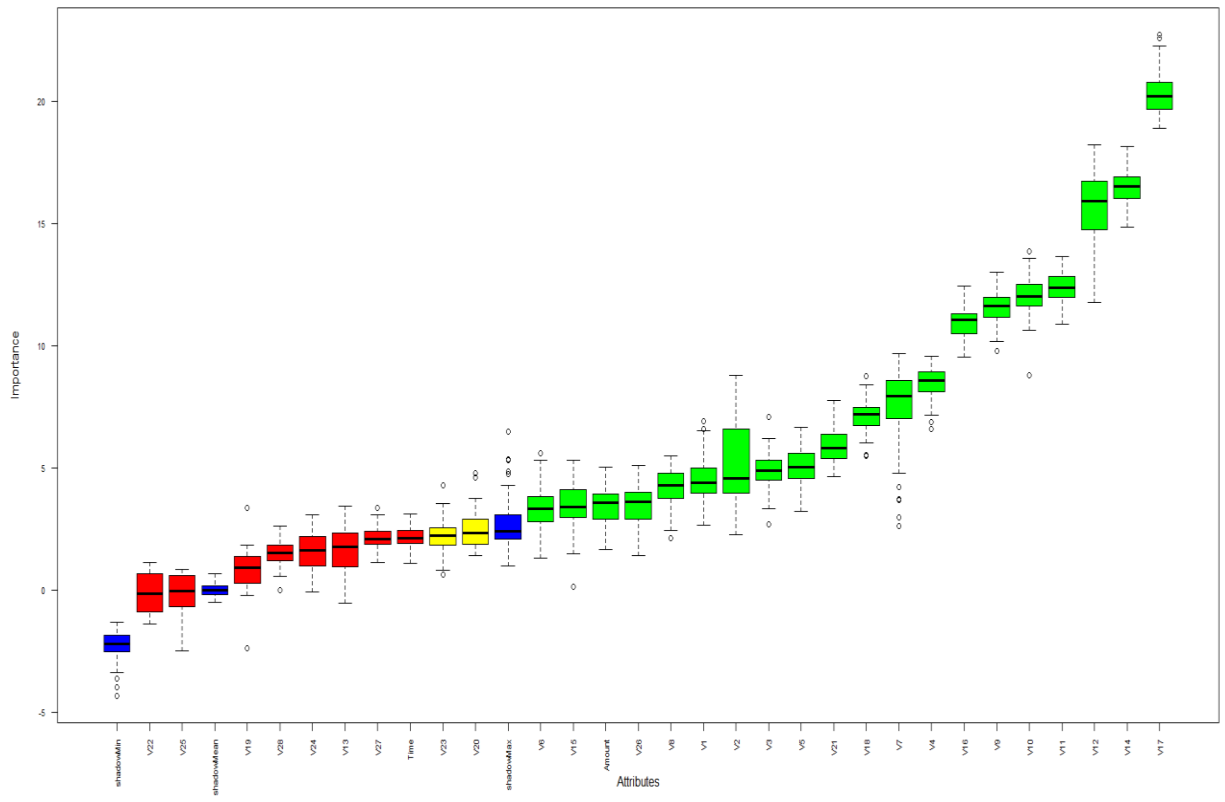
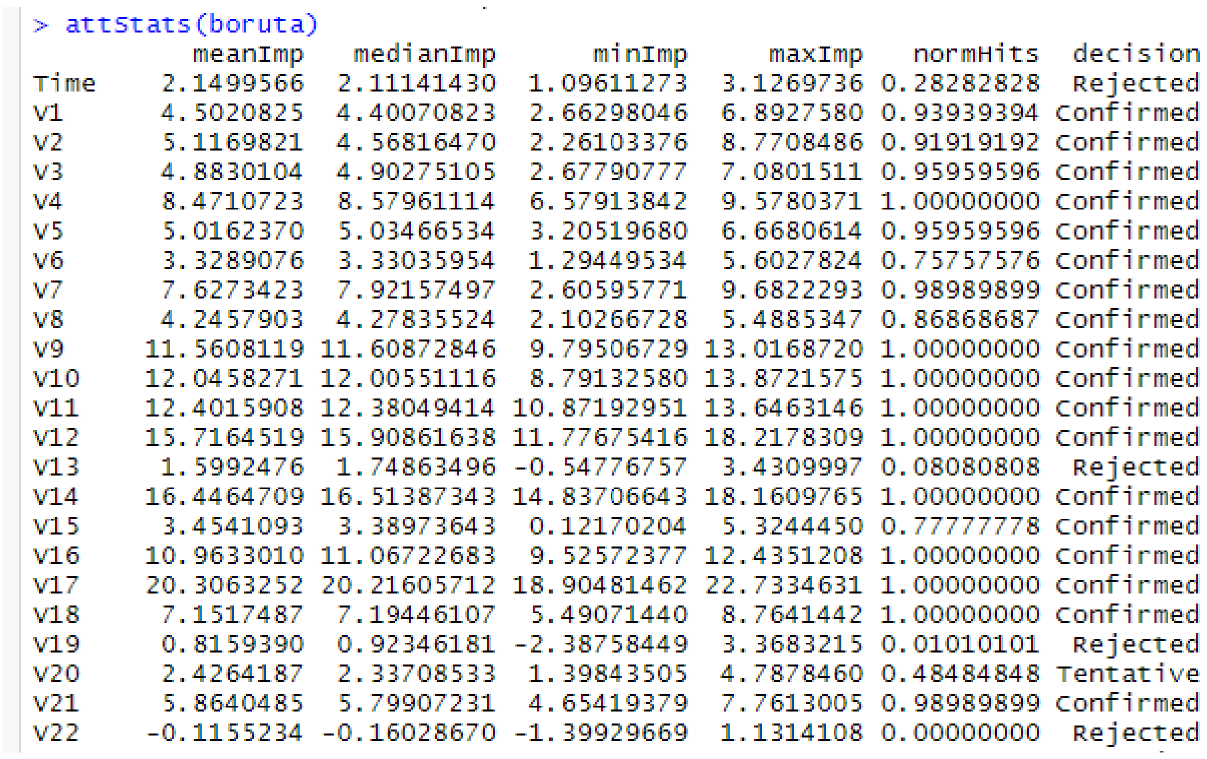


