**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **SR. NO.** | **NAME OF THE PRACTICAL** | **DATE** | **SIGN** |
|  | K-Means Clustering | 10/06/2023 |  |
|  | Apriori Algorithm | 10/06/2023 |  |
|  | Regression  A. Simple Linear Regression  B. Logistic Regression  C. Multiple Regression | 10/06/2023 |  |
|  | Decision Tree | 15/06/2023 |  |
|  | Naïve Bayes Classification | 15/06/2023 |  |
|  | SVM Classification | 15/06/2023 |  |
|  | Text Analysis | 17/06/2023 |  |
|  | Install, Configure and Run Hadoop and HDFS | 17/06/2023 |  |
|  | MapReduce Implementation | 21/06/2023 |  |
|  | Implement an application that stores Big Data in MongoDB and manipulate it using Python | 21/06/2023 |  |

**PRACTICAL NO.1: K-MEANS CLUSTERING**

**FILE USED:** Mall\_Customers.csv

**CODE:**

# Importing the dataset

**dataset=read.csv(‘E:\\BDAPractical\\Mall\_Customers.csv’)**

#Displays contents-6 rows by default

**head(dataset)**

#Column 4th& 5th

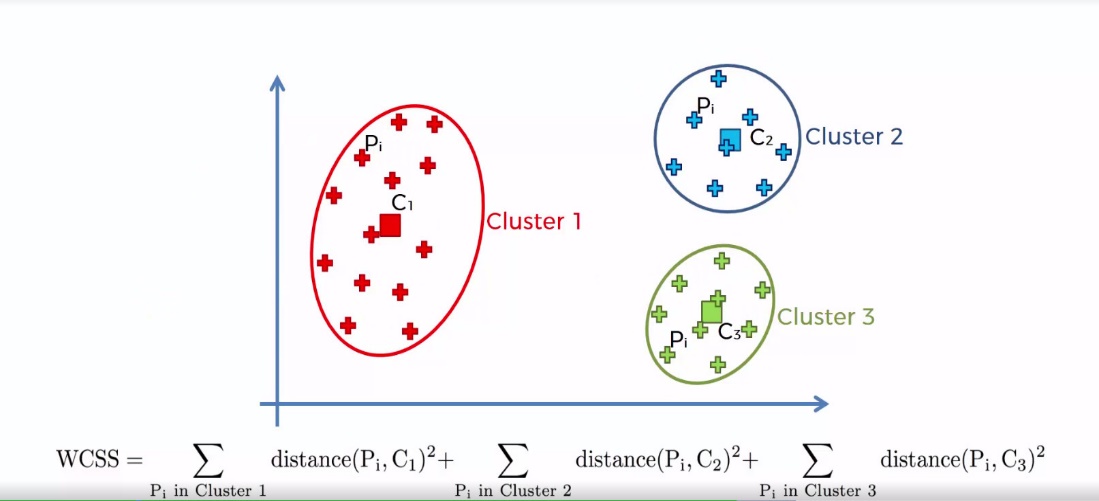
**dataset = dataset[4:5]**

#Displays contents-6 rows by default

**head(dataset)**

#within cluster-sum of squares

**wcss = vector()**



**#**It results in a vector with a number for each cluster

**for (i in 1:10) wcss[i] = sum(kmeans(dataset, i)$withinss)**

#plot(x,y,type=’b’ indicates both points and lines)

**plot(1:10,**

**wcss,**

**type = 'b',**

**main = paste('The Elbow Method'),**

**xlab = 'Number of clusters',**

**ylab = 'WSS')**

# Fitting K-Means to the dataset with no of clusters = 5

**kmeans = kmeans(x = dataset, centers = 5)**

#vector of number ranging from 1 to 5

**y\_kmeans = kmeans$cluster**->

# Visualising the clusters

**library(cluster)**

**clusplot(dataset,**->x

**y\_kmeans,**->y

**lines = 0,**-> no distance lines will appear on plot

**shade = TRUE,**-> according to density

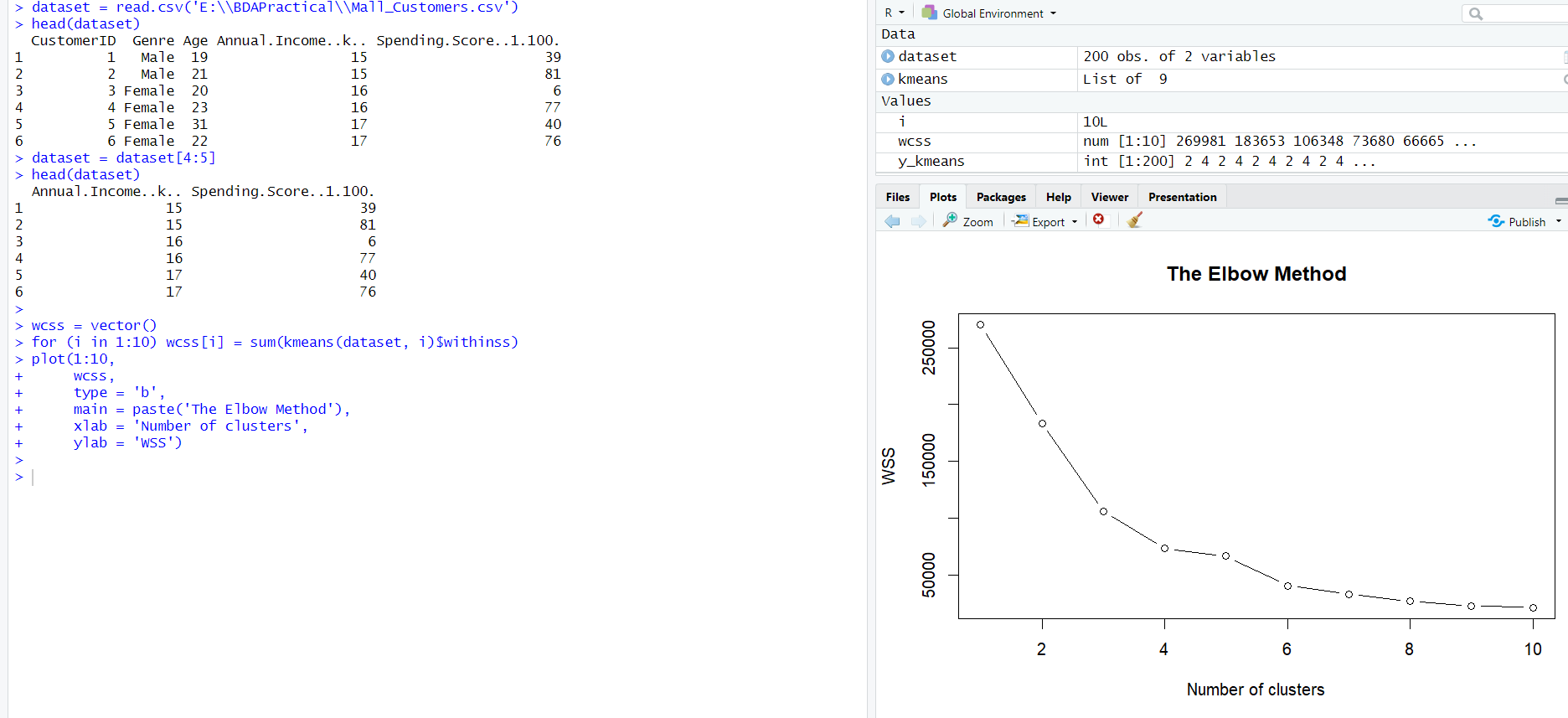
**color = TRUE,** -> according to density

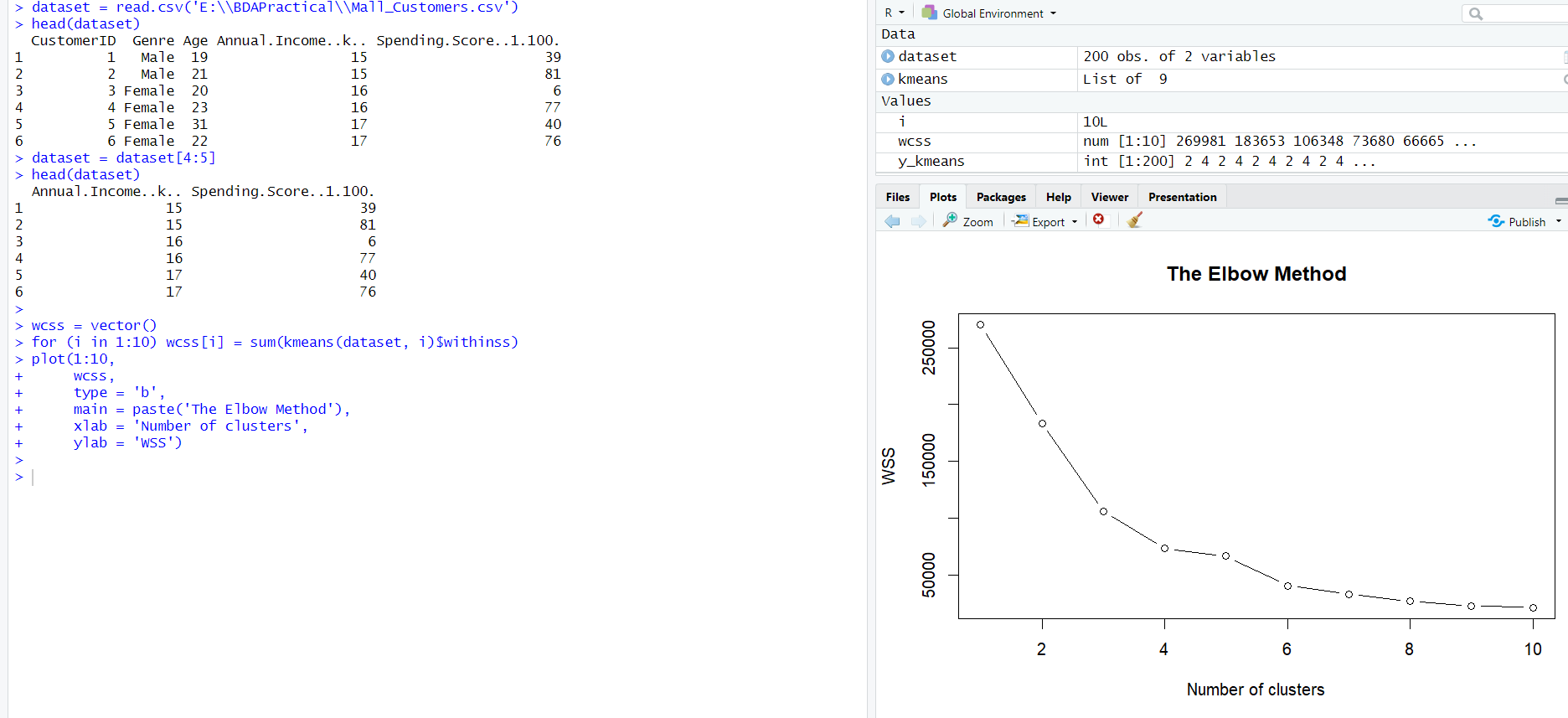
**labels = 2,**-> all points and ellipses are labelled in the plot

