Faculty of Engineering and Technology

DMIHR(DU)

LAB MANUAL

M.Tech – First Year

Semester – II

Introduction to Data Science

Session 2022-2023

Practical List

|  |  |
| --- | --- |
| **Sr. No.** | **Practical List** |
| 1 | Exploratory data analysis (EDA) for a large dataset. |
| 2 | Clustering analysis for customer segmentation. |
| 3 | Linear regression analysis to predict sales. |
| 4 | Principal component analysis (PCA) for dimensionality reduction. |
| 5 | Decision tree analysis to study customer churn. |
| 6 | Time series analysis to forecast the covid-19. |
| 7 | Association rule mining to identify patterns. |
| 8 | Network analysis to identify relationship between variables. |
| 9 | Text mining to extract valuable insights from unstructured data |
| 10 | Anomaly detection to identify unusual patterns. |
| 11 | To construct a Spam filter or Email classifier. |
| 12 | Sentiment Analysis using Exploratory Data Analytics (EDA). |
| 13 | Image classification of IRIS data set. |
| 14 | Analysis and prediction using XGBoost. |
| 15 | Association Rule Mining for prediction. |

Experiment No. – 1

Aim - Exploratory data analysis (EDA) of a large dataset to identify patterns, outliers, and correlations.

Theory:-

**Exploratory Data Analysis** (**EDA**) is the process of analyzing and visualizing the data to get a better understanding of the data and glean insight from it. There are various steps involved when doing EDA but the following are the common steps that a data analyst can take when performing EDA:

* Import the data
* Clean the data
* Process the data
* Visualize the data

1. **Tidyverse**package for tidying up the data set
2. **ggplot2**package for visualizations
3. **corrplot**package for correlation plot
4. Some other basic functions to manipulate data like strsplit (), cbind (), matrix () and so on.

**Program :-**

**Importing the data**

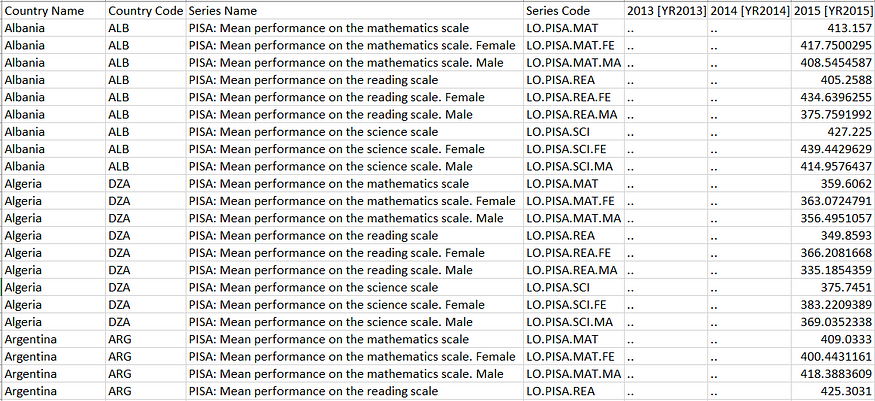
These are 3 ways of importing the data into R.

df.raw <- read.csv(file ='Pisa scores 2013 - 2015 Data.csv', fileEncoding="UTF-8-BOM", na.strings = '..')

df.raw1 <- read.csv(file ='Pisa scores 2013 - 2015 Data.csv')

df.raw2 <- read.csv(file ='Pisa scores 2013 - 2015 Data.csv',na.strings = '..')

**Output :-**



# Cleaning and Processing the data

install.packages("tidyverse")  
library(tidyverse)

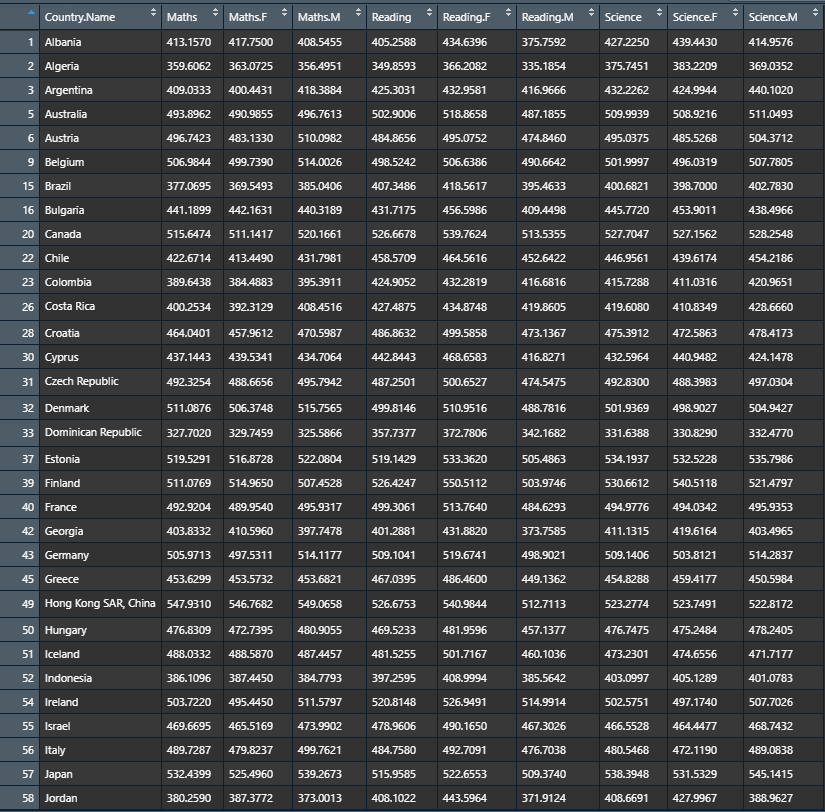
We want to do a few things to clean the dataset:

1. Make sure that each row in the dataset corresponds to ONLY one country: Use spread() function in tidyverse package
2. Make sure that only useful columns and rows are kept: Use drop\_na() and data subsetting
3. Rename the Series Code column for meaningful interpretation: Use rename()

df <- df.raw[1:1161, c(1, 4, 7)] #select relevant rows and cols  
%>% spread(key=Series.Code, value=X2015..YR2015.)   
%>% rename(Maths = LO.PISA.MAT,   
Maths.F = LO.PISA.MAT.FE,  
Maths.M = LO.PISA.MAT.MA,  
Reading = LO.PISA.REA,  
Reading.F = LO.PISA.REA.FE,  
Reading.M = LO.PISA.REA.MA,  
Science = LO.PISA.SCI,  
Science.F = LO.PISA.SCI.FE,  
Science.M = LO.PISA.SCI.MA  
) %>%  
drop\_na()

view(df)

