Netaji Subhas University of Technology

Group Details: **NAME:** Kumar Gaurav – 2021UCA1801
Utkarsh Gupta – 2021UCA1833
Aditya Singh – 2021UCA1862

AI HARDWARE LAB ASSIGNMENT 2

Ques 1: Explore Basic Data Structure in R.

In R, there are several basic data structures that are commonly used for storing and organizing data. These data structures include vectors, matrices, arrays, lists, and data frames. Data structures in R programming are tools for holding multiple values. The idea is to reduce the space and time complexities of different tasks.

R's base data structures are often organized by their dimensionality (1D, 2D, or nD) and whether they're homogeneous (all elements must be of the identical type) or heterogeneous (the elements are often of various types). This gives rise to the six data types which are most frequently utilized in data analysis.

The most essential Data Structures used in R are:

1. **Vectors:** A vector is an ordered collection of basic data types of a given length. The only key thing here is all the elements of a vector must be of the identical data type e.g homogeneous data structures. Vectors are one-dimensional data structures. They can be created using the **c()** function.

   CODE:

```
# R program to illustrate Vector

# Vectors(ordered collection of same data type)
X = c(1, 3, 5, 7, 8)

# Printing those elements in console
print(X)
```

```
Output

Rscript /tmp/QCfEAGbZuc.r
[1] 1 3 5 7 8
```

2. **Lists:** A list is a generic object consisting of an ordered collection of objects. Lists are heterogeneous data structures. These are also one-dimensional data structures. A list can be a list of vectors, list of matrices, a list of characters and a list of functions and so on. They can be created using the **list()** function.

   CODE:

```
# R program to illustrate a List

# The first attributes is a numeric vector containing the employee IDs which is created using the 'c' command here
empId = c(1, 2, 3, 4)
```

```
# The second attribute is the employee name which is created using this line of code here which is the character vector
empName = c("Debi", "Sandeep", "Subham", "Shiba")

# The third attribute is the number of employees which is a single numeric variable.
numberOfEmp = 4

# We can combine all these three different data types into a list containing the details of employees which can be done using a list command
empList = list(empId, empName, numberOfEmp)
print(empList)
```

Output

```
Rscript /tmp/QCfEAGbZuc.r
[[1]]
[1] 1 2 3 4

[[2]]
[1]"Debi"    "Sandeep" "Subham"  "Shiba"

[[3]]
[1] 4
```

3. **Data frames**: Data frames are generic data objects of R which are used to store the tabular data. Dataframes are the foremost popular data objects in R programming because we are comfortable in seeing the data within the tabular form. They are two-dimensional, heterogeneous data structures. These are lists of vectors of equal lengths. They can be created using the **data.frame()** function.

CODE:

```
# R program to illustrate dataframe

# A vector which is a character vector
Name = c("Amiya", "Raj", "Asish")

# A vector which is a character vector
Language = c("R", "Python", "Java")

# A vector which is a numeric vector
Age = c(22, 25, 45)

# To create dataframe use data.frame command and then pass each of the vectors we have created as arguments to the function data.frame()
df = data.frame(Name, Language, Age)
```

```
print(df)
```

```
Output

Rscript /tmp/QCfEAGbZuc.r
Name Language Age
1 Amiya        R  22
2   Raj   Python  25
3 Asish     Java  45
```

4. **Matrices:** A matrix is a rectangular arrangement of numbers in rows and columns. In a matrix, as we know rows are the ones that run horizontally and columns are the ones that run vertically. Matrices are two-dimensional, homogeneous data structures. They can be created using the **matrix()** function.

CODE :

```
# R program to illustrate a matrix

A = matrix(
  # Taking sequence of elements
  c(1, 2, 3, 4, 5, 6, 7, 8, 9),

  # No of rows and columns
  nrow = 3, ncol = 3,

  # By default matrices are  in column-wise order So this parameter decides how to arrange the matrix
  byrow = TRUE
)

print(A)
```

```
Output

Rscript /tmp/QCfEAGbZuc.r
[,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

5. **Arrays:** Arrays are the R data objects which store the data in more than two dimensions. Arrays are n-dimensional data structures. For example, if we create an array of dimensions (2, 3, 3) then it creates 3 rectangular matrices each with 2 rows and 3 columns. They are homogeneous data structures. They can be created using the **array()** function.

CODE:

```
# R program to illustrate an array

A = array(
  # Taking sequence of elements
  c(1, 2, 3, 4, 5, 6, 7, 8),

  # Creating two rectangular matrices each with two rows and two columns
  dim = c(2, 2, 2)
)
print(A)
```

```
Output

Rscript /tmp/QCfEAGbZuc.r
, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1][,2]
[1,]    5    7
[2,]    6    8
```

6. **Factors:** Factors are the data objects which are used to categorize the data and store it as levels. They are useful for storing categorical data. They can store both strings and integers. They are useful to categorize unique values in columns like “TRUE” or “FALSE”, or “MALE” or “FEMALE”, etc.. They are useful in data analysis for statistical modeling.

CODE:

```r
# R program to illustrate factors

# Creating factor using factor()
fac = factor(c("Male", "Female", "Male",
          "Male", "Female", "Male", "Female"))

print(fac)
```

Output

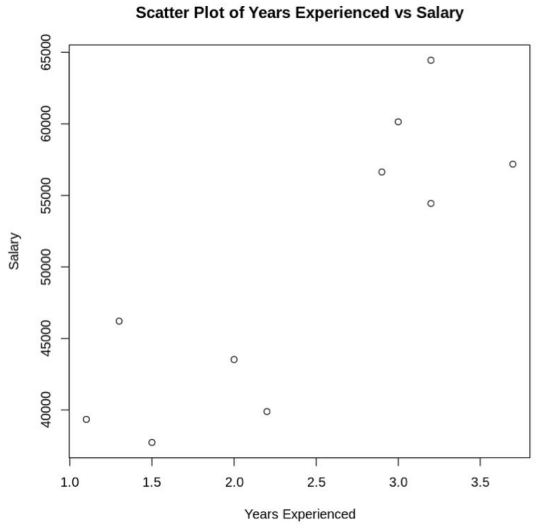
```
Rscript /tmp/QCfEAGbZuc.r
[1] Male   Female Male   Male   Female Male   Female
Levels: Female Male
```

```
Ques 2: Implement Linear Regression in R and Visualize the results
```

Linear Regression In R

Salary dataset where it is given the value of the dependent variable(salary) for every independent variable(years experienced).

```
data <- data.frame(
  Years_Exp = c(1.1, 1.3, 1.5, 2.0, 2.2, 2.9, 3.0, 3.2, 3.2, 3.7),
  Salary = c(39343.00, 46205.00, 37731.00, 43525.00,
             39891.00, 56642.00, 60150.00, 54445.00, 64445.00, 57189.00)
)


# Create the scatter plot
plot(data$Years_Exp, data$Salary,
  xlab = "Years Experienced",
  ylab = "Salary",
  main = "Scatter Plot of Years Experienced vs Salary")
```



Training The Model

```
install.packages('caTools')
library(caTools)
split = sample.split(data$Salary, SplitRatio = 0.7)
trainingset = subset(data, split == TRUE)
testset = subset(data, split == FALSE)

# Fitting Simple Linear Regression to the Training set
lm.r= lm(formula = Salary ~ Years_Exp,
        data = trainingset)
#Summary of the model
summary(lm.r)
```

```
    Installing package into ‘/usr/local/lib/R/site-library’
    (as ‘lib’ is unspecified)

    also installing the dependency ‘bitops’



    Call:
    lm(formula = Salary ~ Years_Exp, data = trainingset)

    Residuals:
        2     3     4     5     6     8     9
     7676 -2831 -2119 -7786  1850 -3396  6604

    Coefficients:
                Estimate Std. Error t value Pr(>|t|)
    (Intercept)    25315       7817   3.239   0.0230 *
    Years_Exp      10164       3204   3.173   0.0247 *
    ---
    Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

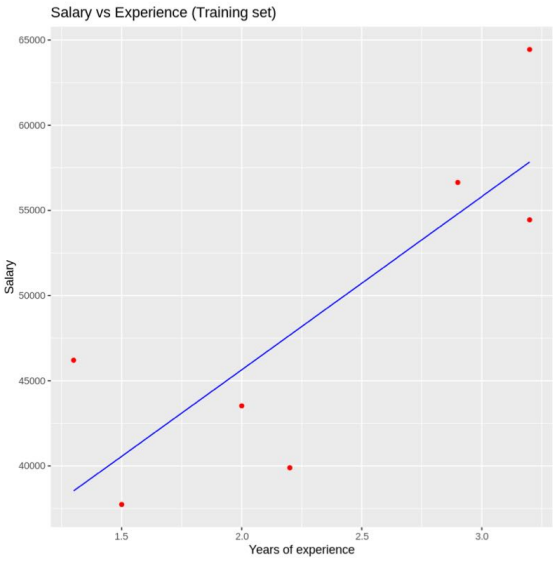
```
Residual standard error: 6174 on 5 degrees of freedom
Multiple R-squared:  0.6681,    Adjusted R-squared:  0.6017
F-statistic: 10.07 on 1 and 5 DF,  p-value: 0.02474
```

