Summer Training Report

on

CREDIT CARD FRAUD DETECTION SYSTEM

A Project Report submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering
IN COMPUTER SCIENCE AND ENGINEERING

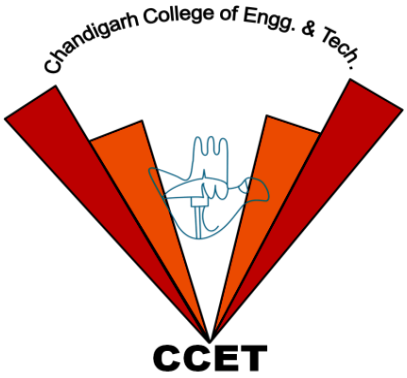
Submitted by

PARINA

(Roll no:CO17343)

Under the supervision of

Dr. Gulshan Goyal - Assistant Professor



CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)

Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University , Chandigarh

Sector-26, Chandigarh. PIN -160019

July, 2020

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc. & Engineering

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled "**CREDIT CARD FRAUD DETECTION SYSTEM**", in fulfillment of the requirement for the award of the degree Bachelor of Engineering in Computer Science & Engineering, submitted in CSE Department, Chandigarh College of Engineering & Technology(Degree wing) affiliated to Punjab University, Chandigarh, is an authentic record of my/our own work carried out during my degree under the guidance of **Dr. Amit Chhabra**. The work reported in this has not been submitted by me for award of any other degree or diploma.

Date : 6/08/2020 PARINA(CO17343)

Place : CCET

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc. & Engineering

CERTIFICATE

JOHNS HOPKINS UNIVERSITY

COURSE CERTIFICATE

06/28/2020

Parina

has successfully completed

R Programming

an online non-credit course authorized by Johns Hopkins University and offered through Coursera

EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

Jeff Leek, PhD; Roger Peng, PhD; Brian Caffo, PhD
Department of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Verify at coursera.org/verify/QD9AEBW8G7XA
Coursera has confirmed the identity of this individual and their participation in the course.

Some online courses may draw on material from courses taught on campus but are not equivalent to on-campus courses. This certificate does not affirm that this learner was enrolled as a student at Johns Hopkins University in any way. It does not confer a JHU grade, course credit or degree; establish any relationship between this learner and JHU or other JHU affiliate; enroll or register this learner at JHU or other JHU affiliate or in any course offered by JHU; or entitle this learner to access or use the resources of JHU or other JHU affiliates beyond the online courses provided by Coursera.

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc. & Engineering

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher **Dr. Amit chhabra** **(Assistant Professor)** as well as our principal **Dr. Manpreet Singh Gujral** who gave me the golden opportunity to do this wonderful project on the topic **CREDIT CARD FRAUD DETECTION SYSTEM**, which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them.

Table of Contents

CHAPTER - 1

R LANGUAGE

1.1 R

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

The R environment

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.
-

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are about eight

packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.

1.2 What is R used for?

- Statistical inference
- Data analysis
- Machine learning algorithm

R by Industry

If we break down the use of R by industry, we see that academics come first. R is a language to do statistic. R is the first choice in the healthcare industry, followed by government and consulting.

R package

The primary uses of R is and will always be, statistic, visualization, and machine learning. The picture below shows which R package got the most questions in Stack Overflow. In the top 10, most of them are related to the workflow of a data scientist: data preparation and communicate the results.

All the libraries of R, almost 12k, are stored in CRAN. CRAN is a free and open source. You can download and use the numerous libraries to perform Machine Learning or time series analysis.

Communicate with R

R has multiple ways to present and share work, either through a markdown document or a shiny app. Everything can be hosted in Rpub, GitHub or the business's website.

Below is an example of a presentation hosted on Rpub

Rstudio accepts markdown to write a document. You can export the documents in different formats:

- Document :
  - HTML
  - PDF/Latex
  - Word
- Presentation
  - HTML
  - PDF beamer

1.3 Why use R?

Data science is shaping the way companies run their businesses. Without a doubt, staying away from Artificial Intelligence and Machine will lead the company to fail. The big question is which tool/language should you use?

They are plenty of tools available in the market to perform data analysis. Learning a new language requires some time investment. The picture below depicts the learning curve compared to the business capability a language offers. The negative relationship implies that there is no free lunch. If you want to give the best insight from the data, then you need to spend some time learning the appropriate tool, which is R.

On the top left of the graph, you can see Excel and PowerBI. These two tools are simple to learn but don't offer outstanding business capability, especially in term of modeling. In the middle, you can see Python and SAS. SAS is a dedicated tool to run a statistical analysis for business, but it is not free. SAS is a click and run software. Python, however, is a language with a monotonous learning curve. Python is a fantastic tool to deploy Machine Learning and AI but lacks communication features. With an identical learning curve, R is a good trade-off between implementation and data analysis.

When it comes to data visualization (DataViz), you'd probably heard about Tableau. Tableau is, without a doubt, a great tool to discover patterns through graphs and charts. Besides, learning Tableau is not time-consuming. One big problem with data visualization is you might end up never finding a pattern or just create plenty of useless charts. Tableau is a good tool for quick visualization of the data or Business Intelligence. When it comes to statistics and decision-making tool, R is more appropriate.

Stack Overflow is a big community for programming languages. If you have a coding issue or need to understand a model, Stack Overflow is here to help. Over the year, the percentage of question-views has increased sharply for R compared to the other languages. This trend is of course highly correlated with the booming age of data science but, it reflects the demand of R language for data science.

In data science, there are two tools competing with each other. R and Python are probably the programming language that defines data science.

1.4 Should you choose R?

Data scientist can use two excellent tools: R and Python. You may not have time to learn them both, especially if you get started to learn data science. Learning statistical modeling and algorithm is far more important than to learn a programming language. A programming language is a tool to compute and communicate your discovery. The most important task in data science is the way you deal with the data: import, clean, prep, feature engineering, feature selection. This should be your primary focus. If you are trying to learn R and Python at the same time without a solid background in statistics, its plain stupid. Data scientist are not programmers. Their job is to understand the data, manipulate it and expose the best approach. If you are thinking about which language to learn, let's see which language is the most appropriate for you.

The principal audience for data science is business professional. In the business, one big implication is communication. There are many ways to communicate: report, web app, dashboard. You need a tool that does all this together.

1.5 SHINY DASHBOARD

Data visualization plays a vital role in life of a Data Scientist. It is easier to visualize complex data and relationships than deciphering them from spreadsheets / tables.

