Title: CREDIT CARD FRAUD DETECTION

1)OBJECTIVE:

In the existing situation, credit card fraud has become a significant concern for the banking sector, leading to substantial financial losses for both individuals and financial institutions. The objective of this project is to develop a brand new credit card fraud detection system using machine learning algorithms implemented in R. By leveraging advanced techniques, the project aims to improve the accuracy and efficiency of fraud detection, thereby reducing fraudulent activities and minimizing financial risks.

Existing Situation:

Credit card fraud poses a significant threat in the banking sector, with sophisticated fraudsters constantly finding new ways to exploit vulnerabilities. Traditional rule-based fraud detection systems often struggle to keep up with evolving fraud patterns, leading to high false positive rates and missed fraudulent transactions. This results in financial losses for both customers and financial institutions, as well as a loss of trust in the banking system.

What We Are Trying to Do:

This project aims to address the limitations of existing credit card fraud detection systems by developing a brand new solution using machine learning techniques in R. By harnessing the power of data analytics and advanced algorithms, we seek to achieve the following objectives:

1) Enhance Fraud Detection Accuracy: By utilizing machine learning algorithms such as logistic regression, random forest, and support vector machines, we aim to build models that can effectively distinguish between legitimate and fraudulent credit card transactions. This will significantly reduce false positives and improve the overall accuracy of fraud detection.

2) Real-Time Fraud Monitoring: Implementing real-time monitoring capabilities will enable the system to analyze credit card transactions as they occur, allowing for immediate detection of suspicious activities. This proactive approach will enable swift action to be taken, minimizing the impact of fraudulent transactions.

3) Improve Efficiency and Cost Savings: By automating the fraud detection process, the project seeks to streamline operations and reduce the manual effort required for investigating potential fraud cases. This will result in cost savings for financial institutions and enable them to allocate resources more effectively.

4) Enhance Customer Experience: By accurately detecting and preventing credit card fraud, the project aims to enhance customer trust and satisfaction. This will create a

safer environment for cardholders to conduct transactions, ultimately improving the overall customer experience within the banking sector.

By undertaking this project, we aim to revolutionize credit card fraud detection in the banking sector, making significant corrections to the existing situation. Through the application of machine learning algorithms in R, we seek to create a more robust and efficient system that will minimize financial losses, protect customer assets, and safeguard the integrity of the banking industry.

2) INTRODUCTION:

Credit card fraud is a significant concern for both credit card companies and cardholders. With the increasing reliance on electronic transactions, the risk of fraudulent activities has also risen. Detecting and preventing credit card fraud is essential to protect the financial interests of individuals and organizations.

The objective of this project is to develop a credit card fraud detection system using the R programming language. R is a powerful statistical programming language widely used for data analysis and machine learning. By leveraging its capabilities, we aim to build an effective and accurate fraud detection system.

Credit card fraud can take various forms, such as stolen card information, unauthorized transactions, or identity theft. These fraudulent activities can lead to financial losses for individuals, businesses, and financial institutions. Therefore, it is crucial to develop robust mechanisms to identify and prevent such frauds.

Machine learning techniques offer promising solutions for credit card fraud detection. By analyzing historical transaction data, patterns and anomalies can be identified, enabling the system to differentiate between legitimate and fraudulent transactions. With the help of R's extensive machine learning libraries and algorithms, we can develop a sophisticated fraud detection system.

The project will involve several stages, including data acquisition, data preparation, data preprocessing, feature engineering, model development, and evaluation. We will employ various machine learning algorithms, such as logistic regression, decision trees, random forests, or support vector machines, to build the detection system. Additionally, we will explore data visualization techniques to gain insights into the data and facilitate fraud pattern identification.

The success of this project will be measured by the system's ability to accurately detect fraudulent transactions while minimizing false positives. A well-performing fraud detection system can save significant financial resources, protect customers' interests, and enhance the overall security of credit card transactions.

In the following sections of this report, we will delve into the details of the data, methodology, implementation, evaluation, and future considerations for the credit card fraud detection project using R

3) ABSTRACT:

What is a Credit card?

A credit card is a financial tool that allows individuals to borrow money from a credit card issuer, typically a bank or a credit card company, to make purchases or payments. It is a type of revolving credit, meaning that the cardholder can borrow up to a predetermined credit limit and repay the borrowed amount over time.

What does Credit Card Fraud mean?

Credit card fraud refers to the unauthorized or fraudulent use of someone else's credit card information to make purchases or transactions without the cardholder's consent. It involves the misuse of credit card details, such as the card number, expiration date, security code, and cardholder's name, for fraudulent purposes.

Types of Credit Card Fraud:

1. **Card Skimming:** Fraudsters use skimming devices to capture credit card information when the card is swiped or inserted into compromised card readers. The stolen data is then used to create counterfeit cards or make unauthorized transactions.
2. **Phishing:** Fraudsters send deceptive emails, text messages, or make phone calls posing as legitimate institutions to trick individuals into revealing their credit card details. They may create fake websites or mimic trusted organizations to gather sensitive information.
3. **Card Not Present (CNP) Fraud:** This type of fraud occurs when credit card information is used for online or over-the-phone transactions where the physical card is not present. Fraudsters obtain card details through various means and use them to make unauthorized purchases.
4. **Identity Theft:** In cases of identity theft, fraudsters acquire personal information, including credit card details, to impersonate the cardholder. They may open new credit card accounts, apply for loans, or make fraudulent transactions using the stolen identity.
5. **Account Takeover:** Fraudsters gain unauthorized access to a legitimate cardholder's account, often through stolen login credentials or by exploiting weak security measures. They may change account details, make unauthorized transactions, or withdraw funds.

6. **Lost or Stolen Cards:** When a physical credit card is lost or stolen, unauthorized individuals may use it to make purchases before the theft is reported and the card is cancelled.
7. **Counterfeit Cards:** Fraudsters create counterfeit credit cards using stolen card information. They reproduce the card details on fake cards and use them for fraudulent transactions.

It's important for cardholders to stay vigilant, protect their card information, monitor their account activity regularly, and report any suspicious transactions or incidents of fraud to their credit card issuer promptly. Credit card companies employ security measures and fraud detection systems to detect and prevent fraudulent activities, but it is essential for cardholders to play an active role in safeguarding their card information.

Rules Based On Credit Card Fraud Detection:

1.Transaction Monitoring: Real-time monitoring of credit card transactions is crucial to detect potential fraud. Rules can be set to flag transactions that exceed certain predefined thresholds, such as unusual high-value purchases or multiple transactions within a short time period.

2.Geographic Restrictions: Rules can be established to restrict or monitor transactions originating from specific geographic regions or countries known for higher fraud rates. Unusual or unexpected transactions from these regions can be flagged for further investigation.

3.Velocity Checks: Rules can be set to monitor the velocity of transactions made by a single card within a specific time frame. Unusually high transaction frequency or a sudden increase in transaction volume can indicate fraudulent activity and trigger an alert.

4.Unusual Spending Patterns: Rules can be designed to identify and flag transactions that deviate from the cardholder's normal spending patterns. Large purchases, purchases in unusual categories, or transactions outside the cardholder's typical spending locations can be considered potential fraud indicators.

5.Card Activation and Usage Rules: Rules can be established to monitor the activation and usage patterns of newly issued credit cards. Unusual activation locations, immediate high-value transactions, or multiple activation attempts can be flagged as potential fraud.

6.Blacklist/Watchlist Checks: Rules can include checks against known fraudsters, stolen card databases, or watchlists maintained by industry organizations or law enforcement agencies. If a card or cardholder is on these lists, the transaction can be flagged as potentially fraudulent.

