Contents

[Objectives: 3](#_Toc106203164)

[Essential libraries: 3](#_Toc106203165)

[Dataset used: 4](#_Toc106203166)

[Data Exploration 4](#_Toc106203167)

[Data Manipulation 7](#_Toc106203168)

[Data Modeling 7](#_Toc106203169)

[Fitting Logistic Regression Model 8](#_Toc106203170)

[Fitting a Decision Tree Model 13](#_Toc106203171)

[Artificial Neural Network 14](#_Toc106203172)

[Gradient Boosting (GBM) 16](#_Toc106203173)

[Summary 19](#_Toc106203174)

[Reference 19](#_Toc106203175)

# **Objectives:**

The aim of this project is to predict whether a credit card transaction is fraudulent or not. It aims to track down credit card transaction data, which is done by detecting anomalies in the transaction data. Credit card fraud detection is typically implemented using an algorithm that detects any anomalies in the transaction data and notifies the cardholder (as a precautionary measure) and the bank about any suspicious transaction.

To build a classifier that can detect credit card fraudulent transactions. We will use a variety of [**machine learning algorithms**](https://data-flair.training/blogs/machine-learning-algorithm/)that will be able to discern fraudulent from non-fraudulent one. By the end of this machine learning project, you will learn how to implement machine learning algorithms to perform classification.

**Algorithms:**

1. **Decision Tree Algorithm**

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving **regression and classification problems** too.

The goal of using a Decision Tree is to create a training model that can be used to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data(training data).

1. **Logistic Regression Algorithm:**

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

1. **Artificial Neural Network:**

Artificial Neural networks (ANN) or neural networks are computational algorithms.

It intended to simulate the behavior of biological systems composed of “neurons”. ANNs are computational models inspired by an animal’s central nervous systems. It is capable of machine learning as well as pattern recognition. These presented as systems of interconnected “neurons” which can compute values from inputs.

1. **Gradient Boosting Classifier:**

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

**Essential libraries:**

Library (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.

Library (Caret):

One of the most powerful and popular packages is the caret library, which follows a consistent syntax for data preparation, model building, and model evaluation, making it easy for data science practitioners. Caret stands for classification and regression training and is arguably the biggest project in R.

Library(data.table):

data.table provides a high performance version of base R’s data.frame with syntax and feature enhancement for ease of use, convenience and programming speed.

Library (caTOOLS):

Contains several basic utility functions including: moving (rolling, running) window statistic functions, read/write for GIF and ENVI binary files, fast calculation of AUC, LogitBoost classifier, base64 encoder/decoder, round-off error free sum and cumsum, etc.

Library(pROC):

pROC is a package for R and S+ specifically dedicated to ROC analysis. It proposes multiple statistical tests to compare ROC curves, and in particular partial areas under the curve, allowing proper ROC interpretation.

Library(Rpart):

Rpart is a powerful machine learning library in R that is used for building classification and regression trees. This library implements recursive partitioning and is very easy to use.

Library(Rpart.plot):

This function combines and extends plot. rpart and text. rpart in the rpart package. It automatically scales and adjusts the displayed tree for best fit. This is a front end to prp , with the most useful arguments of that function.

Library(neuralnet):

Neural Network is just like a human nervous system, which is made up of interconnected neurons, in other words, a neural network is made up of interconnected information processing units. The neural network draws from the parallel processing of information, which is the strength of this method.

Library(gbm, quietly = TRUE):

A Gradient Boosting Machine or GBM combines the predictions from multiple decision trees to generate the final predictions. Keep in mind that all the weak learners in a gradient boosting machine are decision trees.

# **Dataset used**

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

* setwd("D:/3rd semester/DSF/Credit-Card-Dataset")