**main = paste('Clusters of customers'),**

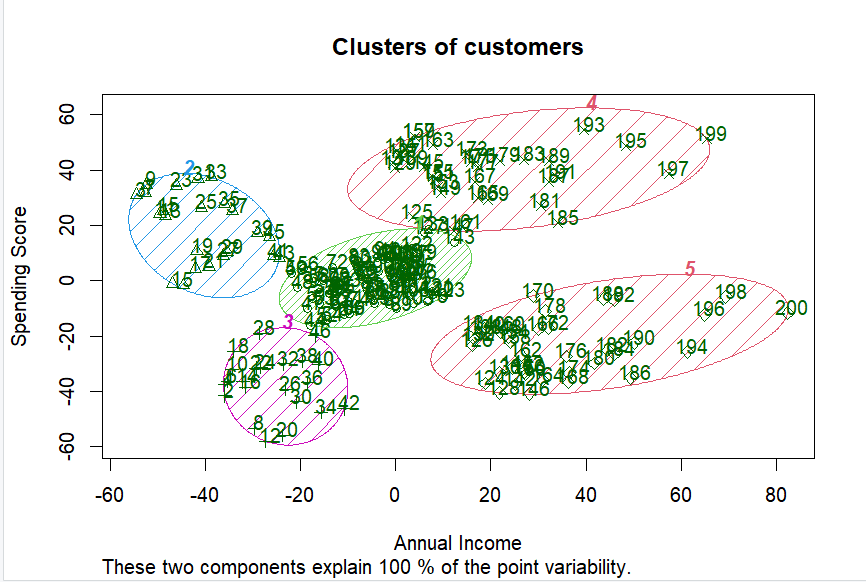
**xlab = 'Annual Income',**

**ylab = 'Spending Score')**

****

****

**OUTPUT:**

****

**PRACTICAL NO.2: APRIORI ALGORITHM**

**FILE USED:** Groceries (Inbuilt)

**CODE:**

**install.packages("arules")**⇒Representing, Manipulating & Analyzing Data & Patterns

**install.packages("arulesViz")**⇒Visualizing Association Rules & Frequent Itemsets

**install.packages("RColorBrewer")**⇒Palette that provides color schemes for maps and other graphics

**library(arules)**

**library(arulesViz)**

**library(RColorBrewer)**

**data("Groceries")**⇒Predefined in RPackage with 9835 records

**Groceries**⇒Display ->9835 rows-> 169 cols

**summary(Groceries)** ⇒Gives statistical information of dataset

**class(Groceries)**⇒Transaction (Returns class attribute of dataset provided)

**rules = apriori(Groceries, parameter = list(supp = 0.02, conf = 0.2))**

⇒Transaction Data, Min. Support, Min. Confidence, For Itemset-1

**summary(rules)**⇒Gives statistical information of rules

**inspect(rules[1:10])**⇒Prints the first 10 strong association rules

**arules::itemFrequencyPlot(Groceries, topN = 20,**⇒Bar Plot for itemFrequencies, 20 items will be plotted

**col = brewer.pal(8, 'Pastel2'),**8-> color with repetition (3 to 8), Pastel2->color scheme

**main = 'Relative Item Frequency Plot',**

**type = "relative",**⇒Considers %value

**ylab = "Item Frequency(Relative)")**

**itemset = apriori(Groceries, parameter = list(minlen=2, maxlen=2, support=0.02, target="frequent itemset") )**

⇒Transaction Data, Get itemset of Length 2, For Itemset-2

**summary(itemset)**

**inspect(itemset[1:10])**⇒10 values

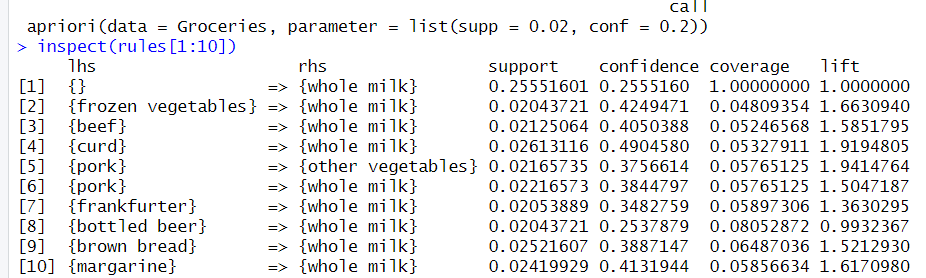
**itemsets\_3 = apriori(Groceries, parameter = list(minlen=3, maxlen=3, support=0.02, target="frequent itemset"))**

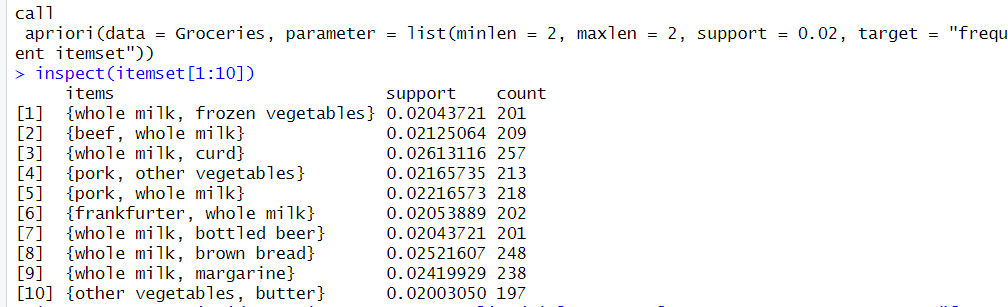
⇒Transaction Data, Get itemset of Length 3, For Itemset-3

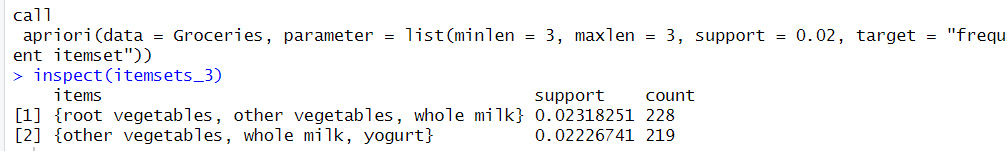
**summary(itemsets\_3)**

**inspect(itemsets\_3)**

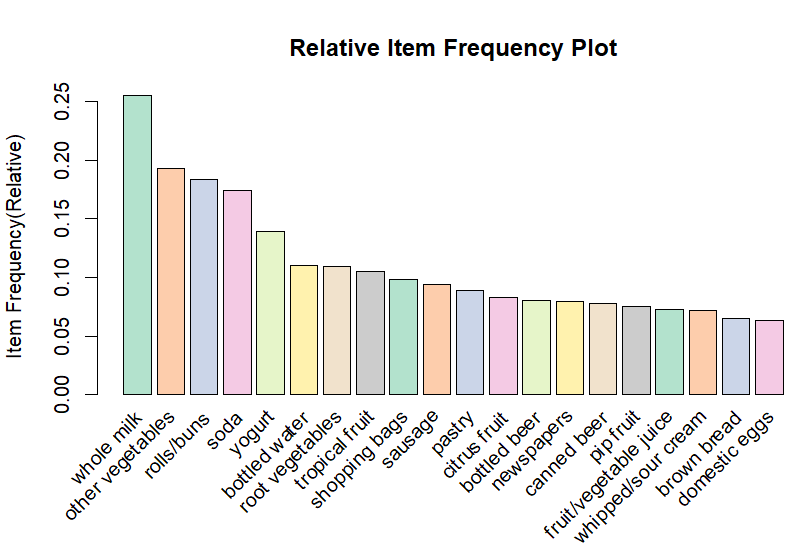
⇒2 values as a result displayed





****

**OUTPUT:**



**PRACTICAL NO.3: REGRESSION**

1. **LINEAR REGRESSION**

**CODE:**

**years\_of\_exp = c(7,5,1,3,5,8,10,12,20,2)**⇒Combine Values in a vector/list

**salary\_in\_lakhs = c(21,13,6,8,13,22,25,27,40,7)**

**employee.data = data.frame(years\_of\_exp, salary\_in\_lakhs)**⇒Data displayed in table format

**employee.data**

# Estimation of the salary of an employee, based on his year of experience.

**model <- lm(salary\_in\_lakhs ~ years\_of\_exp, data = employee.data)**

⇒linear models, formula, data (relationship between both)

**summary(model)**

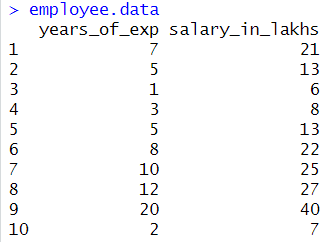
# Visualization of Regression

**plot(years\_of\_exp,salary\_in\_lakhs,data = employee.data,col = "blue",main = "Years of Experience vs Salary in Lakhs", abline(model),xlab = "Years of Experience",ylab = "Salary in Lakhs")**

⇒color of bubble

⇒vertical/ horizontal/ regression lines to a graph

**OUTPUT:**





1. **LOGISTIC REGRESSION**

**FILE USED:**studentmarks.csv

**CODE:**

#fetch the data

**college <- read.csv("E:\\BDAPractical\\studentmarks.csv")**⇒ Assigned to college (400 rows, 4 cols)

**head(college)**⇒ First 6 values

**nrow(college)**⇒ Number of rows

**install.packages("caTools")** # For Logistic regression⇒ Contains several basic utility functions

**library(caTools)**

**split <- sample.split(college, SplitRatio = 0.75)**⇒75% Training Set, 25% Testing Set

**split**

**training\_reg<- subset(college, split == "TRUE")**

**test\_reg<- subset(college, split == "FALSE")**

⇒admit gregpa rank

# Training model

**fit\_logistic\_model<- glm(admit ~ .,**⇒Generalized Linear Model

**data = training\_reg,**

**family = "binomial")**⇒Dependent variable is binary

# Predict test data based on model

**predict\_reg<- predict(fit\_logistic\_model,**

**test\_reg, type = "response")**⇒Output probabilities in normal scale

**predict\_reg**

**cdplot(as.factor(admit)~ gpa, data=college)**⇒Conditional Densities of y changes over variable x

**cdplot(as.factor(admit)~ gre, data=college)**

**cdplot(as.factor(admit)~ rank, data=college)**

⇒single value/ returns factor object

# Changing probabilities

**predict\_reg<- ifelse(predict\_reg>0.5, 1, 0)**⇒Conditional Operator

**predict\_reg**

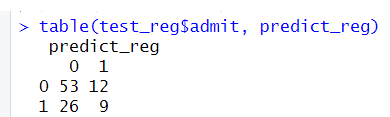
# Evaluating model accuracy

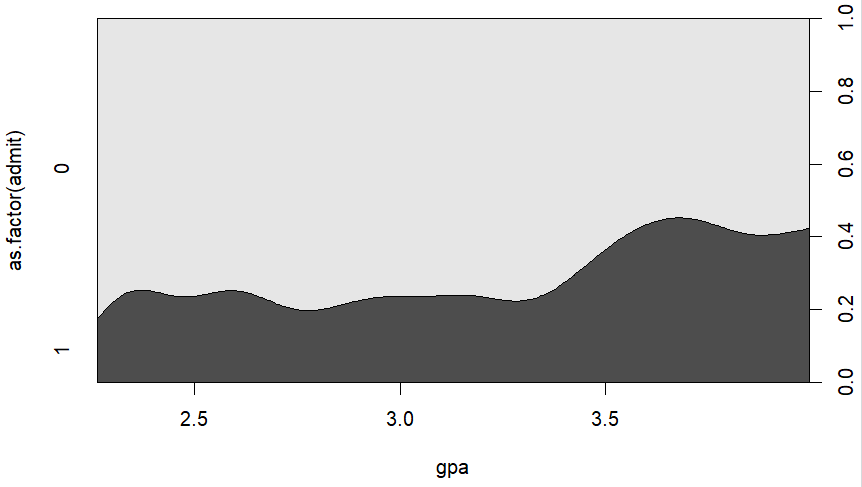
# using confusion matrix

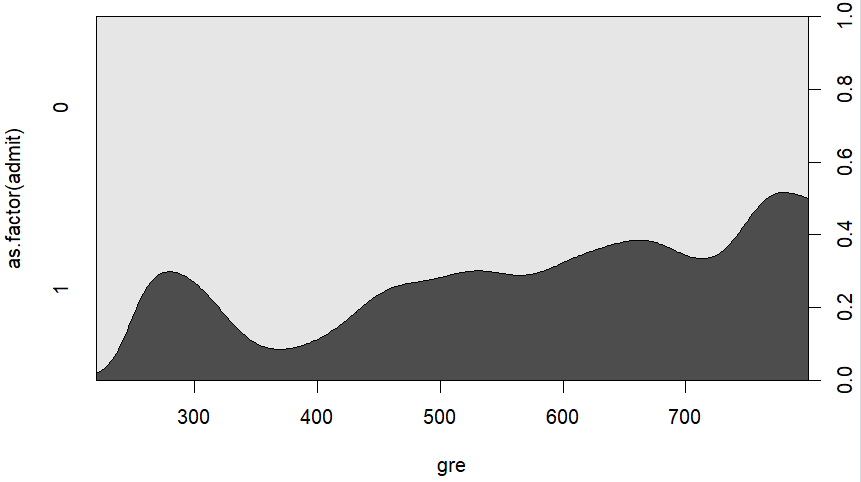
**table(test\_reg$admit, predict\_reg)**