Prediction

```
# Create a data frame with new input values
new_data <- data.frame(Years_Exp = c(4.0, 4.5, 5.0))

# Predict using the linear regression model
predicted_salaries <- predict(lm.r, newdata = new_data)

# Display the predicted salaries
print(predicted_salaries)
```

```
       1        2        3
65972.33 71054.48 76136.63
```

```
# Visualising the Training set results
ggplot() + geom_point(aes(x = trainingset$Years_Ex,
                          y = trainingset$Salary), colour = 'red') +
geom_line(aes(x = trainingset$Years_Ex,
              y = predict(lm.r, newdata = trainingset)), colour = 'blue') +

ggtitle('Salary vs Experience (Training set)') +
xlab('Years of experience') +
ylab('Salary')
```



Ques 3: Implement Logistic Regression in R and Visualize the results.

```
# Installing the package

install.packages("dplyr")

# Loading package
library(dplyr)

# Summary of dataset in package
summary(mtcars)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)


Attaching package: ‘dplyr’


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

    filter, lag


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

    intersect, setdiff, setequal, union


      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

```
# Installing the package

# For Logistic regression
install.packages("caTools")

# For ROC curve to evaluate model
install.packages("ROCR")

# Loading package
library(caTools)
library(ROCR)


# Splitting dataset
split <- sample.split(mtcars, SplitRatio = 0.8)
split

train_reg <- subset(mtcars, split == "TRUE")
test_reg <- subset(mtcars, split == "FALSE")

# Training model
logistic_model <- glm(vs ~ wt + disp,
                      data = train_reg,
                      family = "binomial")
logistic_model

# Summary
summary(logistic_model)
```

```
FALSE · TRUE · TRUE · FALSE · TRUE · TRUE · TRUE · TRUE · TRUE · TRUE · FALSE
```

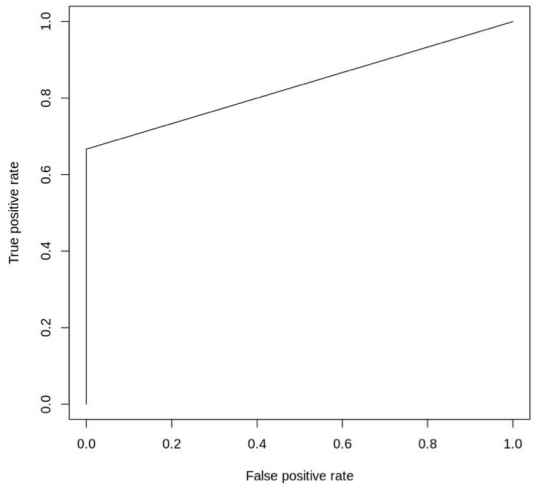
```
Call:  glm(formula = vs ~ wt + disp, family = "binomial", data = train_reg)

Coefficients:
(Intercept)           wt         disp
    1.19918      2.58876     -0.04998

Degrees of Freedom: 23 Total (i.e. Null);  21 Residual
Null Deviance:      33.1
Residual Deviance: 13.5         AIC: 19.5

Call:
glm(formula = vs ~ wt + disp, family = "binomial", data = train_reg)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.19918    2.80968   0.427   0.6695
wt           2.58876    2.04743   1.264   0.2061
disp        -0.04998    0.02680  -1.865   0.0622 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 33.104  on 23  degrees of freedom
Residual deviance: 13.497  on 21  degrees of freedom
AIC: 19.497

Number of Fisher Scoring iterations: 7
```

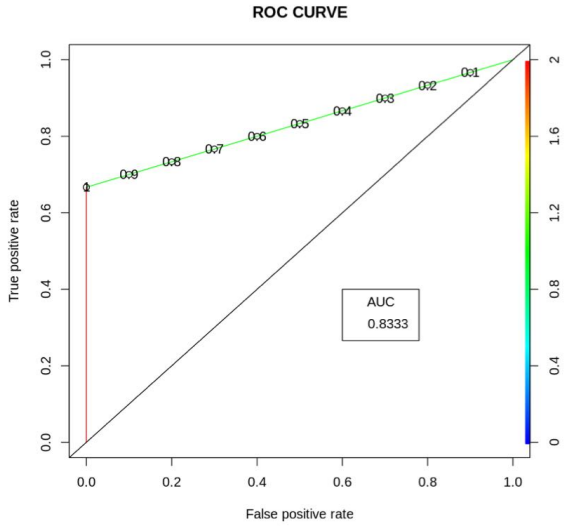
```
predict_reg <- predict(logistic_model,
                       test_reg, type = "response")
predict_reg
```

```
Mazda RX4:         0.496135545203394 Hornet 4 Drive:         0.0331362639942055 Merc 280C:         0.849087190381391 Merc 450SE:         0.1140800375723
      0.000150476553677837 Dodge Challenger:         0.00374789268877345 AMC Javelin:         0.00604117984797943 Fiat X1-9:         0.905494406016653
```