# **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 –

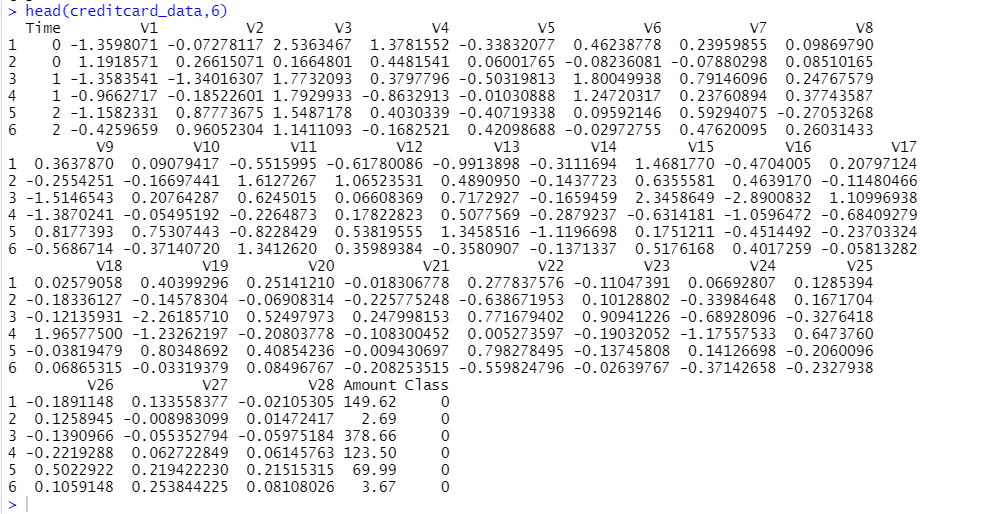
* dim(creditcard\_data)

The dim() is an inbuilt R function that either sets or returns the dimension of the matrix, array, or data frame. The dim() function takes the R object as an argument and returns its dimension, or if you assign the value to the dim() function, then it sets the dimension for that R Object.



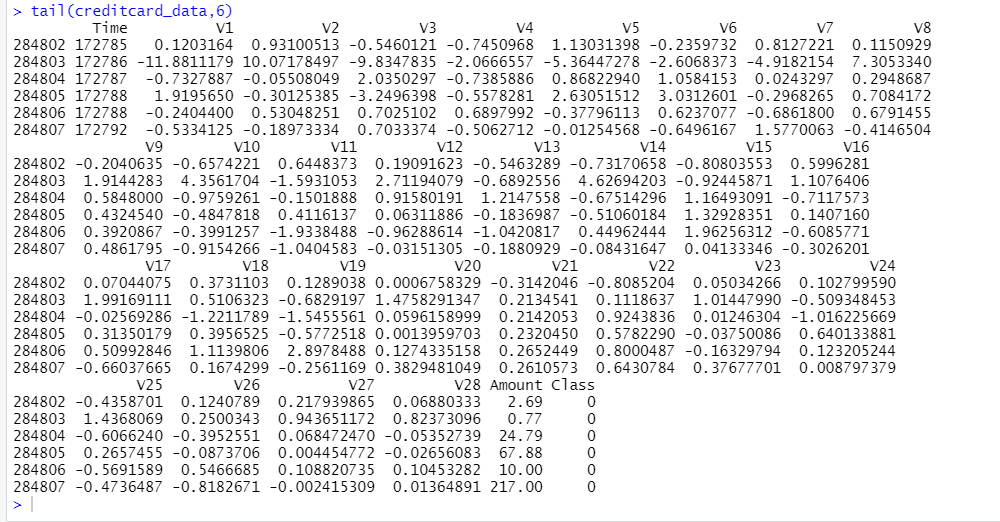
* head(creditcard\_data,6)

Head() is function in R that return last parts of a vector, matrix, table, data frame or function.



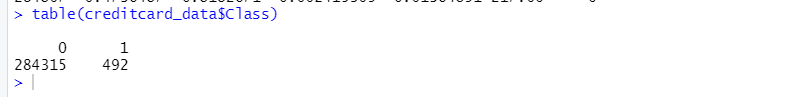
* tail(creditcard\_data,6)

Tail() is function in R that returns last parts of a vector, matrix, table, data frame or function.



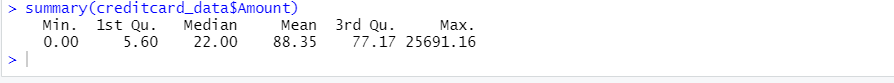
* table(creditcard\_data$Class)

table() function in R Language is used to create a categorical representation of data with variable name and the frequency in the form of a table.