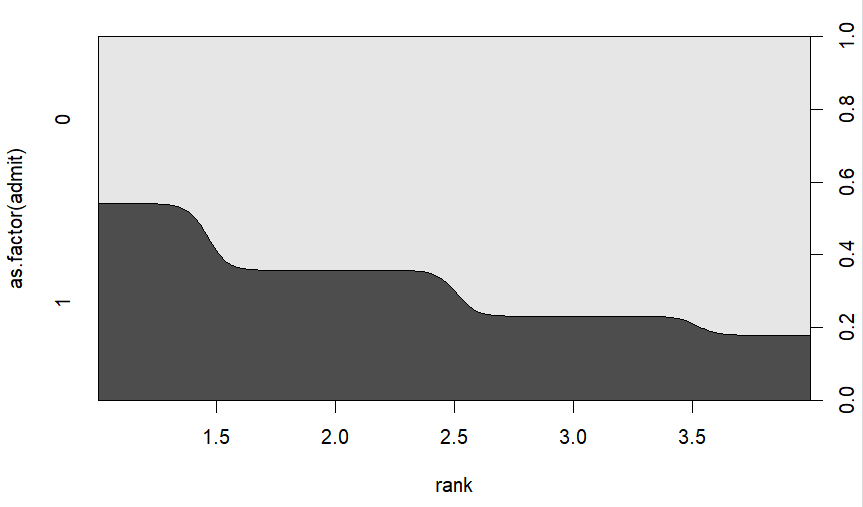
⇒Total 100 values

**OUTPUT:**

****







1. **MULTIPLE REGRESSION**

**FILE USED:**studentmarks.csv

**CODE:**

#fetch the data

**StudentData<- read.csv("E:\\BDAPractical\\studentmarks.csv")**

**head(StudentData)**

**nrow(StudentData)**

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

**library(caTools)**

**split <- sample.split(StudentData, SplitRatio = 0.75)**

**split**

**training\_reg<- subset(StudentData, split == "TRUE")**

**test\_reg<- subset(StudentData, split == "FALSE")**

# Training model

**fit\_MRegressor\_model<- lm(formula = admit ~ gre+gpa+rank,**

**data = training\_reg)**

# Predict test data based on model

**predict\_reg<- predict(fit\_MRegressor\_model,**

**newdata = test\_reg)**

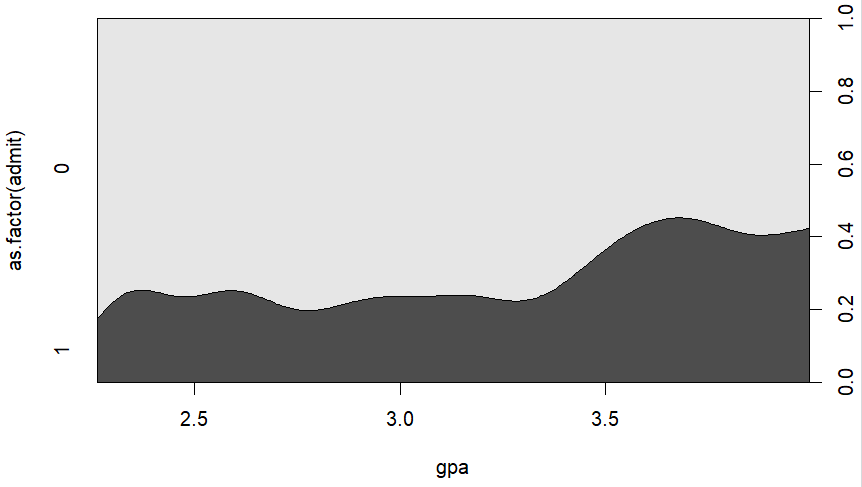
**predict\_reg**

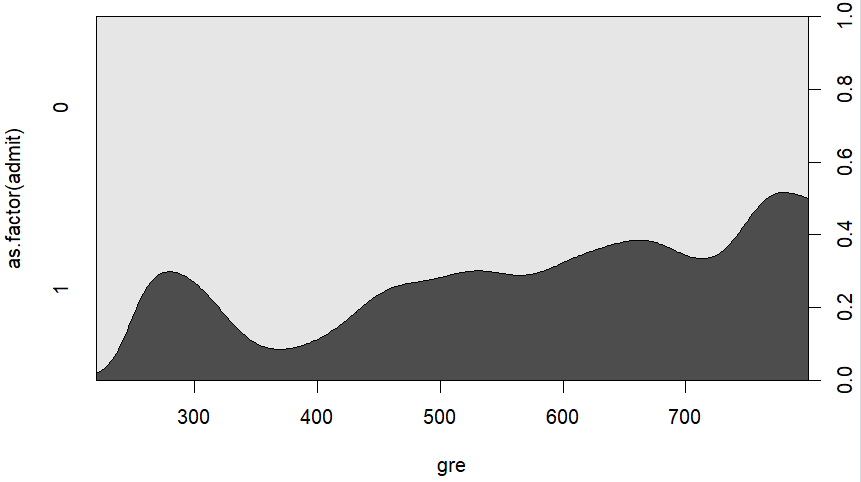
**cdplot(as.factor(admit)~ gpa, data=StudentData)**

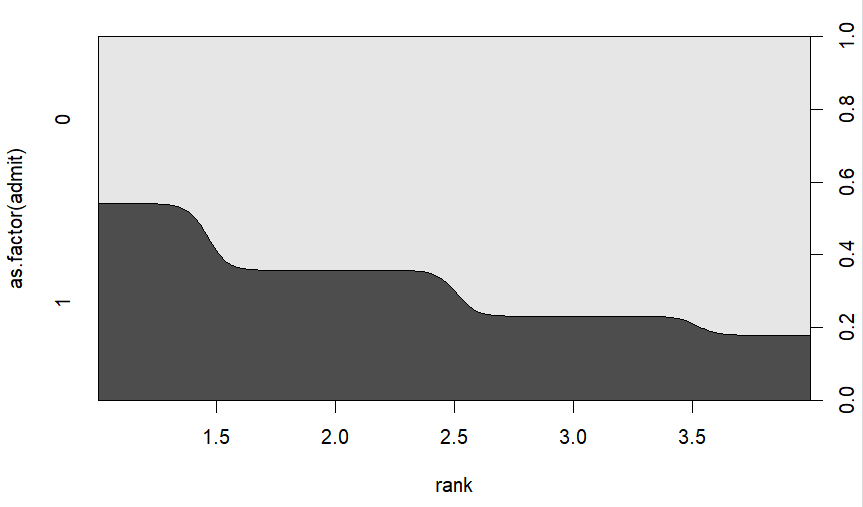
**cdplot(as.factor(admit)~ gre, data=StudentData)**

**cdplot(as.factor(admit)~ rank, data=StudentData)**

**OUTPUT:**







**PRACTICAL NO.4: DECISION TREE**

**FILE USED:** DTdata.csv

**CODE:**

**install.packages("rpart")** #install packages for modeling decision trees

**install.packages("rpart.plot")** #install packages for plotting decision trees

#load libraries

**library(rpart)**

**library(rpart.plot)**

#header and sep for proper alignment and indentation

**play\_decision<-read.table("E:\\BDAPractical\\DTdata.csv",header=TRUE,sep=",")**

**play\_decision**

**summary(play\_decision)**

#Play-OutputVariable, Outlook+Temperature+Humidity+Wind-InputVariable

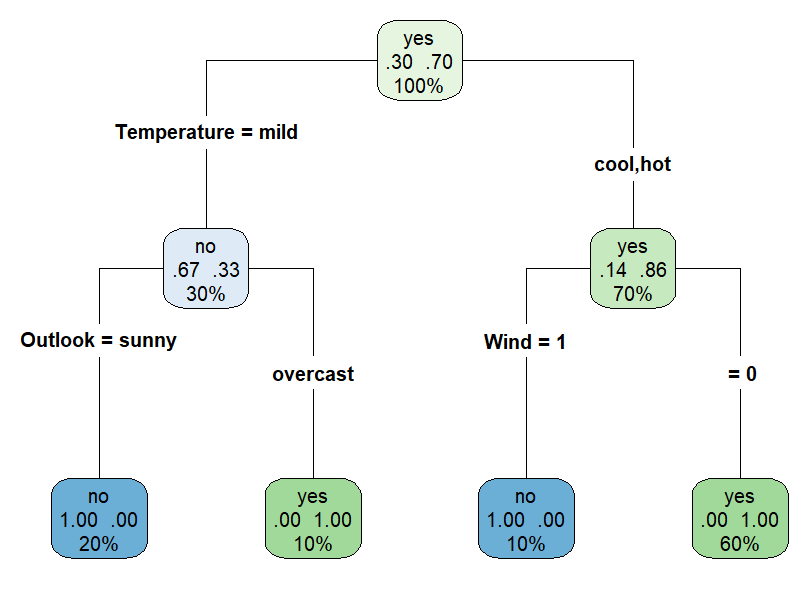
#rpart(formula, classification tree, dataset, controls the tree growth, purity measure-with split either information or gini

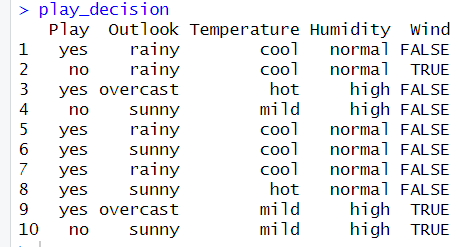
**fit<- rpart (Play~Outlook+Temperature+Humidity+Wind,method="class",data=play\_decision,control=rpart.control(minsplit=1),parms=list(split='information'))**

**summary(fit)**

**rpart.plot(fit,type=4,extra=104)**

**OUTPUT:**





|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Play | Outlook | Temperature | Humidity | Wind | | yes | rainy | cool | normal | FALSE | | yes | overcast | hot | high | FALSE | | yes | rainy | cool | normal | FALSE | | yes | sunny | cool | normal | FALSE | | yes | rainy | cool | normal | FALSE | | yes | sunny | hot | normal | FALSE | | yes | overcast | mild | high | TRUE | | **100% data**  Play=yes  Left(No)-3/10=0.30  Right(Yes)-7/10=0.70 |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Play | Outlook | Temperature | Humidity | Wind | | no | sunny | mild | high | FALSE | | yes | overcast | mild | high | TRUE | | no | sunny | mild | high | TRUE | | **Temperature=mild**  Total Values=3 (30%) with  Left-2/3 play(no)=0.67,  Right-1/3 play(yes)=0.33 |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Play | Outlook | Temperature | Humidity | Wind | | yes | rainy | cool | normal | FALSE | | no | rainy | cool | normal | TRUE | | yes | overcast | hot | high | FALSE | | yes | rainy | cool | normal | FALSE | | yes | sunny | cool | normal | FALSE | | yes | rainy | cool | normal | FALSE | | yes | sunny | hot | normal | FALSE | | **Temperature=cool, hot**  Total Values=7 (70%) with  Left-6/7 play(yes)=0.86  Right-1/7 play(no)=0.14 |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Play | Outlook | Temperature | Humidity | Wind | | no | sunny | mild | high | FALSE | | yes | overcast | mild | high | TRUE | | no | sunny | mild | high | TRUE | | **Temperature=Mild & Outlook=sunny**  Total Values=2(20%)  no=1  yes=0  **Temperature=Mild & Outlook=overcast**  Total Values=1(10%)  no=0  yes=1 |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Play | Outlook | Temperature | Humidity | Wind | | yes | rainy | cool | normal | FALSE | | no | rainy | cool | normal | TRUE | | yes | overcast | hot | high | FALSE | | yes | rainy | cool | normal | FALSE | | yes | sunny | cool | normal | FALSE | | yes | rainy | cool | normal | FALSE | | yes | sunny | hot | normal | FALSE | | **Temperature=Cool, Hot & WIND= 1(TRUE)**  Total Values=1(10%)  no=1  yes=0  **Temperature=Cool, Hot & WIND= 0(FALSE)**  Total Values=6(60%)  no=0  yes=1 |

**PRACTICAL NO.5: NAÏVE BAYES CLASSIFICATION**

**FILE USED:** sample1.csv

**CODE:**

**install.packages("e1071")** #The e1071 package contains the naiveBayes function. It allows numeric and factor variables to be used in the naive bayes model.

**library(e1071)** #load the library

**sample<-read.table("E:\\BDAPractical\\sample1.csv",header=TRUE,sep=",")** #read the data into a table from the file

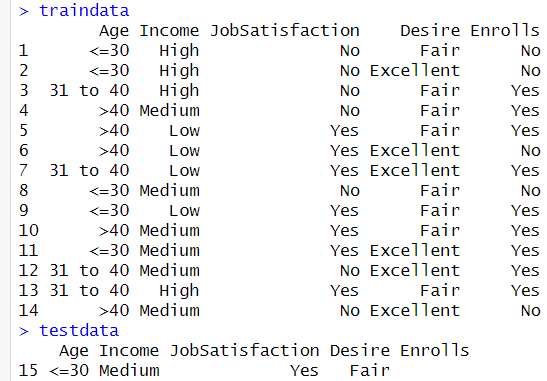
# define the data frames for the Naive Bayes classifier

**traindata<-as.data.frame(sample[1:14,])**

**testdata<-as.data.frame(sample[15,])**

**traindata**

**testdata**



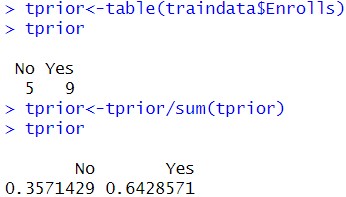
#Compute the prior probabilities P(c) for Enrolls, where C = {Yes,No}.