```
# Changing probabilities
predict_reg <- ifelse(predict_reg >0.5, 1, 0)

# Evaluating model accuracy
# using confusion matrix
table(test_reg$vs, predict_reg)

missing_classerr <- mean(predict_reg != test_reg$vs)
print(paste('Accuracy =', 1 - missing_classerr))

# ROC-AUC Curve
ROCPred <- prediction(predict_reg, test_reg$vs)
ROCPer <- performance(ROCPred, measure = "tpr",
                      x.measure = "fpr")

auc <- performance(ROCPred, measure = "auc")
auc <- auc@y.values[[1]]
auc

# Plotting curve
plot(ROCPer)
plot(ROCPer, colorize = TRUE,
    print.cutoffs.at = seq(0.1, by = 0.1),
    main = "ROC CURVE")
abline(a = 0, b = 1)

auc <- round(auc, 4)
legend(.6, .4, auc, title = "AUC", cex = 1)
```

```
   predict_reg
    0 1
  0 5 0
  1 1 2
[1] "Accuracy = 0.875"
0.833333333333333
```





Ques 4: Implement any Machine learning Algorithm along with feature selection and data visualization on any dataset of your choice.

```
library(dplyr) # for data manipulation
library(stringr) # for data manipulation
library(caret) # for sampling
library(caTools) # for train/test split
library(ggplot2) # for data visualization
library(corrplot) # for correlations
library(Rtsne) # for tsne plotting
library(DMwR) # for smote implementation
library(ROSE)# for ROSE sampling
library(rpart)# for decision tree model
library(Rborist)# for random forest model
library(xgboost) # for xgboost model
```

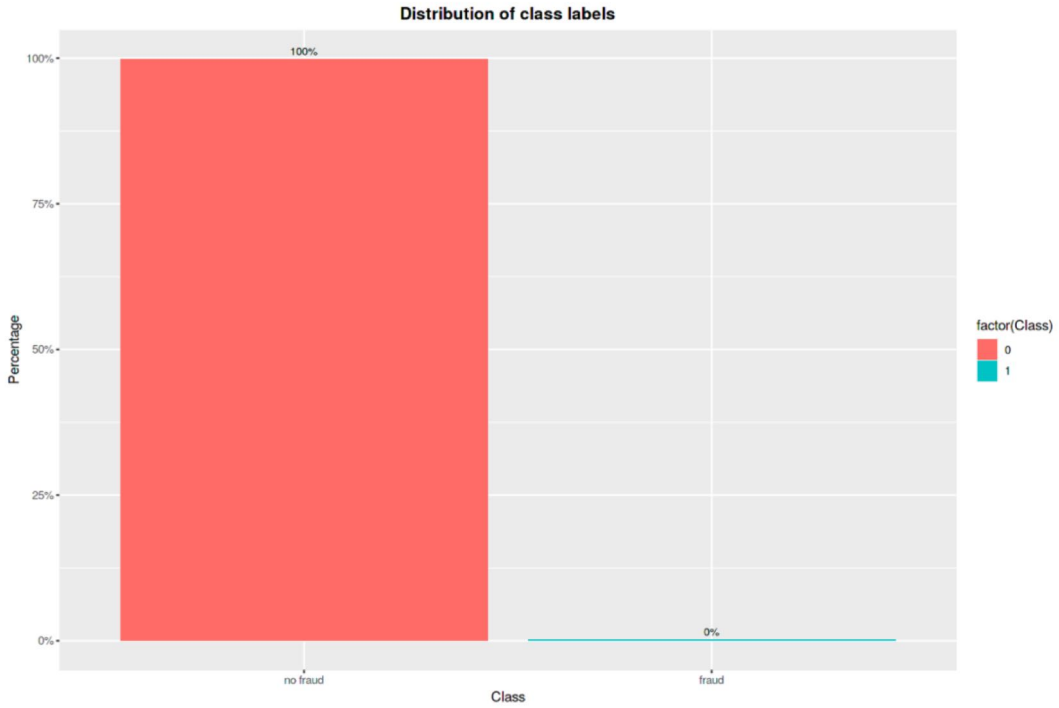
```
# function to set plot height and width
fig <- function(width, heigth){
     options(repr.plot.width = width, repr.plot.height = heigth)
}
```

In [3]:

```
# loading the data
df = read.csv('../input/creditcardfraud/creditcard.csv')
```

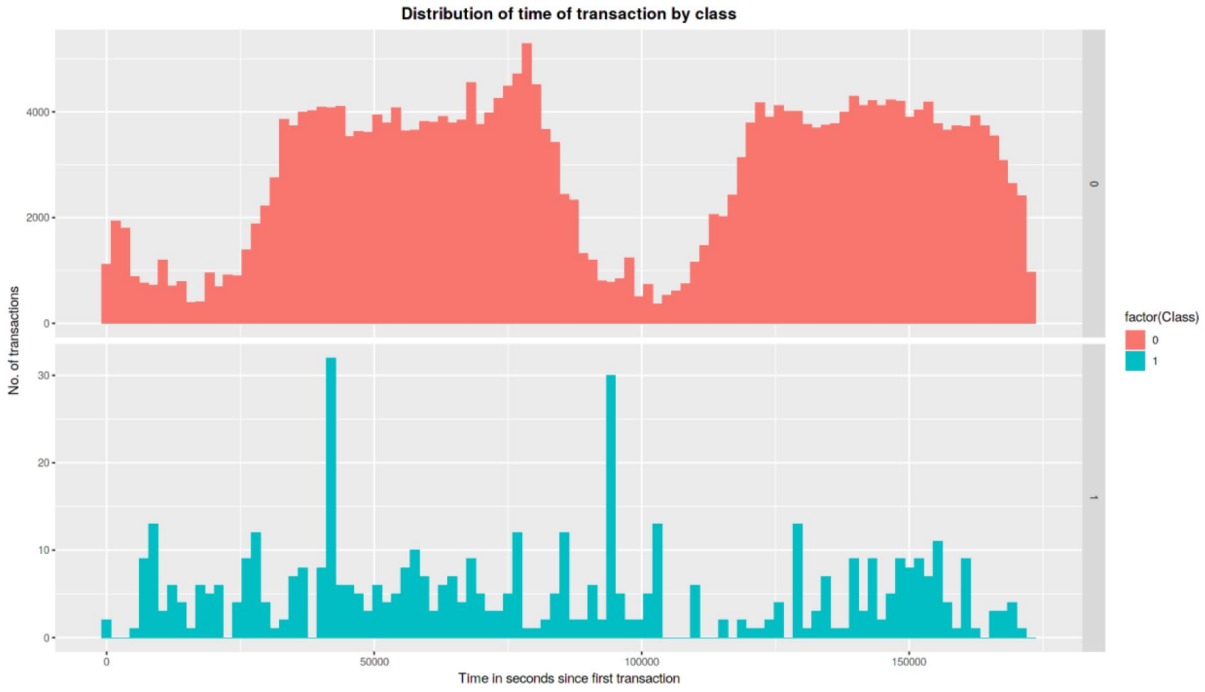
```
fig(12, 8)
common_theme <- theme(plot.title = element_text(hjust = 0.5,
face = "bold"))

ggplot(data = df, aes(x = factor(Class),
                      y = prop.table(stat(count)), fill =
factor(Class),
                      label =
scales::percent(prop.table(stat(count))))) +
```

```
    geom_bar(position = "dodge") +
    geom_text(stat = 'count',
              position = position_dodge(.9),
              vjust = -0.5,
              size = 3) +
    scale_x_discrete(labels = c("no fraud", "fraud"))+
    scale_y_continuous(labels = scales::percent)+
    labs(x = 'Class', y = 'Percentage') +
    ggtitle("Distribution of class labels") +
    common_theme
```

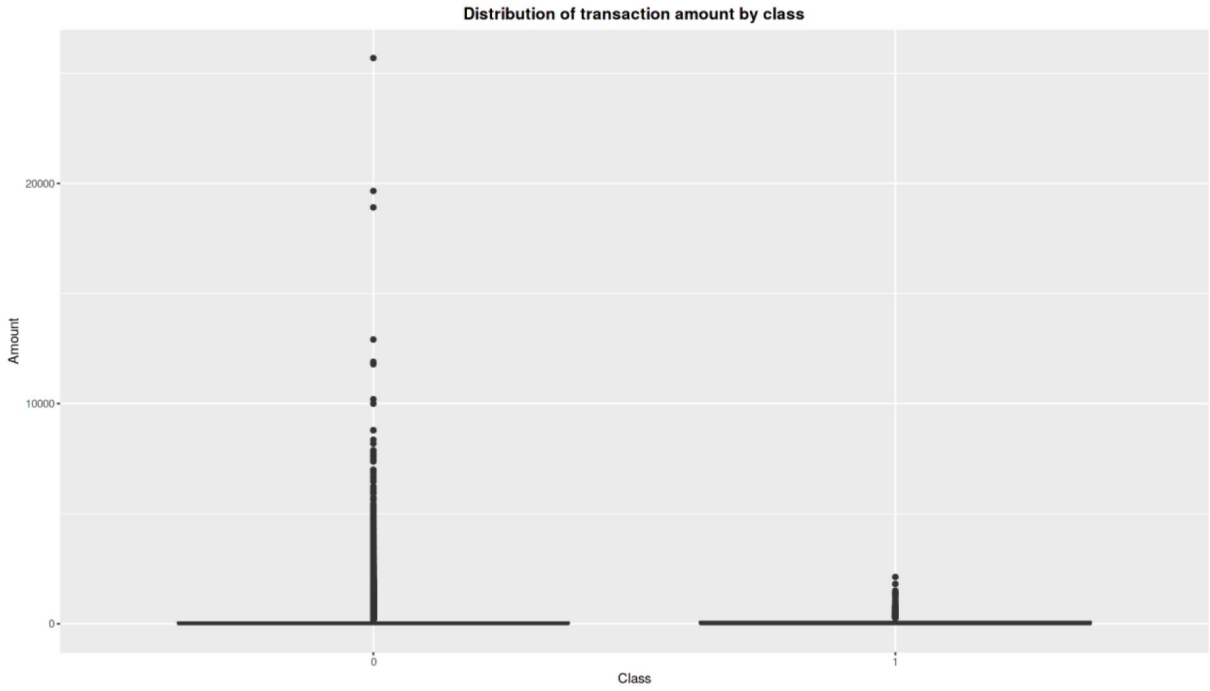