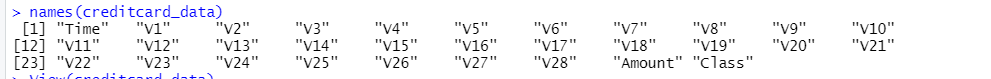
* summary(creditcard\_data$Amount)

The summary is a built-in R function used to produce result summaries of various model fitting functions. The summary() function implores specific methods that depend on the class of the first argument.



* names(creditcard\_data)

names() function in R Language is used to get or set the name of an Object. This function takes object i.e. vector, matrix, or data frame as argument along with the value that is to be assigned as name to the object. The length of the value vector passed must be exactly equal to the length of the object to be named.



* var(creditcard\_data$Amount)

var() function in R Language computes the sample variance of a vector. It is the measure of how much value is away from the mean value.



* sd(creditcard\_data$Amount)

sd() function is used to compute the standard deviation of given values in R. It is the square root of its variance.



# **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:

* head(creditcard\_data)

Head function I use to display head.

* creditcard\_data$Amount=scale(creditcard\_data$Amount)

scale() function in R Language is a generic function which centers and scales the columns of a numeric matrix. The center parameter takes either numeric alike vector or logical value. If the numeric vector is provided, then each column of the matrix has the corresponding value from center subtracted from it.

* NewData=creditcard\_data[,-c(1)]

Make another variable named NewData and then pass all the columns as in creditcard\_data except first column.

* head(NewData)

This will display head of new data.

# **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 –

* set.seed(123)

Setting a seed in R means to initialize a pseudorandom number generator. Most of the simulation methods in Statistics require the possibility to generate pseudorandom numbers that mimic the properties of independent generations of a uniform distribution in the interval.

* data\_sample = sample.split(NewData$Class,SplitRatio=0.80)

Split() is a built-in R function that divides a vector or data frame into groups according to the function's parameters.

* train\_data = subset(NewData,data\_sample==TRUE)

Attributing 80% of data to train data.

* test\_data = subset(NewData,data\_sample==FALSE)

Remaining 20% is attributed to test data.

* dim(train\_data)
* dim(test\_data)

Getting dimensions of train data and test data.



# **Fitting Logistic Regression Model**

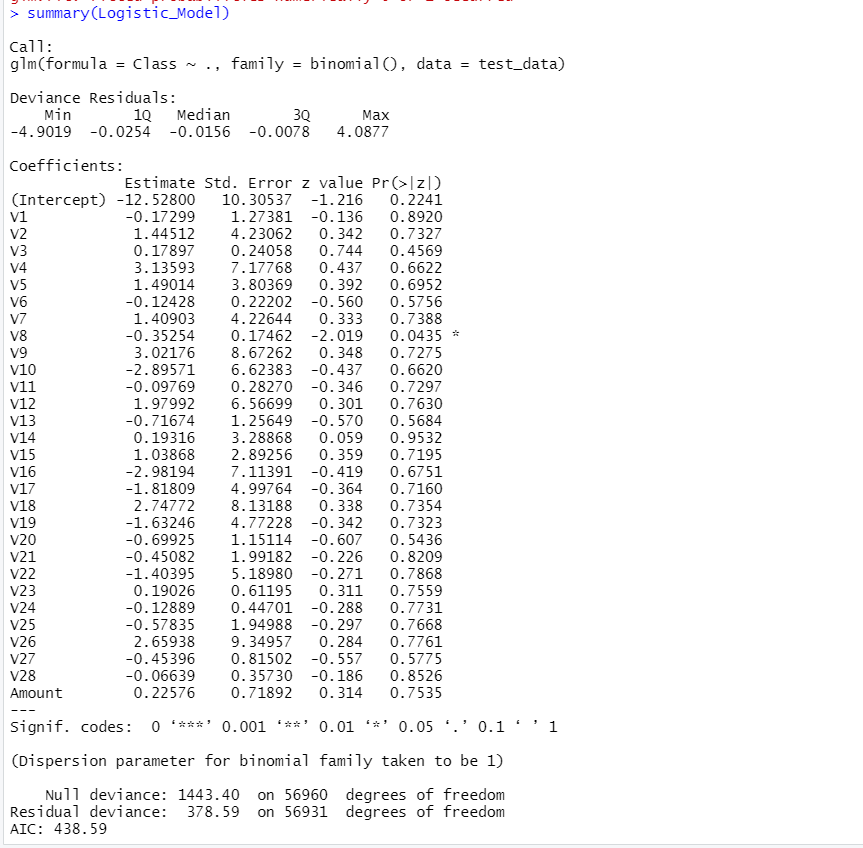
In this section of credit card fraud detection project, we will fit our first model. We will begin with logistic regression. A logistic regression is used for modeling the outcome probability of a class such as pass/fail, positive/negative and in our case – fraud/not fraud. We proceed to implement this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

* Logistic\_Model=glm(Class~.,test\_data,family=binomial())

Glm is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and description of the error distribution.

