

# **CREDIT CARD FRAUD DETECTION USING R**

## **A MINI PROJECT REPORT**

Submitted in partial fulfilment of  
**BACHELORS OF TECHNOLOGY**  
**IN**  
**ARTIFICIAL INTELLIGENCE & DATA SCIENCE**

by

**P. NAVYA (21BQ1A5442)**

Under the guidance of

**K. RAJANI, M. Tech, (Ph. D), Assoc Prof.**



**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**  
(Affiliated to JNTU Kakinada, Approved by AICTE, New Delhi)  
Accredited by NAAC with 'A' Grade, Accredited by NBA  
An ISO 9001:2015 Certified Institution, Nambur:522508, Andhra Pradesh, India

## **DECLARATION**

I hereby declare that this project report titled **CREDIT CARD FRAUD DETECTION USING R** done by **P.Navya**. I affirm that this report has been completed as a Mini Project in R Programming in **Vasireddy Venkatadri Institute of Technology**. I further declare that the information presented in this report is authentic, and any external sources used for reference have been duly acknowledged through appropriate citations and references. The ideas, concepts, and conclusions presented herein are the product of my own research and analysis.

**Signature Of the Candidate**

**(P.Navya)**

## **CERTIFICATE**

This is to certify that the project entitled **CREDIT CARD FRAUD DETECTION USING R** is the bona fide work carried out by **P.Navya(21BQ1A5442)** in partial fulfilment of the requirements for the award of the degree Bachelor of Technology in Artificial Intelligence & Data Science from **Vasireddy Venkatadri Institute Of Technology** during the year 2022-23 under supervision and guidance of (Mrs. K. Rajani).

Signature of the Guide

(K. Rajani)

Signature of the H.O.D

(Dr. T. Sudhir)

## **ACKNOWLEDGEMENT**

We greatly indebted to Dr T. Sudhir, Professor & Head of the Department, for providing motivation and valuable suggestions at every stage of our course of study.

It is our pleasure to express our deep and sincere gratitude to our mini project guide Mrs. K. Rajani, M. Tech, (Ph. D), Assoc Professor, for extending her sincere and heartfelt guidance throughout this project work.

We extend our sincere thanks to all other Teaching and Non-Teaching Faculty members of the department for their cooperation and encouragement throughout our course of study.

We affectionately acknowledge our friends for their motivation and suggestions which helps us in successfully completing our course.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our Parents and Family Members.

P. Navya - 21BQ1A5442

## **ABSTRACT**

In today's digital era, where electronic transactions have become the norm, credit card fraud has emerged as a pressing concern for both individuals and financial institutions. As technology advances, so do the methods employed by fraudsters to exploit vulnerabilities in payment systems. To combat this growing menace and protect consumers, the development of robust fraud detection systems has become essential.

This project aims to address the critical issue of credit card fraud detection by leveraging the power of R, a versatile programming language widely used for data analysis and machine learning. By employing advanced analytical techniques and machine learning algorithms, we seek to build a reliable and efficient fraud detection system that can identify and prevent fraudulent credit card transactions in real-time.

The primary objective of this project is to create a predictive model that can accurately classify transactions as either fraudulent or legitimate. To achieve this, we will utilize historical credit card transaction data, which includes a mixture of both fraudulent and non-fraudulent instances. By extracting meaningful patterns and features from the data, we will train a machine learning model capable of distinguishing between genuine transactions and fraudulent activities.

The fraud detection process will involve several key steps. First, we will explore and pre-process the dataset, which may involve tasks such as data cleaning, feature selection, and normalization. Next, we will delve into exploratory data analysis to gain insights into the distribution of fraudulent transactions and identify potential patterns or anomalies.

To develop an effective fraud detection model, we will employ various machine learning techniques, such as logistic regression, decision trees, random forests, Artificial Neural Networks and finally, Gradient Boosting Classifier and. These algorithms will be trained on labelled data, enabling them to learn patterns and make predictions on unseen instances. We will evaluate the performance of each model using appropriate metrics and select the one that exhibits the highest accuracy and robustness in detecting credit card fraud.

By successfully developing a credit card fraud detection model in R, this project aims to contribute to the ongoing efforts to combat financial fraud. Detecting and preventing fraudulent transactions not only protects individuals from financial losses but also helps maintain trust and integrity in the financial ecosystem.

## APPLICATIONS

**Financial Institutions:** Credit card fraud detection is crucial for financial institutions such as banks, credit card companies, and payment processors. Detecting fraudulent transactions helps these institutions protect their customers from financial losses, maintain trust in their services, and prevent potential damage to their reputation.

**E-commerce and Online Payments:** With the rapid growth of e-commerce, online payments have become a common method of transaction. Credit card fraud detection plays a vital role in securing online transactions, ensuring that fraudulent activities are detected and prevented, thereby safeguarding both merchants and customers.

**Retail Industry:** Credit card fraud detection is essential for retailers, particularly those operating in brick-and-mortar stores. By identifying fraudulent transactions in real-time, retailers can reduce the risk of financial losses, improve operational efficiency, and protect their customers' payment information.

**Insurance Companies:** Insurance providers face challenges related to fraudulent claims. By applying credit card fraud detection techniques, insurance companies can identify and investigate potentially fraudulent claims, leading to cost savings and maintaining the integrity of their operations.

**Government Agencies:** Government agencies that handle financial transactions, such as tax authorities or social security offices, can benefit from credit card fraud detection to identify fraudulent activities and prevent misuse of public funds.

**Mobile Payment Apps:** Mobile payment applications have gained popularity in recent years. Implementing credit card fraud detection in these apps is crucial to ensure secure transactions and protect users' financial data.

**Travel and Hospitality Industry:** Credit card fraud detection is significant in the travel and hospitality sector, where fraudulent activities can occur during hotel bookings, airline ticket purchases, or car rentals. Detecting and preventing fraud helps protect businesses and customers from potential losses.

**Healthcare Sector:** The healthcare industry faces challenges related to medical insurance fraud and fraudulent billing. Credit card fraud detection techniques can be employed to detect and prevent such fraudulent activities, reducing financial losses and maintaining the integrity of the healthcare system.