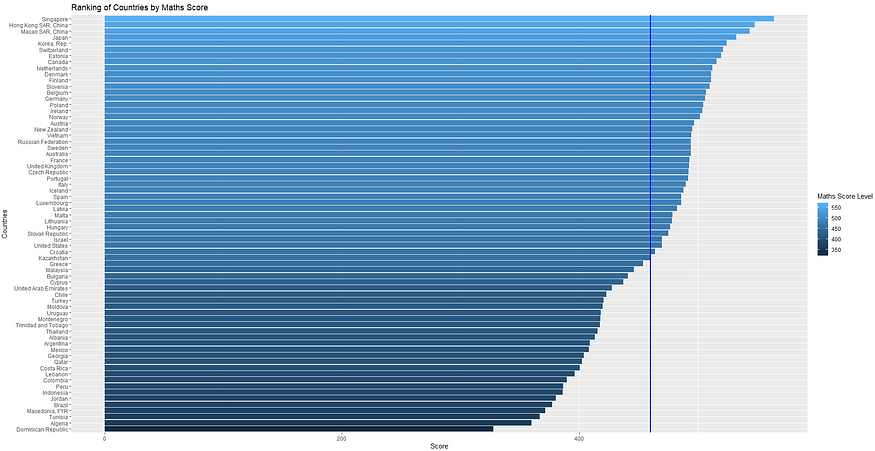
**Output :-**



# ****Visualizing the data****

install.packages("ggplot2")  
library(ggplot2)#Ranking of Maths Score by Countriesggplot(data=df,aes(x=reorder(Country.Name,Maths),y=Maths)) +   
 geom\_bar(stat ='identity',aes(fill=Maths))+  
 coord\_flip() +   
 theme\_grey() +   
 scale\_fill\_gradient(name="Maths Score Level")+  
 labs(title = 'Ranking of Countries by Maths Score',  
 y='Score',x='Countries')+   
 geom\_hline(yintercept = mean(df$Maths),size = 1, color = 'blue')

**Output:- Barplot**



**Correlation Plot**

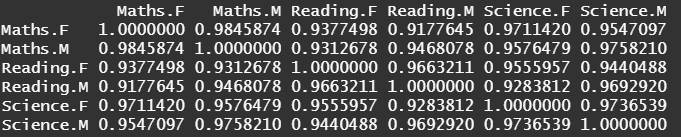
df = df[,c(1,3,4,6,7,9,10)] #select relevant columns

#To create correlation plot, simply use cor():

res = cor(df[,-1]) # -1 here means we look at all columns except the first column

res

**Output:-**



**Conclusion : -** Exploratory data analysis (EDA) of a large dataset to identify patterns, outliers, and correlations is executed successfully .

Experiment No. – 2

Aim :- Clustering analysis to segment customers into different groups based on their demographic and behaviours patterns .

Theory :-

Customer Segmentation is the process of splitting a customer base into multiple groups of individuals that share a similarity in ways a product is or can be marketed to them such as gender, age, interests, demographics, economic status, geography, behavioral patterns, spending habits and much more. Customer Segmentation is one the most important applications of unsupervised learning. Using clustering techniques, companies can identify the several segments of customers allowing them to target the potential user base.

Companies use the clustering process to foresee or map customer segments with similar behavior to identify and target potential user base.

## **K-MEAN ALGORITHM**

Summing up the K-means clustering –

• We specify the number of clusters that we need to create.

• The algorithm selects k objects at random from the dataset. This object is the initial cluster or mean.

• The closest centroid obtains the assignment of a new observation. We base this assignment on the Euclidean Distance between object and the centroid.

• k clusters in the data points update the centroid through calculation of the new mean valus present in all the data points of the cluster. The kth cluster’s centroid has a length of p that co-ntains means of all variables for observations in the k-th cluster.

We denote the number of variables with p.

• Iterative minimization of the total within the sum of squares. Then through the iterative minimization of the total sum of the square, the assignment stops wavering when we achieve maximum iteration. The default value is 10 that the R software uses for the maximum iterations.

Determining Optimal Clusters: While working with clusters, we need to specify the number of clusters to use. we would like to utilize the optimal number of clusters. To determining the

optimal clusters, there are methods –

**ELBOW PLOT**

The main goal behind cluster partitioning methods like k-means is to define the clusters such

that the intra-cluster variation stays minimum.

Minimize (sum W(Ck)), k=1…k

Where Ck represents the kth cluster and W(Ck) denotes the intra-cluster variation (this is the distance between data points in the same cluster). With the measurement of the total intra-cluster variation, one can evaluate the compactness of the clustering boundary. We can then proce-ed to define the optimal clusters as follows –

1. Calculate the clustering algorithm for several values of k. This can be done by creating a

variation within k from 1 to 10 clusters.

2. Calculate the total intra-cluster sum of square (iss).

3. Plot iss based on the number of k clusters. This plot denotes the appropriate number of clusters required in our model.

4. In the plot, the location of a bend or a knee is the indication of the optimum number of

clusters

**Program :-**

**UPLOADING DATA SHEET**

**library**(readr)

Mall\_Customers <- read\_csv("~/KAMILIMU/Data Science/customer-segmentation-dataset/customer-segmentation-dataset/Mall\_Customers.csv")