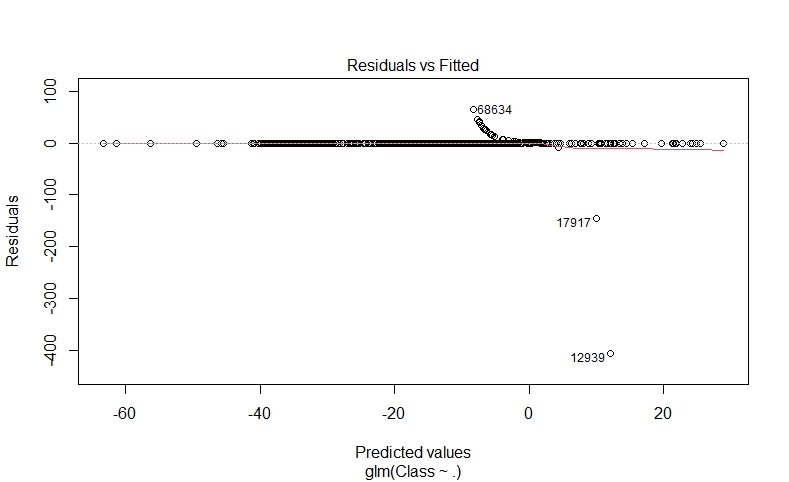
* summary(Logistic\_Model)

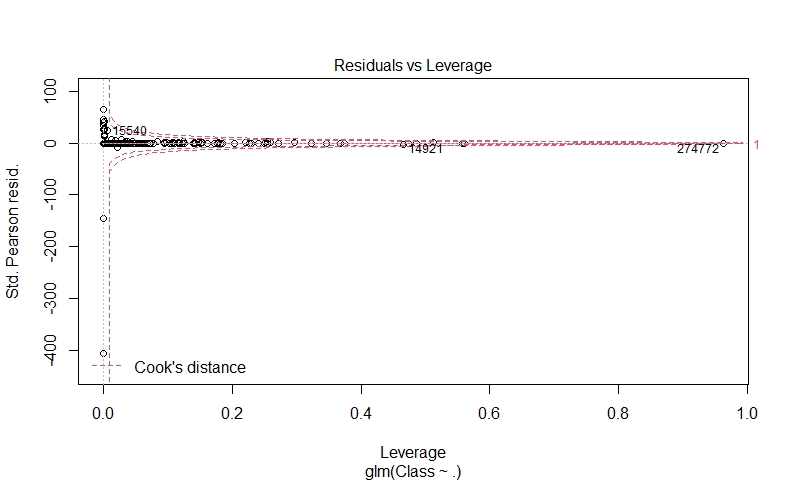
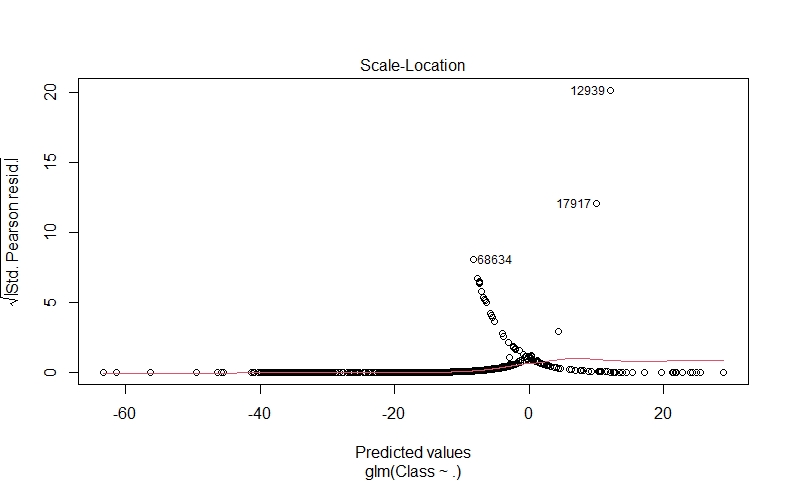
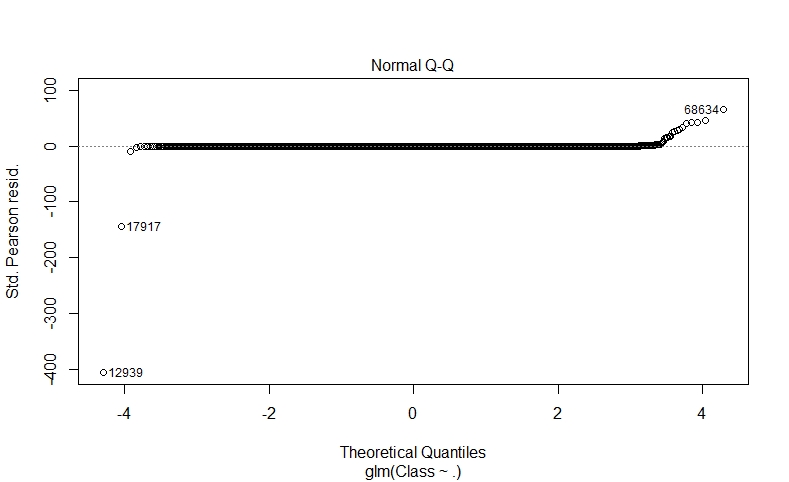
Summarizing logistic model.