## INTRODUCTION

To get started with credit fraud detection in R, we need to have a basic understanding of R programming and data manipulation. Once we have that, we can import your time series data into R, perform some initial exploratory data analysis, and then start building models to analyse our data.

### Platform: R STUDIO

R was specifically designed for statistical analysis, which makes it highly suitable for data science applications. Although the learning curve for programming with R can be steep, especially for people without prior programming experience, the tools now available for carrying out text analysis in R make it easy to perform powerful, cutting-edge text analytics using only a few simple commands. One of the keys to R's explosive growth has been its densely populated collection of extension software libraries, known in R terminology as packages, supplied and maintained by R's extensive user community. Each package extends the functionality of the base R language and core packages, and in addition to functions and data must include documentation and examples, often in the form of vignettes demonstrating the use of the package. The best known package repository, the Comprehensive R Archive Network (CRAN), currently has over 10,000 packages that are published.

Text analysis in particular has become well established in R. There is a vast collection of dedicated text processing and text analysis packages, from low-level string operations to advanced text modelling techniques such as fitting Latent Dirichlet Allocation models, R provides it all. One of the main advantages of performing text analysis in R is that it is often possible, and relatively easy, to switch between different packages or to combine them. Recent efforts among the R text analysis developers' community are designed to promote this interoperability to maximize flexibility and choice among users. As a result, learning the basics for text analysis in R provides access to a wide range of advanced text analysis features.

## PROJECT SPECIFICATION

 R Studio version 1.2.5033

### DATASET

 creditcard.csv

For our R project we utilized historical credit card transaction data(creditcard.csv), which includes a mixture of both fraudulent and non-fraudulent instances.

The dataset here contains transactions made by credit cards in September 2013 by european cardholders. This dataset from Kaggle is available here. 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.

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659
10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755
10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.323729
11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023
12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087
12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617	0.747731
12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333	-0.26798
13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527	-0.73798
14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.55974	0.160842	1.23309	0.345173
15	1.492936	-1.02935	0.454795	-1.43803	-1.55543	-0.72096	-1.08066	-0.05313	-1.97868	1.638076

V11	V12	V13	V14	V15	V16	V17	V18	V19	V20
-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791	0.403993	0.251412
1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336	-0.14578	-0.06908
0.624501	0.066084	0.717293	-0.16595	2.345865	-2.89008	1.109969	-0.12136	-2.26186	0.52498
-0.22649	0.178228	0.507757	-0.28792	-0.63142	-1.05965	-0.68409	1.965775	-1.23262	-0.20804
-0.82284	0.538196	1.345852	-1.11967	0.175121	-0.45145	-0.23703	-0.03819	0.803487	0.408542
1.341262	0.359894	-0.35809	-0.13713	0.517617	0.401726	-0.05813	0.068653	-0.03319	0.084968
-1.41691	-0.15383	-0.75106	0.167372	0.050144	-0.44359	0.002821	-0.61199	-0.04558	-0.21963
-0.61947	0.291474	1.757964	-1.32387	0.686133	-0.07613	-1.22213	-0.35822	0.324505	-0.15674
-0.70512	-0.11045	-0.28625	0.074355	-0.32878	-0.21008	-0.49977	0.118765	0.570328	0.052736
1.017614	0.83639	1.006844	-0.44352	0.150219	0.739453	-0.54098	0.476677	0.451773	0.203711
1.199644	-0.67144	-0.51395	-0.09505	0.23093	0.031967	0.253415	0.854344	-0.22137	-0.38723
-0.25912	-0.32614	-0.09005	0.362832	0.928904	-0.12949	-0.80998	0.359985	0.707664	0.125992
0.227666	-0.24268	1.205417	-0.31763	0.725675	-0.81561	0.873936	-0.84779	-0.68319	-0.10276
-0.77366	0.323387	-0.01108	-0.17849	-0.65556	-0.19993	0.124005	-0.9805	-0.98292	-0.1532
0.844555	0.792944	0.370448	-0.73498	0.406796	-0.30306	-0.15587	0.778265	2.221868	-1.58212
-0.79398	-0.77041	1.047627	-1.0666	1.106953	1.660114	-0.27927	-0.41999	0.432535	0.263451
-0.45031	0.936708	0.70838	-0.46865	0.354574	-0.24663	-0.00921	-0.59591	-0.57568	-0.11391
0.324098	0.277192	0.252624	-0.2919	-0.18452	1.143174	-0.92871	0.68047	0.025436	-0.04702
0.91723	0.970117	-0.26657	-0.47913	-0.52661	0.472004	-0.72548	0.075081	-0.40687	-2.19685
1.077542	-0.63205	-0.41696	0.052011	-0.04298	-0.16643	0.304241	0.554432	0.05423	-0.38791










V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
-0.01831	0.277838	-0.11047	0.066928	0.128539	-0.18911	0.133558	-0.02105	149.62	0
-0.22578	-0.63867	0.101288	-0.33985	0.16717	0.125895	-0.00898	0.014724	2.69	0
0.247998	0.771679	0.909412	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	0
-0.1083	0.005274	-0.19032	-1.17558	0.647376	-0.22193	0.062723	0.061458	123.5	0
-0.00943	0.798278	-0.13746	0.141267	-0.20601	0.502292	0.219422	0.215153	69.99	0
-0.20825	-0.55982	-0.0264	-0.37143	-0.23279	0.105915	0.253844	0.08108	3.67	0
-0.16772	-0.27071	-0.1541	-0.78006	0.750137	-0.25724	0.034507	0.005168	4.99	0
1.943465	-1.01545	0.057504	-0.64971	-0.41527	-0.05163	-1.20692	-1.08534	40.8	0
-0.07343	-0.26809	-0.20423	1.011592	0.373205	-0.38416	0.011747	0.142404	93.2	0
-0.24691	-0.63375	-0.12079	-0.38505	-0.06973	0.094199	0.246219	0.083076	3.68	0
-0.0093	0.313894	0.02774	0.500512	0.251367	-0.12948	0.04285	0.016253	7.8	0
0.049924	0.238422	0.00913	0.99671	-0.76731	-0.49221	0.042472	-0.05434	9.99	0
-0.23181	-0.48329	0.084668	0.392831	0.161135	-0.35499	0.026416	0.042422	121.5	0
-0.03688	0.074412	-0.07141	0.104744	0.548265	0.104094	0.021491	0.021293	27.5	0
1.151663	0.222182	1.020586	0.028317	-0.23275	-0.23556	-0.16478	-0.03015	58.8	0
0.499625	1.35365	-0.25657	-0.06508	-0.03912	-0.08709	-0.181	0.129394	15.99	0
-0.02461	0.196002	0.013802	0.103758	0.364298	-0.38226	0.092809	0.037051	12.99	0
-0.1948	-0.67264	-0.15686	-0.88839	-0.34241	-0.04903	0.079692	0.131024	0.89	0
-0.5036	0.98446	2.458589	0.042119	-0.48163	-0.62127	0.392053	0.949594	46.8	0
-0.17765	-0.17507	0.040002	0.295814	0.332931	-0.22038	0.022298	0.007602	5	0