Fig 5.6: Boxplot of divided attributes



```
> attStats(boruta)
          meanImp   medianImp      minImp     maxImp   normHits  decision
Time    2.1499566  2.11141430  1.09611273  3.1269736 0.28282828  Rejected
V1      4.5020825  4.40070823  2.66298046  6.8927580 0.93939394 Confirmed
V2      5.1169821  4.56816470  2.26103376  8.7708486 0.91919192 Confirmed
V3      4.8830104  4.90275105  2.67790777  7.0801511 0.95959596 Confirmed
V4      8.4710723  8.57961114  6.57913842  9.5780371 1.00000000 Confirmed
V5      5.0162370  5.03466534  3.20519680  6.6680614 0.95959596 Confirmed
V6      3.3289076  3.33035954  1.29449534  5.6027824 0.75757576 Confirmed
V7      7.6273423  7.92157497  2.60595771  9.6822293 0.98989899 Confirmed
V8      4.2457903  4.27835524  2.10266728  5.4885347 0.86868687 Confirmed
V9     11.5608119 11.60872846  9.79506729 13.0168720 1.00000000 Confirmed
V10    12.0458271 12.00551116  8.79132580 13.8721575 1.00000000 Confirmed
V11    12.4015908 12.38049414 10.87192951 13.6463146 1.00000000 Confirmed
V12    15.7164519 15.90861638 11.77675416 18.2178309 1.00000000 Confirmed
V13     1.5992476  1.74863496 -0.54776757  3.4309997 0.08080808  Rejected
V14    16.4464709 16.51387343 14.83706643 18.1609765 1.00000000 Confirmed
V15     3.4541093  3.38973643  0.12170204  5.3244450 0.77777778 Confirmed
V16    10.9633010 11.06722683  9.52572377 12.4351208 1.00000000 Confirmed
V17    20.3063252 20.21605712 18.90481462 22.7334631 1.00000000 Confirmed
V18     7.1517487  7.19446107  5.49071440  8.7641442 1.00000000 Confirmed
V19     0.8159390  0.92346181 -2.38758449  3.3683215 0.01010101  Rejected
V20     2.4264187  2.33708533  1.39843505  4.7878460 0.48484848 Tentative
V21     5.8640485  5.79907231  4.65419379  7.7613005 0.98989899 Confirmed
V22    -0.1155234 -0.16028670 -1.39929669  1.1314108 0.00000000  Rejected
```

Fig5.7: Division of attributes through boruta

5.4 Selecting the 10 Most Important Variables

From the above obtained important variables, now we need to use the top 10 important variables to check the accuracy and compare it with the accuracy founded without using important variables.

```
# build random forest model using every variable
rfModel <- randomForest(Class ~ . , data = train)



options(repr.plot.width=5, repr.plot.height=4)
varImpPlot(rfModel,
sort = T,
n.var=10,
main="Top 10 Most Important Variables")
```