**tprior<-table(traindata$Enrolls)**

**tprior**

**tprior<-tprior/sum(tprior)**

**tprior**



#compute conditional probabilities P(A|C), where A ={Age, Income, JobSatisfaction,Desire} and C ={Yes, No}.

**ageCounts<-table(traindata[,c("Enrolls","Age")])**

**ageCounts<-ageCounts/rowSums(ageCounts)**

**ageCounts**

**incomeCounts<-table(traindata[,c("Enrolls","Income")])**

**incomeCounts<-incomeCounts/rowSums(incomeCounts)**

**incomeCounts**

**jsCounts<-table(traindata[,c("Enrolls","JobSatisfaction")])**

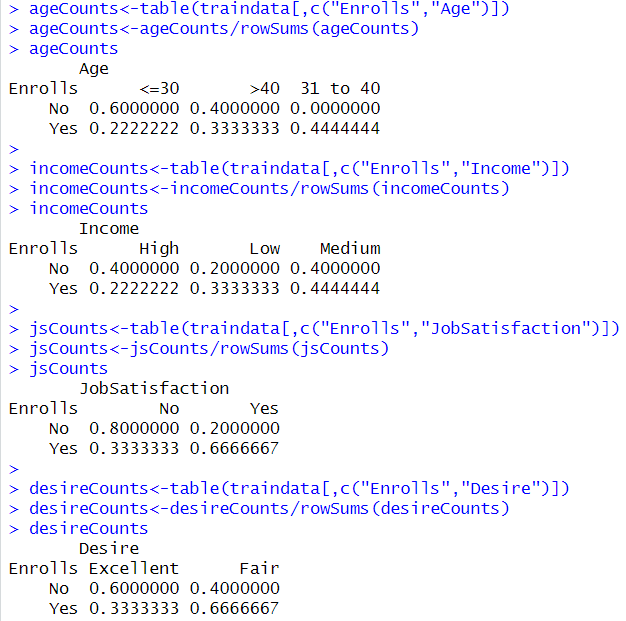
**jsCounts<-jsCounts/rowSums(jsCounts)**

**jsCounts**

**desireCounts<-table(traindata[,c("Enrolls","Desire")])**

**desireCounts<-desireCounts/rowSums(desireCounts)**

**desireCounts**



#probability P(c|A) is determined by the product of P(a|c,) times the (c1) where c1 =Yes and c2 =No.

prob\_Yes<-**ageCounts["Yes",testdata[,c("Age")]]\*incomeCounts["Yes",testdata[,c("Income")]]\*jsCounts["Yes",testdata[,c("JobSatisfaction")]]\*desireCounts["Yes",testdata[,c("Desire")]]\*tprior["Yes"]**

**prob\_Yes**



**prob\_No<-ageCounts["No",testdata[,c("Age")]]\*incomeCounts["No",testdata[,c("Income")]]\*jsCounts["No",testdata[,c("JobSatisfaction")]]\*desireCounts["No",testdata[,c("Desire")]]\*tprior["No"]**

**prob\_No**



#The larger value of P(Yes|A) and P(No|A) determines the predicted result of the output variable.

**max(prob\_Yes,prob\_No)**

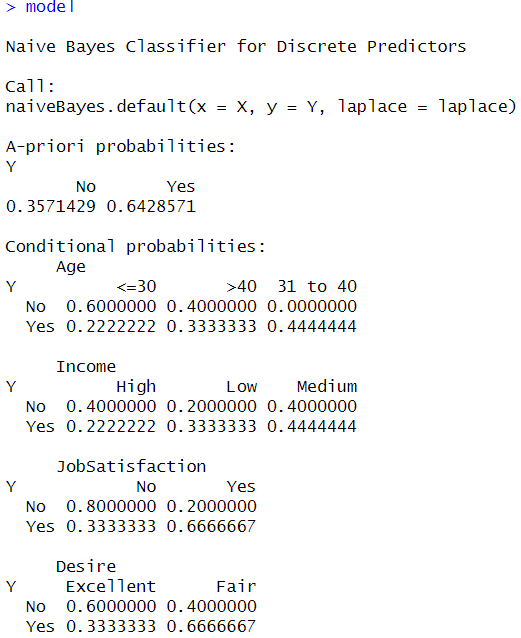


#Naive Bayes function computes the conditional probabilities. The function takes the form of naive Bayes (formula, data,… ), where the arguments are defined as follows.

# formula: A formula of the form class ~ xl + x2 + ... assuming xl, x2 ... are conditionally independent & data: A data frame of factors

**model<-naiveBayes(Enrolls~Age+Income+JobSatisfaction+Desire,traindata)**

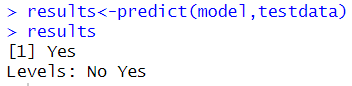
**model**



#predicting the outcome of Enrolls with the test data shows the result is Enrolls= Yes

**results<-predict(model,testdata)**

**results**



**PRACTICAL NO.6: SVM CLASSIFICATION**

**CODE:**

**set.seed(10111)** #to make sure that we get the same results for randomization

#a matrix x is created using the matrix() function. The rnorm() function generates 40 random numbers. These numbers are arranged into a matrix with 20 rows and 2 columns.

**x = matrix(rnorm(40), 20, 2)**

#A vector y is created using the rep() function. It contains 20 elements, with the first 10 being -1 and the second 10 being 1.

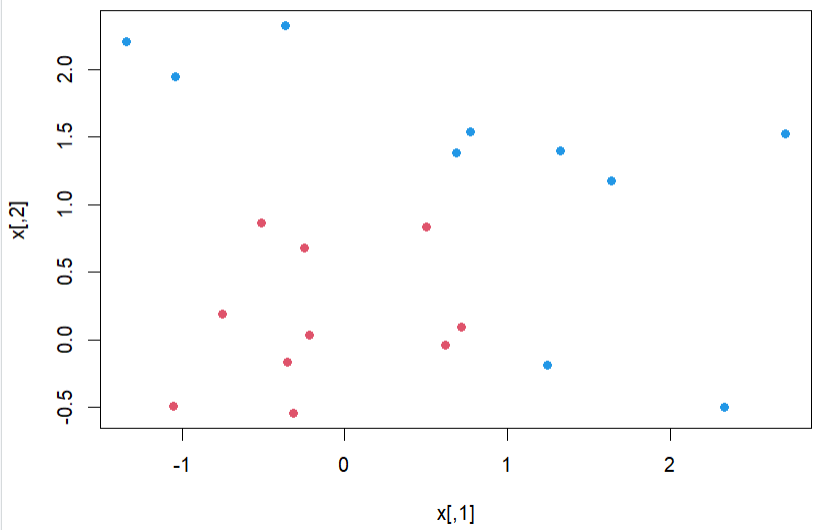
**y = rep(c(-1, 1), c(10, 10))**

#The code modifies values of x for the rows where y = 1. The rows are selected using the logical expression y == 1, and the values in those rows are increased by 1 using the + operator.

**x[y == 1,] = x[y == 1,] + 1**

#the plot() function is used to create a scatter plot of the data. The col argument sets the color of the points based on the values in y, with -1 being blue and 1 being red. The pch argument sets the shape of the points to a filled circle.

**plot(x, col = y + 3, pch = 19)**

****

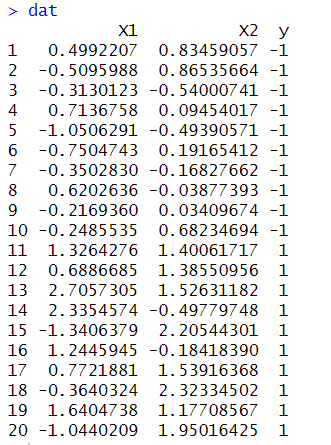
#e1071 package provides functions for statistical learning and data mining.

**library(e1071)**

#creates a data frame called "dat" with two columns: "x" and "y". The "x" column is assumed to already exist in the workspace, while the "y" column is created by converting an existing variable "y" into a factor using the "as.factor()" function.

**dat = data.frame(x, y = as.factor(y))**

**dat**

****

#The formula "y ~ ." specifies that the response variable is "y". The "kernel" argument specifies that a linear kernel should be used, while the "cost" argument sets the cost parameter to 10. The "scale" argument is set to FALSE, which means that the data will not be scaled before fitting the model.

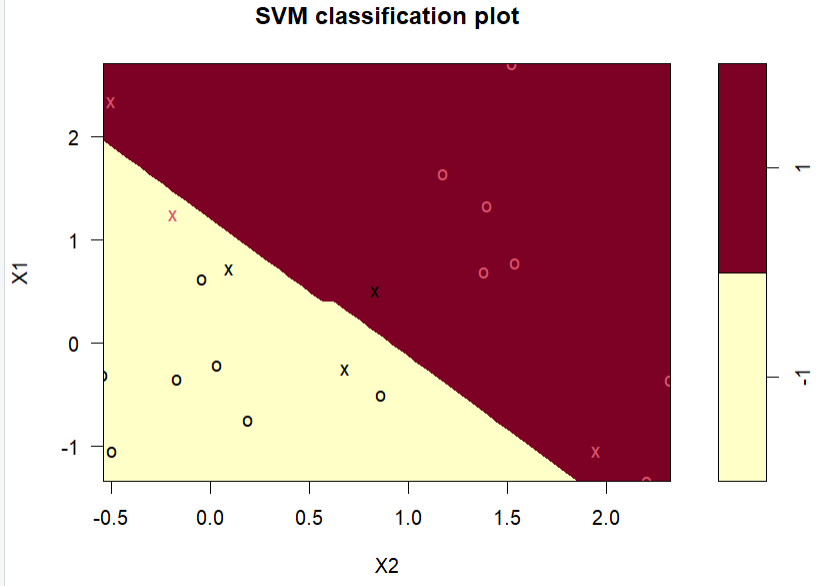
**svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)**

#prints the SVM model object to the console

**print(svmfit)**

#The plot will show the decision boundary of the SVM model and the support vectors. The svmfit object is the result of fitting an SVM model to the data dat. The dat object contains the data used to fit the SVM model.

**plot(svmfit, dat)**

****

**PRACTICAL NO.7: TEXT ANALYSIS**

**STEPS:**

1. **INSTALLING AND LOADING R PACKAGES**
2. **READING FILE DATA INTO R**
3. **CLEANING UP TEXT DATA**
4. **BUILDING THE TERM DOCUMENT MATRIX**
5. **GENERATE THE WORD CLOUD**
6. **WORD ASSOCIATION**
7. **SENTIMENT SCORES**
8. **EMOTION CLASSIFICATION**
9. **INSTALLING AND LOADING R PACKAGES:**

# Install

**install.packages("tm")** # for text miningoperations like removing numbers, special characters, punctuations and stop words

**install.packages("SnowballC")** # for text stemming, which is the process of reducing words to their base or root form.

**install.packages("wordcloud")** # word-cloud generator

**install.packages("RColorBrewer")** # color palettes

**install.packages("syuzhet")** # for sentiment scores and emotion classification

**install.packages("ggplot2")** # for plotting graphs

# Load

**library("tm")**

**library("SnowballC")**

**library("wordcloud")**

**library("RColorBrewer")**

**library("syuzhet")**

**library("ggplot2")**

1. **READING FILE DATA INTO R:**

# Read the text file from local machine, choose file interactively

**text <- readLines("E://BDAPractical//textanalysissample.txt")**

# Load the vector as a corpus(Collection of text documents to apply text mining) and storing in TextDoc variable

**TextDoc<- Corpus(VectorSource(text))**

1. **CLEANING UP TEXT DATA:**

#tm\_map() function to replace special characters like /, @ and | with a space.

#content\_transformer is a function which modifies the content of an R object and gsub(pattern, replacement, x) is a global substitution.