Note: These are the sample 20 transactions from the original dataset

The dataset contains only numerical input variables which are the result of a PCA transformation. Due to confidentiality issues, we do not have access to 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-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

## PACKAGES REQUIRED

-  ranger
-  caret
-  rpart
-  caTools
-  pROC
-  neuralnet
-  gbm

## DATA EXPLORATION

Firstly we imported the datasets that contain transactions made by credit cards. To perform analysis, reading of data set is done using command “read.csv”

```
# Importing Datasets
```

```
creditcard_data <- read.csv
```

```
("C:/Users/Desktop/22/Rproject/creditcard.csv")
```

Then we explored the data that is contained in the creditcard\_data dataframe. After displaying the creditcard\_data using the head() function as well as the tail() function, we proceeded to explore the other components of this dataframe.

```
# Data Exploration
```

```
dim(creditcard_data)
```

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

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

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

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

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

```
sd(creditcard_data$Amount)
```

**output:**

```
> 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
	V19	V20	V21	V22	V23	V24
1	0.40399296	0.25141210	-0.018306778	0.277837576	-0.11047391	0.06692807
2	-0.14578304	-0.06908314	-0.225775248	-0.638671953	0.10128802	-0.33984648
3	-2.26185710	0.52497973	0.247998153	0.771679402	0.90941226	-0.68928096
4	-1.23262197	-0.20803778	-0.108300452	0.005273597	-0.19032052	-1.17557533
5	0.80348692	0.40854236	-0.009430697	0.798278495	-0.13745808	0.14126698
6	-0.03319379	0.08496767	-0.208253515	-0.559824796	-0.02639767	-0.37142658
	V25	V26	V27	V28	Amount	Class
1	0.1285394	-0.1891148	0.133558377	-0.02105305	149.62	0
2	0.1671704	0.1258945	-0.008983099	0.01472417	2.69	0
3	-0.3276418	-0.1390966	-0.055352794	-0.05975184	378.66	0
4	0.6473760	-0.2219288	0.062722849	0.06145763	123.50	0
5	-0.2060096	0.5022922	0.219422230	0.21515315	69.99	0
6	-0.2327938	0.1059148	0.253844225	0.08108026	3.67	0

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

Time	V1	V2	V3	V4	V5
284802	172785	0.1203164	0.93100513	-0.5460121	-0.7450968
284803	172786	-11.8811179	10.07178497	-9.8347835	-2.0666557
284804	172787	-0.7327887	-0.05508049	2.0350297	-0.7385886
284805	172788	1.9195650	-0.30125385	-3.2496398	-0.5578281
284806	172788	-0.2404400	0.53048251	0.7025102	0.6897992
284807	172792	-0.5334125	-0.18973334	0.7033374	-0.5062712

	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	V17
284802	0.19091623	-0.5463289	-0.73170658	-0.80803553	0.5996281	0.07044075
284803	2.71194079	-0.6892556	4.62694203	-0.92445871	1.1076406	1.99169111
284804	0.91580191	1.2147558	-0.67514296	1.16493091	-0.7117573	-0.02569286
284805	0.06311886	-0.1836987	-0.51060184	1.32928351	0.1407160	0.31350179
284806	-0.96288614	-1.0420817	0.44962444	1.96256312	-0.6085771	0.50992846
284807	-0.03151305	-0.1880929	-0.08431647	0.04133346	-0.3026201	-0.66037665

	V18	V19	V20	V21	V22	V23
284802	0.3731103	0.1289038	0.0006758329	-0.3142046	-0.8085204	0.05034266
284803	0.5106323	-0.6829197	1.4758291347	0.2134541	0.1118637	1.01447990
284804	-1.2211789	-1.5455561	0.0596158999	0.2142053	0.9243836	0.01246304
284805	0.3956525	-0.5772518	0.0013959703	0.2320450	0.5782290	-0.03750086
284806	1.1139806	2.8978488	0.1274335158	0.2652449	0.8000487	-0.16329794
284807	0.1674299	-0.2561169	0.3829481049	0.2610573	0.6430784	0.37677701

	V24	V25	V26	V27	V28	Amount
284802	0.102799590	-0.4358701	0.1240789	0.217939865	0.06880333	2.69
284803	-0.509348453	1.4368069	0.2500343	0.943651172	0.82373096	0.77
284804	-1.016225669	-0.6066240	-0.3952551	0.068472470	-0.05352739	24.79
284805	0.640133881	0.2657455	-0.0873706	0.004454772	-0.02656083	67.88
284806	0.123205244	-0.5691589	0.5466685	0.108820735	0.10453282	10.00
284807	0.008797379	-0.4736487	-0.8182671	-0.002415309	0.01364891	217.00

Class

284802	0
284803	0
284804	0
284805	0
284806	0
284807	0

```
> 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" "V7"
[9] "V8" "V9" "V10" "V11" "V12" "V13" "V14" "V15"
[17] "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23"
[25] "V24" "V25" "V26" "V27" "V28" "Amount" "Class"
```

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

```
[1] 62560.07
```

```
> sd(creditcard_data$Amount)
```

```
[1] 250.1201
```

## DATA MANIPULATION

In this section of the project, the data is scaled using the `scale()` function. we applied this to the amount component of our `creditcard_data` amount. With the help of scaling, the data is structured according to a specified range. Therefore, there are no extreme values in the dataset that might interfere with the functioning of the model.

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

## DATA MODELLING

After standardizing the entire dataset, the dataset is splitted into training set as well as test set with a split ratio of 0.80. This means that 80% of the data will be attributed to the `train_data` whereas 20% will be attributed to the `test_data`. we then found the dimensions using the `dim()` function.