7.Anomaly Detection: Advanced machine learning techniques can be used to build models that learn patterns from historical data and identify anomalies in real-time transactions.

Unusual behaviour or transactions that significantly deviate from the learned patterns can be flagged for investigation.

4) METHODOLOGIES :

i) Data science approach:

1.Data Collection: Gather relevant data, including historical credit card transactions, cardholder information, transaction details (amount, location, time, etc.), and any known fraud labels or indicators.

2.Data Preprocessing: Clean and preprocess the data to handle missing values, outliers, and inconsistencies. Perform data transformations, feature engineering, and normalization to prepare the data for analysis.

3.Exploratory Data Analysis (EDA): Conduct exploratory analysis to gain insights into the data. Visualize transaction patterns, identify trends, and understand the characteristics of fraudulent transactions compared to legitimate ones.

4.Feature Selection: Select the most relevant features that contribute to fraud detection. This can be done using techniques like correlation analysis, feature importance, or dimensionality reduction methods.

5.Model Development: Build machine learning models using various algorithms such as logistic regression, decision trees, random forests, gradient boosting, or neural networks. Train the models using the labelled dataset, where fraudulent and non-fraudulent transactions are labelled accordingly.

6.Model Evaluation: Evaluate the performance of the models using appropriate metrics such as accuracy, precision, recall, F1-score, or area under the receiver operating characteristic (ROC) curve. Utilize techniques like cross-validation or holdout validation to ensure robustness.

7.Model Maintenance and Updates: Regularly update the models with new data and adapt them to changing fraud patterns. Monitor the model's performance over time and make necessary adjustments to ensure its effectiveness.

ii) Software Analysis:

What is R language?

R is a statistical language created by statisticians. Thus, it excels in statistical computation. R is the most used programming language for developing statistical tools. R is a statistical package and mathematical programming language. Unlike Stata, SAS, SPSS, MATLAB, and

other statistical packages, it is totally open source. Students can easily install on their own computers and use them after they graduate. Unlike Excel, in R you can easily write scripts that make your analysis reproducible. RStudio is an integrated development environment (IDE) for R. You still need to install R to use RStudio, but it is a more helpful, graphical environment for working with R. Windows for scripts, files, packages, and plots, in addition to the console, make it easier to keep track of what you are doing. It has many add-ins to make R more powerful. RStudio is especially good for making reports and presentations with the packages.

Techniques of R use in this project:

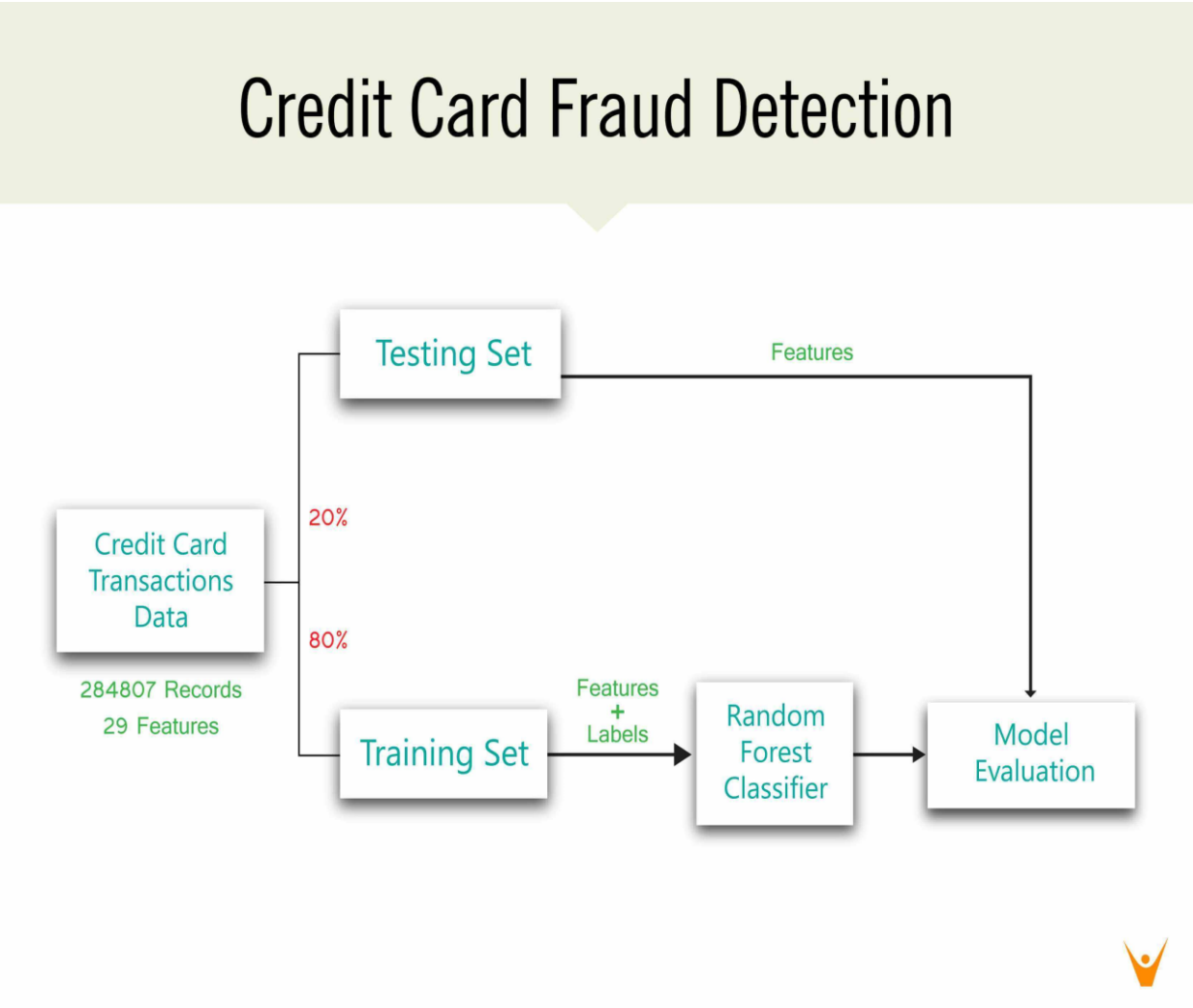
In a credit card fraud detection project using the R language, various techniques can be applied at different stages of the project. Here's an overview of how these techniques can be used:

1. **Data Loading:** The first step is to load the credit card transaction data into R. This can be done using functions like **read.csv()** or specific data-loading functions provided by packages like **readr** or **data.table**.
2. **Data Exploration**: Once the data is loaded, exploratory data analysis techniques can be applied to gain insights into the dataset. This involves summarizing and visualizing the data using functions like **summary()**, **head()**, **tail()**, and various plotting functions from packages like **ggplot2**. Data exploration helps understand the distribution of variables, identify missing values or outliers, and detect any patterns or anomalies in the data.
3. **Data Manipulation:** R provides powerful packages like **dplyr** and **tidyr** for data manipulation tasks. These packages offer functions for filtering, transforming, aggregating, and reshaping data. Techniques like cleaning missing values, handling outliers, and creating derived variables can be applied using these packages to prepare the data for modeling.
4. **Data Modelling:** Several modeling techniques can be utilized for credit card fraud detection in R. Here are a few commonly used ones:
    - **Logistic Regression:** Logistic regression is a popular technique for binary classification problems. R provides functions like **glm()** to fit logistic regression models to the data. The model can be trained on historical transaction data, with the target variable indicating fraudulent or non-fraudulent transactions.
    - **Decision Trees:** Decision trees are intuitive models that can capture non-linear relationships in the data. Packages like **rpart** or **party** can be used to build decision tree models in R. These models can be trained on transaction data, considering various features as input, and the target variable as the outcome.
    - **Artificial Neural Networks:** Artificial Neural Networks (ANNs) are powerful models for complex patterns and relationships in data. R offers packages like **neuralnet**, **keras**, or **tensorflow** that enable the construction and training of ANN models for credit card fraud detection. ANNs can learn from historical transaction data to predict the likelihood of fraud.