* plot(Logistic\_Model)

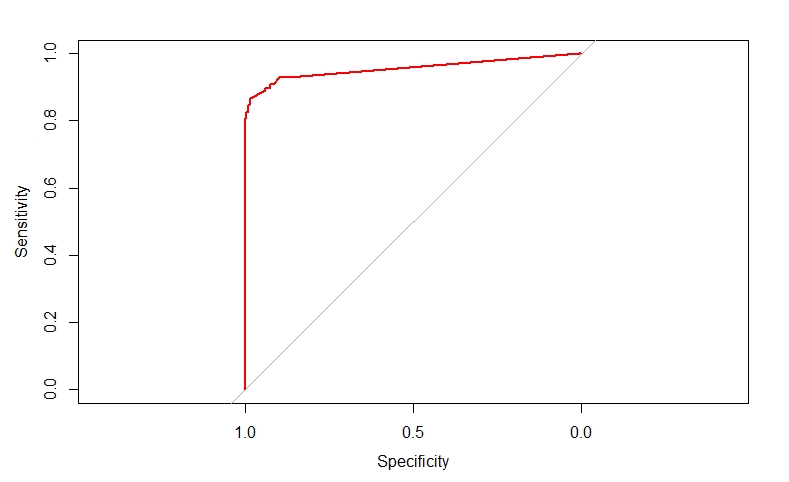
After we have summarised our model, we will visual it through the following plots





In order to assess the performance of our model, we will delineate the ROC curve. ROC is also known as Receiver Optimistic Characteristics.

* lr.predict <- predict(Logistic\_Model,train\_data, probability = TRUE)
* auc.gbm = roc(test\_data$Class, lr.predict, plot = TRUE, col = "blue")



# **Fitting a Decision Tree Model**

In this section, we will implement a decision tree algorithm. [**Decision Trees**](https://data-flair.training/blogs/r-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.

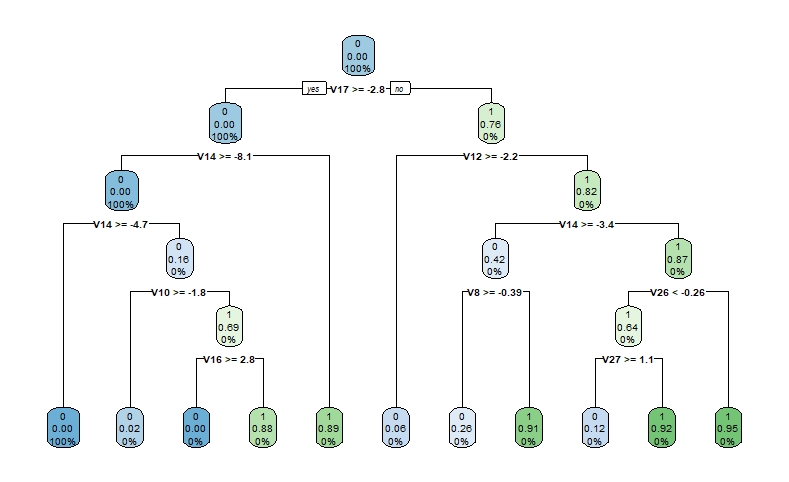
* decisionTree\_model <- rpart(Class ~ . , creditcard\_data, method = 'class')