There are several tools for visualizing data such as Tableau, Qlik, Dygraphs, Kibana etc. If I talk specifically about R, it provides three plotting systems:

- The Base Plotting System.
- The Lattice System.
- The ggplot2 System.

But, writing codes for plotting graphs in R time & again can get very tiring. Also, it is very difficult to create an interactive visualization for story narration using above packages. These problems can be resolved by dynamically creating interactive plots in R using Shiny with minimal effort.

If you use R, chances are that you might have come across Shiny. It is an open package from RStudio, used to build interactive web pages with R. It provides a very powerful way to share your analysis in an interactive manner with the community. The best part about shiny is that you don't need any knowledge of HTML, CSS or JavaScript to get started.

Today, I will walk you through all the steps involved in creating a shiny app as well as deploying it online to make it accessible to everyone. This article will provide you a good understanding n how shiny apps work and how they can be useful. To provide you a hands on experience on creating Shiny Apps on your own I will be using the Loan Prediction III Practice Problem. And am sure by the end of this article you will be able to create Shiny apps yourself.

Shiny: Overview

Shiny is an open package from RStudio, which provides a web application framework to create interactive web applications (visualization) called "Shiny apps". The ease of working with Shiny has what popularized it among R users. These web applications seamlessly display R objects (like plots, tables etc.) and can also be made live to allow access to anyone.

Shiny provides automatic reactive binding between inputs and outputs which we will be discussing in the later parts of this article. It also provides extensive pre-built widgets which make it possible to build elegant and powerful applications with minimal effort.

Any shiny app is built using two components:

1.UI.R: This file creates the user interface in a shiny application. It provides interactivity to the shiny app by taking the input from the user and dynamically displaying the generated output on the screen.

2. Server.R: This file contains the series of steps to convert the input given by user into the desired output to be displayed.

3. Setting up shiny

Before we proceed further you need to set up Shiny in your system. Follow these steps to get started.

1. Create a new project in R Studio

2. Select type as Shiny web application.

3. It creates two scripts in R Studio named ui.R and server.R.

4. Each file needs to be coded separately and the flow of input and output between two is possible.

5. Writing "ui.R"

If you are creating a shiny application, the best way to ensure that the application interface runs smoothly on different devices with different screen resolutions is to create it using fluid page. This ensures that the page is laid out dynamically based on the resolution of each device.

The user interface can be broadly divided into three categories:

- Title Panel: The content in the title panel is displayed as metadata, as in top left corner of above image which generally provides name of the application and some other relevant information.
- Sidebar Layout: Sidebar layout takes input from the user in various forms like text input, checkbox input, radio button input, drop down input, etc. It is represented in dark background in left section of the above image.
- Main Panel: It is part of screen where the output(s) generated as a result of performing a set of operations on input(s) at the server.R is / are displayed.

Writing SERVER.R

This acts as the brain of web application. The server.R is written in the form of a function which maps input(s) to the output(s) by some set of logical operations. The inputs taken in ui.R file are accessed using $ operator (input$InputName). The outputs are also referred using the $ operator (output$OutputName). We will be discussing a few examples of server.R in the coming sections of the article for better understanding.

Deploying the Shiny app on the Web

The shiny apps which you have created can be accessed and used by anyone only if, it is deployed on the web. You can host your shiny application on "Shinyapps.io". It provides free of cost platform as a service [PaaS] for deployment of shiny apps, with some restrictions though like only 25 hours of usage in a month, limited memory space, etc. You can also use your own server for deploying shiny apps

CHAPTER – 2

R Packages

2.1 R packages

A package is a suitable way to organize your own work and, if you want to, share it with others. Typically, a package will include code (not only R code!), documentation for the package and the functions inside, some tests to check everything works as it should, and data sets.

The basic information about a package is provided in the DESCRIPTION file, where you can find out what the package does, who the author is, what version the documentation belongs to, the date, the type of license its use, and the package dependencies.

Besides finding the DESCRIPTION files such as cran.r-project.org or stat.ethz.ch, you can also access the description file inside R with the command packageDescription("package"), via the documentation of the package help(package = "package"), or online in the repository of the package.

R packages are collections of functions and data sets developed by the community. They increase the power of R by improving existing base R functionalities, or by adding new ones. For example, if you are usually working with data frames, probably you will have heard about dplyr or data.table, two of the most popular R packages.

2.2 PACKAGES WE USED

2.2.1 Ranger

A fast implementation of Random Forests, particularly suited for high dimensional data. Ensembles of classification, regression, survival and probability prediction trees are supported. Data from genome-wide association studies can be analyzed efficiently. In addition to data frames, datasets of class 'gwaa.data' (R package 'GenABEL') and 'dgCMatrix' (R package 'Matrix') can be directly analyzed.

Version 0.12.1

License GPL-3

URL https://github.com/imbs-hl/ranger

2.2.2 Caret

The caret package (short for **C**lassification **A**nd **RE**gression **T**raining) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

Version 6.0-86
License GPL (>= 2)

URL https://github.com/topepo/caret/

2.2.3 Data.table

This vignette introduces the data.table syntax, its general form, how to *subset* rows, *select and compute* on columns, and perform aggregations *by group*. Familiarity with data.frame data structure from base R is useful, but not essential to follow this vignette.

Data manipulation operations such as *subset*, *group*, *update*, *join* etc., are all inherently related. Keeping these *related operations together* allows for:

- *concise* and *consistent* syntax irrespective of the set of operations you would like to perform to achieve your end goal.
- performing analysis *fluidly* without the cognitive burden of having to map each operation to a particular function from a potentially huge set of functions available before performing the analysis.
- *automatically* optimising operations internally, and very effectively, by knowing precisely the data required for each operation, leading to very fast and memory efficient code.

Version: 1.13.0

License: MPL-2.0

URL: http://r-datatable.com,

https://Rdatatable.gitlab.io/data.table,

https://github.com/Rdatatable/data.table

CHAPTER – 3
ABOUT THE PROJECT

3.1 OUR PROJECT – Detect Credit Card Fraud

Today we will discuss the Credit Card Fraud Detection Project using Machine Learning and R concepts. In this R Project, we will learn how to perform detection of credit cards. We will go through the various algorithms like Decision Trees, Logistic Regression, Artificial Neural Networks and finally, Gradient Boosting Classifier. For carrying out the credit card fraud detection, we will make use of the Card Transactions dataset that contains a mix of fraud as well as non-fraudulent transactions.