- **Gradient Boosting:** Gradient boosting algorithms like XGBoost or LightGBM can be employed for credit card fraud detection. These algorithms build an ensemble of weak models (e.g., decision trees) iteratively, focusing on misclassified instances. R packages like **xgboost** or **lightgbm** provide functions for training and evaluating gradient boosting models.

5. **Model Evaluation:** After building the models, they need to be evaluated to assess their performance. Techniques like cross-validation, precision, recall, F1-score, ROC curve analysis, or confusion matrix can be applied to evaluate the models' accuracy and effectiveness in detecting fraud. R packages like **caret**, **pROC**, or **yardstick** can be used for model evaluation.
6. **Model Selection and Deployment:** Based on the model evaluation results, the best-performing model can be selected. The selected model can then be deployed for real-time scoring of new credit card transactions. R packages like **plumber** or **APItools** can help in creating APIs that accept transaction data and provide fraud predictions based on the trained models.

5) FLOWCHART:



6) LIBRARIES USED:

- *Ranger*
- *caret*
- *dat.table*
- *readxl*
- *carTools*
- *pROC*
- *rpart*
- *rpart.plot*
- *neuralnet*
- *qbm*

7)DETAILED ANALYSIS:

i)Data Description:

The credit card fraud detection project utilizes a dataset that consists of credit card transaction records. The dataset contains relevant information about each transaction, allowing us to analyze and detect fraudulent activities. Here is a description of the data attributes:
Time: This attribute represents the number of seconds elapsed between the current transaction and the first transaction in the dataset. It provides a temporal dimension to the transactions.

V1, V2, ..., V28: These attributes are anonymized numerical values resulting from a dimensionality reduction process, such as Principal Component Analysis (PCA). They represent transformed features of the original transaction data to maintain privacy.

Amount: This attribute denotes the transaction amount, providing information about the monetary value involved in each transaction.

Class: This is the binary class label that indicates whether a transaction is fraudulent (Class = 1) or legitimate (Class = 0).

The dataset contains a combination of fraudulent and legitimate transactions, allowing us to train and evaluate our fraud detection system effectively. The anonymization of features ensures the privacy and confidentiality of the original data while still enabling us to extract meaningful insights.

It is essential to note that the dataset may be imbalanced, meaning the number of legitimate transactions might significantly outnumber the fraudulent ones. Imbalanced data poses challenges during model training and evaluation, requiring appropriate techniques to handle class imbalance issues.

About Data:

Detecting Credit Card Fraud

Importing Datasets

library(ranger)

> library(ranger)

#The "ranger" library in R is a package that offers a fast implementation of random forests for classification and regression tasks. It provides efficient and scalable algorithms for building robust predictive models on large datasets.

> library(caret)

#By using the "library(caret)" command, the "caret" package is loaded, which provides a unified interface for performing various machine learning tasks, such as data preprocessing, model training, and evaluation, in R.

Loading required package: ggplot2

Loading required package: lattice

> library(data.table)

#The "data.table" library in R is designed to efficiently handle large datasets by providing high-performance data manipulation operations. It extends the functionality of data frames and offers optimised operations for tasks such as subsetting, grouping, and merging data tables.

>

> library(readxl)

#The "readxl" library in R is used for reading Microsoft Excel files (.xls and .xlsx) into R as data frames, making it easy to work with Excel data. It provides functions to import specific sheets, specify ranges, and handle various data types, offering flexibility in accessing and analysing Excel data within R.

ii)Data Loading:

Now we will load our data:

> creditcard_data <- read_excel("C:\\Users\\ankita\\Desktop\\data_set2.xlsx")-

> print(creditcard_data)

A tibble: 284,807 × 31

Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14

<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 0 -1.36 -0.0728 2.54 1.38 -0.338 0.462 0.240 0.0987 0.364 0.0908 -0.552 -0.618 -0.991 -0.311

2 0 1.19 0.266 0.166 0.448 0.0600 -0.0824 -0.0788 0.0851 -0.255 -0.167 1.61 1.07 0.489 -0.144

3 1 -1.36 -1.34 1.77 0.380 -0.503 1.80 0.791 0.248 -1.51 0.208 0.625 0.0661 0.717 -0.166

4 1 -0.966 -0.185 1.79 -0.863 -0.0103 1.25 0.238 0.377 -1.39 -0.0550 -0.226 0.178 0.508 -0.288

5 2 -1.16 0.878 1.55 0.403 -0.407 0.0959 0.593 -0.271 0.818 0.753 -0.823 0.538 1.35 -1.12

6 2 -0.426 0.961 1.14 -0.168 0.421 -0.0297 0.476 0.260 -0.569 -0.371 1.34 0.360 -0.358 -0.137

7 4 1.23 0.141 0.0454 1.20 0.192 0.273 -0.00516 0.0812 0.465 -0.0993 -1.42 -0.154 -0.751 0.167

8 7 -0.644 1.42 1.07 -0.492 0.949 0.428 1.12 -3.81 0.615 1.25 -0.619 0.291 1.76 -1.32

9 7 -0.894 0.286 -0.113 -0.272 2.67 3.72 0.370 0.851 -0.392 -0.410 -0.705 -0.110 -0.286 0.0744

10 9 -0.338 1.12 1.04 -0.222 0.499 -0.247 0.652 0.0695 -0.737 -0.367 1.02 0.836 1.01 -0.444

ℹ 284,797 more rows

ℹ 16 more variables: V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>, V22 <dbl>,

V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>, Class <dbl>

ℹ Use `print(n = ...)` to see more rows

>

iii)Data Exploration:

> is.null(creditcard_data)

[1] FALSE

>

> dim(creditcard_data)

[1] 284807 31

>

> head(creditcard_data)

A tibble: 6 × 31

Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14

<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 0 -1.36 -0.0728 2.54 1.38 -0.338 0.462 0.240 0.0987 0.364 0.0908 -0.552 -0.618 -0.991 -0.311

2 0 1.19 0.266 0.166 0.448 0.0600 -0.0824 -0.0788 0.0851 -0.255 -0.167 1.61 1.07 0.489 -0.144

3 1 -1.36 -1.34 1.77 0.380 -0.503 1.80 0.791 0.248 -1.51 0.208 0.625 0.0661 0.717 -0.166

4 1 -0.966 -0.185 1.79 -0.863 -0.0103 1.25 0.238 0.377 -1.39 -0.0550 -0.226 0.178 0.508 -0.288

5 2 -1.16 0.878 1.55 0.403 -0.407 0.0959 0.593 -0.271 0.818 0.753 -0.823 0.538 1.35 -1.12

6 2 -0.426 0.961 1.14 -0.168 0.421 -0.0297 0.476 0.260 -0.569 -0.371 1.34 0.360 -0.358 -0.137

ℹ 16 more variables: V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>, V22 <dbl>,

V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>, Class <dbl>

>

> tail(creditcard_data)

A tibble: 6 × 31

Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14

<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 172785 0.120 0.931 -0.546 -0.745 1.13 -0.236 0.813 0.115 -0.204 -0.657 0.645 0.191 -0.546 -0.732