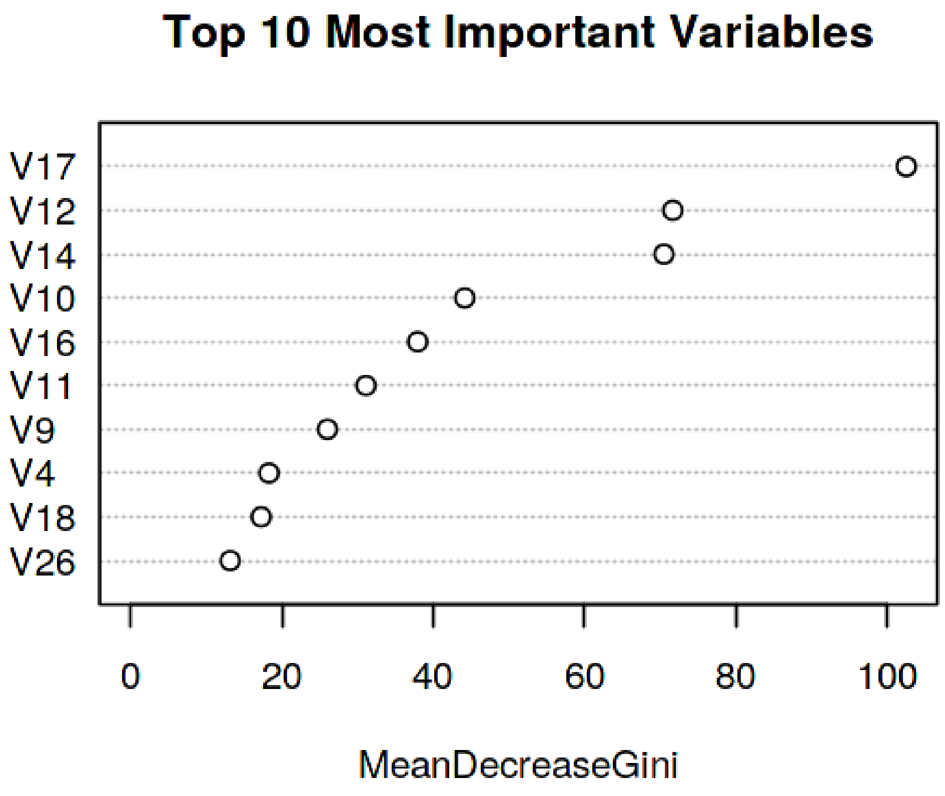


Fig5.8: Top 10 Most Important Variables

.

5.5 Splitting of Dataset

The dataset was bifurcated into training (70% of total) dataset and a testing (30% of total) dataset.

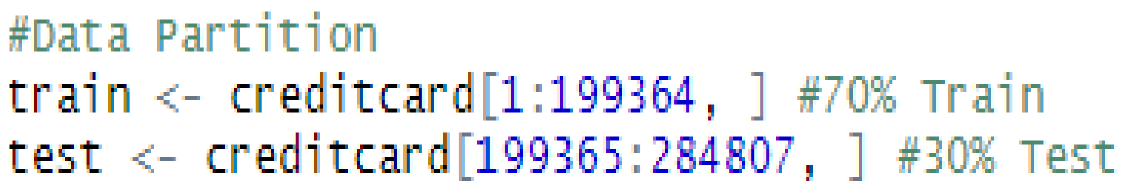

```
#Data Partition
train <- creditcard[1:199364, ] #70% Train
test <- creditcard[199365:284807, ] #30% Test
```

Fig 5.9: Splitting of Dataset

5.6 Applying Random Forest

This section gives a brief overview of random forests and some comments about the features of the method.

Overview

We assume that the user knows about the construction of single classification trees. Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

Features of Random Forests

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.

- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

How random forests work?

To understand and use the various options, further information about how they are computed is useful. Most of the options depend on two data objects generated by random forests.

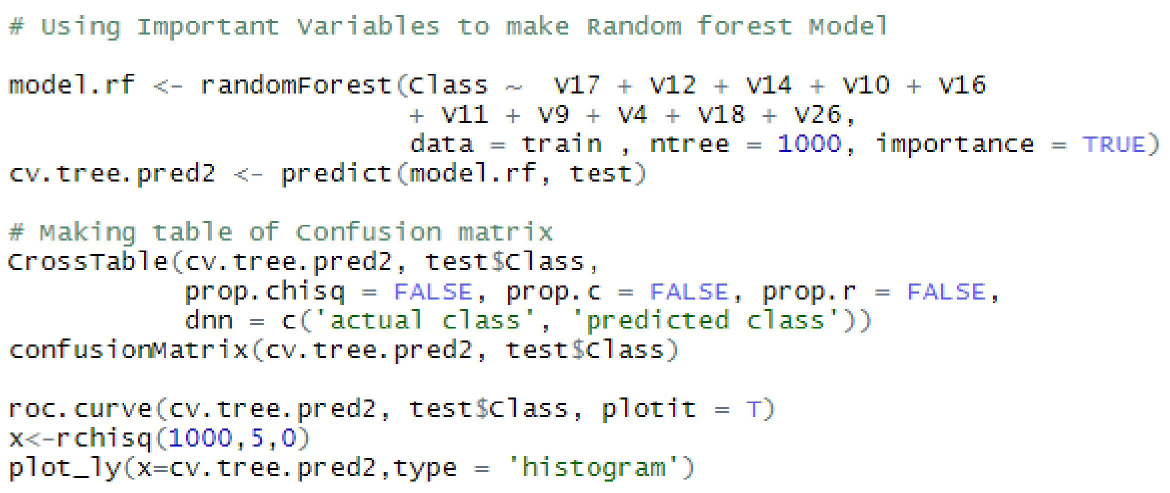
When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Say, we have 1000 observation in the complete population with 10 variables. Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observation and 5 randomly chosen initial variables to build a CART model. It will repeat the process (say) 10 times and then make a final prediction on each observation. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction.

Random forest gives much more accurate predictions when compared to simple CART/CHAID or regression models in many scenarios. These cases generally have high number of predictive variables and huge sample size. This is because it captures the variance of several input variables at the same time and enables high number of observations to participate in the prediction. In some of the coming articles, we will talk more about the algorithm in more detail and talk about how to build a simple random forest on R.