3.2 What is Credit Card Fraud?

Credit card fraud is an inclusive term for fraud committed using a payment card, such as a credit card or debit card. The purpose may be to obtain goods or services, or to make payment to another account which is controlled by a criminal. The Payment Card Industry Data Security Standard (PCI DSS) is the data security standard created to help businesses process card payments securely and reduce card fraud.

Credit card fraud can be authorised, where the genuine customer themselves processes a payment to another account which is controlled by a criminal, or unauthorised, where the account holder does not provide authorisation for the payment to proceed and the transaction is carried out by a third party. In 2018, unauthorised financial fraud losses across payment cards and remote banking totalled £844.8 million in the United Kingdom. Whereas banks and card companies prevented £1.66 billion in unauthorised fraud in 2018. That is the equivalent to £2 in every £3 of attempted fraud being stopped.

Credit cards are more secure than ever, with regulators, card providers and banks taking considerable time and effort to collaborate with investigators worldwide to ensure fraudsters aren't successful. Cardholders' money is usually protected from scammers with regulations that make the card provider and bank accountable. The technology and security measures behind credit cards are becoming increasingly sophisticated making it harder for fraudsters to steal money.

3.3 The Dataset

The datasets contains transactions made by credit cards in September 2013 by european cardholders.
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-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

3.4 Prerequisites

Before starting with this R project source code, we should be familiar with the library of R that is ranger , caret and data.table .

Ranger , caret and data.table . are the R packages that are necessary for this project in R. To install them, simply run this load command in your R console

CHAPTER – 4

STEPS FOR BULIDING CREDIT CARD FRAUD DETECTION SYSTEM

4.1 Importing the Datasets

We are importing the datasets that contain transactions made by credit cards-

Code:

```
library(ranger)
library(caret)
library(data.table)
creditcard_data <- read.csv("/home/dataflair/data/Credit Card/creditcard.csv")
```

Input Screenshot:

```
library(ranger)
library(caret)
```

```
## Loading required package: lattice
```

```
library(data.table)
creditcard_data <- read.csv("/home/dataflair/data/Credit Card/creditcard.csv")
```

4.2. Data Exploration

In this section of the fraud detection ML project, we will explore the data that is contained in the creditcard_data dataframe. We will proceed by displaying the creditcard_data using the head() function as well as the tail() function. We will then proceed to explore the other components of this dataframe –

Code:

```
dim(creditcard_data)
head(creditcard_data,6)
```

Output Screenshot

:

```
dim(creditcard_data)
```

```
## [1] 284807     31
```

```
head(creditcard_data,6)
```

```
##   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
```

Code

:

```
1.  tail(creditcard_data,6)
```

Output Screenshot:

```
tail(creditcard_data,6)
```

```
##          Time          V1          V2         V3         V4          V5
## 284802 172785   0.1203164  0.93100513 -0.5460121 -0.7450968  1.13031398
## 284803 172786 -11.8811179 10.07178497 -9.8347835 -2.0666557 -5.36447278
## 284804 172787  -0.7327887 -0.05508049  2.0350297 -0.7385886  0.86822940
## 284805 172788   1.9195650 -0.30125385 -3.2496398 -0.5578281  2.63051512
## 284806 172788  -0.2404400  0.53048251  0.7025102  0.6897992 -0.37796113
## 284807 172792  -0.5334125 -0.18973334  0.7033374 -0.5062712 -0.01254568
##                V6         V7         V8         V9         V10         V11
## 284802 -0.2359732  0.8127221  0.1150929 -0.2040635 -0.6574221  0.6448373
## 284803 -2.6068373 -4.9182154  7.3053340  1.9144283  4.3561704 -1.5931053
## 284804  1.0584153  0.0243297  0.2948687  0.5848000 -0.9759261 -0.1501888
## 284805  3.0312601 -0.2968265  0.7084172  0.4324540 -0.4847818  0.4116137
## 284806  0.6237077 -0.6861800  0.6791455  0.3920867 -0.3991257 -1.9338488
## 284807 -0.6496167  1.5770063 -0.4146504  0.4861795 -0.9154266 -1.0404583
##                V12        V13         V14         V15         V16
## 284802  0.19091623 -0.5463289 -0.73170658 -0.80803553  0.5996281
## 284803  2.71194079 -0.6892556  4.62694203 -0.92445871  1.1076406
## 284804  0.91580191  1.2147558 -0.67514296  1.16493091 -0.7117573
## 284805  0.06311886 -0.1836987 -0.51060184  1.32928351  0.1407160
## 284806 -0.96288614 -1.0420817  0.44962444  1.96256312 -0.6085771
## 284807 -0.03151305 -0.1880929 -0.08431647  0.04133346 -0.3026201
```

Code:

```
1. table(creditcard_data$Class)
2. summary(creditcard_data$Amount)
3. names(creditcard_data)
4. var(creditcard_data$Amount)
```

Output Screenshot:

```
table(creditcard_data$Class)
```

```
##
##      0      1
## 284315    492
```

```
summary(creditcard_data$Amount)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.00     5.60    22.00    88.35    77.17 25691.16
```

```
names(creditcard_data)
```

```
##  [1] "Time"   "V1"     "V2"     "V3"     "V4"     "V5"     "V6"
##  [8] "V7"     "V8"     "V9"     "V10"    "V11"    "V12"    "V13"
## [15] "V14"    "V15"    "V16"    "V17"    "V18"    "V19"    "V20"
## [22] "V21"    "V22"    "V23"    "V24"    "V25"    "V26"    "V27"
## [29] "V28"    "Amount" "Class"
```

```
var(creditcard_data$Amount)
```

```
## [1] 62560.07
```

Code:

```
1. sd(creditcard_data$Amount)
```

Output Screenshot:


```
1. sd(creditcard_data$Amount)
```

4.3. Data Manipulation

In this section of the R data science project, we will scale our data using the scale() function. We will apply this to the amount component of our creditcard_data amount. Scaling is also known as feature standardization. With the help of scaling, the data is structured according to a specified range. Therefore, there are no extreme values in our dataset that might interfere with the functioning of our model. We will carry this out as follows:

Code:

```
head(creditcard_data)
```

Output Screenshot:

```
head(creditcard_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
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767
```

Code:

```
creditcard_data$Amount=scale(creditcard_data$Amount)
NewData=creditcard_data[,-c(1)]
head(NewData)
```