2 172786 -11.9 10.1 -9.83 -2.07 -5.36 -2.61 -4.92 7.31 1.91 4.36 -1.59 2.71 -0.689 4.63

3 172787 -0.733 -0.0551 2.04 -0.739 0.868 1.06 0.0243 0.295 0.585 -0.976 -0.150 0.916 1.21 -0.675

4 172788 1.92 -0.301 -3.25 -0.558 2.63 3.03 -0.297 0.708 0.432 -0.485 0.412 0.0631 -0.184 -0.511

5 172788 -0.240 0.530 0.703 0.690 -0.378 0.624 -0.686 0.679 0.392 -0.399 -1.93 -0.963 -1.04 0.450

6 172792 -0.533 -0.190 0.703 -0.506 -0.0125 -0.650 1.58 -0.415 0.486 -0.915 -1.04 -0.0315 -0.188 -0.0843

ℹ 16 more variables: V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>, V22 <dbl>,

V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>, Class <dbl>

> 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" "V8" "V9" "V10" "V11"

[13] "V12" "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23"

[25] "V24" "V25" "V26" "V27" "V28" "Amount" "Class"

>

> var(creditcard_data$Amount)

[1] 62560.07

iv) Data Manipulation:

Here we will scale our data using scale().

We will apply this to amount component of our creditcard_amount.Scaling is also known as feature standardization. With the help of scaling, data is distrituted according to specified range. Therefore, there are no extreme values in our dataset that interfere with the functioning of our model. We will carry this out as follow:

>

> head(creditcard_data)

A tibble: 6 × 31

Time V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14

<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 0 -1.36 -0.0728 2.54 1.38 -0.338 0.462 0.240 0.0987 0.364 0.0908 -0.552 -0.618 -0.991 -0.311

2 0 1.19 0.266 0.166 0.448 0.0600 -0.0824 -0.0788 0.0851 -0.255 -0.167 1.61 1.07 0.489 -0.144

3 1 -1.36 -1.34 1.77 0.380 -0.503 1.80 0.791 0.248 -1.51 0.208 0.625 0.0661 0.717 -0.166

4 1 -0.966 -0.185 1.79 -0.863 -0.0103 1.25 0.238 0.377 -1.39 -0.0550 -0.226 0.178 0.508 -0.288

5 2 -1.16 0.878 1.55 0.403 -0.407 0.0959 0.593 -0.271 0.818 0.753 -0.823 0.538 1.35 -1.12

6 2 -0.426 0.961 1.14 -0.168 0.421 -0.0297 0.476 0.260 -0.569 -0.371 1.34 0.360 -0.358 -0.137

i 16 more variables: V15 <dbl>, V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>, V22 <dbl>,

V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>, Class <dbl>

>

> creditcard_data$Amount=scale(creditcard_data$Amount)

> NewData=creditcard_data[,-c(1)]

> head(NewData)

A tibble: 6 × 30

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15

<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

1 -1.36 -0.0728 2.54 1.38 -0.338 0.462 0.240 0.0987 0.364 0.0908 -0.552 -0.618 -0.991 -0.311 1.47

2 1.19 0.266 0.166 0.448 0.0600 -0.0824 -0.0788 0.0851 -0.255 -0.167 1.61 1.07 0.489 -0.144 0.636

3 -1.36 -1.34 1.77 0.380 -0.503 1.80 0.791 0.248 -1.51 0.208 0.625 0.0661 0.717 -0.166 2.35

4 -0.966 -0.185 1.79 -0.863 -0.0103 1.25 0.238 0.377 -1.39 -0.0550 -0.226 0.178 0.508 -0.288 -0.631

5 -1.16 0.878 1.55 0.403 -0.407 0.0959 0.593 -0.271 0.818 0.753 -0.823 0.538 1.35 -1.12 0.175

6 -0.426 0.961 1.14 -0.168 0.421 -0.0297 0.476 0.260 -0.569 -0.371 1.34 0.360 -0.358 -0.137 0.518

ℹ 15 more variables: V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>, V22 <dbl>, V23 <dbl>,

V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl[,1]>, Class <dbl>

>

v)Data Modelling:

After we have standardized our entire dataset, we will split our dataset into training set and testing set.

The spilt ratio will be 80% Training data and 20% Testing data.

```
> 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)
>
> is.null(train_data)
[1] FALSE
```

>

> is.null(test_data)

[1] FALSE

>

> dim(train_data)

[1] 227846 30

>

> dim(test_data)

[1] 56961 30

>

vi) Data Prediction:

Using Logistic Regression for Prediction:

We will use our first model as Logistic Regression.

Logistic regression is used for modelling the outcome probability of a class such as pass/fail, positive/negative and in our case – fraud/not fraud.

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

Warning message:

glm.fit: fitted probabilities numerically 0 or 1 occurred

> save.image("C:\\Users\\ankita\\Desktop\\ankita_workspace")

>

> summary(Logistic_Model)

Call:

glm(formula = Class ~ ., family = binomial(), data = test_data)

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-12.52800	10.30537	-1.216	0.2241

V1	-0.17299	1.27381	-0.136	0.8920
V2	1.44512	4.23062	0.342	0.7327
V3	0.17897	0.24058	0.744	0.4569
V4	3.13593	7.17768	0.437	0.6622
V5	1.49014	3.80369	0.392	0.6952
V6	-0.12428	0.22202	-0.560	0.5756
V7	1.40903	4.22644	0.333	0.7388
V8	-0.35254	0.17462	-2.019	0.0435 *
V9	3.02176	8.67262	0.348	0.7275
V10	-2.89571	6.62383	-0.437	0.6620
V11	-0.09769	0.28270	-0.346	0.7297
V12	1.97992	6.56699	0.301	0.7630
V13	-0.71674	1.25649	-0.570	0.5684
V14	0.19316	3.28868	0.059	0.9532
V15	1.03868	2.89256	0.359	0.7195
V16	-2.98194	7.11391	-0.419	0.6751
V17	-1.81809	4.99764	-0.364	0.7160
V18	2.74772	8.13188	0.338	0.7354
V19	-1.63246	4.77228	-0.342	0.7323
V20	-0.69925	1.15114	-0.607	0.5436
V21	-0.45082	1.99182	-0.226	0.8209
V22	-1.40395	5.18980	-0.271	0.7868
V23	0.19026	0.61195	0.311	0.7559
V24	-0.12889	0.44701	-0.288	0.7731
V25	-0.57835	1.94988	-0.297	0.7668
V26	2.65938	9.34957	0.284	0.7761
V27	-0.45396	0.81502	-0.557	0.5775
V28	-0.06639	0.35730	-0.186	0.8526
Amount	0.22576	0.71892	0.314	0.7535

Signif. codes: 0 ‘*’ 0.001 ‘*’ 0.01 ‘’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1443.40 on 56960 degrees of freedom
Residual deviance: 378.59 on 56931 degrees of freedom
AIC: 438.59

Number of Fisher Scoring iterations: 17

- After we have summarised our model, we will visual it through the following plots – plot(Logistic_Model)

Warning messages:

1: In sqrt(crit * p * (1 - hh)/hh) : NaNs produced

2: In sqrt(crit * p * (1 - hh)/hh) : NaNs produced

>

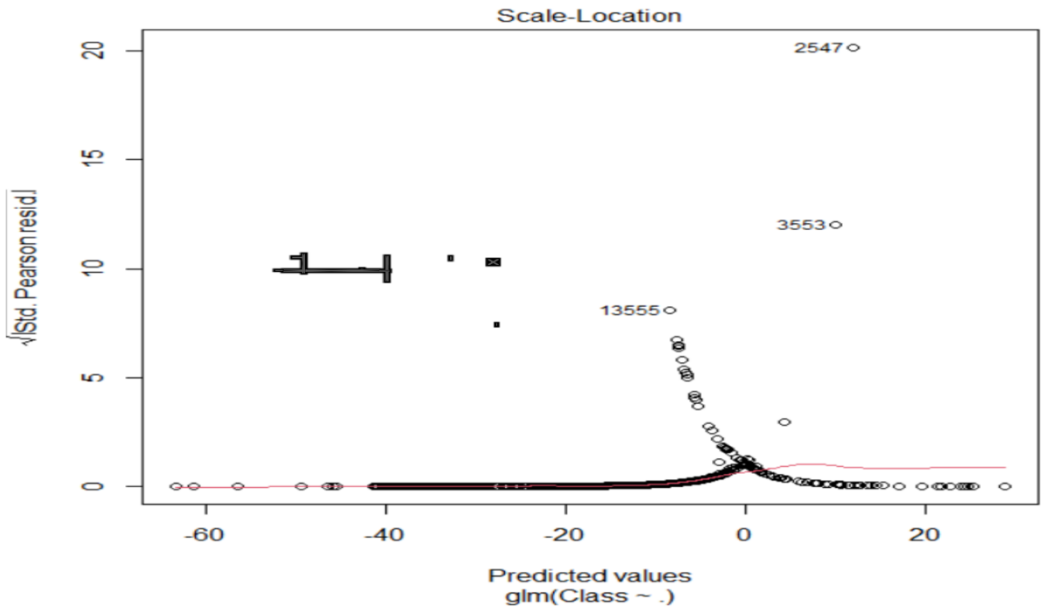
> plot(Logistic_Model)

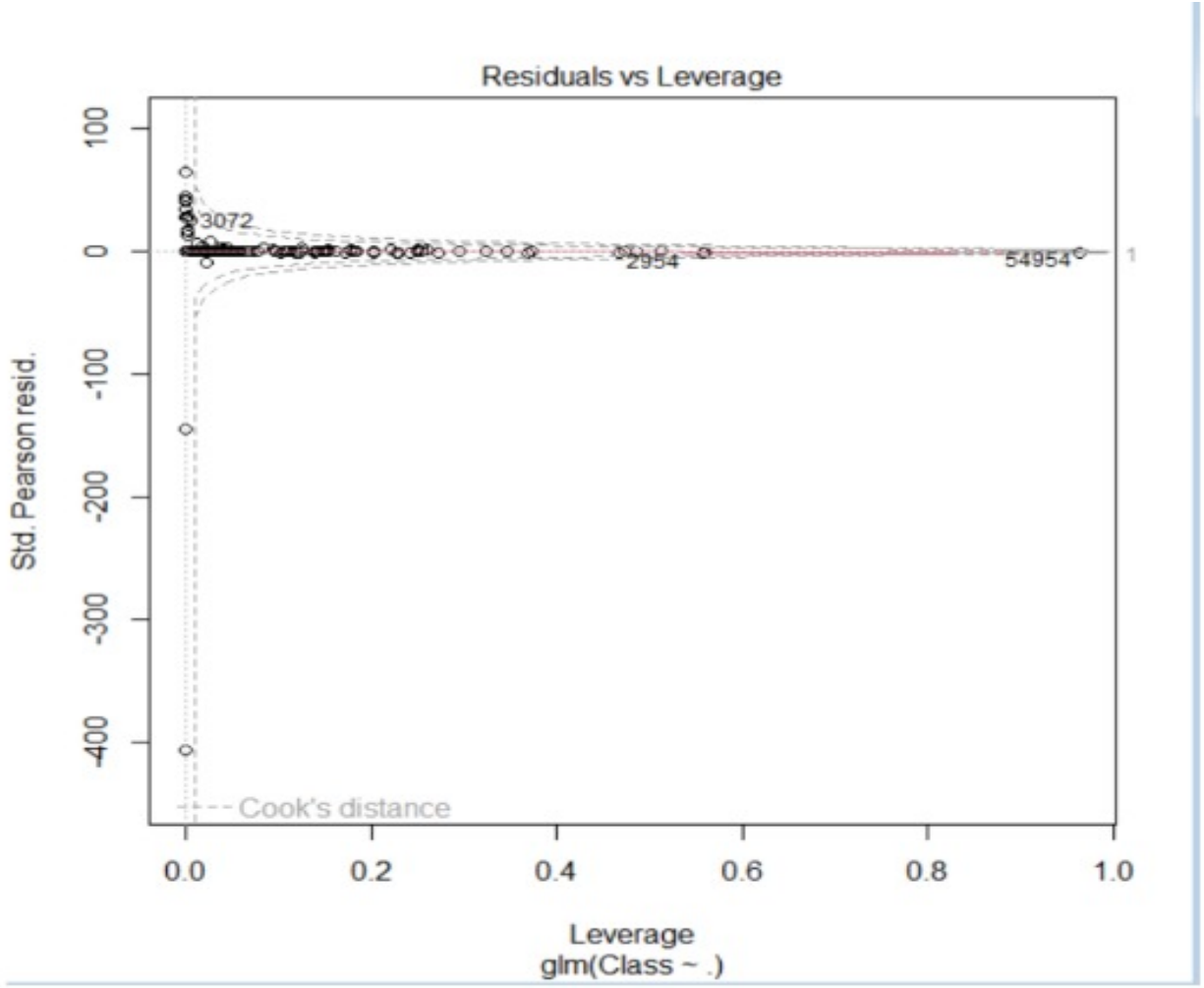
Waiting to confirm page change...

Waiting to confirm page change...

Waiting to confirm page change...

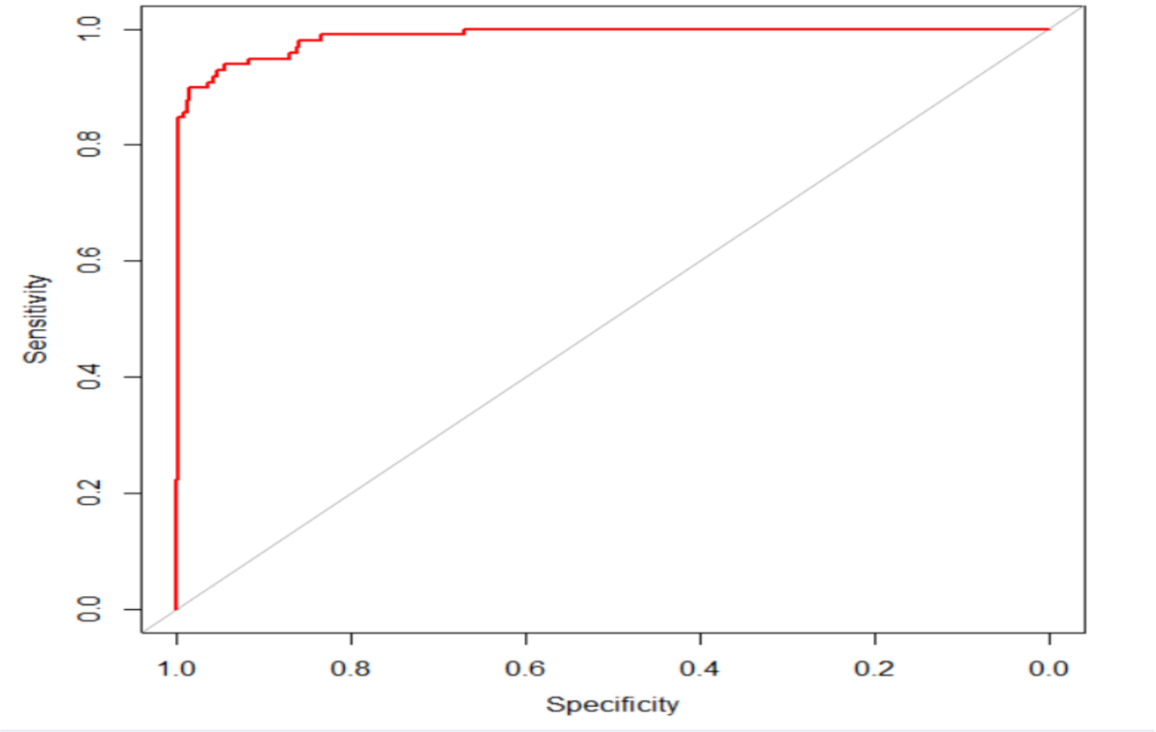
Waiting to confirm page change...