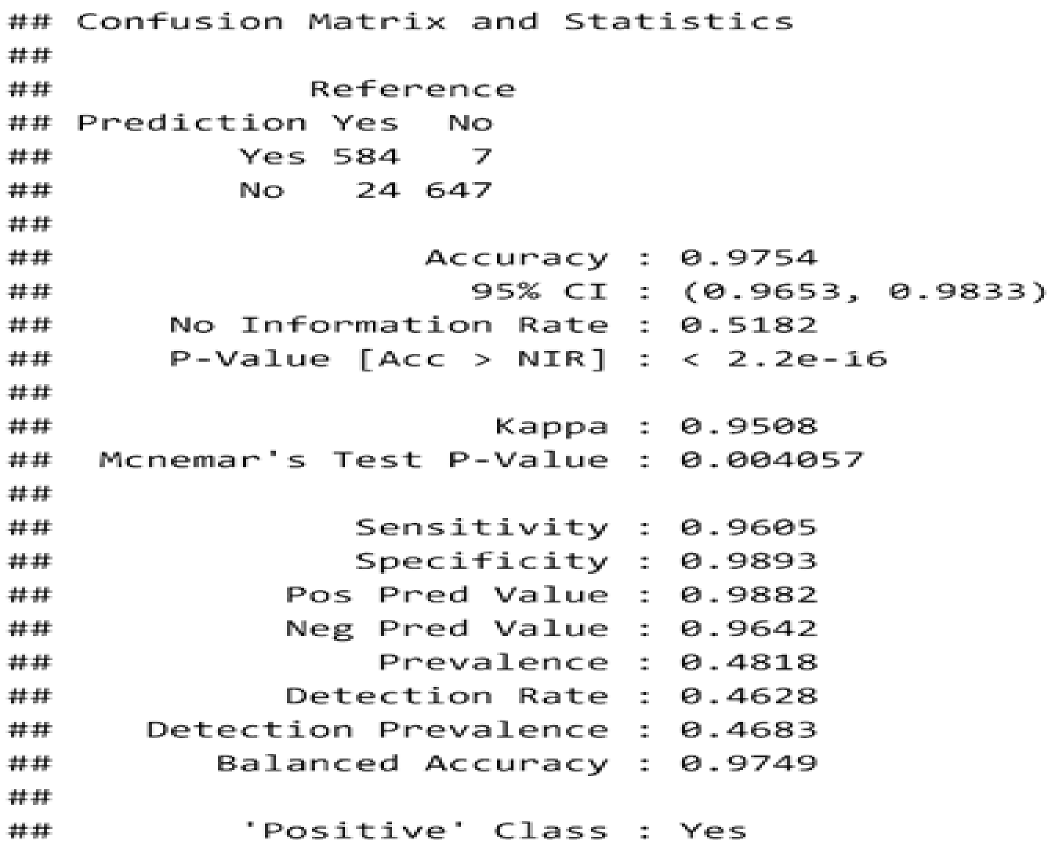
Now Random Forest model is made using the top 10 important variables by using the below code, and accuracy is founded.



```
# Using Important Variables to make Random forest Model

model.rf <- randomForest(Class ~  V17 + V12 + V14 + V10 + V16
                          + V11 + V9 + V4 + V18 + V26,
                          data = train , ntree = 1000, importance = TRUE)
cv.tree.pred2 <- predict(model.rf, test)

# Making table of Confusion matrix
CrossTable(cv.tree.pred2, test$Class,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual class', 'predicted class'))
confusionMatrix(cv.tree.pred2, test$Class)

roc.curve(cv.tree.pred2, test$Class, plotit = T)
x<-rchisq(1000,5,0)
plot_ly(x=cv.tree.pred2,type = 'histogram')
```

Fig 5.10: Random Forest Model of 10 Most Important Variables

The obtained result of above code is shown below:



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 584   7
##        No   24 647
##
##                Accuracy : 0.9754
##                  95% CI : (0.9653, 0.9833)
##     No Information Rate : 0.5182
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9508
##  Mcnemar's Test P-Value : 0.004057
##
##             Sensitivity : 0.9605
##             Specificity : 0.9893
##          Pos Pred Value : 0.9882
##          Neg Pred Value : 0.9642
##              Prevalence : 0.4818
##          Detection Rate : 0.4628
##    Detection Prevalence : 0.4683
##       Balanced Accuracy : 0.9749
##
##        'Positive' Class : Yes
```

Fig 5.11: Result of Random Forest Model

5.7 Applying Support Vector Machine

"Support Vector Machine" (SVM) is a supervised macine learning algoritm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point I n-dimensional space with the value of each feature being the value of a particular coordinate,

Then, we perform classification by finding the hyper-plane that differentiate two classes very well (look at the below snapshot)

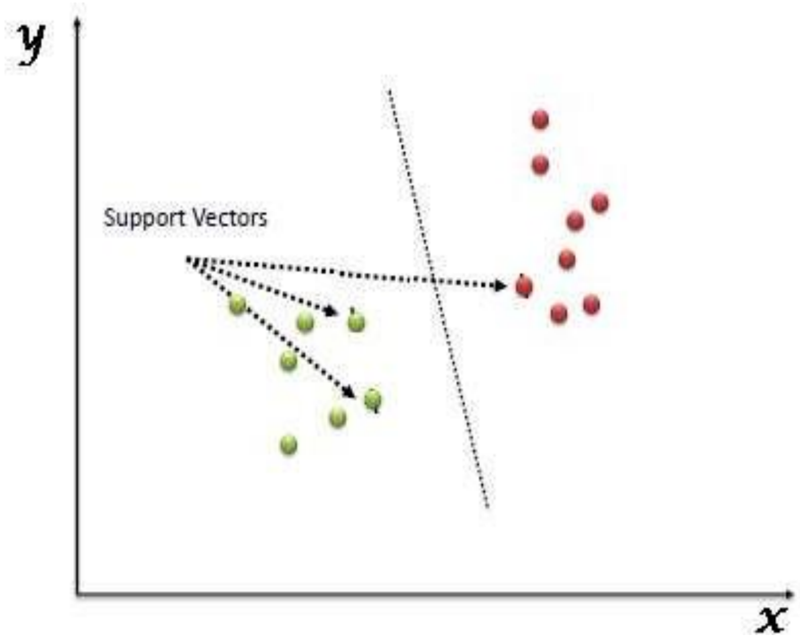


Fig 5.12: SVM Hyper plane

Support Vectors are simply the coordinates of individual observations. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane / line).

How does it work?

Above, we got accustomed to the process of segregating te two classes with a hyper-plane. Now let's see how can we identify hyper-plane?