Output Screenshot:

```
creditcard_data$Amount=scale(creditcard_data$Amount)
NewData=creditcard_data[,-c(1)]
head(NewData)
```

```
##           V1          V2        V3         V4          V5          V6
## 1 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6 -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
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767
```

4.4. Data Modeling

After we have standardized our entire dataset, we will split our dataset into training set as well as test set with a split ratio of 0.80. This means that 80% of our data will be attributed to the train_data whereas 20% will be attributed to the test data. We will then find the dimensions using the dim() function –

Code:

```
library(caTools)
set.seed(123)
data_sample = sample.split(NewData$Class,SplitRatio=0.80)
train_data = subset(NewData,data_sample==TRUE)
test_data = subset(NewData,data_sample==FALSE)
dim(train_data)
dim(test_data)
```

Output Screenshot:

```
library(caTools)
set.seed(123)
data_sample = sample.split(NewData$Class,SplitRatio=0.80)
train_data = subset(NewData,data_sample==TRUE)
test_data = subset(NewData,data_sample==FALSE)
dim(train_data)
```

```
## [1] 227846     30
```

```
dim(test_data)
```

```
## [1] 56961    30
```

```
  Logistic_Model=glm(Class~.,test_data,family=binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
  summary(Logistic_Model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial(), data = test_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.9019  -0.0254  -0.0156  -0.0078   4.0877
```

4.5. Fitting Logistic Regression Model

In this section of credit card fraud detection project, we will fit our first model. We will begin with logistic regression. A logistic regression is used for modeling the outcome probability of a class such as pass/fail, positive/negative and in our case – fraud/not fraud. We proceed to implement this model on our test data as follows –

Code:

```
1.  Logistic_Model=glm(Class~.,test_data,family=binomial())
2.  summary(Logistic_Model)
```

Output Screenshot:

```
Logistic_Model=glm(Class~.,test_data,family=binomial())
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(Logistic_Model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial(), data = test_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.9019  -0.0254  -0.0156  -0.0078   4.0877
```

After we have summarised our model, we will visual it through the following plots –

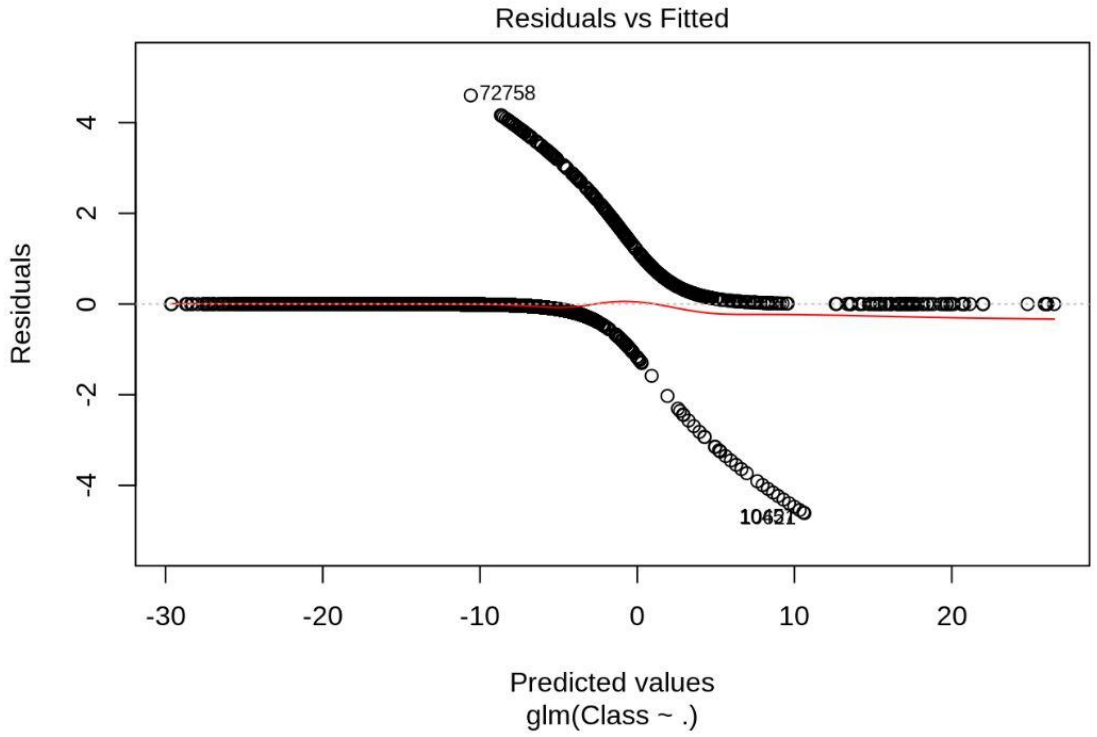
Code:

```
plot(Logistic_Model)
```

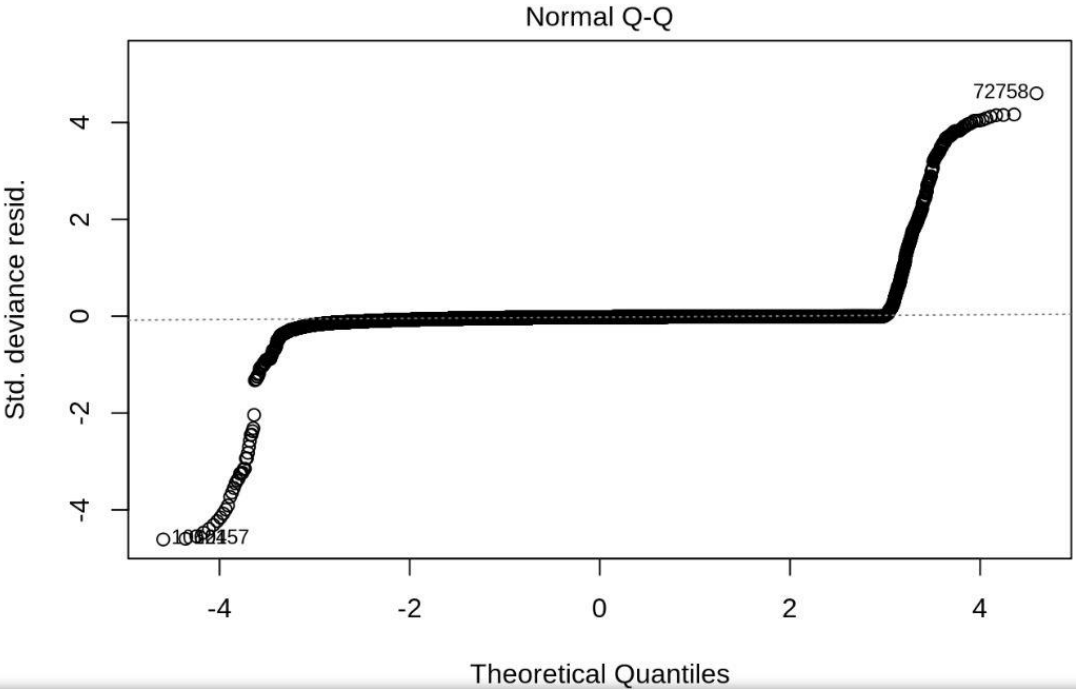
Input Screenshot:

```
plot(Logistic_Model)
```