R’s [rpart package](https://cran.r-project.org/web/packages/rpart/index.html" \t "_blank) provides a powerful framework for growing classification and regression trees.

* predicted\_val <- predict(decisionTree\_model, creditcard\_data, type = 'class')
* probability <- predict(decisionTree\_model, creditcard\_data, type = 'prob')

The predict() function in R is used to predict the values based on the input data. All the modeling aspects in the R program will make use of the predict function in its own way, but note that the functionality of the predict() function.

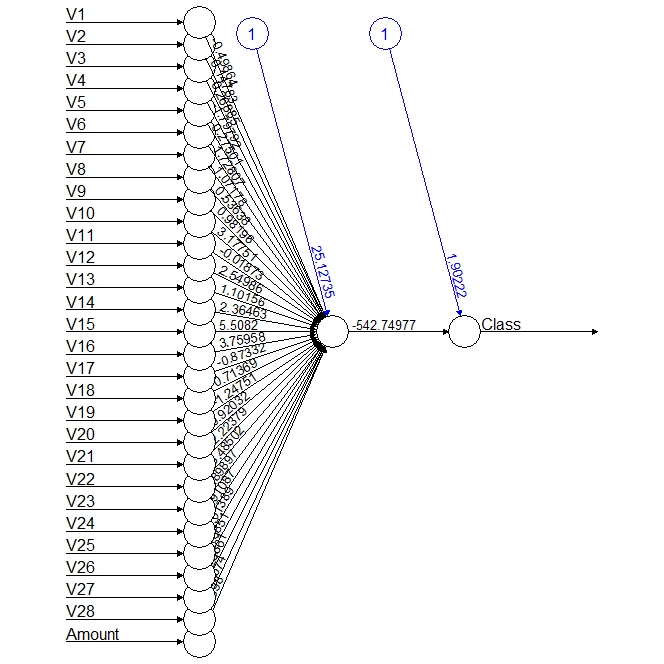
* rpart.plot(decisionTree\_model)



# **Artificial Neural Network**

[Artificial Neural Networks](https://data-flair.training/blogs/artificial-neural-network/) 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.

* ANN\_model =neuralnet (Class~.,train\_data,linear.output=FALSE)
* plot(ANN\_model)



* predANN=compute(ANN\_model,test\_data)

compute() stores results in a remote temporary table. Compute forces computation of lazy tables, leaving data in the remote source

* resultANN=predANN$net.result
* resultANN=ifelse(resultANN>0.5,1,0)

We set a threshold as 0.5, that is, values above 0.5 will correspond to 1 and the rest will be 0.

# **Gradient Boosting (GBM)**

[Gradient Boosting](https://data-flair.training/blogs/gradient-boosting-algorithm/) 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 –

* 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))

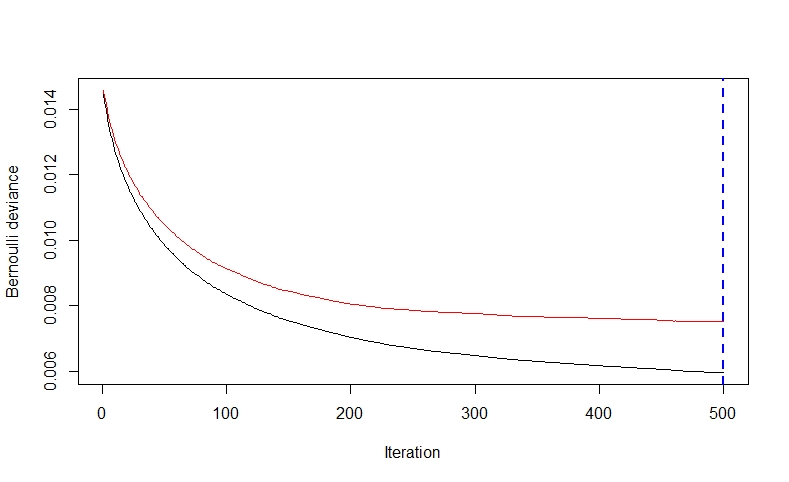
)