```
# Data Modelling  
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)
```

## FITTING LOGISTIC REGRESSION MODEL

In this section of the project, we fit the first model. we began with logistic regression and used it for modeling the outcome probability of fraud/not fraud. we proceeded to implement this model on the test data. Once we summarised the model, we visualized it through plots. In order to assess the performance of the model, we portrayed the Receiver Optimistic Characteristics or ROC curve. For this, we first imported the ROC package and then plotted the ROC curve to analyze its performance.

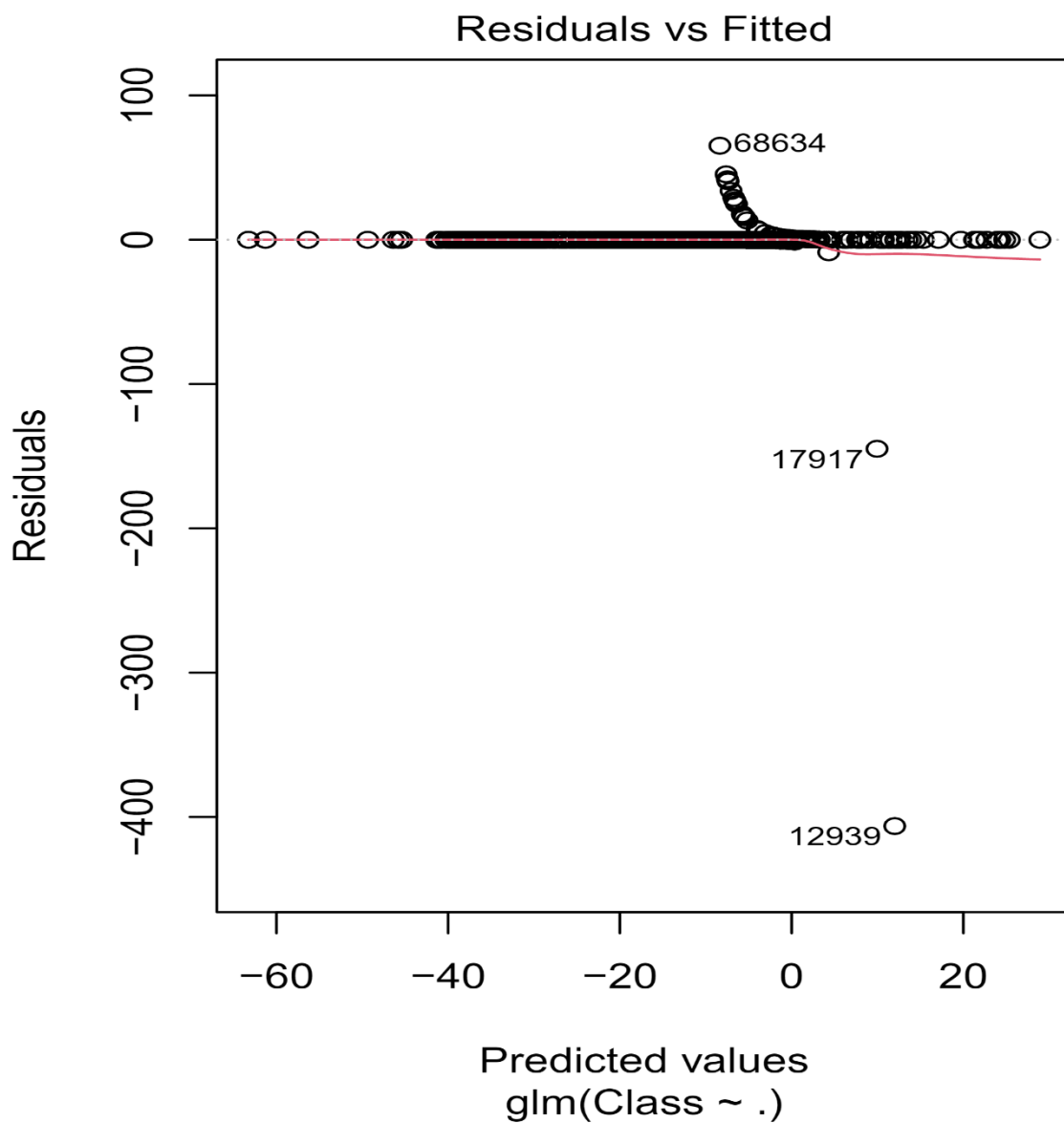
```
# Fitting Logistic Regression Model
```

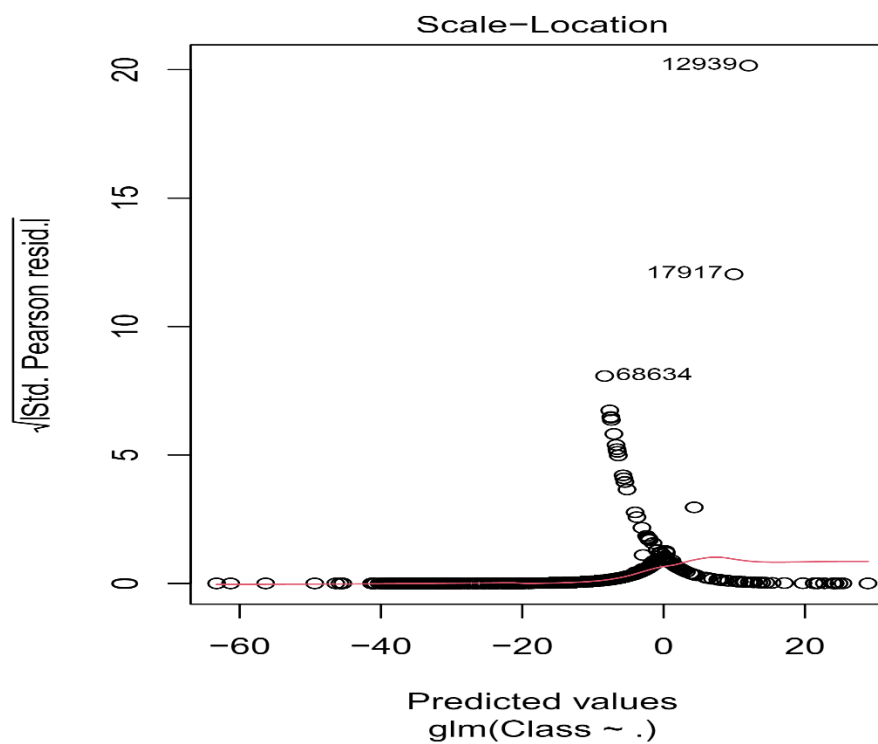
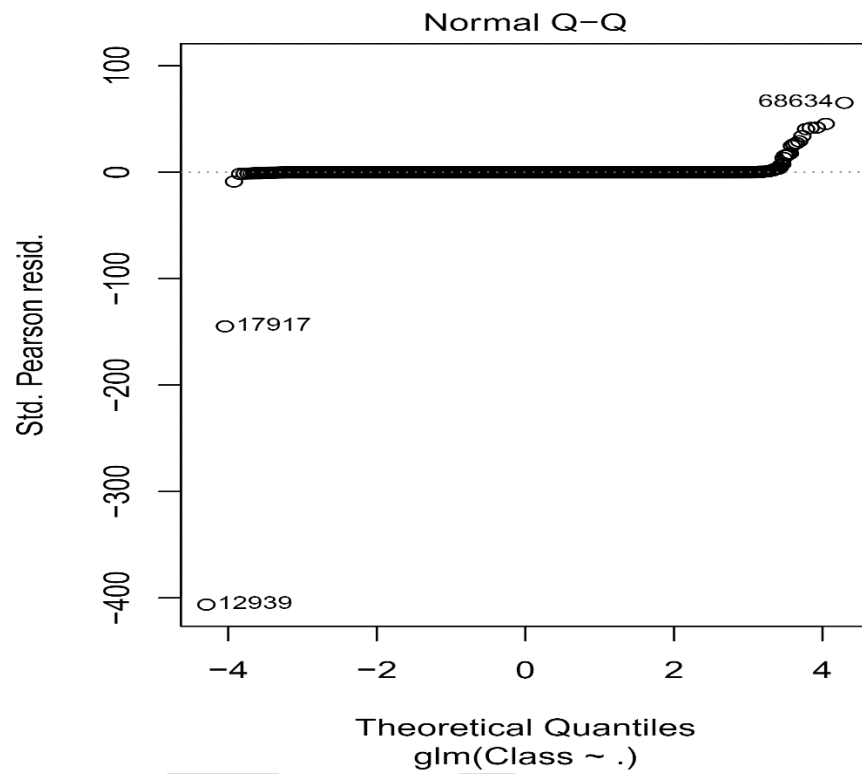
```
Logistic_Model=glm(Class~.,test_data,family=binomial())
```