**toSpace<- content\_transformer(function (x , pattern ) gsub(pattern, " ", x))**

**TextDoc<- tm\_map(TextDoc, toSpace, "/")**

**TextDoc<- tm\_map(TextDoc, toSpace, "@")**

**TextDoc<- tm\_map(TextDoc, toSpace, "\\|")**

# Convert the text to lower case

**TextDoc<- tm\_map(TextDoc, content\_transformer(tolower))**

# Remove numbers

**TextDoc<- tm\_map(TextDoc, removeNumbers)**

# Remove english common stopwords (the, is, at, on)

**TextDoc<- tm\_map(TextDoc, removeWords, stopwords("english"))**

# Remove your own stop word

# specify your custom stopwords as a character vector

**TextDoc<- tm\_map(TextDoc, removeWords, c("s", "company", "team"))**

# Remove punctuations

**TextDoc<- tm\_map(TextDoc, removePunctuation)**

# Eliminate extra white spaces

**TextDoc<- tm\_map(TextDoc, stripWhitespace)**

# Text stemming - which reduces words to their root form

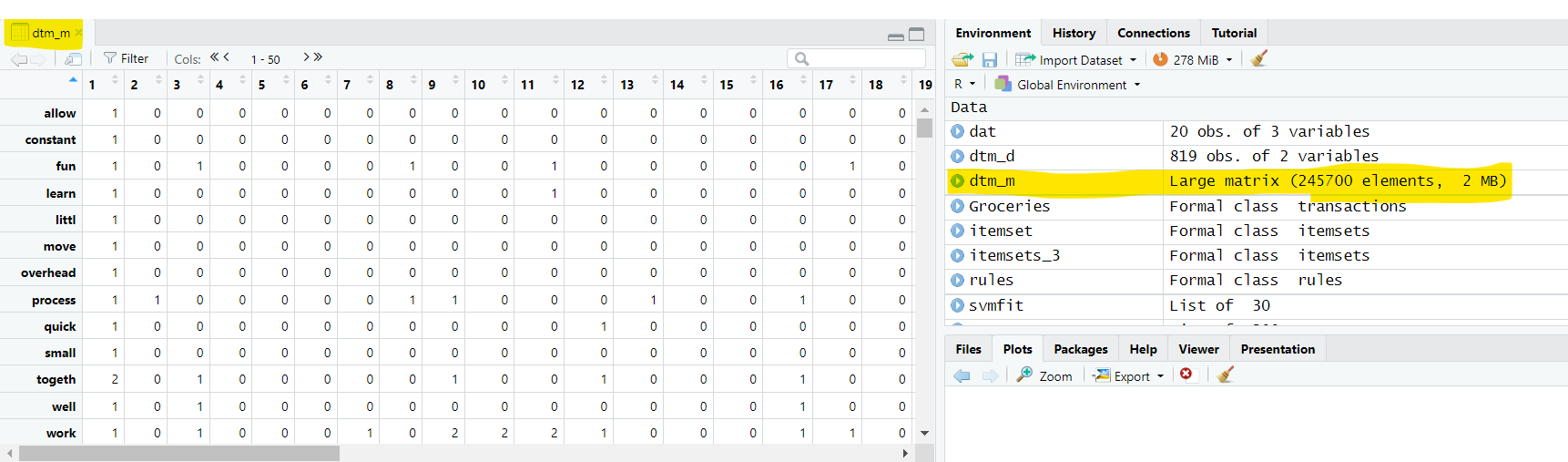
**TextDoc<- tm\_map(TextDoc, stemDocument)**

1. **BUILDING THE TERM DOCUMENT MATRIX:**

# TermDocumentMatrix() helps to build a Document Matrix, i.e. a table containing the frequency of words.

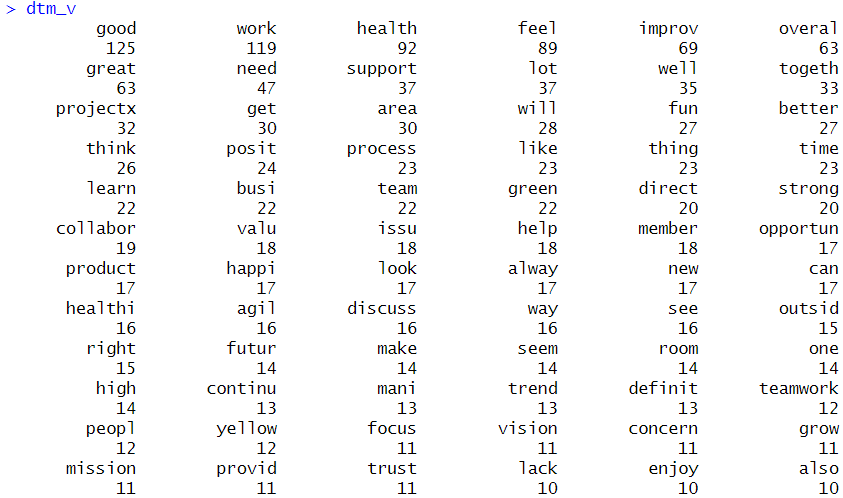
**TextDoc\_dtm<- TermDocumentMatrix(TextDoc)**

**dtm\_m<- as.matrix(TextDoc\_dtm)**

****

# Sort by decreasing value of frequency

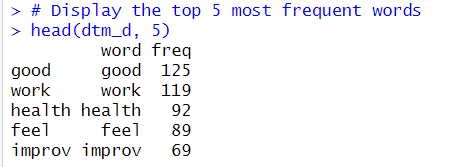
**dtm\_v<- sort(rowSums(dtm\_m),decreasing=TRUE)**

****

**dtm\_d<- data.frame(word = names(dtm\_v),freq=dtm\_v)**

# Display the top 5 most frequent words

**head(dtm\_d, 5)**

****

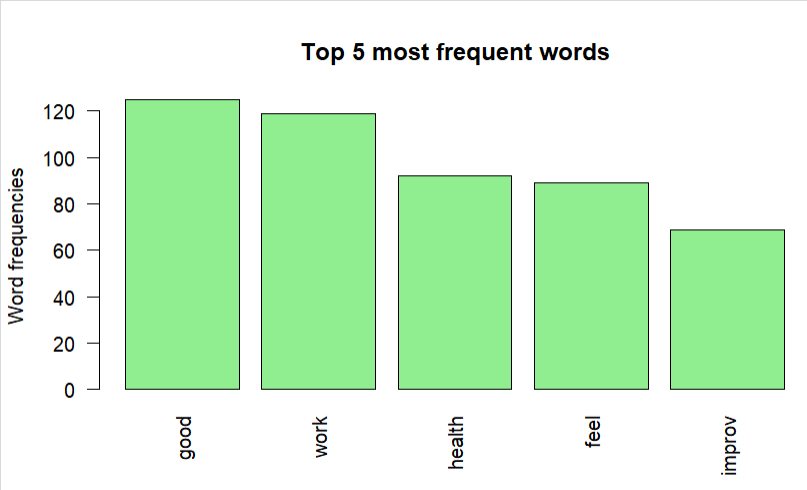
# Plot the most frequent words

#barplot(starting 5 frequent values, style of axis labels,x labels, color, title, y labels)

**barplot(dtm\_d[1:5,]$freq, las = 2, names.arg = dtm\_d[1:5,]$word,**

**col ="lightgreen", main ="Top 5 most frequent words",**

**ylab = "Word frequencies")**

****

1. **GENERATE THE WORD CLOUD:**

**Word Cloud:** An image composed of keywords found within a body of text, where the size of each word indicates its frequency in that body of text.

**words** – words to be plotted, **freq** – frequencies of words, **min.freq** – words whose frequency is at or above this threshold value (5) is plotted, **max.words** – the maximum number of words to display on the plot, **random.order** –the words are plotted in order of decreasing frequency, **rot.per** – the percentage of words that are displayed as vertical text (with 90-degree rotation). **colors** – changes word colors going from lowest to highest frequencies

#generate word cloud

**set.seed(1234)**

**wordcloud(words = dtm\_d$word, freq = dtm\_d$freq, min.freq = 5,**

**max.words=100, random.order=FALSE, rot.per=0.40,**

**colors=brewer.pal(8, "Dark2"))**

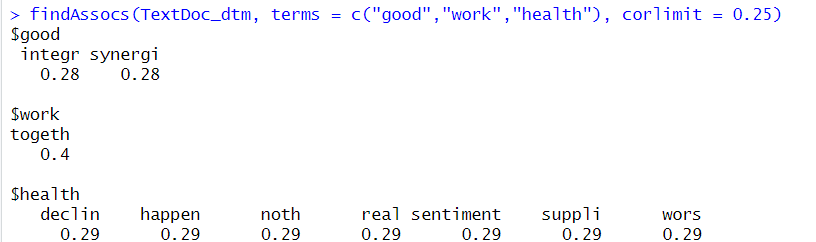
****

1. **WORD ASSOCIATION:**

# Find associations

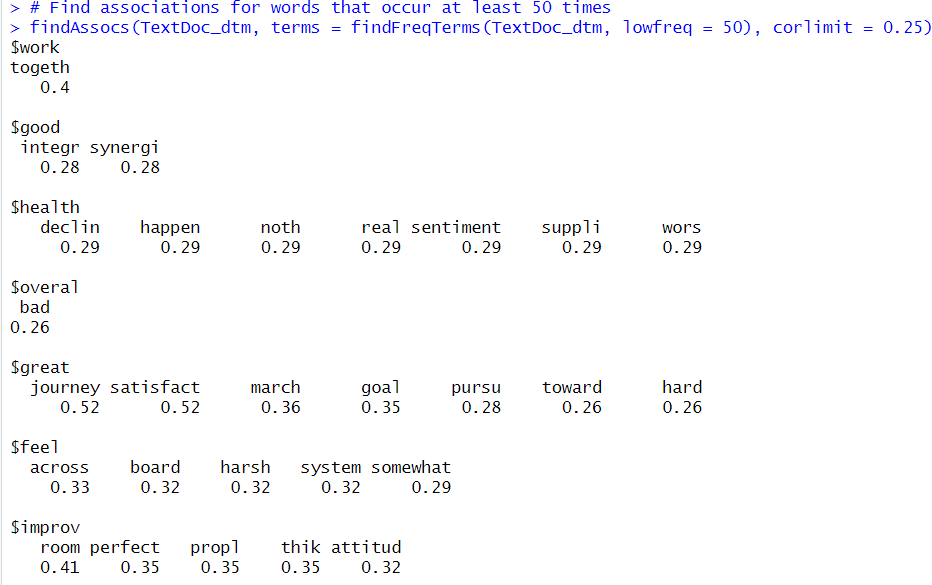
**findAssocs(TextDoc\_dtm, terms = c("good","work","health"), corlimit = 0.25)**

**#**This script shows which words are most frequently associated with the top three terms (corlimit = 0.25 is the lower limit/threshold.

****

# Find associations for words that occur at least 50 times

**findAssocs(TextDoc\_dtm, terms = findFreqTerms(TextDoc\_dtm, lowfreq = 50), corlimit = 0.25)**

****

1. **SENTIMENT SCORES:**

Sentiments can be classified as positive, neutral or negative. The get\_sentiment function accepts two arguments: **a character vector** (of sentences or words) and a **method.** The selected method determines which of the four available sentiment extraction methods will be used. **The four methods are syuzhet (this is the default), bing, afinn and nrc.**

The scale of sentiment scores generated by different methods:

**Syuzhet-** decimal and ranges from -1(indicating most negative) to +1(indicating most positive).

**bing**– binary scale with -1 indicating negative and +1 indicating positive sentiment

**afinn**– integer scale ranging from -5 to +5

**syuzhet\_vector<- get\_sentiment(text, method="syuzhet")**

# see the first row of the vector

**head(syuzhet\_vector)**

# see summary statistics of the vector

**summary(syuzhet\_vector)**

# bing

**bing\_vector<- get\_sentiment(text, method="bing")**

**head(bing\_vector)**

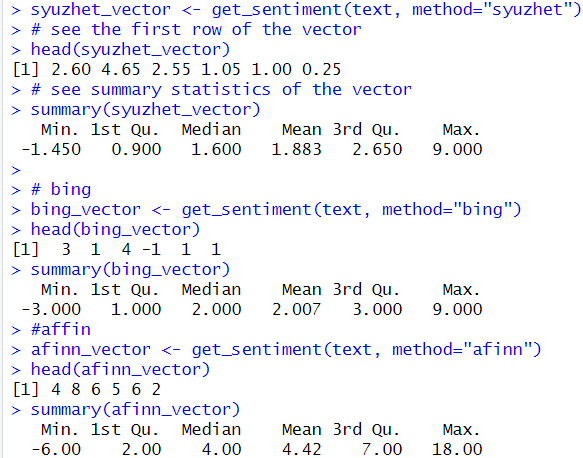
**summary(bing\_vector)**

#affin

**afinn\_vector<- get\_sentiment(text, method="afinn")**

**head(afinn\_vector)**

**summary(afinn\_vector)**



Because these different methods use different scales, it’s better to convert their output to a common scale before comparing them. This basic scale conversion can be done easily using R’s built-in sign function, which converts all positive number to 1, all negative numbers to -1 and all zeros remain 0

#compare the first row of each vector using sign function (rbind-rowbind)

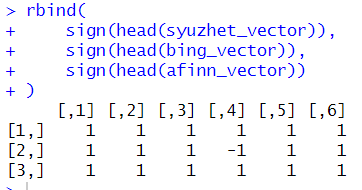
**rbind(**

**sign(head(syuzhet\_vector)),**

**sign(head(bing\_vector)),**

**sign(head(afinn\_vector))**

**)**

****

1. **EMOTION CLASSIFICATION:**

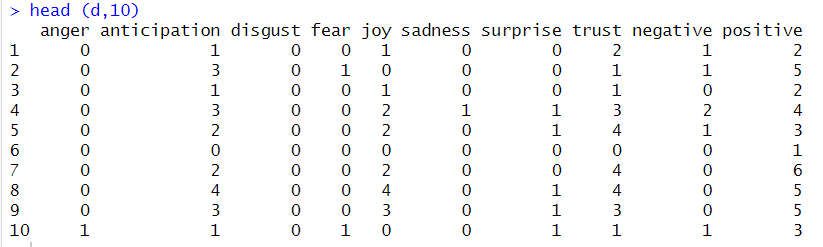
Emotion classification is built on the NRC Word-Emotion Association Lexicon. The NRC Emotion Lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive).