Scale-Location
√|Std. Pearson resid.|
2547
3553
13555
0
5
10
15
20
-60
-40
-20
0
20
Predicted values
glm(Class ~ .)


Residuals vs Leverage
Std. Pearson resid.
3072
2954
54954
1
100
0
-100
-200
-300
-400
Cook's distance
0.0
0.2
0.4
0.6
0.8
1.0
Leverage
glm(Class ~ .)

- 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.

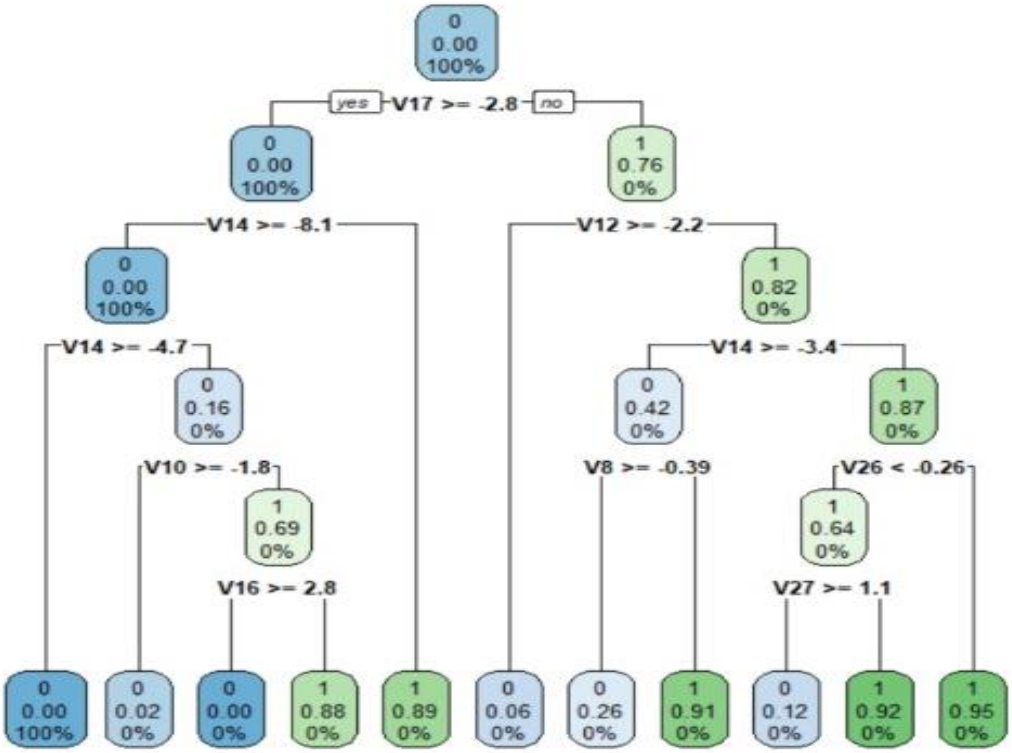
```
> library(pROC)”


Type 'citation("pROC")' for a citation.


Attaching package: ‘pROC’


The following objects are masked from ‘package:stats’:

    cov, smooth, var

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

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

Setting levels: control = 0, case = 1

Setting direction: controls < cases

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

Setting levels: control = 0, case = 1

Setting direction: controls < cases
```



vii)Fitting Decision Tree:

```
>
> 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)
>
```

>



viii)Artificial Neural Network:

Artificial Neural Networks are a type of machine learning algorithm that are modeled after the human nervous system.

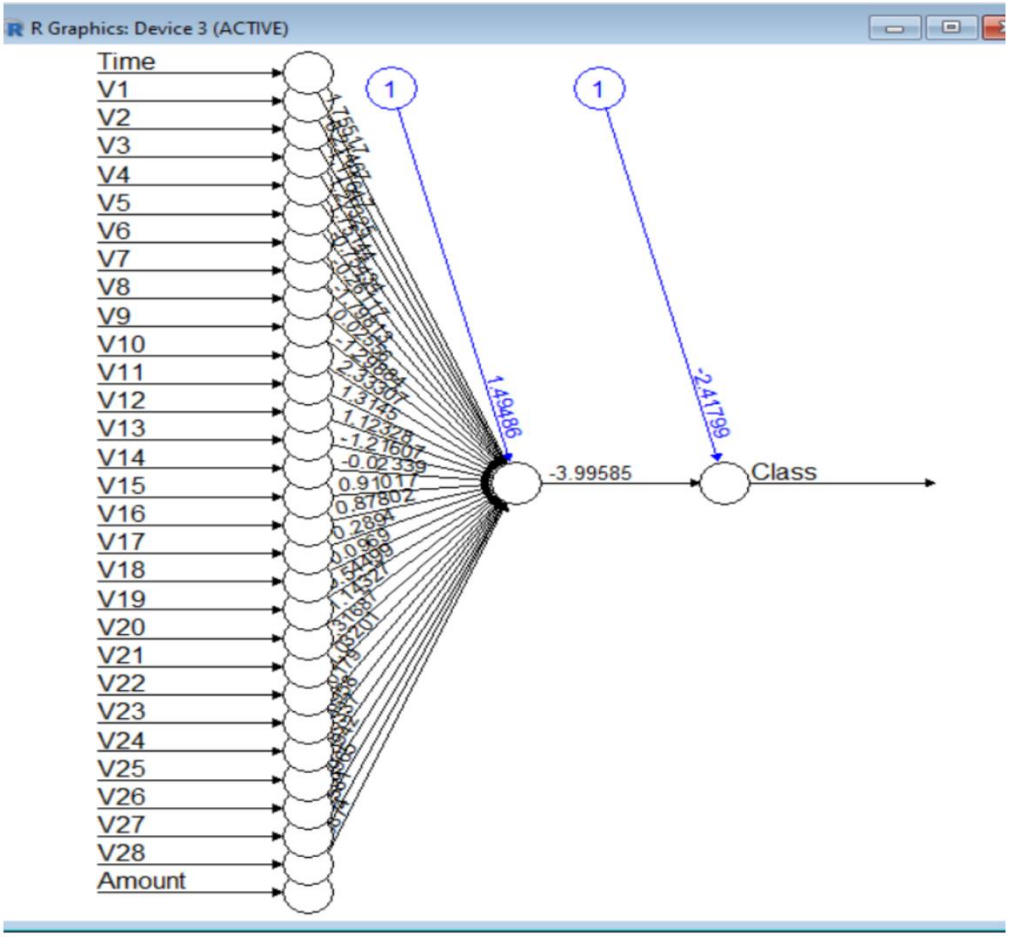
The ANN models are able to learn the patterns using the historical data and are able to perform classification on the input data.

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.