```
summary(Logistic_Model)
```

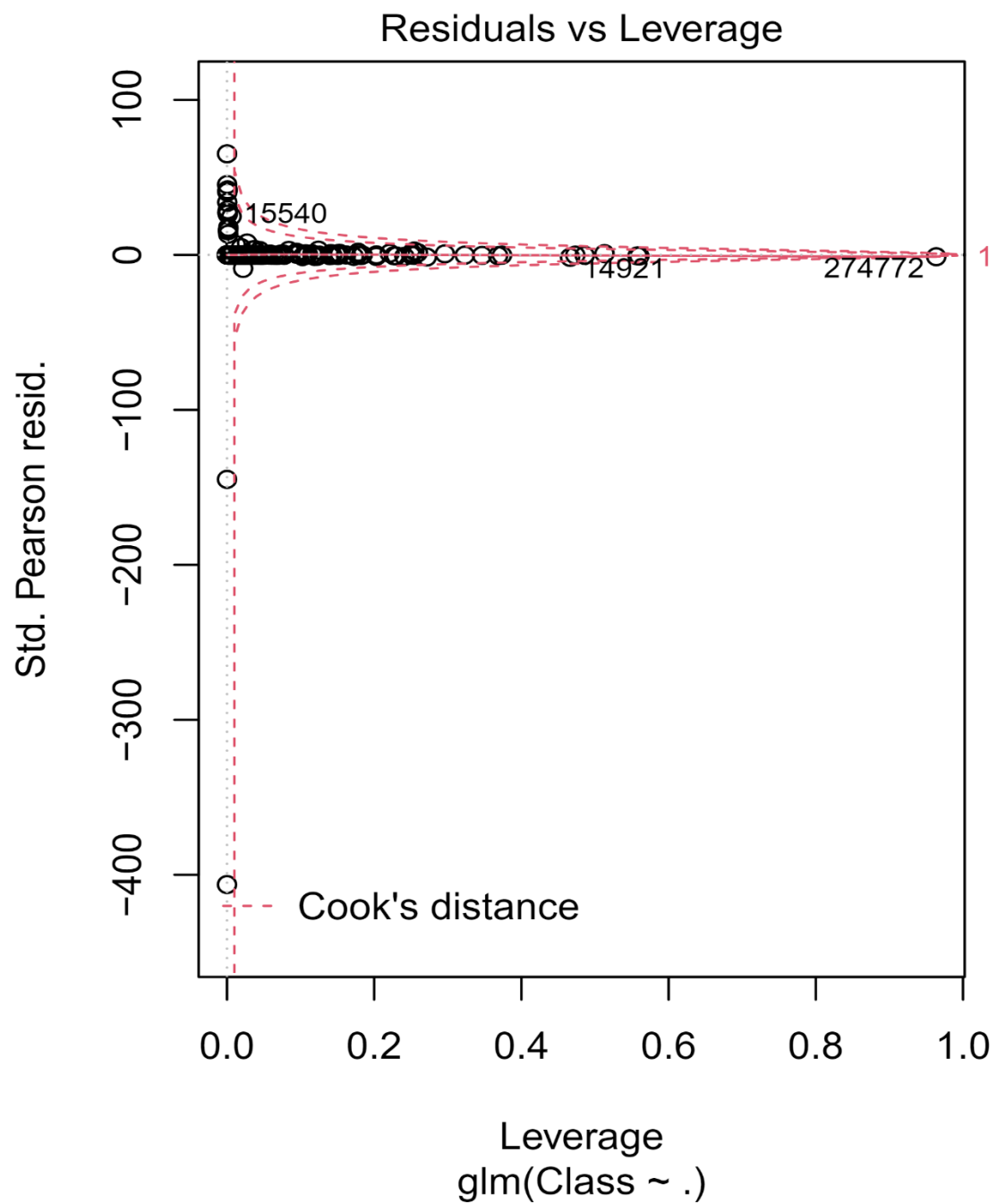
```
# Visualizing summarized model through the following plots
```