```
fig(14, 8)
df %>%
  ggplot(aes(x = Time, fill = factor(Class))) + geom_histogram(bins =
100)+
  labs(x = 'Time in seconds since first transaction', y = 'No. of
```

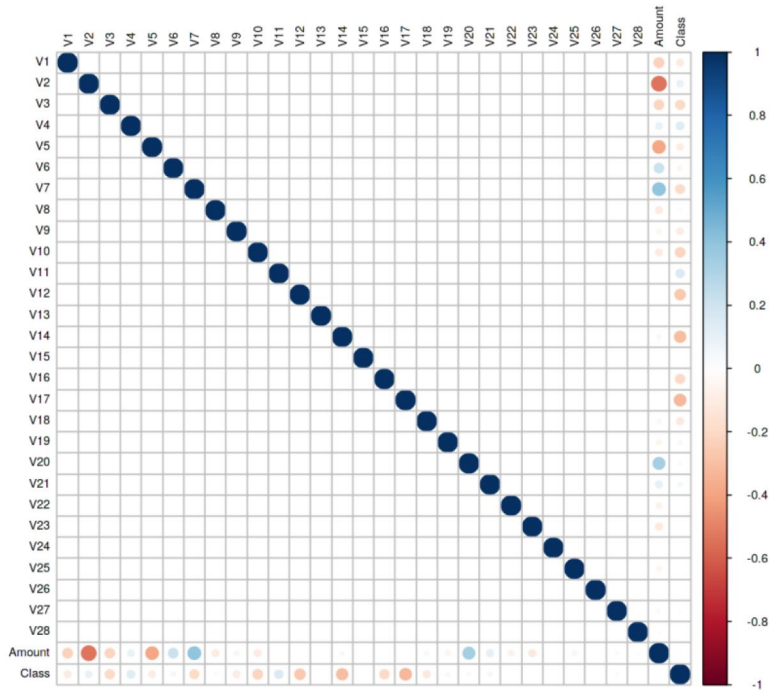
```
transactions') +
  ggtitle('Distribution of time of transaction by class') +
  facet_grid(Class ~ ., scales = 'free_y') + common_theme
```



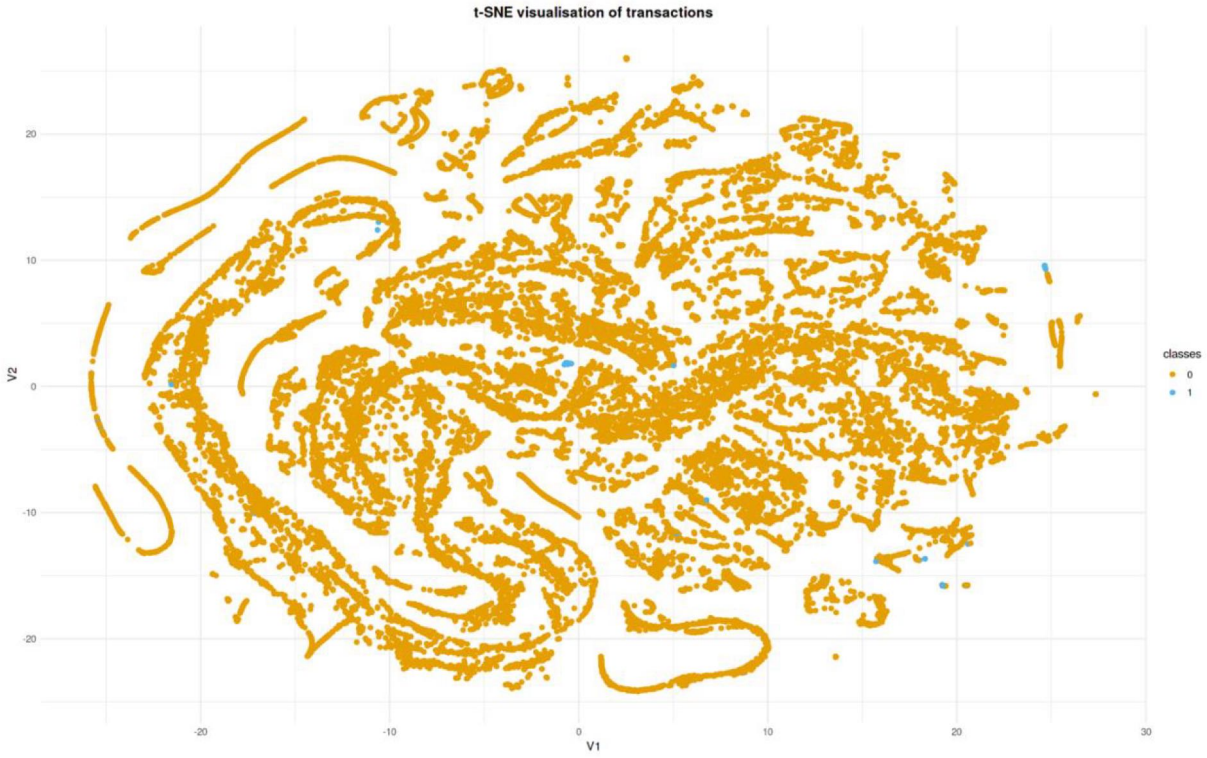