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

```
Error in neuralnet(Class ~ ., train_data, linear.output = FALSE) :
  could not find function "neuralnet"
> ANN_model=neuralnet(Class~.,train_data,linear.output=FALSE)
Error in neuralnet(Class ~ ., train_data, linear.output = FALSE) :
  could not find function "neuralnet"
> ANN_model=neuralnet(Class~.,train_data,linear.output=FALSE)
Error in neuralnet(Class ~ ., train_data, linear.output = FALSE) :
  could not find function "neuralnet"
> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.icts.res.in/bin/windows/contrib/4.3/neuralnet_1.44.2.zip'
Content type 'application/zip' length 123780 bytes (120 KB)
downloaded 120 KB

package ‘neuralnet’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\ankita\AppData\Local\Temp\RtmponVkpV\downloaded_packages
> ANN_model=neuralnet(Class~.,train_data,linear.output=FALSE)
Error in neuralnet(Class ~ ., train_data, linear.output = FALSE) :
  could not find function "neuralnet"
> plot.nnet(ANN_model)
Error in plot.nnet(ANN_model) : could not find function "plot.nnet"
> formula <- Class ~ .
> ANN_model <- neuralnet(formula, data = train_data, linear.output = FALSE)
Error in neuralnet(formula, data = train_data, linear.output = FALSE) :
  could not find function "neuralnet"
> plot(ANN_model)
>
```



> library(gbm, quietly=TRUE)

Loaded gbm 2.1.8.1

>

>

ix)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.

> library(gbm, quietly=TRUE)