)



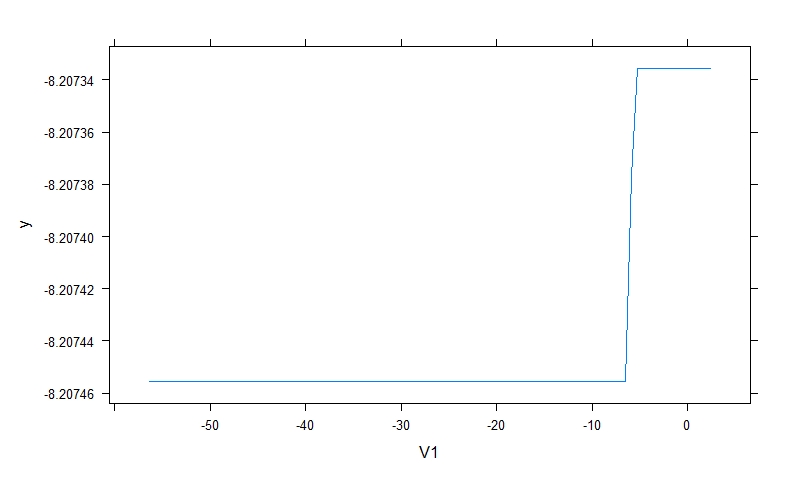
* gbm.iter = gbm.perf(model\_gbm, method = "test")

Determine best iteration based on test data



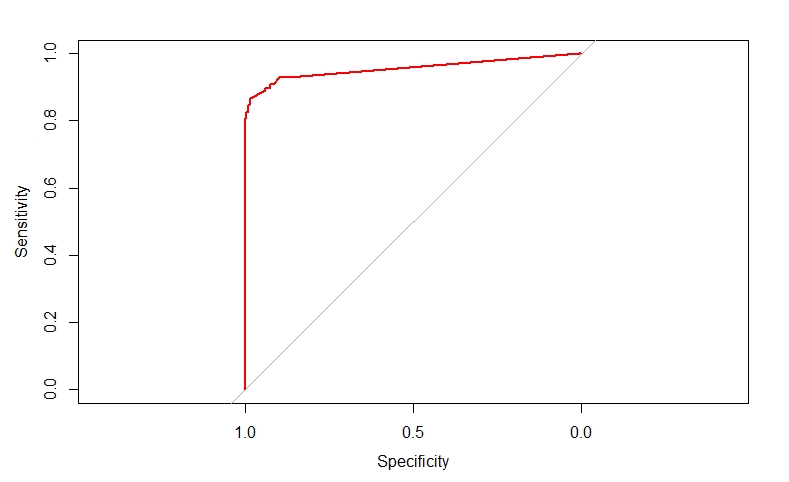
* model.influence = relative.influence(model\_gbm, n.trees = gbm.iter, sort. = TRUE)
* plot(model\_gbm)

Plot the gbm model

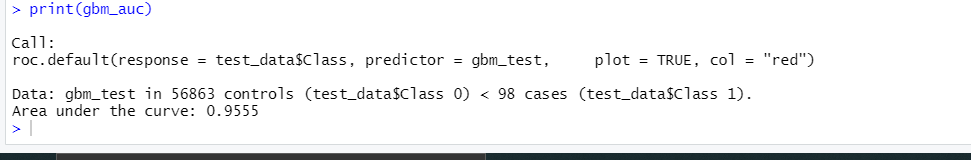


# Plot and calculate AUC on test data

* gbm\_test = predict(model\_gbm, newdata = test\_data, n.trees = gbm.iter)
* gbm\_auc = roc(test\_data$Class, gbm\_test, plot = TRUE, col = "red")



* print(gbm\_auc)



# **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.

# **Reference**

https://www.rdocumentation.org/packages/recommenderlab/versions/0.2-7