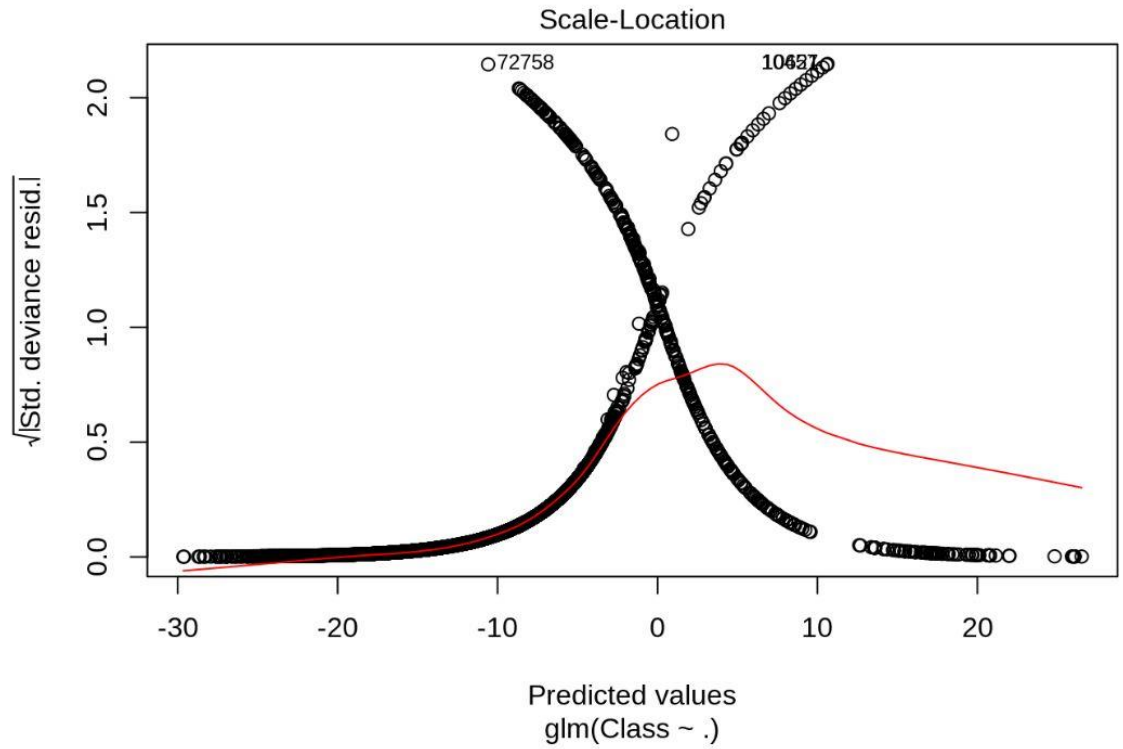
Output:



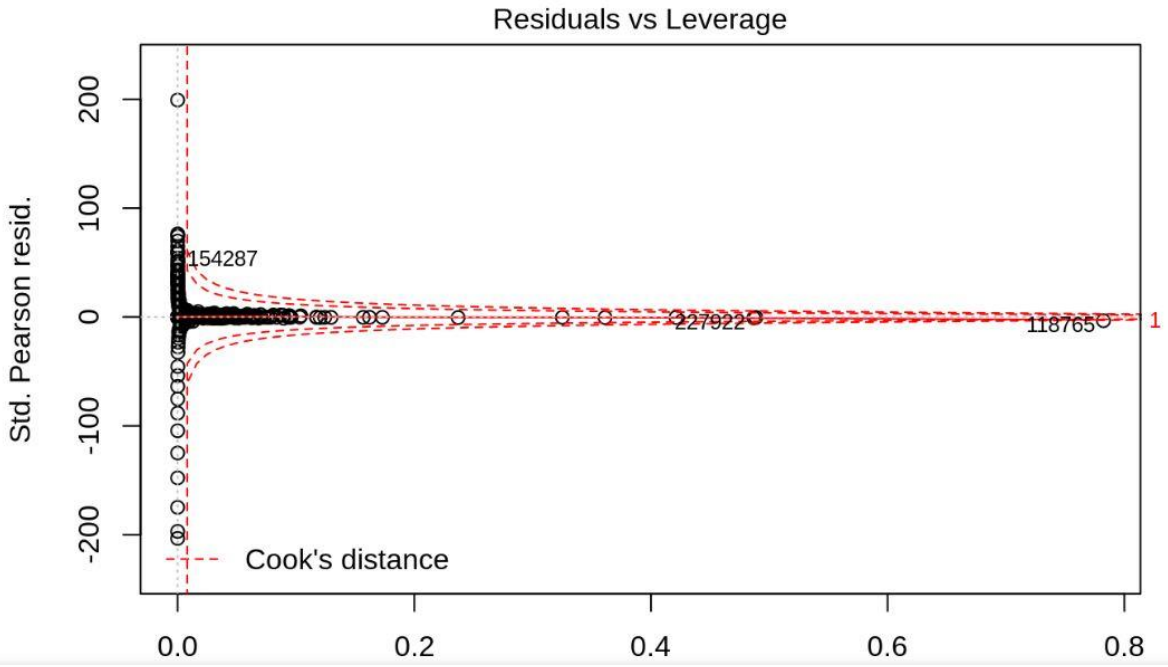
Output:



Output:



Output:



In order to assess the performance of our model, we will delineate the ROC curve. ROC is also known as Receiver Optimistic Characteristics. For this, we will first import the ROC package and then plot our ROC curve to analyze its performance.