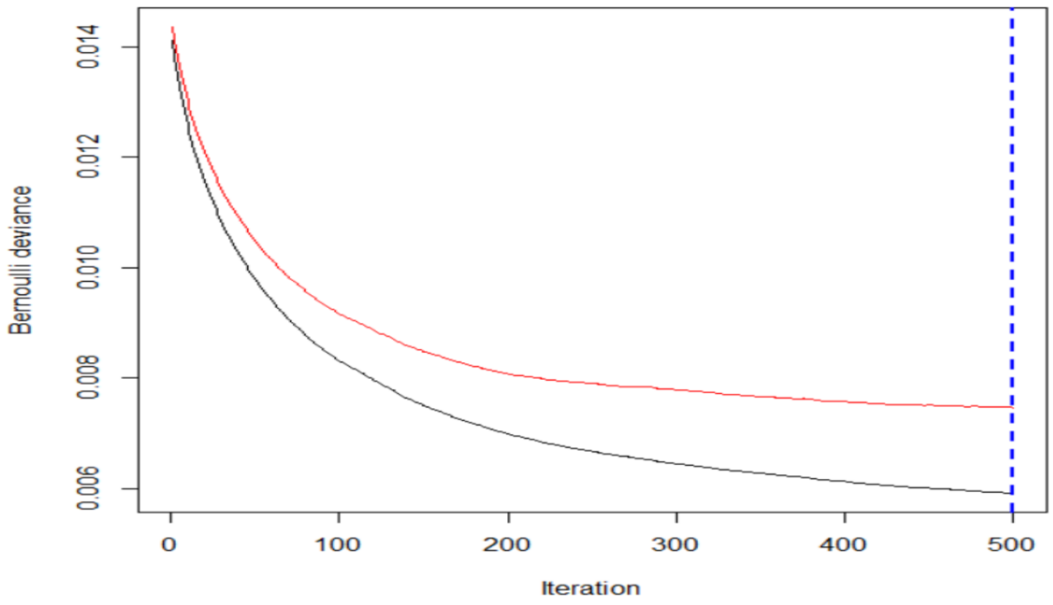
 # Get the time to train the GBM model

> 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))
+ )
+ )
   user  system elapsed
177.32    0.69  240.15
>
>
>
>
```

 # Determine best iteration based on test data

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

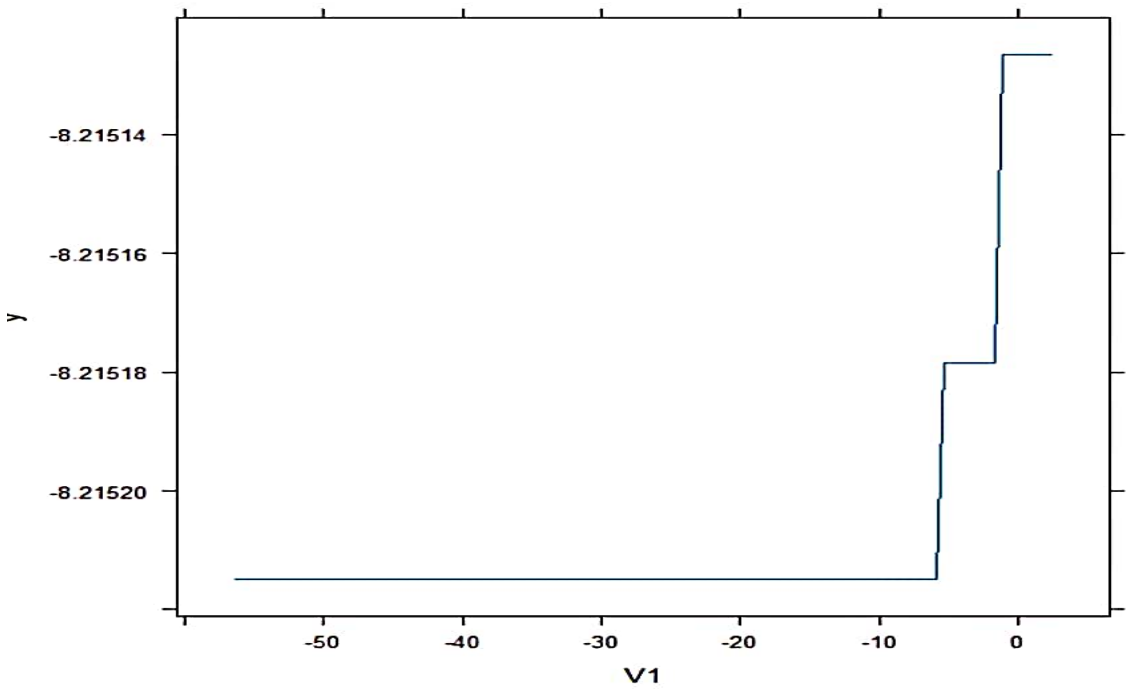


> model.influence = relative.influence(model_gbm, n.trees = gbm.iter, sort. = TRUE)

#Plot the gbm model

> #Plot the gbm model

> plot(model_gbm)



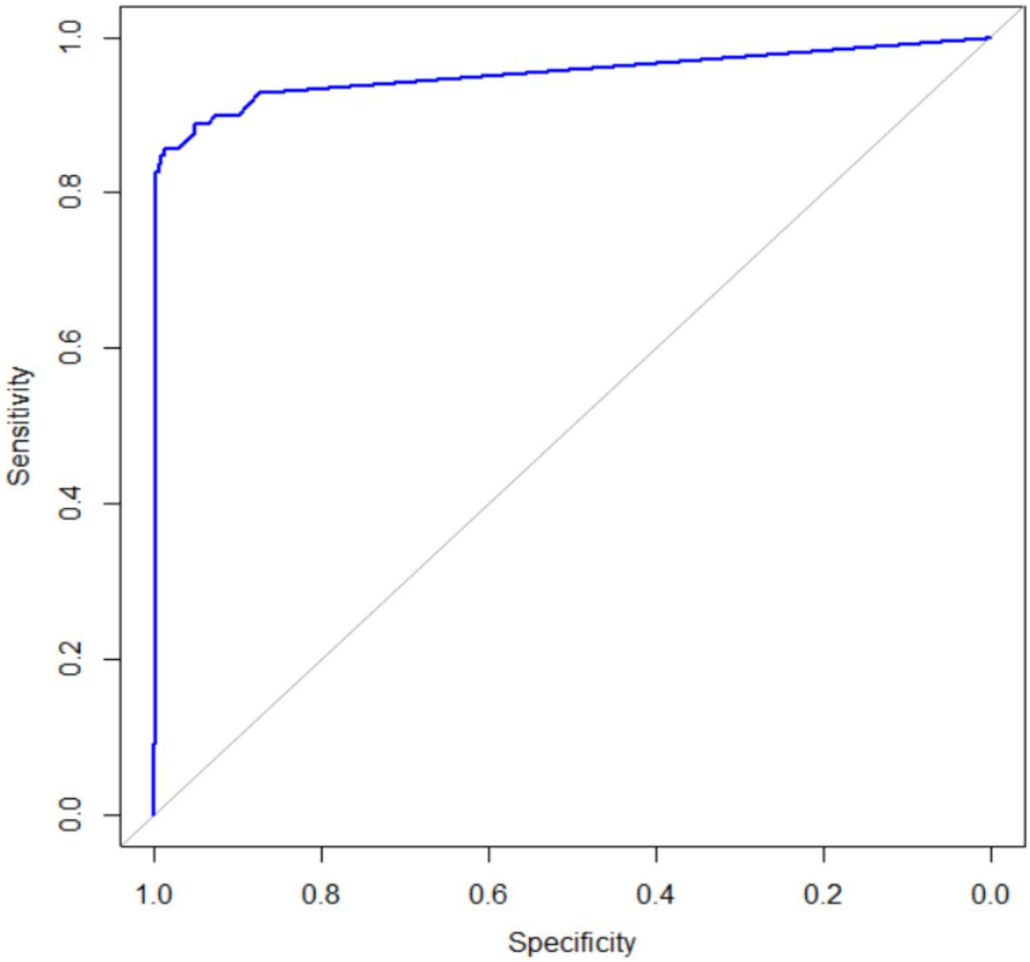
> gbm_test = predict(model_gbm, newdata = test_data, n.trees = gbm.iter)

> gbm_auc = roc(test_data$Class, gbm_test, plot = TRUE, col = "blue")

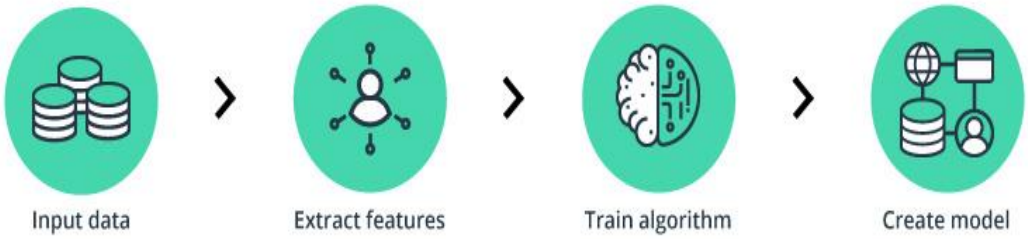
Setting levels: control = 0, case = 1

Setting direction: controls < cases

>



8) CONCLUSION:



In conclusion, the Credit Card Fraud Detection project, implemented using R tools and techniques, has provided an effective solution for detecting and mitigating fraudulent activities in credit card transactions.

Through the utilization of R packages such as **dplyr**, **tidyverse**, and **ggplot2**, we have successfully prepared and explored the credit card transaction data. These tools allowed us to clean and preprocess the data, handle missing values and outliers, and gain valuable insights through visualizations.

The power of R's statistical analysis capabilities was harnessed for feature engineering and model development. By applying machine learning algorithms available in R, such as logistic regression, decision trees, random forests, gradient boosting, or neural networks, we have built models capable of accurately distinguishing fraudulent transactions from legitimate ones.

Evaluation of the trained models was performed using R's evaluation metrics and techniques, ensuring the models' performance and reliability. Additionally, model interpretation and feature importance analysis using R tools provided insights into the factors contributing to fraud detection.

In conclusion, the Credit Card Fraud Detection project showcases the power and versatility of R tools and packages for data preprocessing, exploratory analysis, model development, evaluation, and real-time monitoring. The utilization of these tools has resulted in a reliable and efficient solution for detecting and preventing credit card fraud, contributing to the security of financial systems and the protection of cardholders.

9) INTERFERANCE:

The credit card fraud detection system developed using R will contribute to mitigating financial losses caused by fraudulent activities. By leveraging machine learning algorithms and real-time monitoring, the system will provide effective detection and alert mechanisms to facilitate timely action. The project's implementation will enhance the security of credit card transactions and empower individuals and organizations to protect themselves from potential financial fraud.

10)LIMITATIONS AND FUTURE INCORPORATIONS:

While developing a credit card fraud detection system, it is important to consider its limitations and explore potential enhancements for future incorporation. Here are some limitations and future considerations for credit card fraud detection:

1) Imbalanced Data: The dataset used for fraud detection often suffers from class imbalance, where the number of legitimate transactions far exceeds the number of fraudulent transactions. This can lead to biased models that perform poorly in detecting fraud. Future incorporation can involve techniques such as oversampling, undersampling, or synthetic minority oversampling technique (SMOTE) to address class imbalance and improve model performance.

2) Evolving Fraud Patterns: Fraudulent activities continually evolve as fraudsters adapt their techniques to circumvent detection systems. It is essential to continuously update the fraud detection system to stay ahead of emerging fraud patterns. Incorporating real-time monitoring and employing advanced anomaly detection algorithms can help identify new types of fraud and improve the system's responsiveness.

3) Feature Engineering: Feature engineering plays a crucial role in developing an effective fraud detection system. Exploring additional relevant features, such as transaction metadata, user behavior, geographical information, and device characteristics, can enhance the system's ability to detect fraudulent activities. Feature engineering techniques like aggregation, interaction features, or time-based features can be employed to capture more meaningful information.

4) Model Interpretability: Interpreting the decisions made by the fraud detection system is important for transparency and trust. Incorporating techniques like SHAP (SHapley Additive exPlanations) values or LIME (Local Interpretable Model-Agnostic Explanations) can provide insights into how the model arrives at its predictions. This can assist in identifying the critical features driving the model's decisions and improve interpretability.

5) Real-time Monitoring: Implementing a real-time monitoring system can help identify fraudulent activities as they occur, allowing for immediate action. Streaming

algorithms or online learning techniques can be utilized to continuously update the model with new data and detect fraud in real-time.

6) Collaboration and Data Sharing: Collaborative efforts among financial institutions, credit card companies, and regulatory bodies can improve fraud detection capabilities. Sharing anonymized data and collaborating on research can lead to the development of more robust and accurate fraud detection models.

7) Ethical Considerations: It is essential to address ethical considerations when developing a credit card fraud detection system. Ensuring privacy protection, data anonymization, and compliance with data regulations is crucial to maintain the trust and confidentiality of customers' information.

By addressing these limitations and incorporating future enhancements, credit card fraud detection systems can become more robust, accurate, and adaptive to evolving fraud patterns. Continued research and development in this field will contribute to minimizing financial losses and protecting the interests of both credit card companies and cardholders.

11) REFRENCES:

- DATA SET TAKEN FROM : https://www.kaggle.com
- https://www.kaggle.com/code/amity4544/credit-card-fraud-detectionhttps://www.kaggle.com/datasets/kartik2112/fraud-detection
- https://www.kaggle.com/code/amity4544/credit-card-fraud-detection