The **get\_nrc\_sentiment**function, returns a data frame with each row representing a sentence from the original file. The data frame has **ten columns** (one column for each of the eight emotions, one column for **positive sentiment** and one for **negative sentiment**).

**d<-get\_nrc\_sentiment(text)**

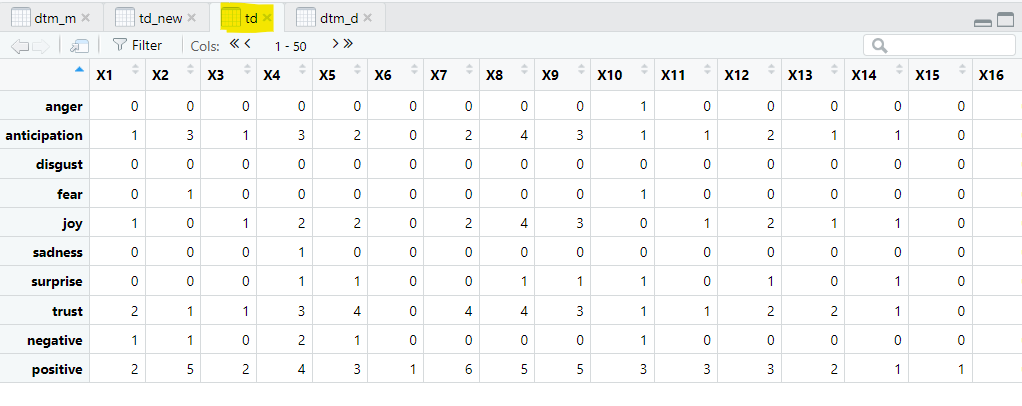
# head(d,10) - to see top 10 lines of the get\_nrc\_sentimentdataframe

**head (d,10)**

****

#transpose

**td<-data.frame(t(d))**

****

#The function rowSums computes column sums across rows for each level of a grouping variable.

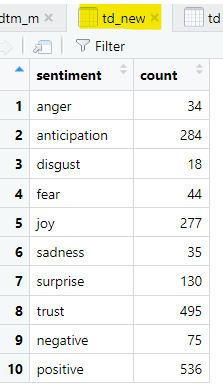
**td\_new<- data.frame(rowSums(td[2:253]))**

#Transformation and cleaning

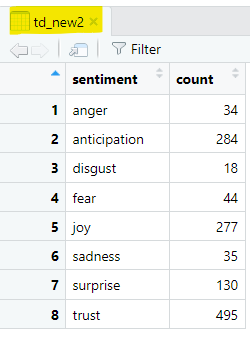
**names(td\_new)[1] <- "count"**

**td\_new<- cbind("sentiment" = rownames(td\_new), td\_new)**

**rownames(td\_new) <- NULL**

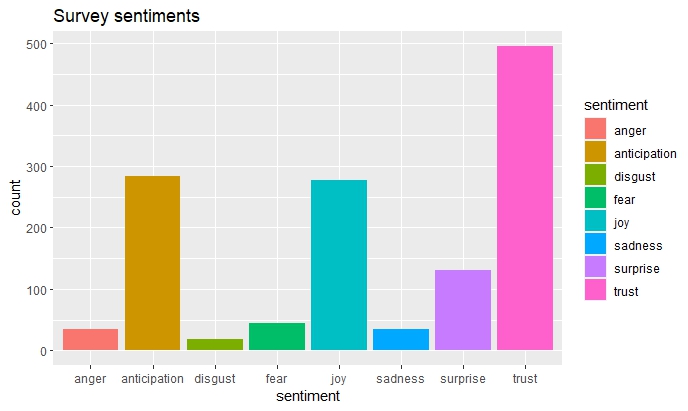
****

**td\_new2<-td\_new[1:8,]**

****

#Plot One - count of words associated with each sentiment

**quickplot(sentiment, data=td\_new2, weight=count, geom="bar", fill=sentiment, ylab="count")+ggtitle("Survey sentiments")**

****

This bar chart demonstrates that words associated with the positive emotion of “trust” occurred about five hundred times in the text, whereas words associated with the negative emotion of “disgust” occurred less than 25 times.

#Plot Two - count of words associated with each sentiment, expressed as a percentage

**barplot(**

**sort(colSums(prop.table(d[, 1:8]))),**

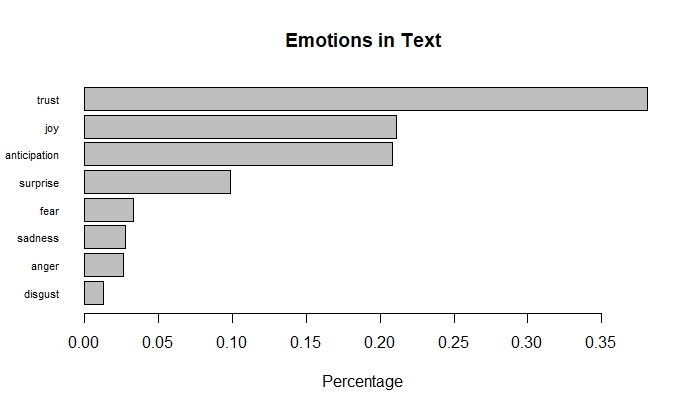
**horiz = TRUE,**

**cex.names = 0.7,**

**las = 1,**

**main = "Emotions in Text", xlab="Percentage"**

**)**

****

This bar plot allows for a quick and easy comparison of the proportion of words associated with each emotion in the text. The emotion “trust” has the longest bar and shows that words associated with this positive emotion constitute just over 35% of all the meaningful words in this text. On the other hand, the emotion of “disgust” has the shortest bar and shows that words associated with this negative emotion constitute less than 2% of all the meaningful words in this text.

**PRACTICAL NO.8: Install, Configure and Run Hadoop and HDFS.**

**STEPS TO INSTALL HADOOP:**

**1. INSTALL JAVA JDK 20**

**2. DOWNLOAD HADOOP. EXTRACT & PLACE UNDER C DRIVE**

**3. SET PATH IN ENVIRONMENT VARIABLES**

**4. CONFIG FILES UNDER HADOOP DIRECTORY**

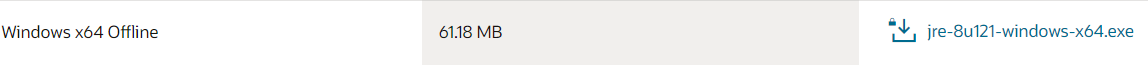
**5. CREATE FOLDER DATANODE AND NAMENODE UNDER DATA DIRECTORY**

**6. EDIT HDFS & YARN FILES**

**7. SET JAVA HOME ENVIRONMENT IN HADOOP ENVIRONMENT**

**8. SETUP COMPLETE. TEST BY EXECUTING start-all.cmd**

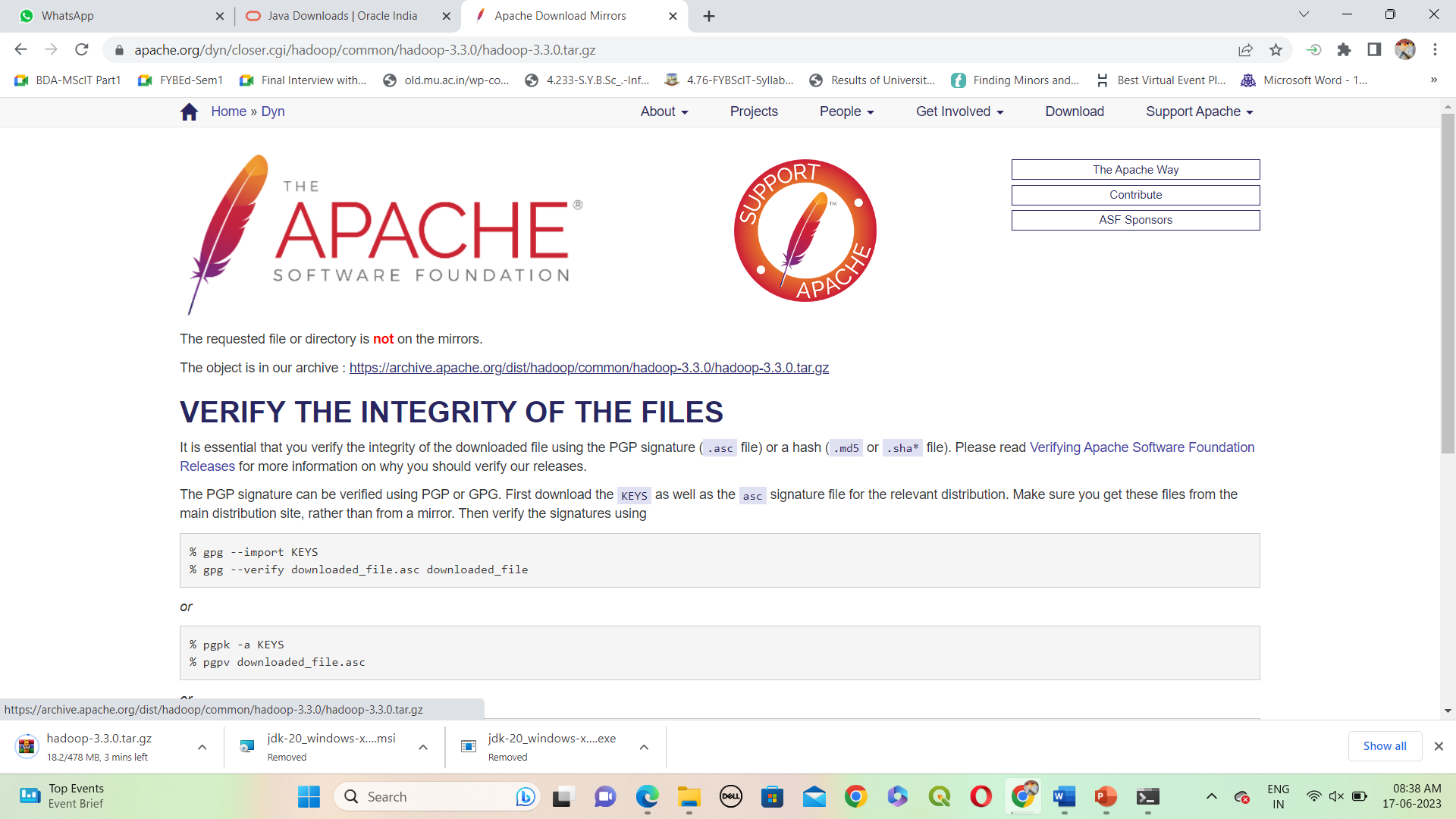
**1. INSTALL JAVA JDK 20:**

* Download Link: <https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html>
* Java SE Development Kit 8u121:
* Java SE Runtime Environment 8u121:
* Open cmd and type ->javac -version