Code:

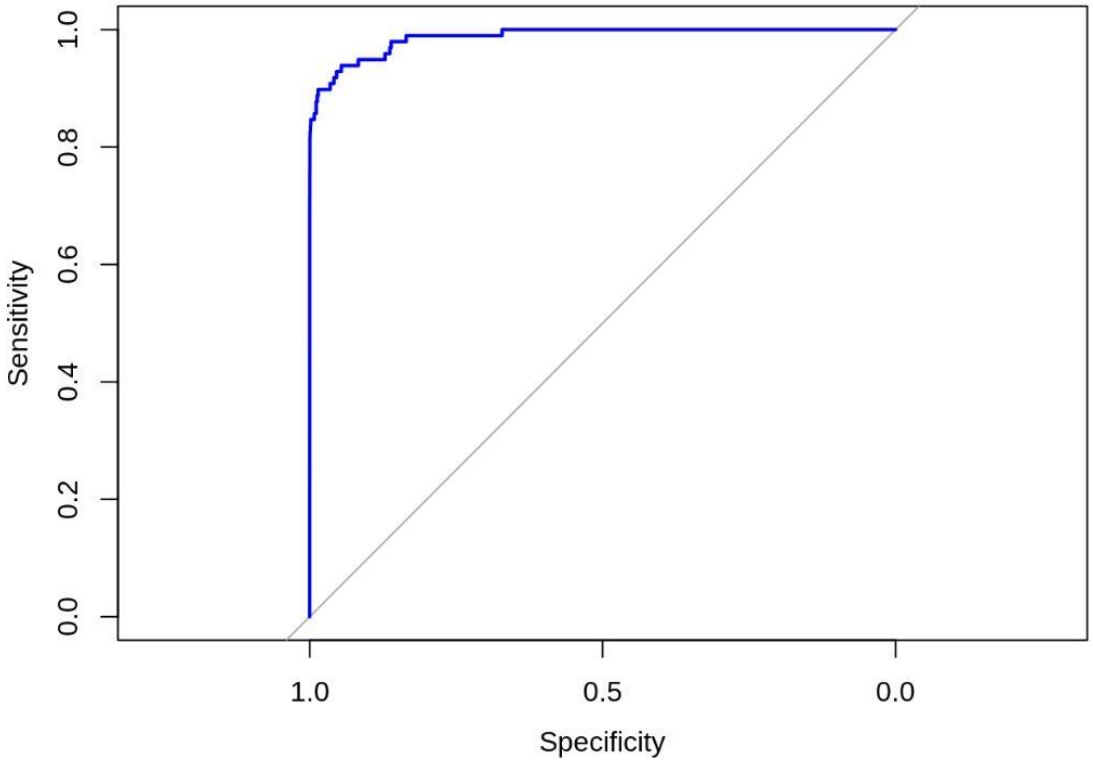
```
library(pROC)
lr.predict <- predict(Logistic_Model,train_data, probability = TRUE)
auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
```

Output Screenshot:

```
Logistic_Model=glm(Class~.,train_data,family=binomial())
summary(Logistic_Model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial(), data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6108  -0.0292  -0.0194  -0.0125   4.6021
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.651305   0.160212 -53.999  < 2e-16 ***
## V1           0.072540   0.044144   1.643 0.100332
## V2           0.014818   0.059777   0.248 0.804220
```

Output:



4.6. Fitting a Decision Tree Model

In this section, we will implement a decision tree algorithm. Decision Trees to plot the outcomes of a decision. These outcomes are basically a consequence through which we can conclude as to what class the object belongs to. We will now implement our decision tree model and will plot it using the rpart.plot() function. We will specifically use the recursive parting to plot the decision tree.

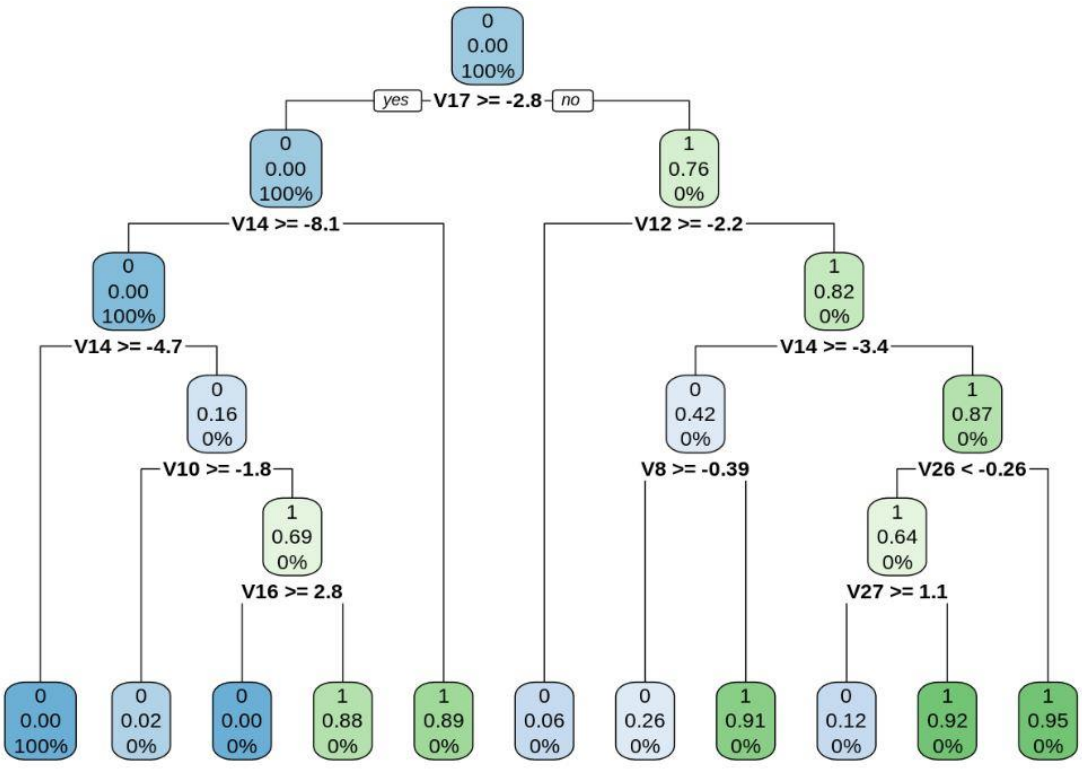
Code:

```
library(rpart)
library(rpart.plot)
decisionTree_model <- rpart(Class ~ . , creditcard_data, method = 'class')
predicted_val <- predict(decisionTree_model, creditcard_data, type = 'class')
probability <- predict(decisionTree_model, creditcard_data, type = 'prob')
rpart.plot(decisionTree_model)
```