```
plot(Logistic_Model)
```







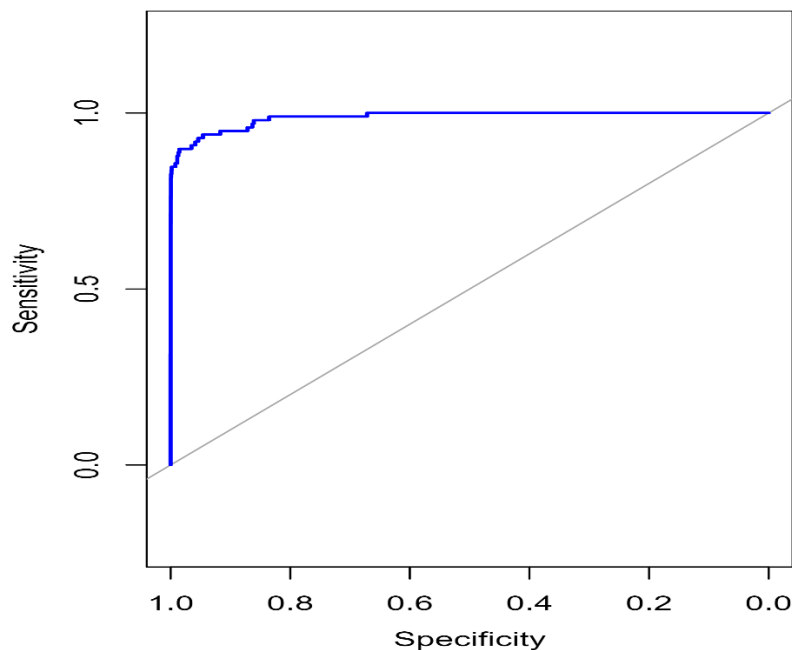


```
# ROC Curve to assess the performance of the model
```

```
library(pROC)
```

```
lr.predict <- predict(Logistic_Model, test_data, probability = TRUE)
```

```
auc.gbm = roc(test_data$Class, lr.predict, plot = TRUE, col = "blue")
```



## FITTING A DECISION TREE MODEL

Next, we implemented a decision tree algorithm to plot the outcomes of a decision through which we could conclude as to what class the object belongs to. Then we implemented the decision tree model and plotted it using the `rpart.plot()` function. we specifically used the recursive parting to plot the decision tree.

```
# Fitting a Decision Tree Model
```

```
library(rpart)
```

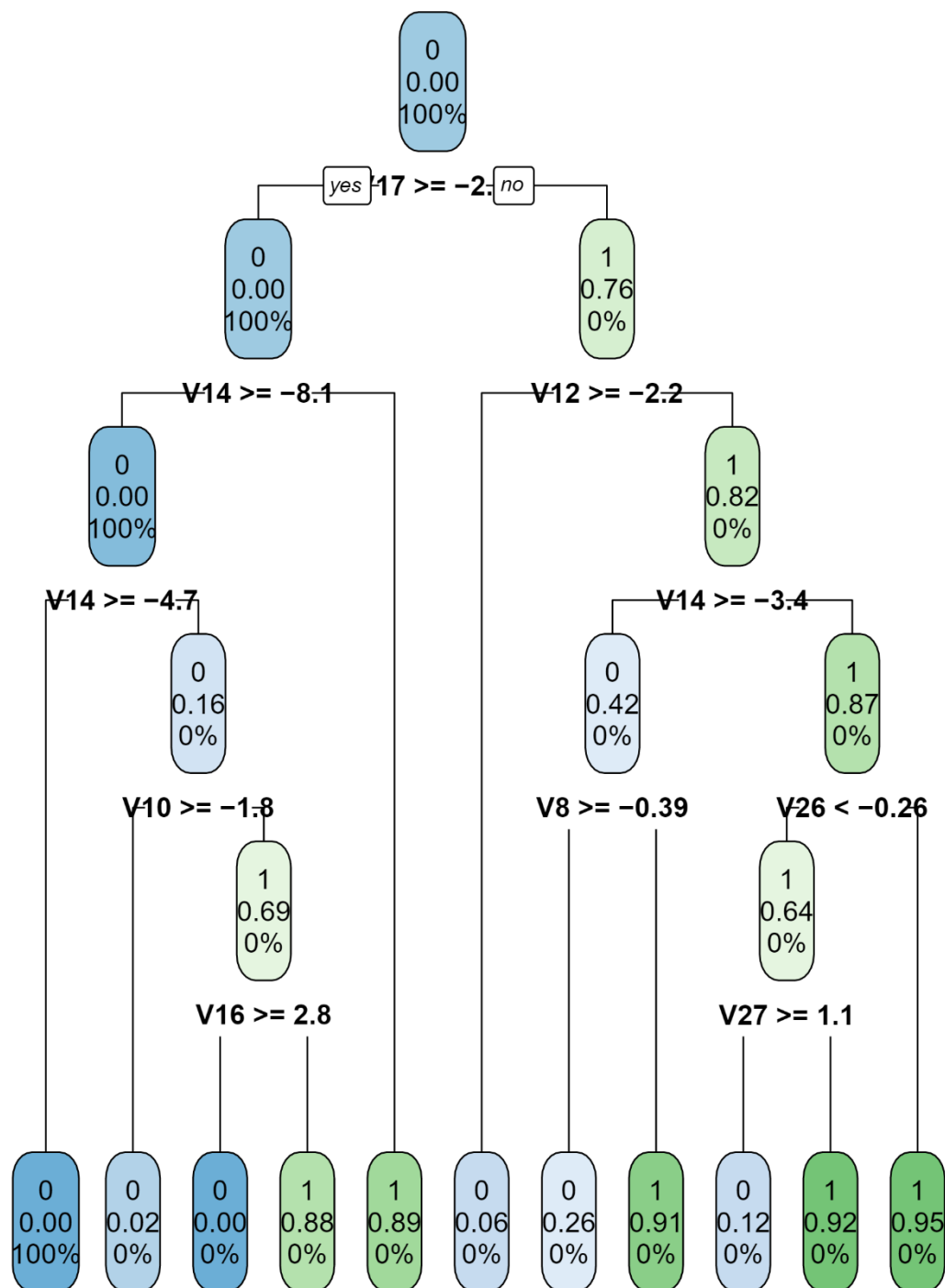
```
library(rpart.plot)
```