## Parsed with column specification:

## cols(

## CustomerID = col\_double(),

## Gender = col\_character(),

## Age = col\_double(),

## `Annual Income (k$)` = col\_double(),

## `Spending Score (1-100)` = col\_double()

## )

*#table(Mall\_Customers$`Annual Income (k$)`)*

*#table(Mall\_Customers$`Spending Score (1-100)`)*

**DATA SUMMARY**

View(Mall\_Customers)

summary(Mall\_Customers) *#summary of the data set*

**Output :-**

## CustomerID Gender Age Annual Income (k$)

## Min. : 1.00 Length:200 Min. :18.00 Min. : 15.00

## 1st Qu.: 50.75 Class :character 1st Qu.:28.75 1st Qu.: 41.50

## Median :100.50 Mode :character Median :36.00 Median : 61.50

## Mean :100.50 Mean :38.85 Mean : 60.56

## 3rd Qu.:150.25 3rd Qu.:49.00 3rd Qu.: 78.00

## Max. :200.00 Max. :70.00 Max. :137.00

## Spending Score (1-100)

## Min. : 1.00

## 1st Qu.:34.75

## Median :50.00

## Mean :50.20

## 3rd Qu.:73.00

## Max. :99.00

any(is.na(Mall\_Customers))*#To find missing values*

**Output :-**

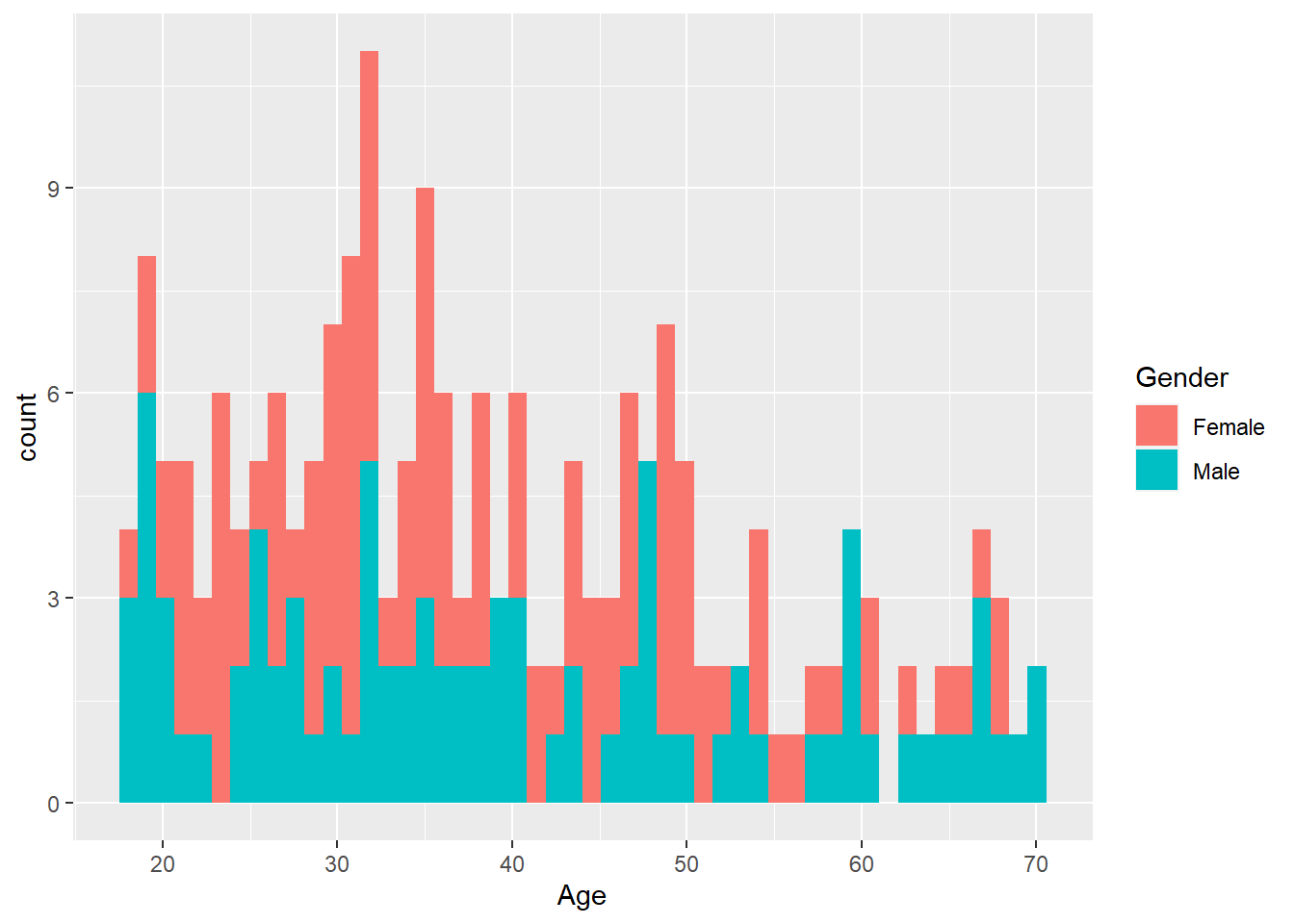
## [1] FALSE

FALSE indicated that there is no missing data. Therefore, there is no data cleaning that was done, and we head to data visulaization.