Input Screenshot:

```
library(rpart)
library(rpart.plot)
decisionTree_model <- rpart(Class ~ . , creditcard_data, method = 'class')
predicted_val <- predict(decisionTree_model, creditcard_data, type = 'class')
probability <- predict(decisionTree_model, creditcard_data, type = 'prob')

rpart.plot(decisionTree_model)
```

Output:



4.7. Artificial Neural Network

Artificial Neural Networks are a type of machine learning algorithm that are modeled after the human nervous system. The ANN models are able to learn the patterns using the historical data and are able to perform classification on the input data. We import the neuralnet package that would allow us to implement our ANNs. Then we proceeded to plot it using the plot() function. Now, in the case of Artificial Neural Networks, there is a range of values that is between 1 and 0. We set a threshold as 0.5, that is, values above 0.5 will correspond to 1 and the rest will be 0. We implement this as follows –

Code:

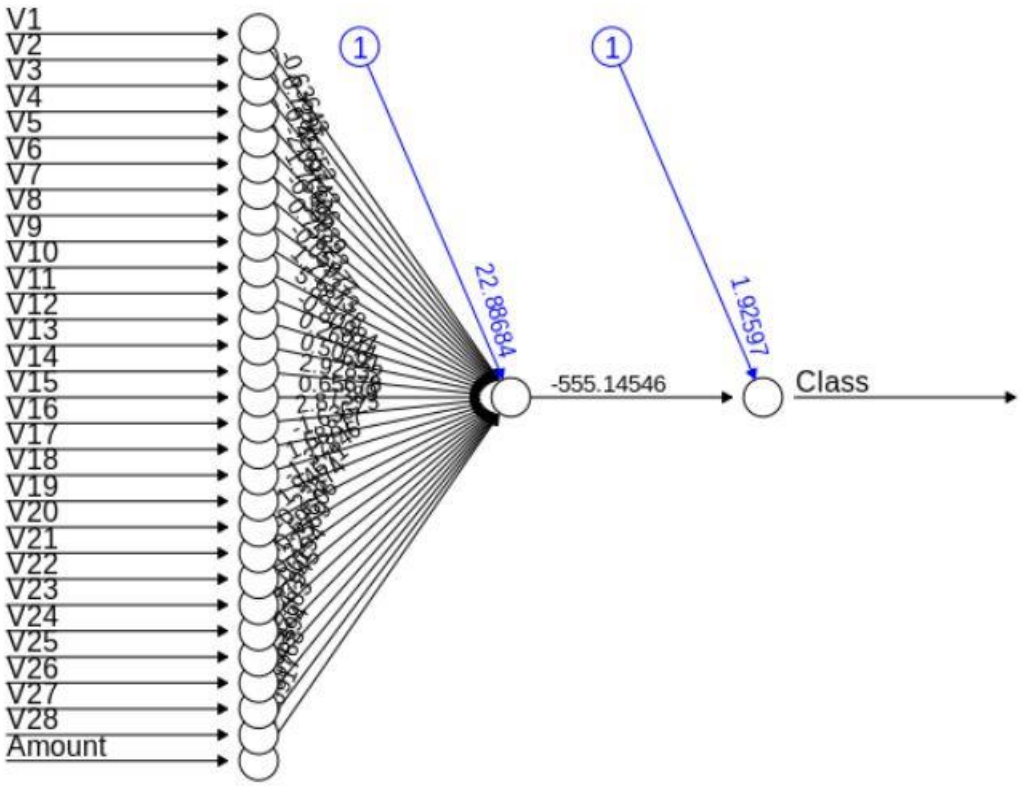
```
library(neuralnet)
ANN_model =neuralnet (Class~.,train_data,linear.output=FALSE)
plot(ANN_model)

predANN=compute(ANN_model,test_data)
resultANN=predANN$net.result
resultANN=ifelse(resultANN>0.5,1,0)
```

Input Screenshot

```
library(neuralnet)
ANN_model =neuralnet (Class~.,train_data,linear.output=FALSE)
plot(ANN_model)
```

```
predANN=compute(ANN_model,test_data)
resultANN=predANN$net.result
resultANN=ifelse(resultANN>0.5,1,0)
```

Output:



4.8. Gradient Boosting (GBM)

Gradient Boosting is a popular machine learning algorithm that is used to perform classification and regression tasks. This model comprises of several underlying ensemble models like weak decision trees. These decision trees combine together to form a strong model of gradient boosting. We will implement gradient descent algorithm in our model as follows –

Code:

```
library(gbm, quietly=TRUE)

# Get the time to train the GBM model
system.time(
       model_gbm <- gbm(Class ~ .
                , distribution = "bernoulli"
                , data = rbind(train_data, test_data)
                , n.trees = 500
                , interaction.depth = 3
                , n.minobsinnode = 100
                , shrinkage = 0.01
                , bag.fraction = 0.5
                , train.fraction = nrow(train_data) / (nrow(train_data) + nrow(test_data))
)
)
# Determine best iteration based on test data
gbm.iter = gbm.perf(model_gbm, method = "test")
```

Input Screenshot:

```
library(gbm, quietly=TRUE)
```

```
## Loaded gbm 2.1.5
```

```
# Get the time to train the GBM model
system.time(
    model_gbm <- gbm(Class ~ .
        , distribution = "bernoulli"
        , data = rbind(train_data, test_data)
        , n.trees = 500
        , interaction.depth = 3
        , n.minobsinnode = 100
        , shrinkage = 0.01
        , bag.fraction = 0.5
        , train.fraction = nrow(train_data) / (nrow(train_data) + nrow(test_data))
        )
)
```

```
##    user  system elapsed
## 345.781   0.144 345.971
```

```
# Determine best iteration based on test data
gbm.iter = gbm.perf(model_gbm, method = "test")
```