**2. DOWNLOAD HADOOP. EXTRACT & PLACE UNDER C DRIVE:**

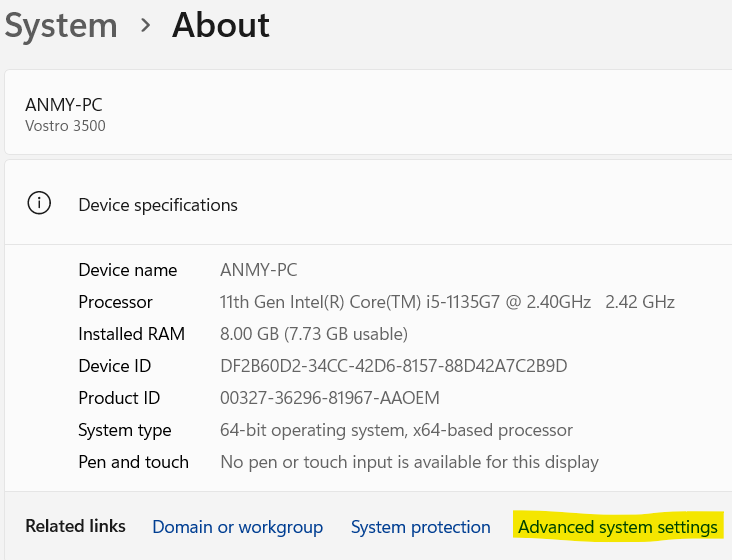
* Download Link: https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz



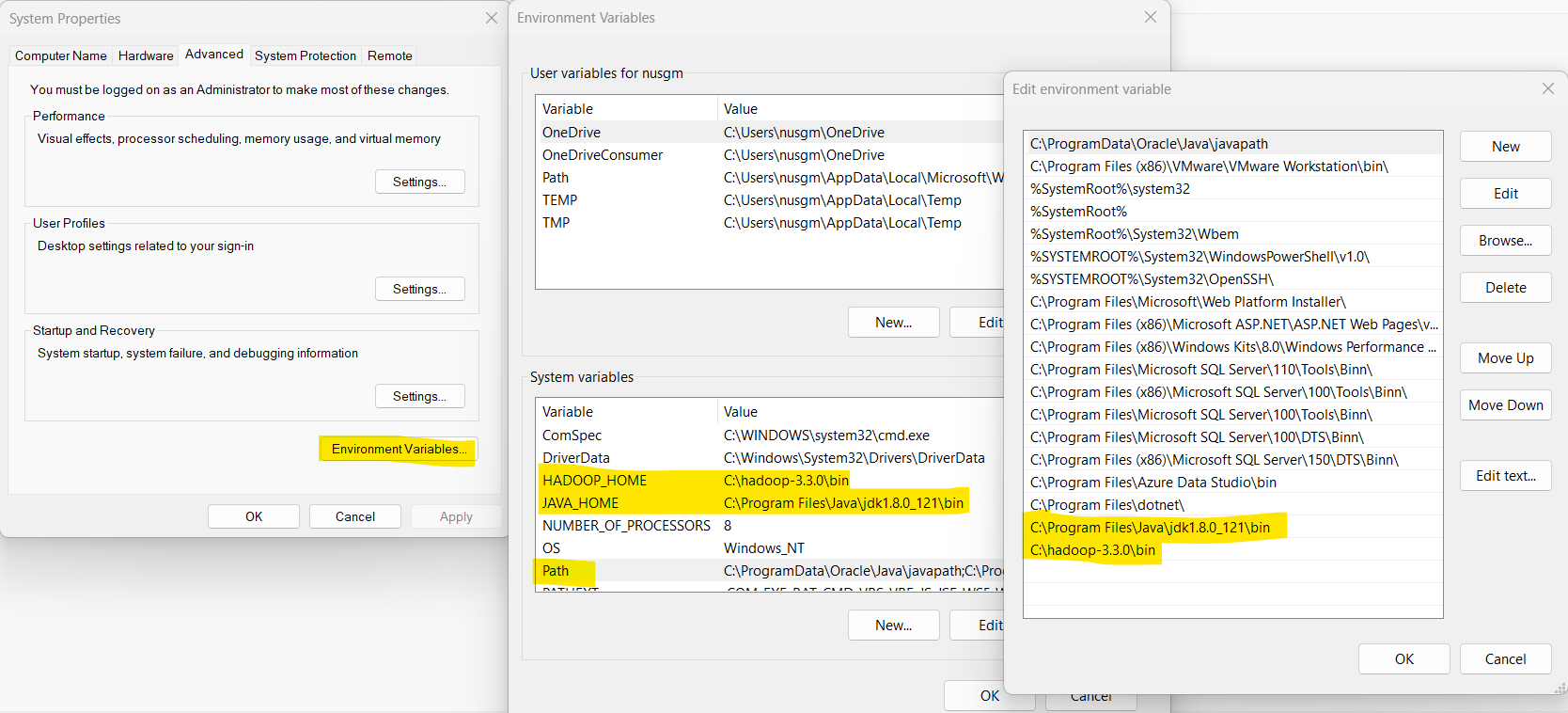
* Right click on hadoop-3.3.0.tar->Extract to C:\hadoop-3.3.0

1. **SET PATH IN ENVIRONMENT VARIABLES:**

* Right Click on This PC->Properties->Advanced System Settings->Environment Variables

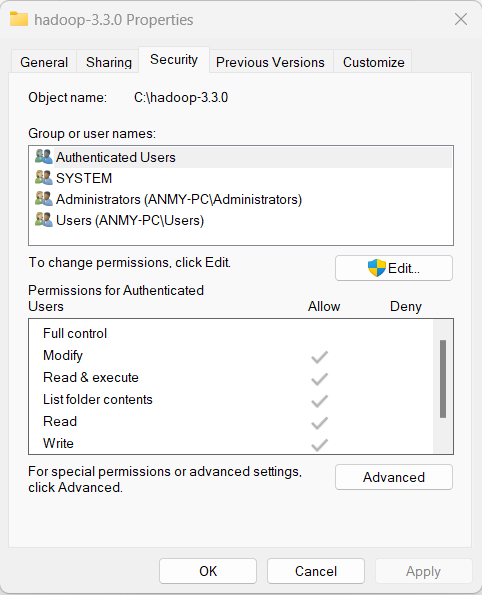


* Set the path JAVA\_HOME & HADOOP\_HOME under Environment variable



1. **CONFIG FILES UNDER HADOOP DIRECTORY:**

* Set the permission of hadoop-3.3.0 folder accessible by everyone.

****

* **Edit file C:\hadoop-3.3.0\etc\hadoop\core-site.xml.**
* **Paste the xml code and save:**

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

* **Edit file C:\hadoop-3.3.0\etc\hadoop\mapred-site.xml.**
* **Paste the xml code and save:**

<configuration>

<property>

<name>mapreduce.framework.name</name>

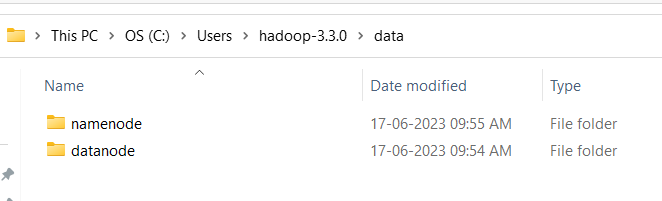
<value>yarn</value>

</property>

</configuration>

1. **CREATE FOLDER DATANODE AND NAMENODE UNDER DATA DIRECTORY:**

* **Create folder “data” under “C:\hadoop-3.3.0”**
* **Create folder “datanode” under “C:\hadoop-3.3.0\data”**
* **Create folder “namenode” under “C:\hadoop-3.3.0\data”**



1. **EDIT HDFS & YARN FILES:**

* **Edit file C:\hadoop-3.3.0\etc\hadoop\hdfs-site.xml.**
* **Paste the xml code and save:**

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>/hadoop-3.3.0/data/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/hadoop-3.3.0/data/datanode</value>

</property>

</configuration>

* **Edit file C:\hadoop-3.3.0\etc\Hadoop\yarn-site.xml,**
* **Paste the xml code and save:**

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>127.0.0.1:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>127.0.0.1:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

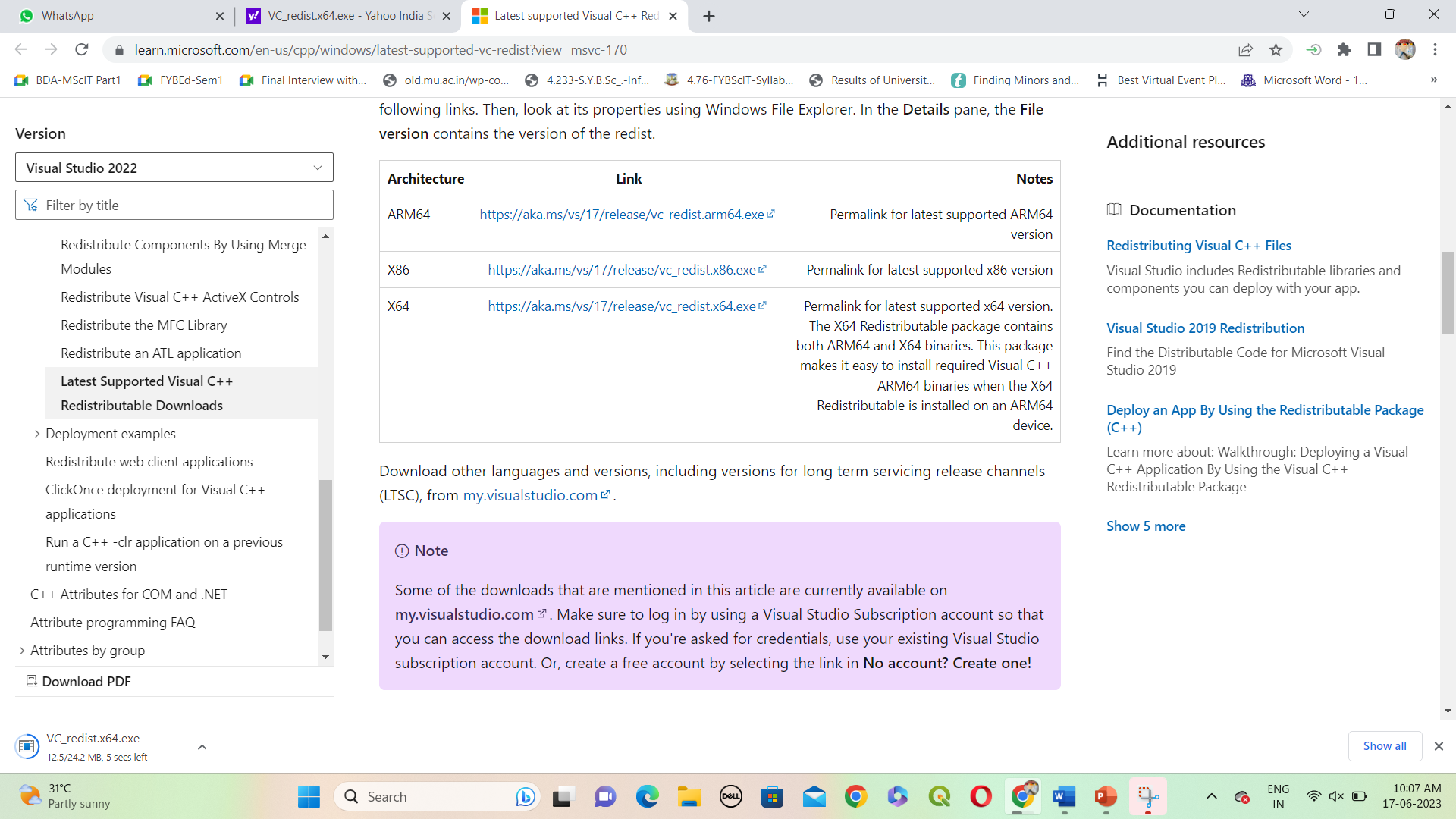
<value>127.0.0.1:8031</value>

</property>

</configuration>

1. **SET JAVA HOME ENVIRONMENT IN HADOOP ENVIRONMENT:**

* **Edit file C:/hadoop-3.3.0/etc/hadoop/hadoop-env.cmd.**
* **Find “JAVA\_HOME=%JAVA\_HOME%” and replace it as set JAVA\_HOME=""**
* **Download and run VC\_redist.x64.exe**