```
fig(14, 8)
ggplot(df, aes(x = factor(Class), y = Amount)) + geom_boxplot() +
labs(x = 'Class', y = 'Amount') +
ggtitle("Distribution of transaction amount by class") + common_theme
```




```
fig(14, 8)
correlations <- cor(df[,-1],method="pearson")
corrplot(correlations, number.cex = .9, method = "circle", type = "full",
tl.cex=0.8,tl.col = "black")
```




```
fig(16, 10)

# Use 10% of data to compute t-SNE
tsne_subset <- 1:as.integer(0.1*nrow(df))
tsne <- Rtsne(df[tsne_subset,-c(1, 31)], perplexity = 20, theta = 0.5, pca
= F, verbose = F, max_iter = 500, check_duplicates = F)

classes <- as.factor(df$Class[tsne_subset])
tsne_mat <- as.data.frame(tsne$Y)
ggplot(tsne_mat, aes(x = V1, y = V2)) + geom_point(aes(color = classes)) +
theme_minimal() + common_theme + ggtitle("t-SNE visualisation of
transactions") + scale_color_manual(values = c("#E69F00", "#56B4E9"))
```




```
set.seed(123)
split <- sample.split(df$Class, SplitRatio = 0.7)
train <-  subset(df, split == TRUE)
test <- subset(df, split == FALSE)
```

```
# downsampling
set.seed(9560)
down_train <- downSample(x = train[, -ncol(train)],
                         y = train$Class)
table(down_train$Class)
```