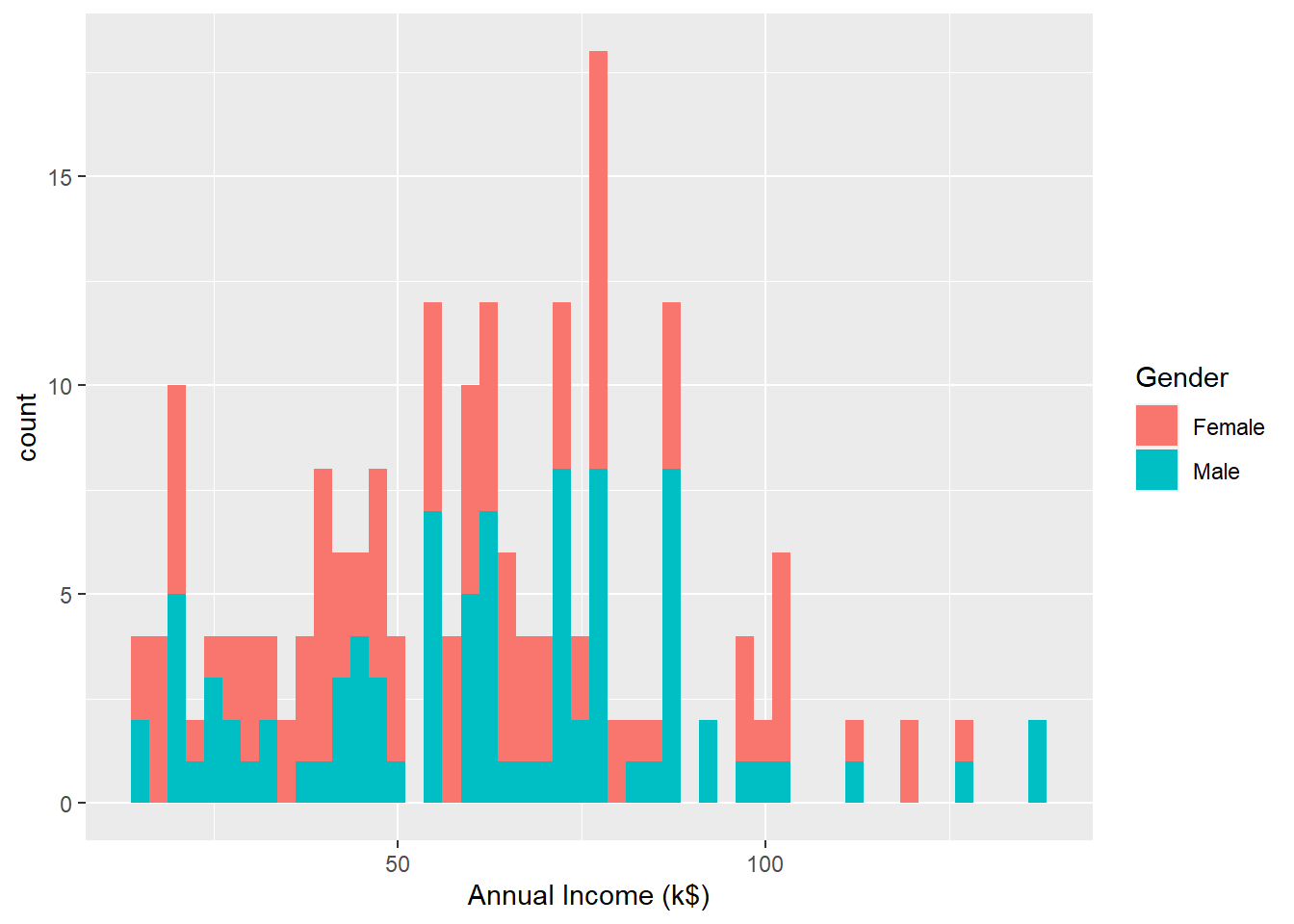
## **DATA VISUALIZATION**

**library**(ggplot2)

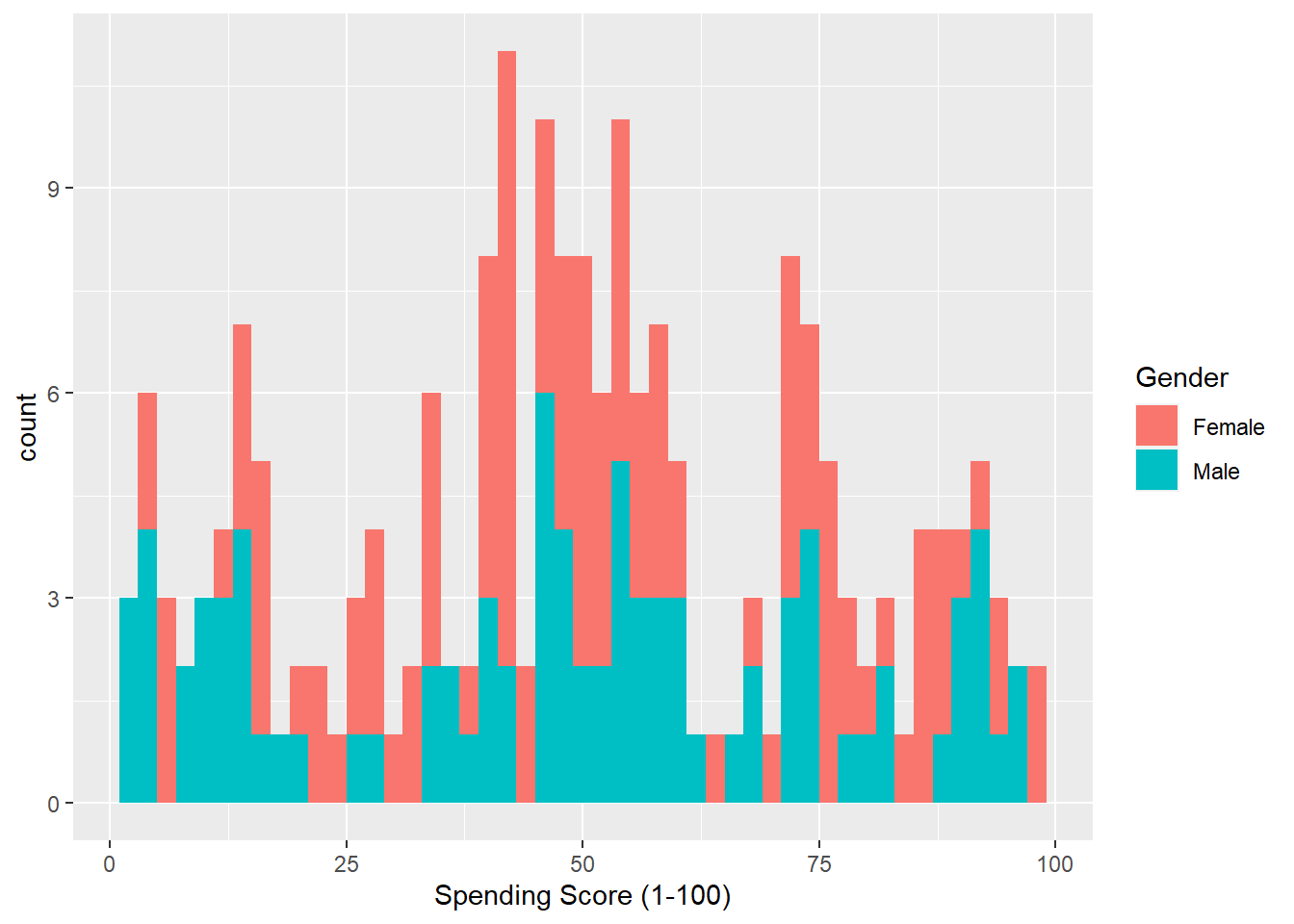
ggplot(Mall\_Customers,aes(x= Age, fill=Gender))+geom\_histogram(bins = 50) *# Histogram of Age filling Gender*



ggplot(Mall\_Customers,aes(x= `Annual Income (k$)`,fill=Gender)) +geom\_histogram(bins = 50) *# Histogram of Annual income filling Gender*



ggplot(Mall\_Customers,aes(x= `Spending Score (1-100)`,fill=Gender)) +geom\_histogram(bins=50) *# Histogram of Spending Score filling Gender*



**Conclusion :-** Clustering analysis to segment customers into different groups based on their demographic and behaviours patterns is executed successfully.