```
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)
```



Decision Tree Model

## 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

```
# Artificial Neural Network
```

```
library(neuralnet)
```

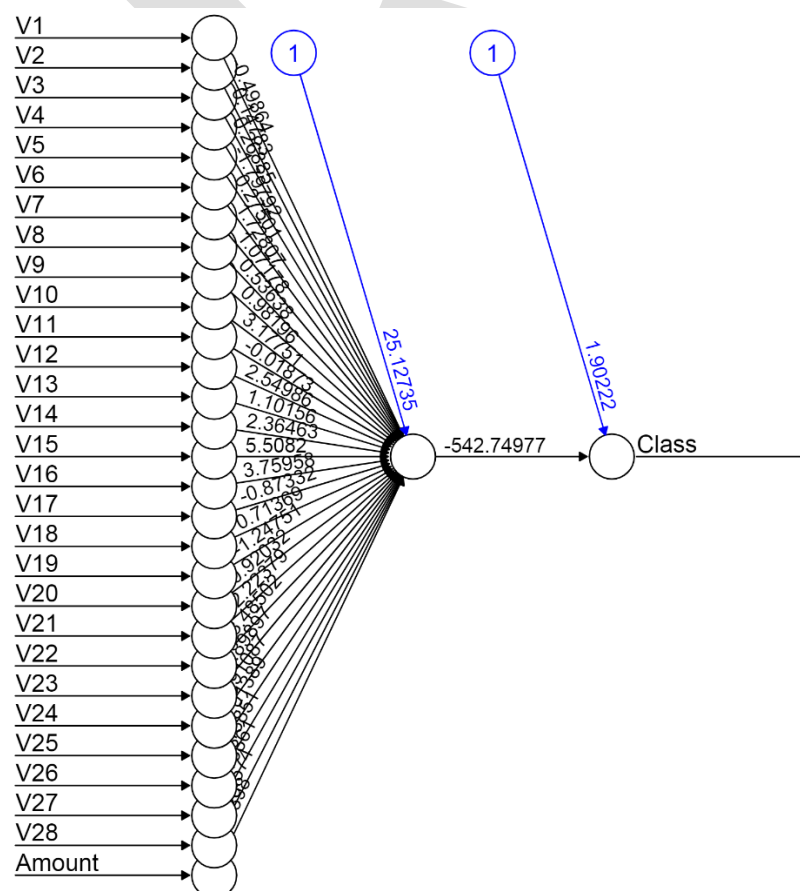
```
ANN_model =neuralnet (Class~.,train_data,linear.output=FALSE)
```

```
plot(ANN_model)
```

```
predANN=compute(ANN_model,test_data)
```

```
resultANN=predANN$net.result
```

```
resultANN=ifelse(resultANN>0.5,1,0)
```



## 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

```
# Gradient Boosting (GBM)

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))))