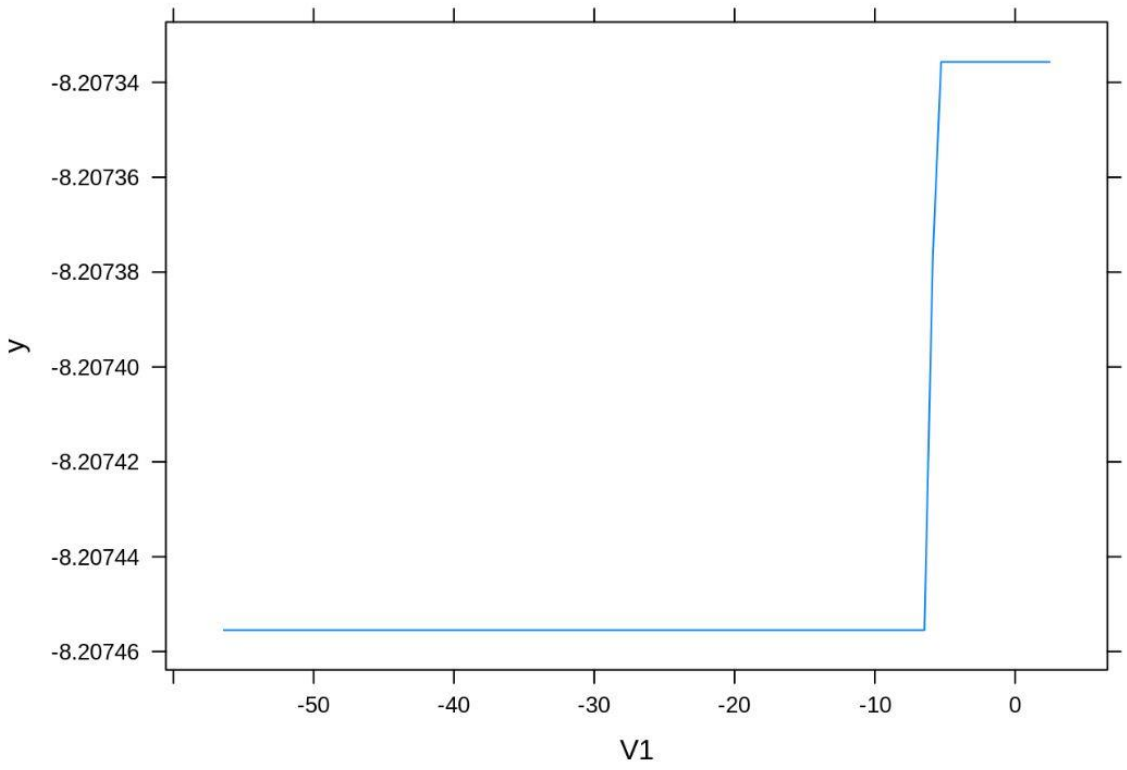
Code:

```
model.influence = relative.influence(model_gbm, n.trees = gbm.iter, sort. = TRUE)
#Plot the gbm model

plot(model_gbm)
```

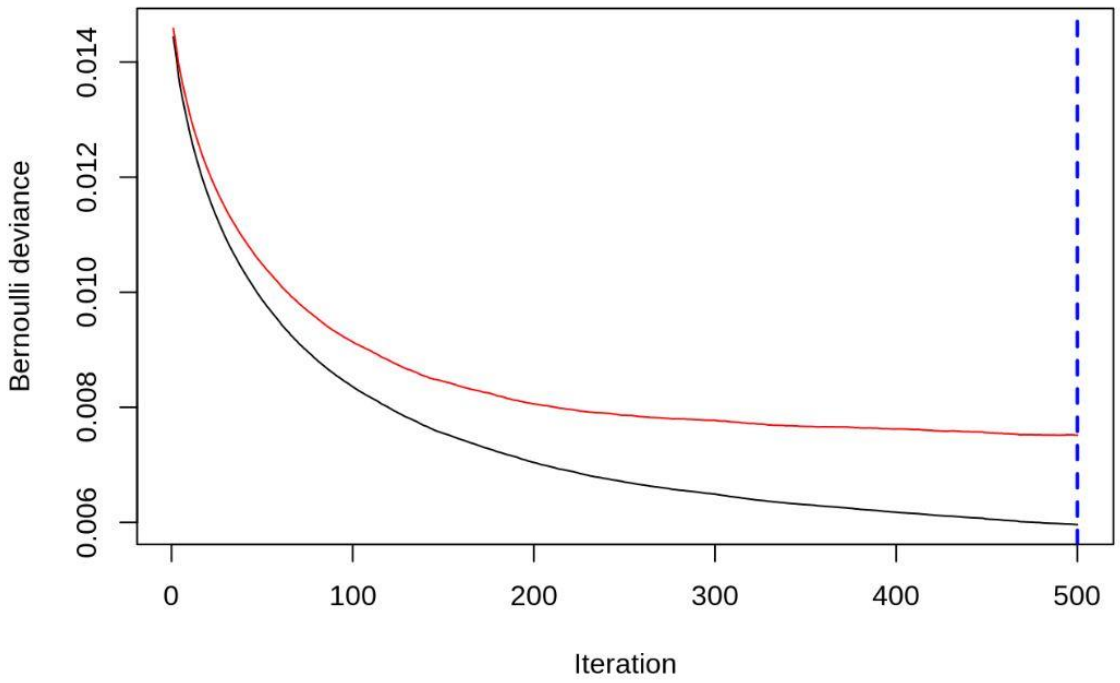
Input Screenshot:

```
model.influence = relative.influence(model_gbm, n.trees = gbm.iter, sort. = TRUE)
#Plot the gbm model

plot(model_gbm)
```

Output:



Output:



Code:

```
# Plot and calculate AUC on test data
gbm_test = predict(model_gbm, newdata = test_data, n.trees = gbm.iter)
gbm_auc = roc(test_data$Class, gbm_test, plot = TRUE, col = "red")
```

Output Screenshot:

```
# Plot and calculate AUC on test data
gbm_test = predict(model_gbm, newdata = test_data, n.trees = gbm.iter)
gbm_auc = roc(test_data$Class, gbm_test, plot = TRUE, col = "red")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Code:

```
print(gbm_auc)
```

Output Screenshot:

```
print(gbm_auc)
```

```
##
## Call:
## roc.default(response = test_data$Class, predictor = gbm_test,     plot = TRUE, col = "red")
##
## Data: gbm_test in 56863 controls (test_data$Class 0) < 98 cases (test_data$Class 1).
## Area under the curve: 0.9555
```

CHATPER – 5

SUMMARY

Concluding our R Data Science project, we learnt how to develop our credit card fraud detection model using machine learning. We used a variety of ML algorithms to implement this model and also plotted the respective performance curves for the models. We learnt how data can be analyzed and visualized to discern fraudulent transactions from other types of data.