Experiment No. – 3

Aim :- Show the Linear regression analysis to predict sales (based on the basis of the amount of money spent in the three advertising medias (youtube, facebook and newspaper)

Theory :-

The mathematical formula of the linear regression can be written as follow:

y = b0 + b1\*x + e

We read this as “y is modeled as beta1 (b1) times x, plus a constant beta0 (b0), plus an error term e.”

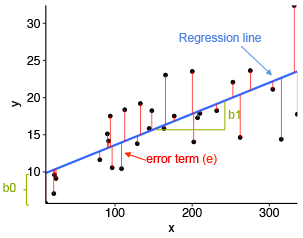
When you have multiple predictor variables, the equation can be written as y = b0 + b1\*x1 + b2\*x2 + ... + bn\*xn,

where:

* b0 is the intercept,
* b1, b2, …, bn are the regression weights or coefficients associated with the predictors x1, x2, …, xn.
* e is the *error term* (also known as the *residual errors*), the part of y that can be explained by the regression model.

The figure below illustrates a simple linear regression model, where:

* the best-fit regression line is in blue
* the intercept (b0) and the slope (b1) are shown in green
* the error terms (e) are represented by vertical red lines



From the scatter plot above, it can be seen that not all the data points fall exactly on the fitted regression line. Some of the points are above the blue curve and some are below it; overall, the residual errors (e) have approximately mean zero.

The sum of the squares of the residual errors are called the **Residual Sum of Squares** or **RSS**.

The average variation of points around the fitted regression line is called the **Residual Standard Error** (**RSE**). This is one the metrics used to evaluate the overall quality of the fitted regression model. The lower the RSE, the better it is.

Since the mean error term is zero, the outcome variable y can be approximately estimated as follow:

y ~ b0 + b1\*x

Mathematically, the beta coefficients (b0 and b1) are determined so that the RSS is as minimal as possible. This method of determining the beta coefficients is technically called **least squares** regression or **ordinary least squares** (OLS) regression.

Once, the beta coefficients are calculated, a t-test is performed to check whether or not these coefficients are significantly different from zero. A non-zero beta coefficients means that there is a significant relationship between the predictors (x) and the outcome variable (y).

**Program :-**

## **Loading Required R packages**

* tidyverse for easy data manipulation and visualization
* caret for easy machine learning workflow

**library**(tidyverse)

**library**(caret)

theme\_set(theme\_bw())

## **Preparing the data**

# Load the data

data("marketing", package = "datarium")

# Inspect the data

sample\_n(marketing, 3)

Output:-

## youtube facebook newspaper sales

## 58 163.4 23.0 19.9 15.8

## 157 112.7 52.2 60.6 18.4

## 81 91.7 32.0 26.8 14.2

# Split the data into training and test set

set.seed(123)

training.samples <- marketing$sales %>%

createDataPartition(p = 0.8, list = FALSE)

train.data <- marketing[training.samples, ]

test.data <- marketing[-training.samples, ]

## **Computing linear regression**

The R function lm() is used to compute linear regression model.

# Build the model

model <- lm(sales ~., data = train.data)

# Summarize the model

summary(model)

# Make predictions

predictions <- model %>% predict(test.data)

# Model performance # (a) Prediction error, RMSE

RMSE(predictions, test.data$sales)

# (b) R-square

R2(predictions, test.data$sales)

### **Simple linear regression**

The **simple linear regression** is used to predict a continuous outcome variable (y) based on one single predictor variable (x).

In the following example, we’ll build a simple linear model to predict sales units based on the advertising budget spent on youtube. The regression equation can be written as sales = b0 + b1\*youtube.

The R function lm() can be used to determine the beta coefficients of the linear model, as follow:

model <- lm(sales ~ youtube, data = train.data)

summary(model)$coef

**Output:-**

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 8.3839 0.62442 13.4 5.22e-28

## youtube 0.0468 0.00301 15.6 7.84e-34

Predictions can be easily made using the R function predict(). In the following example, we predict sales units for two youtube advertising budget: 0 and 1000.

newdata <- data.frame(youtube = c(0, 1000))

model %>% predict(newdata)

## 1 2

## 8.38 55.19

### **Multiple linear regression**

**Multiple linear regression** is an extension of simple linear regression for predicting an outcome variable (y) on the basis of multiple distinct predictor variables (x).

For example, with three predictor variables (x), the prediction of y is expressed by the following equation: y = b0 + b1\*x1 + b2\*x2 + b3\*x3

The regression beta coefficients measure the association between each predictor variable and the outcome. “b\_j” can be interpreted as the average effect on y of a one unit increase in “x\_j”, holding all other predictors fixed.

In this section, we’ll build a multiple regression model to predict sales based on the budget invested in three advertising medias: youtube, facebook and newspaper. The formula is as follow: sales = b0 + b1\*youtube + b2\*facebook + b3\*newspaper

You can compute the multiple regression model coefficients in R as follow:

model <- lm(sales ~ youtube + facebook + newspaper, data = train.data)

summary(model)$coef

Note that, if you have many predictor variables in your data, you can simply include all the available variables in the model using ~.:

model <- lm(sales ~., data = train.data)

summary(model)$coef

**Output:-**

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 3.39188 0.44062 7.698 1.41e-12

## youtube 0.04557 0.00159 28.630 2.03e-64

## facebook 0.18694 0.00989 18.905 2.07e-42

## newspaper 0.00179 0.00677 0.264 7.92e-01

**Conclusion :-** The Linear regression analysis to predict sales (based on the basis of the amount of money spent in the three advertising medias (youtube, facebook and newspaper) is implemented successfully.

Experiment No. – 4

**Aim :-** Principal component analysis (PCA) to identify the most important variables that contribute to customer satisfaction.