- Identify the right hyper-plane (Scenario-1): Here, we have three hyper- planes (A, B and C). Now, identify the right hyper-plane to classify star.

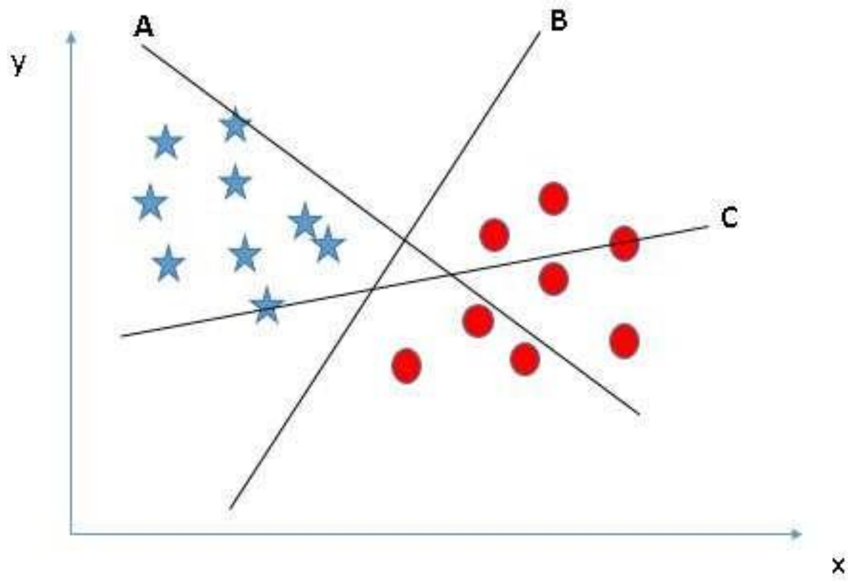


Fig 5.13: Hyper plane Identification

You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

- Identify the right hyper-plane (Scenario-2): Here, we have three hyper- planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

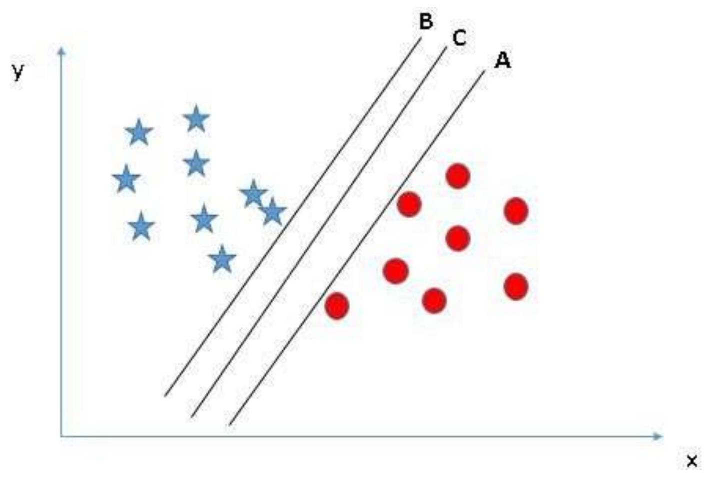


Fig 5.14: Hyper plane Segregated

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin.

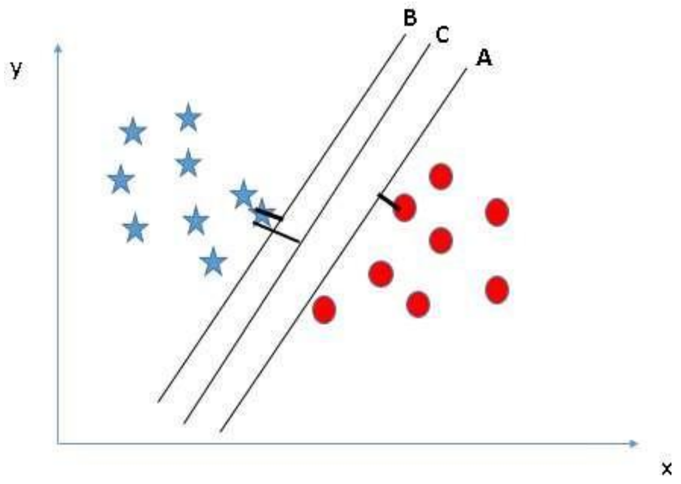


Fig 5.15: Hyper plane with distances

Above, you can see that the margin for hyper-plane c is high as compared to both A and B.Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chanceof miss-classification.

➢ Identify the right hyper-plane (Scenario-3):

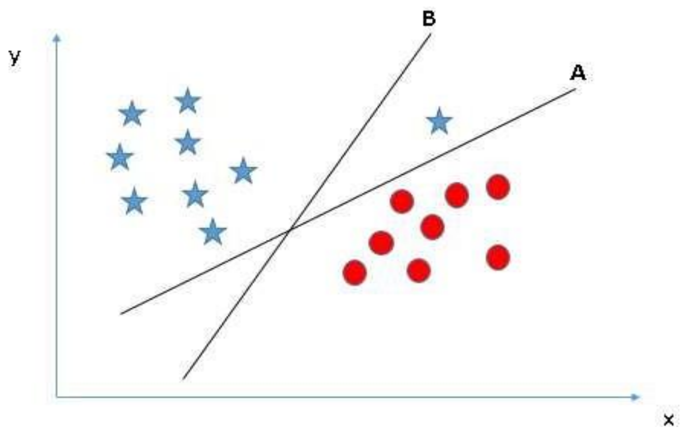


Fig 5.16: Right Hyper plane

Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, theright hyper-plane is **A.**

- Can we classify two classes (Scenario-4)?

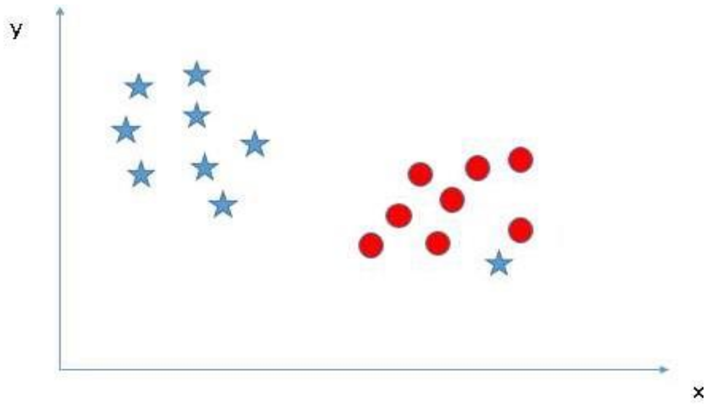


Fig 5.17: Hyper plane Classification

As I have already mentioned, one star at other end is like an outlier for starclass. SVM has a featuretoignoreoutliersandfindthehyper-plane that has maximum that has maximum margin. Hence, we can say , SVM is Robust to outliers..

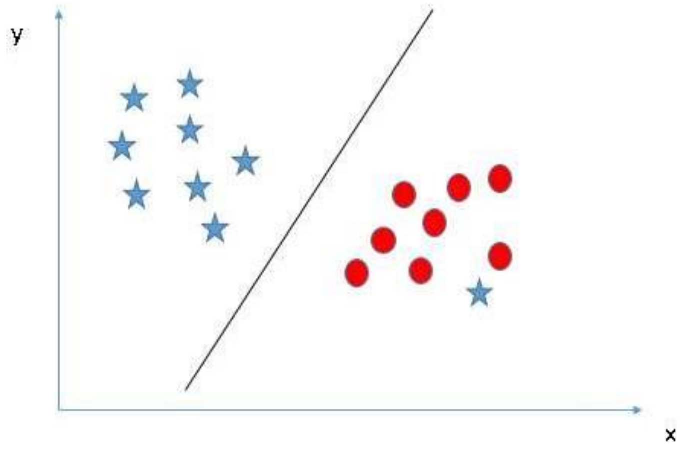


Fig 5.18: SVM with robust identifier

- **Find the hyper-plane to segregate two classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so, how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.