```
Not_Fraud     Fraud
      344       344
```

In [20]:

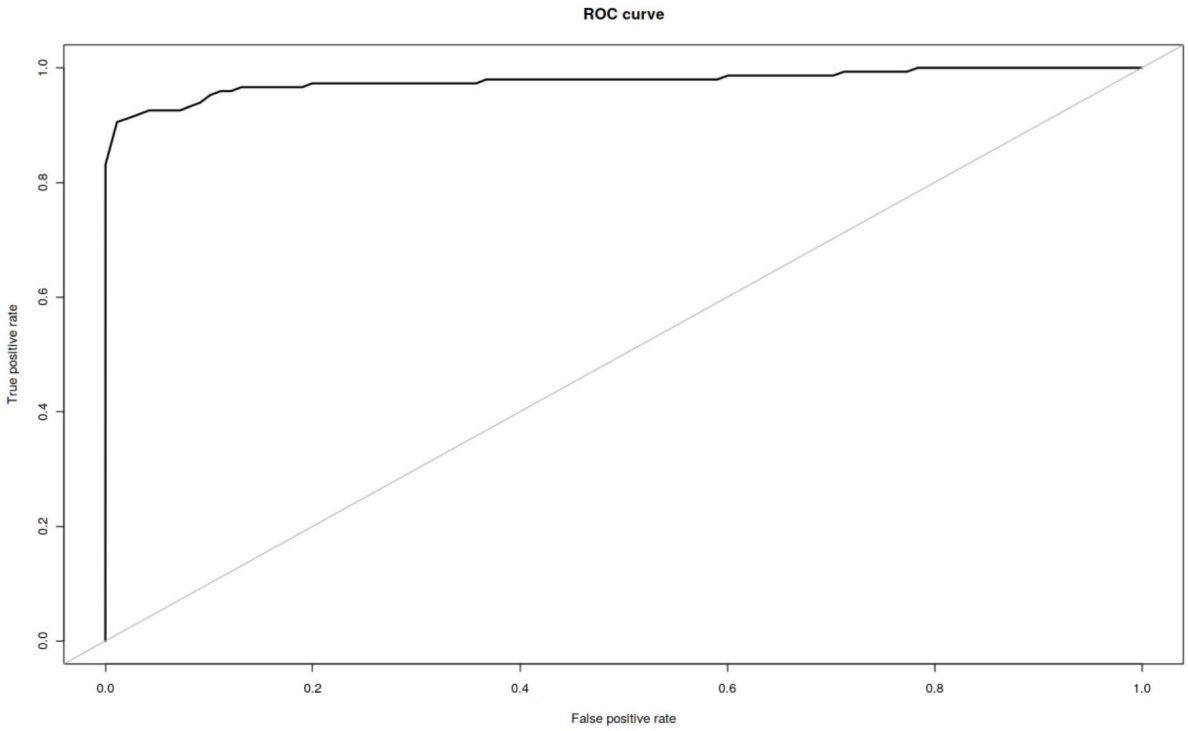
```
upsampling
set.seed(9560)
up_train <- upSample(x = train[, -ncol(train)],
```

XGBoost¶

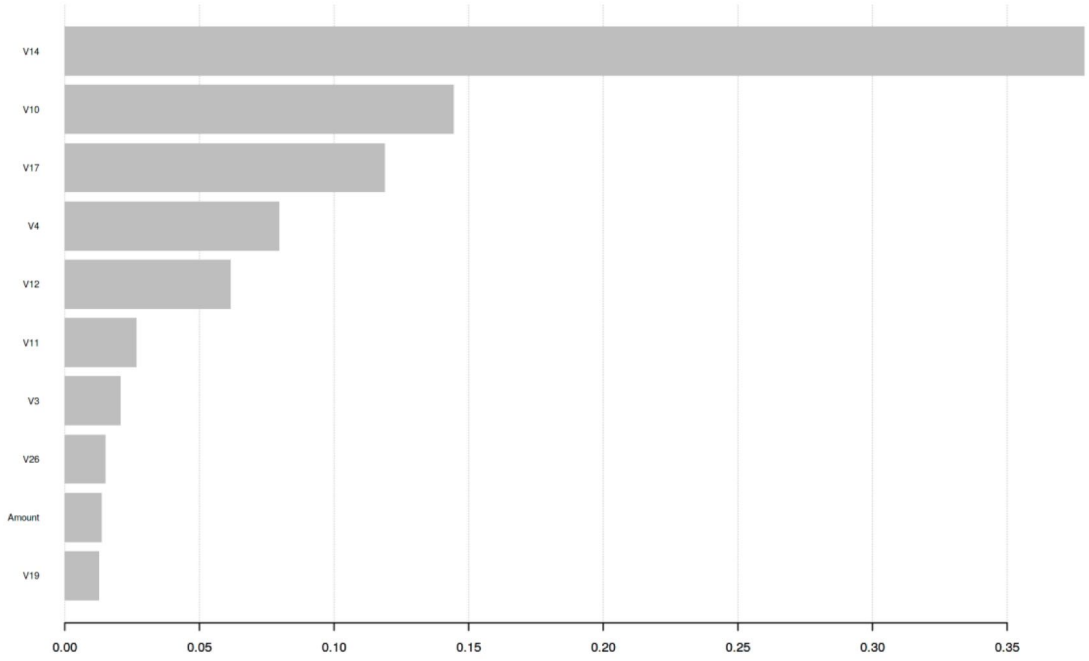
```
# Convert class labels from factor to numeric

labels <- up_train$Class

y <- recode(labels, 'Not_Fraud' = 0, "Fraud" = 1)
```

In [32]:

```
set.seed(42)
xgb <- xgboost(data = data.matrix(up_train[,-30]),
 label = y,
 eta = 0.1,
 gamma = 0.1,
 max_depth = 10,
 nrounds = 300,
 objective = "binary:logistic",
 colsample_bytree = 0.6,
 verbose = 0,
 nthread = 7,
)


xgb_pred <- predict(xgb, data.matrix(test[,-30]))

roc.curve(test$Class, xgb_pred, plotit = TRUE)

Area under the curve (AUC): 0.977
```




```
names <- dimnames(data.matrix(up_train[,-30]))[[2]]

# Compute feature importance matrix
importance_matrix <- xgb.importance(names, model = xgb)
# Nice graph
xgb.plot.importance(importance_matrix[1:10,])
```


V14
V10
V17
V4
V12
V11
V3
V26
Amount
V19
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35