**Theory :-**

**Principal component analysis(PCA) in**[**R programming**](https://www.geeksforgeeks.org/introduction-to-r-programming-language/)is an analysis of the linear components of all existing attributes. Principal components are linear combinations (orthogonal transformation) of the original predictor in the dataset. It is a useful technique for EDA(Exploratory data analysis) and allows you to better visualize the variations present in a dataset with many variables.

## **R – Principal Component Analysis**

**First principal component** captures the maximum variance in dataset. It determines the direction of higher variability. **Second principal component** captures the remaining variance in data and is uncorrelated with PC1. The correlation between PC1 and PC2 should be zero. So, all succeeding principal components follow the same concept. They capture the remaining variance without being correlated to the previous principal component.

### **The Dataset**

The dataset **mtcars**(motor trend car road test) comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles. It comes pre-installed with dplyr package in R.

**Program :-**

# Installing required package

install.packages("dplyr")

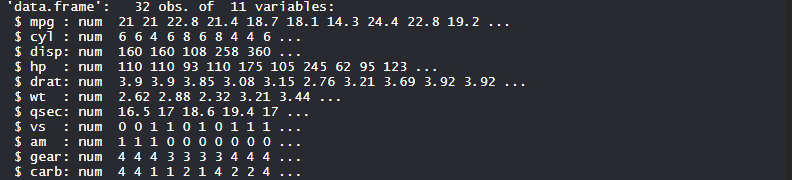
# Loading the package

library(dplyr)

# Importing excel file

str(mtcars)

**Output :-**



## **Principal Component Analysis with R language using dataset**

We perform Principal component analysis on **mtcars** which consists of 32 car brands and 10 variables.

# Loading Data

data(mtcars)

# Apply PCA using prcomp function

# Need to scale / Normalize as

# PCA depends on distance measure

my\_pca <- prcomp(mtcars, scale = TRUE,center = TRUE, retx = T)

names(my\_pca)

# Summary

summary(my\_pca)

my\_pca

# View the principal component loading

# my\_pca$rotation[1:5, 1:4]

my\_pca$rotation

# See the principal components

dim(my\_pca$x)

my\_pca$x

# Plotting the resultant principal components

# The parameter scale = 0 ensures that arrows

# are scaled to represent the loadings

biplot(my\_pca, main = "Biplot", scale = 0)

# Compute standard deviation

my\_pca$sdev

# Compute variance

my\_pca.var <- my\_pca$sdev ^ 2

my\_pca.var

# Proportion of variance for a scree plot

propve <- my\_pca.var / sum(my\_pca.var)

propve

# Plot variance explained for each principal component

plot(propve, xlab = "principal component",

ylab = "Proportion of Variance Explained",

ylim = c(0, 1), type = "b",

main = "Scree Plot")

# Plot the cumulative proportion of variance explained

plot(cumsum(propve),

xlab = "Principal Component",

ylab = "Cumulative Proportion of Variance Explained",

ylim = c(0, 1), type = "b")

# Find Top n principal component

# which will atleast cover 90 % variance of dimension

which(cumsum(propve) >= 0.9)[1]

# Predict mpg using first 4 new Principal Components

# Add a training set with principal components

train.data <- data.frame(disp = mtcars$disp, my\_pca$x[, 1:4])

# Running a Decision tree algporithm

## Installing and loading packages

install.packages("rpart")

install.packages("rpart.plot")

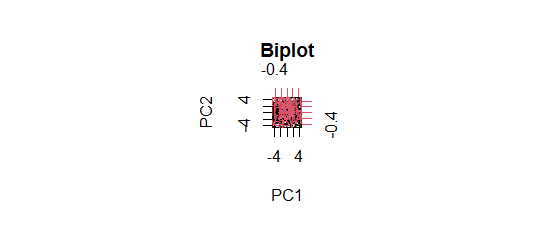
library(rpart)

library(rpart.plot)

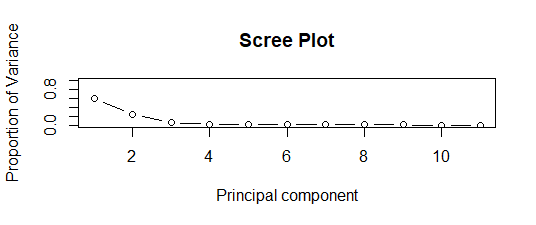
rpart.model <- rpart(disp ~ ., data = train.data, method = "anova")

rpart.plot(rpart.model)

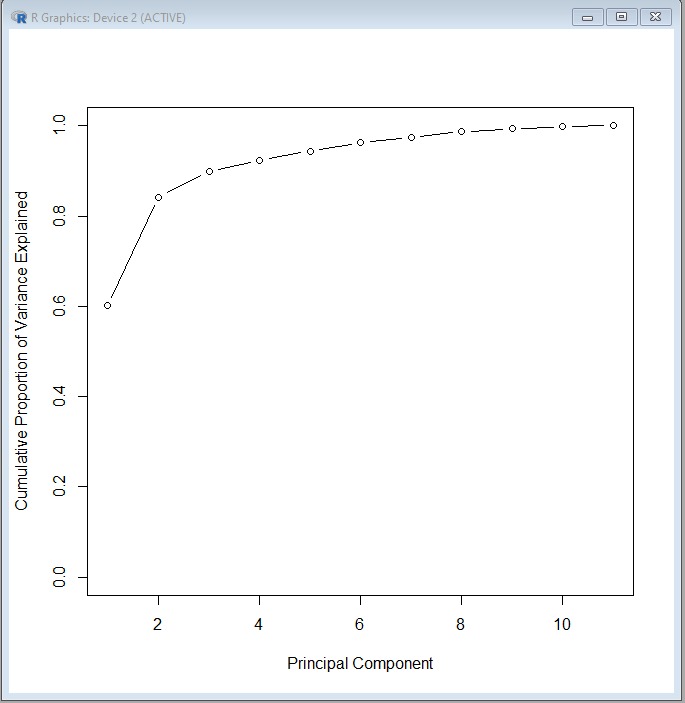
**Output:-**



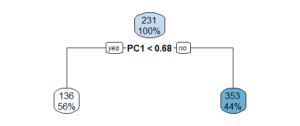
* The resultant principal components are plotted as **Biplot**. Scale value 0 represents that arrows are scaled representing loadings.
* **Variance explained for each principal component**



* **Scree Plot** represents the proportion of variance and a principal component. Below 2 principal components, there is a maximum proportion of variance as clearly seen in the plot.
* **Cumulative proportion of variance**

****

* **Scree Plot** represents the Cumulative proportion of variance and a principal component. Above 2 principal components, there is a maximum cumulative proportion of variance as clearly seen in the plot.
* **Decision tree model**



* **Decision tree** model was built to predict **disp** using other variables in the dataset and using ANOVA method. The decision tree plot is plotted and displays the information.