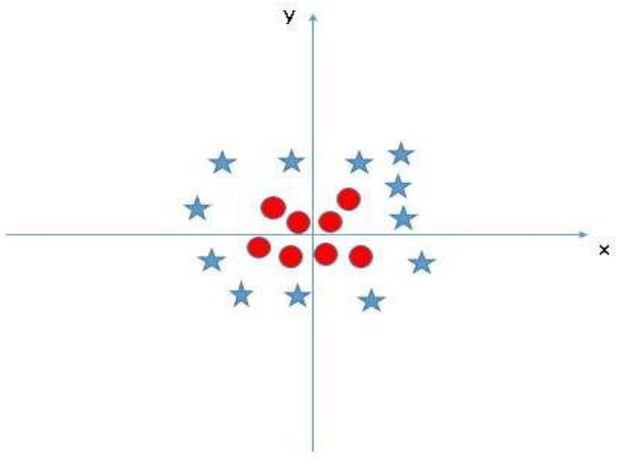


Fig 5.19: Hyper plane to segregate two classes

SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

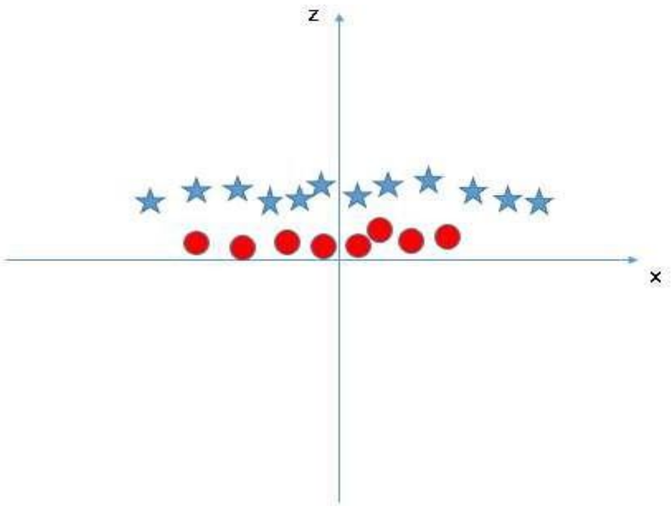


Fig 5.20: Hyper plane to segregate two classes

In above plot, points to consider are:

- Allvaluesforzwouldbepositivealwaysbecausezisthesquared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lowervalueofzandstarrelativelyawayfromthe origin result to higher value of z

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the kernel **trick**. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you've defined.

.

When we look at the hyper-plane in original input space it looks like a circle:

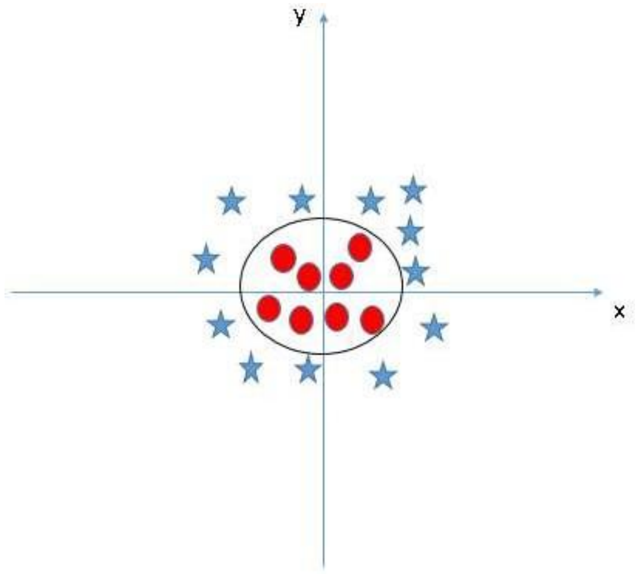


Fig 5.21: Hyper-plane in original input space

Pros and Cons associated with SVM

- Pros:

    a) It works really well with clear margin of separation

    b) It is effective in high dimensional spaces.

    c) It is effective in cases where number of dimensions is greater than the number of samples.

    d) It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

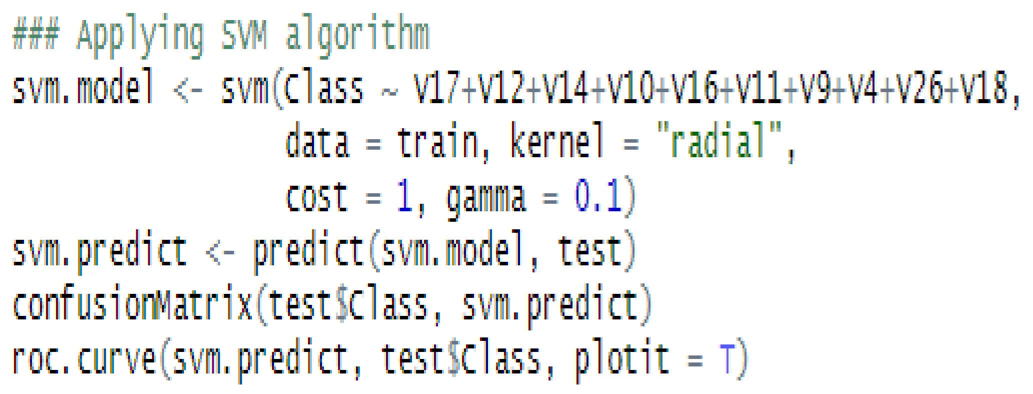
- Cons:

    a) It doesn't perform well, when we have large data set because the required training time is higher

    b) It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping

    c) SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

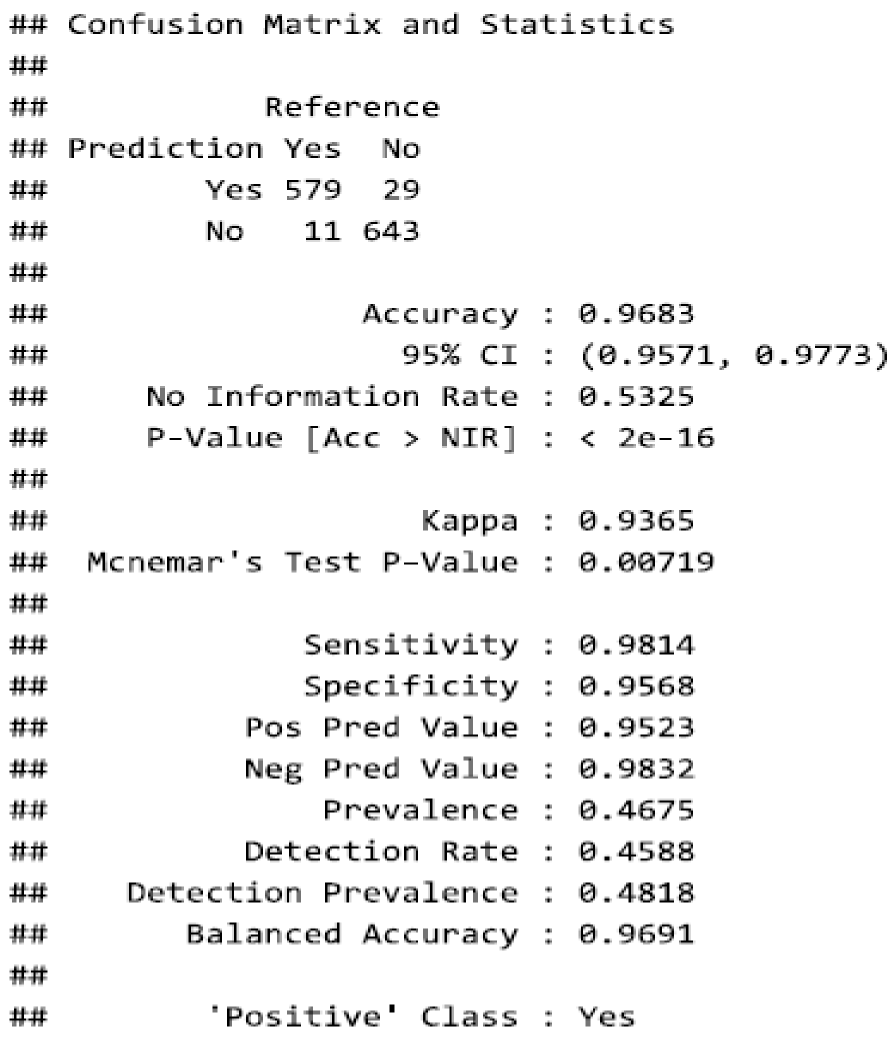
Now we apply SVM algorithm:



```
### Applying SVM algorithm
svm.model <- svm(Class ~ V17+V12+V14+V10+V16+V11+V9+V4+V26+V18,
                 data = train, kernel = "radial",
                 cost = 1, gamma = 0.1)
svm.predict <- predict(svm.model, test)
confusionMatrix(test$Class, svm.predict)
roc.curve(svm.predict, test$Class, plotit = T)
```

Fig 5.22: Applying SVM on Top 10 important Variable

The obtained result of above code is shown below:



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Yes  No
##        Yes 579  29
##        No   11 643
##
##                Accuracy : 0.9683
##                  95% CI : (0.9571, 0.9773)
##     No Information Rate : 0.5325
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9365
##  Mcnemar's Test P-Value : 0.00719
##
##             Sensitivity : 0.9814
##             Specificity : 0.9568
##          Pos Pred Value : 0.9523
##          Neg Pred Value : 0.9832
##              Prevalence : 0.4675
##          Detection Rate : 0.4588
##    Detection Prevalence : 0.4818
##       Balanced Accuracy : 0.9691
##
##        'Positive' Class : Yes
```

Fig 5.23: Result of SVM Model

CHAPTER 6

COMPARISON OF MINOR AND MAJOR PROJECT

After execution of the Random Forest and SVM models, now we compare the results of our minor and major project.

As we have applied three different algorithms on the Credit card dataset taken from Kaggle. On Some calculations these Algorithms provides Accuracy, Precision, and Recall values of the implementation process, which are tabulated as:

Models	Accuracy	Sensitivity	Specificity	Detection Rate
K-Nearest Neighbour	0.6767	0.6332	0.7171	0.3051
Support Vector Machine	0.9683	0.9814	0.9568	0.4588
Random Forest	0.9754	0.9605	0.9893	0.4628

Table No: 1: Comparison Table Of the Three Models

The accuracy values of the models on the studied credit card transaction dataset are shown in above table. It can be seen that all the models perform well on this data set, with Random Forest achieves the best performance with accuracy of 97.54%, while with SVM obtains the accuracy value of 96.83%.

While comparing the values with our KNN model the obtained value of accuracy is 67.67%. It is expected that the methods like RF and SVM perform better than the methods like K-Nearest Neighbor.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1Conclusion

Finally, as the results in table shows that highest accuracy in given by the Random forest algorithm. These results are formed on taking different fraud data rates in the dataset before applying to the algorithms.

The Random forest algorithm will perform better with a larger number of training data, but speed during testing and application will suffer. Application of more pre-processing techniques would also help. The SVM algorithm still suffers from the imbalanced dataset problem and requires more preprocessing to give better results at the results shown by SVM is great but it could have been better if more preprocessing have been done on the data.

7.2Future Scope

- More Algorithms can be applied on the same dataset to get good results.

- Combinations of two or more algorithms of data mining are very helpful in determining the best results. This combination of algorithms is known as Hybrid Algorithm.

- Also by making classifiers in present algorithms and applying new upcoming algorithms may be helpful for improvement.

- Choosing only those attributes which are most important may improve the outcomes.

REFERENCES

[1] https://www.kaggle.com.

[2] Nancy Demla and Alankrita Aggarwal, “Credit Card Fraud Detection using SVM and Reduction of False Alarms”, International Journal of Innovations in Engineering and Technology, H.C.T.M College, Haryana, August 2016.

[3] Nana Kwame Gyamfi and Dr Jamal-Deen Abdulai, “Bank Fraud Detection Using Support Vector Machine”, Kumasi Technical University and University of Ghana, November 2018.

[4] Navanshu Khare and Saad Yunus Sait, “Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models”, International Journal of Pure and Applied Mathematics, SRM Institute of Science and Technology, Kattankulathur Campus, Chennai, Tamil Nadu, 2018.

[5] Devi Meenakshi et al, “Credit Card Fraud Detection Using Random Forest”, International Research Journal of Engineering and Technology (IRJET), March 2019.

[6] Vaishnave Jonnalagadda et al, “Credit card fraud detection using Random Forest Algorithm”, International Journal of Advance Research, Ideas and Innovations in Technology, SRM Institute of Science and Technology, Chennai, Tamil Nadu, 2019.

[7] Abhilasha Kulkarni et al, “Credit Card Fraud Detection Using Random Forest and Local Outlier Factor”, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Marathwada Mitra Mandals College Of Engineering, Pune, 2019.

[8] B.Mohan Kumar et al, “Credit Card Fraud Detection Using Random Forest Technique”, International Journal of Innovative Research in Science, Engineering and Technology, Sri Ramakrishna Engineering College, Coimbatore,Tamilnadu, April 2019.