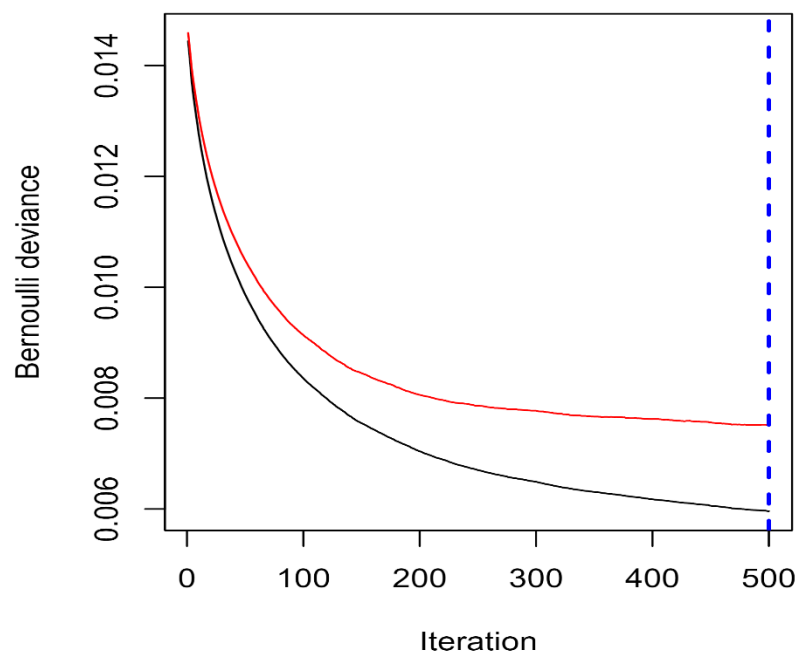
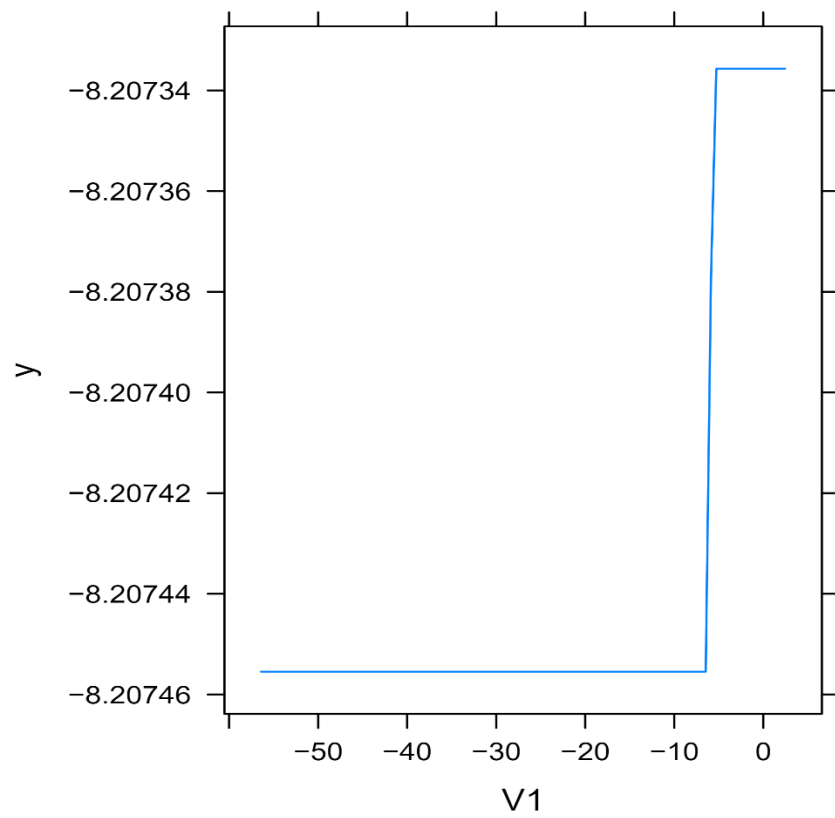
# Determine best iteration based on test data

gbm.iter = gbm.perf(model_gbm, method = "test")

model.influence = relative.influence(model_gbm, n.trees = gbm.iter, sort. = TRUE)

#Plot the gbm model

plot(model_gbm)
```



## AUC-ROC CURVE

In the last section of the project, we calculated and plotted an ROC curve measuring the sensitivity and specificity of the model. The print command plots the curve and calculates the area under the curve. The area of a ROC curve can be a test of the sensitivity and accuracy of a model.

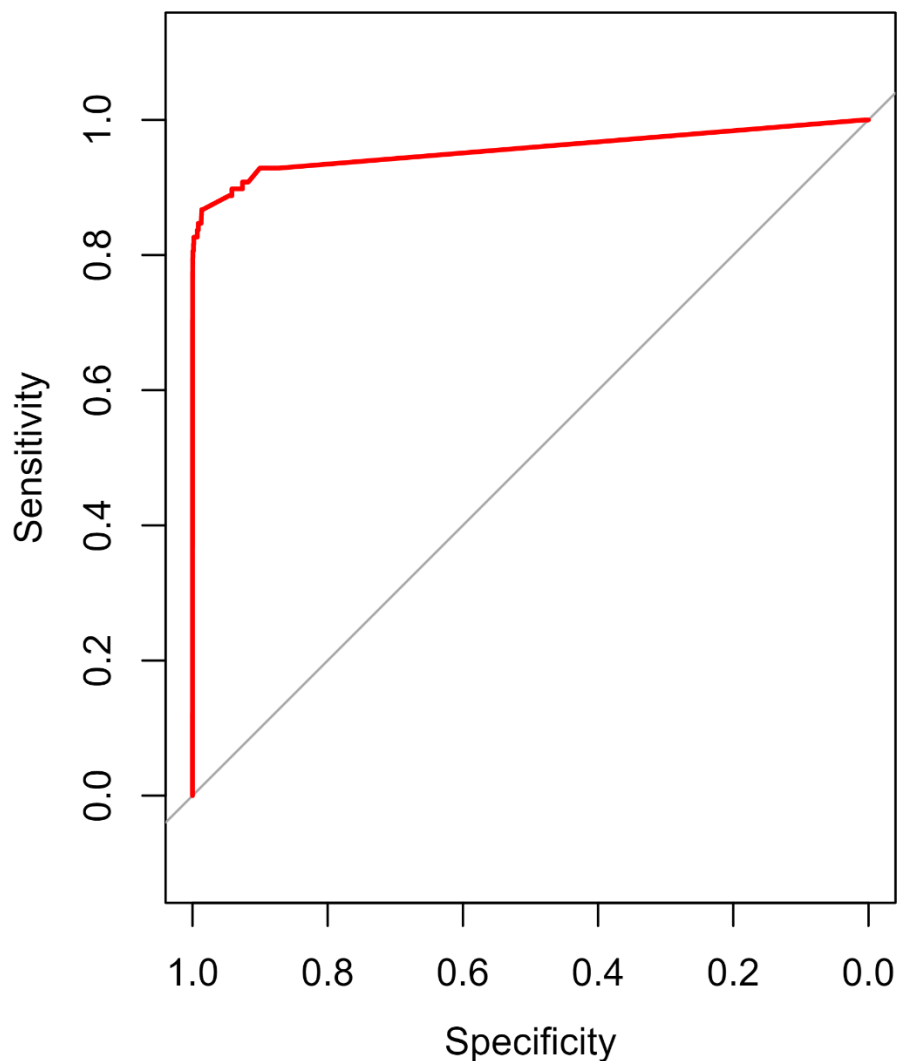
```
# Plot and calculate AUC on test data
```

```
library(pROC)
```

```
gbm_test = predict(model_gbm, newdata = test_data, n.trees = gbm.iter)
```

```
gbm_auc = roc(test_data$Class, gbm_test, plot = TRUE, col = "red")
```

```
print(gbm_auc)
```



## CONCLUSION

Concluding our R Data Science project, we learnt how to develop a 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 also learnt how data can be analyzed and visualized to discern fraudulent transactions from other types of data.