**Conclusion :-** Principal component analysis (PCA) to identify the most important variables that contribute to customer satisfaction is executed successfully.

Experiment No. – 5

Aim :- Decision tree analysis to understand the factors that influence customer churn and develop strategies to prevent it.

Theory :-

Customer churn occurs when customers or subscribers stop doing business with a company or service, also known as customer attrition. It is also referred as loss of clients or customers. One industry in which churn rates are particularly useful is the telecommunications industry, because most customers have multiple options from which to choose within a geographic location.

Similar concept with [predicting employee turnover](https://towardsdatascience.com/predict-employee-turnover-with-python-da4975588aa3), we are going to predict customer churn using [telecom dataset](https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/).

* customerID
* gender (female, male)
* SeniorCitizen (Whether the customer is a senior citizen or not (1, 0))
* Partner (Whether the customer has a partner or not (Yes, No))
* Dependents (Whether the customer has dependents or not (Yes, No))
* tenure (Number of months the customer has stayed with the company)
* PhoneService (Whether the customer has a phone service or not (Yes, No))
* MultipleLines (Whether the customer has multiple lines r not (Yes, No, No phone service)
* InternetService (Customer’s internet service provider (DSL, Fiber optic, No)
* OnlineSecurity (Whether the customer has online security or not (Yes, No, No internet service)
* OnlineBackup (Whether the customer has online backup or not (Yes, No, No internet service)
* DeviceProtection (Whether the customer has device protection or not (Yes, No, No internet service)
* TechSupport (Whether the customer has tech support or not (Yes, No, No internet service)
* streamingTV (Whether the customer has streaming TV or not (Yes, No, No internet service)
* streamingMovies (Whether the customer has streaming movies or not (Yes, No, No internet service)
* Contract (The contract term of the customer (Month-to-month, One year, Two year)
* PaperlessBilling (Whether the customer has paperless billing or not (Yes, No))
* PaymentMethod (The customer’s payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic)))
* MonthlyCharges (The amount charged to the customer monthly — numeric)
* TotalCharges (The total amount charged to the customer — numeric)
* Churn ( Whether the customer churned or not (Yes or No))

The raw data contains 7043 rows (customers) and 21 columns (features). The “Churn” column is our target.

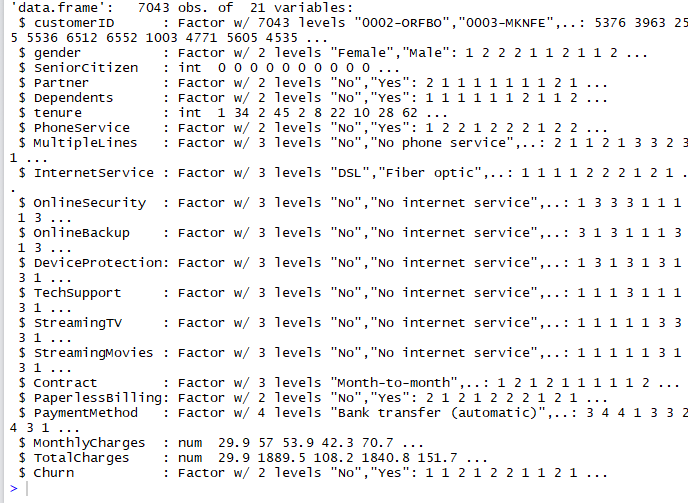
**Program :-**

**Data Preprocessing**

The data was downloaded from [IBM Sample Data Sets](https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/). Each row represents a customer, each column contains that customer’s attributes:

library(plyr)  
library(corrplot)  
library(ggplot2)  
library(gridExtra)  
library(ggthemes)  
library(caret)  
library(MASS)  
library(randomForest)  
library(party)churn <- read.csv('Telco-Customer-Churn.csv')  
str(churn)

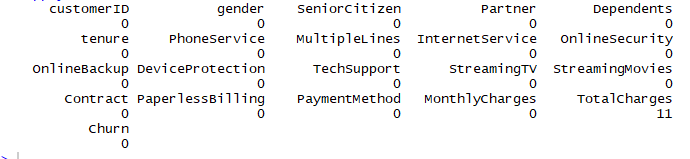
Output :-



We use sapply to check the number if missing values in each columns. We found that there are 11 missing values in “TotalCharges” columns. So, let’s remove all rows with missing values.

sapply(churn, function(x) sum(is.na(x)))

Output:-



churn <- churn[complete.cases(churn), ]

**Look at the variables, we can see that we have some wrangling to do**.

1. We will change “No internet service” to “No” for six columns, they are: “OnlineSecurity”, “OnlineBackup”, “DeviceProtection”, “TechSupport”, “streamingTV”, “streamingMovies”.

cols\_recode1 <- c(10:15)  
for(i in 1:ncol(churn[,cols\_recode1])) {  
churn[,cols\_recode1][,i] <- as.factor(mapvalues  
(churn[,cols\_recode1][,i], from =c("No internet service"),to=c("No")))  
}

2.We will change “No phone service” to “No” for column “MultipleLines”

churn$MultipleLines <- as.factor(mapvalues(churn$MultipleLines,   
from=c("No phone service"),to=c("No")))

3. Since the minimum tenure is 1 month and maximum tenure is 72 months, we can group them into five tenure groups: “0–12 Month”, “12–24 Month”, “24–48 Months”, “48–60 Month”, “> 60 Month”

min(churn$tenure); max(churn$tenure)

**Output:-**

[1] 1

**[1] 72**

group\_tenure <- function(tenure){  
 if (tenure >= 0 & tenure <= 12){  
 return('0-12 Month')  
 }else if(tenure > 12 & tenure <= 24){  
 return('12-24 Month')  
 }else if (tenure > 24 & tenure <= 48){  
 return('24-48 Month')  
 }else if (tenure > 48 & tenure <=60){  
 return('48-60 Month')  
 }else if (tenure > 60){  
 return('> 60 Month')  
 }  
}churn$tenure\_group <- sapply(churn$tenure,group\_tenure)  
churn$tenure\_group <- as.factor(churn$tenure\_group)

**4. Change the values in column “SeniorCitizen” from 0 or 1 to “No” or “Yes”.**

churn$SeniorCitizen <- as.factor(mapvalues(churn$SeniorCitizen,  
 from=c("0","1"),  
 to=c("No", "Yes")))

**5. Remove the columns we do not need for the analysis.**

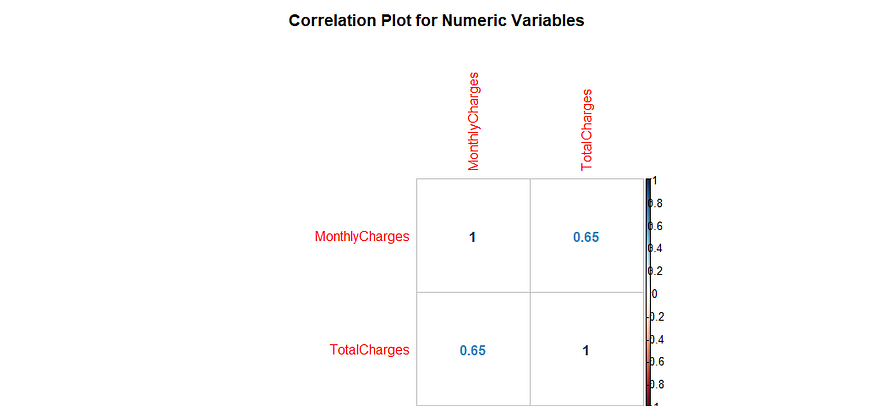
churn$customerID <- NULL  
churn$tenure <- NULL

# Exploratory data analysis and feature selection

**Correlation between numeric variables**

numeric.var <- sapply(churn, is.numeric)  
corr.matrix <- cor(churn[,numeric.var])  
corrplot(corr.matrix, main="\n\nCorrelation Plot for Numerical Variables", method="number")

**Output :-**



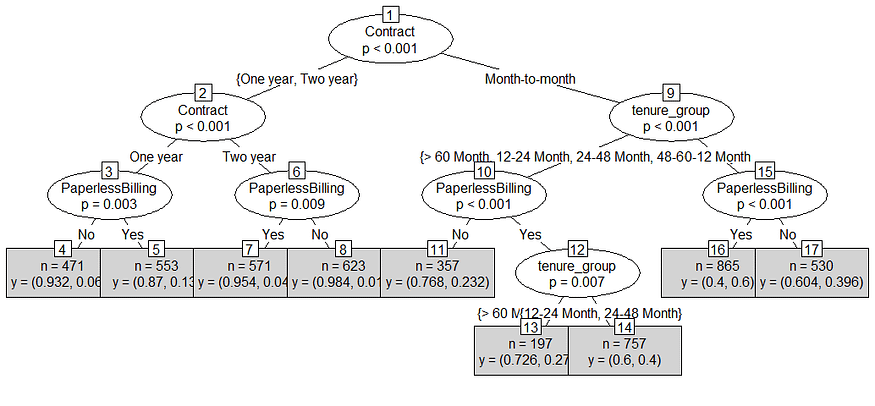
# ****Decision Tree****

# ****Decision Tree visualization****

we are going to use only three variables for plotting Decision Trees, they are “Contract”, “tenure\_group” and “PaperlessBilling”.

tree <- ctree(Churn~Contract+tenure\_group+PaperlessBilling, training)  
plot(tree)

**Output :-**

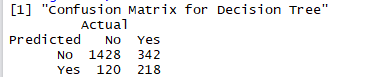


**Decision Tree Confusion Matrix**

We are using all the variables to product confusion matrix table and make predictions.

pred\_tree <- predict(tree, testing)  
print("Confusion Matrix for Decision Tree"); table(Predicted = pred\_tree, Actual = testing$Churn)

**Output :-**



**Decision Tree Accuracy**

p1 <- predict(tree, training)  
tab1 <- table(Predicted = p1, Actual = training$Churn)  
tab2 <- table(Predicted = pred\_tree, Actual = testing$Churn)  
print(paste('Decision Tree Accuracy',sum(diag(tab2))/sum(tab2)))

**Output:-**

[1] Decision Tree Accuracy 0.780834914611006

Conclusion :- Decision tree analysis to understand the factors that influence customer churn and develop strategies to prevent it is implemented successfully.

Experiment No. – 6

Aim :- Time series analysis to forecast the covid-19 pandemic situation .

**Theory :-**

**Time Series Analysis in R** is used to see how an object behaves over a period of time.

In [R Programming Language](https://www.geeksforgeeks.org/r-programming-language-introduction/), it can be easily done by the **ts()** function with some parameters. Time series takes the data vector and each data is connected with a timestamp value as given by the user. This function is mostly used to learn and forecast the behavior of an asset in business for a period of time. For example, sales analysis of a company, inventory analysis, price analysis of a particular stock or market, population analysis, etc.

**Syntax:**  objectName <- ts(data, start, end, frequency)

**where,**

* **data** – represents the data vector
* **start** – represents the first observation in time series
* **end** – represents the last observation in time series
* **frequency** – represents number of observations per unit time. For example, frequency=1 for monthly data.

Taking the total number of positive cases of COVID-19 cases weekly from 22 January 2020 to 15 April 2020 the world in data vector.

**Program :-**

# Weekly data of COVID-19 positive cases from

# 22 January, 2020 to 15 April, 2020

x <- c(580, 7813, 28266, 59287, 75700,

87820, 95314, 126214, 218843, 471497,

936851, 1508725, 2072113)

# library required for decimal\_date() function

library(lubridate)

# output to be created as png file

png(file ="timeSeries.png")

# creating time series object

# from date 22 January, 2020

mts <- ts(x, start = decimal\_date(ymd("2020-01-22")), frequency = 365.25 / 7)

# plotting the graph

plot(mts, xlab ="Weekly Data",

ylab ="Total Positive Cases",

main ="COVID-19 Pandemic",

col.main ="darkgreen")

# saving the file

dev.off()

**Output:-**