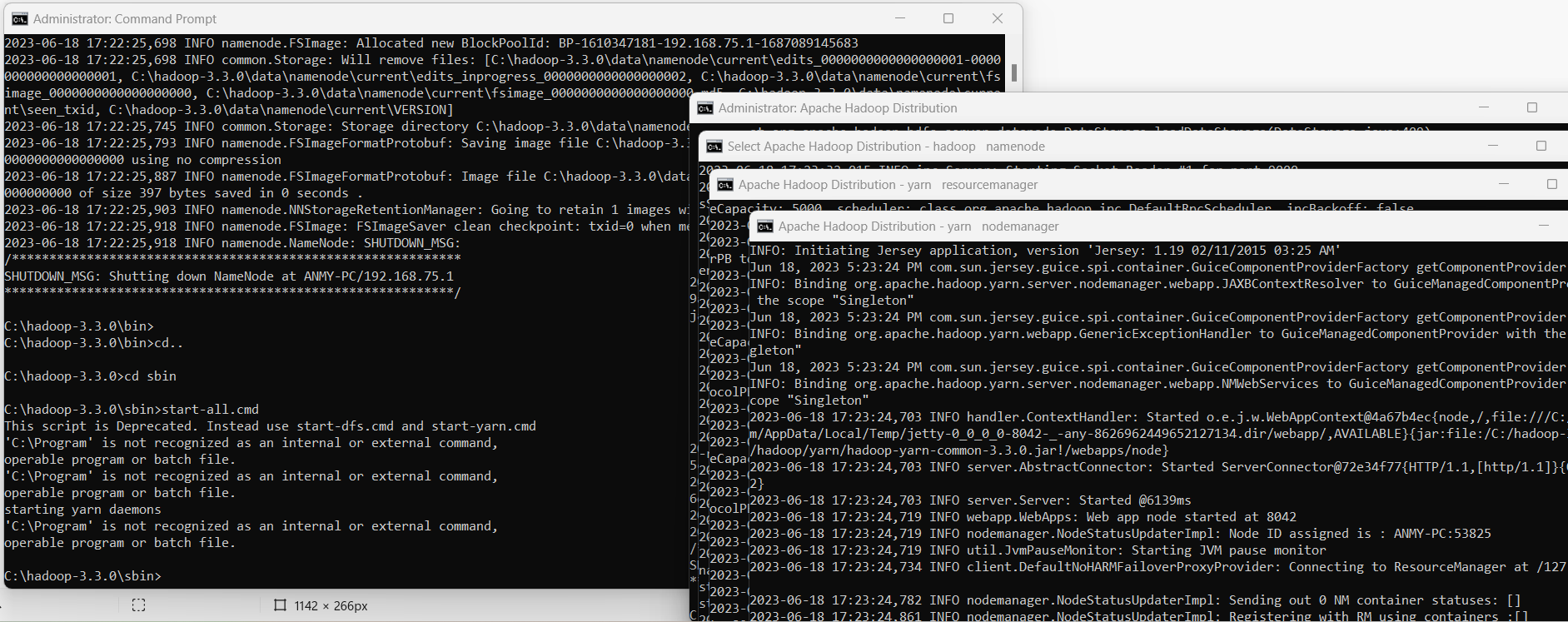
* **Download bin folder from: https://github.com/s911415/apache-hadoop-3.1.0-winutils**
* **Copy the bin folder to C:\hadoop-3.3.0. Replace the existing bin folder.**
* **Copy "hadoop-yarn-server-timelineservice-3.0.3.jar"from~\hadoop-3.0.3\share\hadoop\yarn\timelineservice to ~\hadoop-3.0.3\share\hadoop\yarn folder.**

1. **SETUP COMPLETE. TEST BY EXECUTING start-all.cmd:**

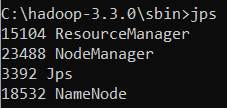
* **Format the NameNode**
* **Open cmd ‘Run as Administrator’ and type command “hdfsnamenode –format”**

**C:\hadoop-3.3.0\bin>hdfs namenode -format**

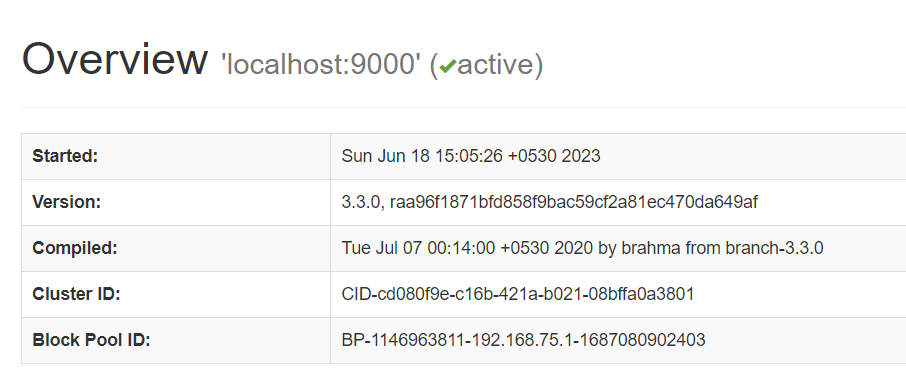
* **Change directory to C:\Hadoop-3.3.0\sbin & type start-all.cmd**

****

* **You will get 4 more running threads for DATANODE, NAMENODE, RESOURCE MANAGER & NODE MANAGER.**

****

* **Run http://localhost:9870/ from any browser:**



**PRACTICAL NO.9: MAPREDUCE IMPLEMENTATION**

* MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. It is the heart of [Apache Hadoop](https://www.ibm.com/analytics/hadoop).
* The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform:
  + The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).
  + The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples.

**STEPS:**

1. **Starting the Services:**

Open a command prompt as administrator and run the following commands:

**C:\hadoop-3.3.0\sbin>start-dfs.cmd**

**C:\hadoop-3.3.0\sbin>start-yarn.cmd**

**C:\hadoop-3.3.0\bin>cd\**

**C:\>hadoop dfsadmin -safemode leave**

DEPRECATED: Use of this script to execute hdfs command is deprecated.

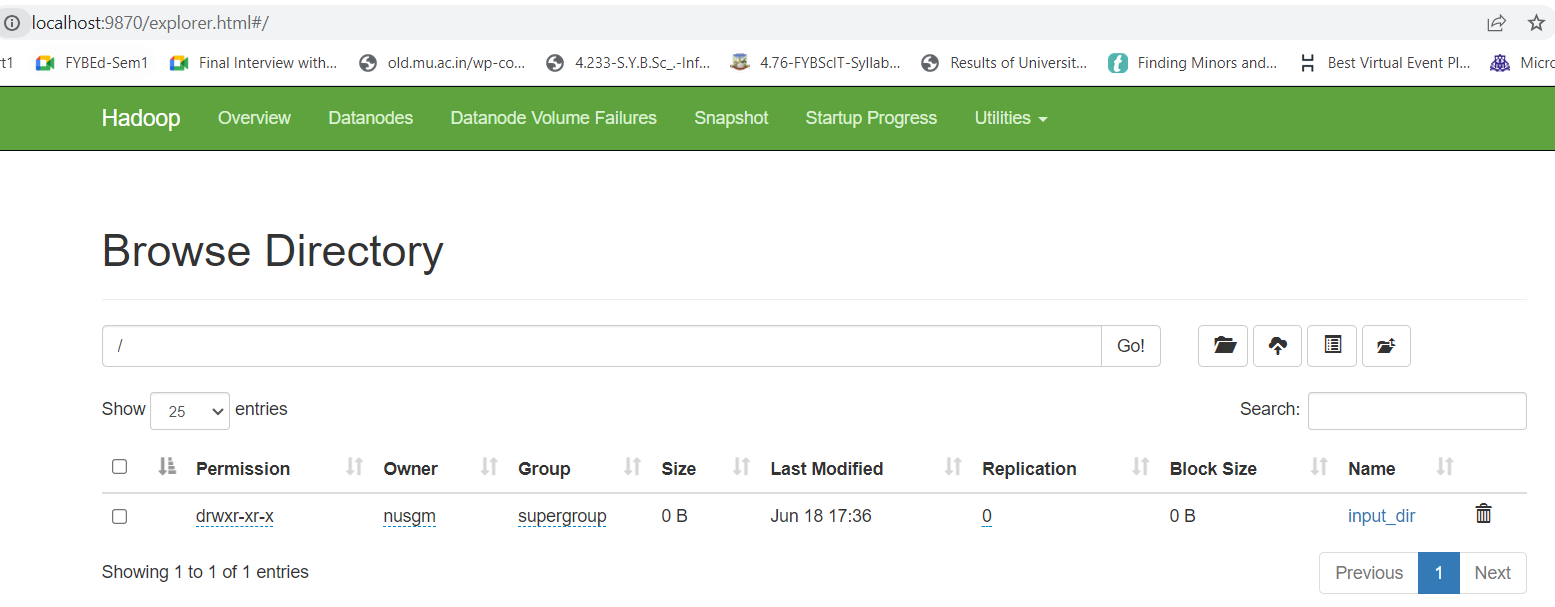
Instead use the hdfs command for it.

Safe mode is OFF

1. **Create a folder in HDFS:**

**C:\>hadoop fs -mkdir /input\_dir**

Check it by giving the following URL at browser: <http://localhost:9870>



1. **Steps to copy the file named input\_file.txt in the input directory (input\_dir)of HDFS:**

Make a file in C:\ Drive with the name as input\_file.txt and write the following contents in it:

Hadoop Window version is easy compared to Ubuntu version.

**C:\>hadoop fs -put C:/input\_file.txt /input\_dir**

**Graphical user interface, text, application

Description automatically generated**

Verify input\_file.txt available in HDFS input directory (input\_dir): **C:\>hadoop fs -ls /input\_dir/**

Text

Description automatically generated

Verify contents of the copied file: **C:\>hadoop dfs -cat /input\_dir/input\_file.txt**

Text

Description automatically generated

1. **Run MapReduceClient.jar and also provide input and output directories:**

C:\>hadoop jar C:/hadoop-3.3.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar wordcount /input\_dir /output\_dir

Text

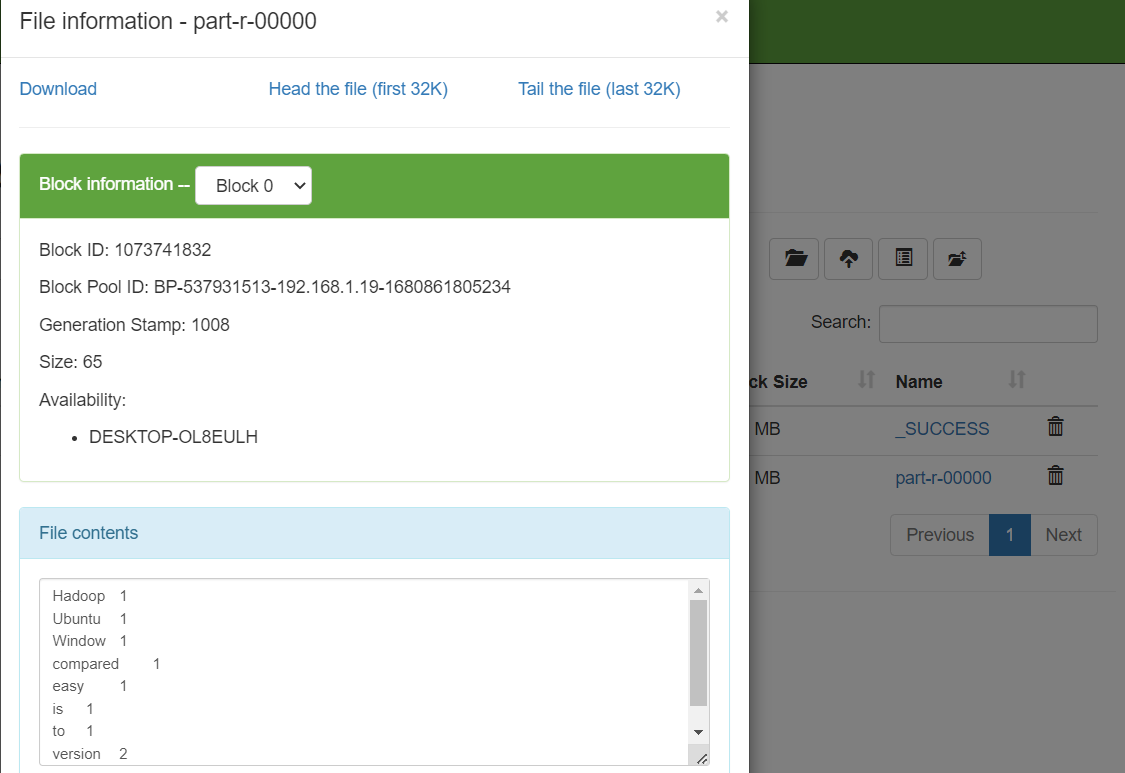
Description automatically generated

Check the output\_dir on browser as follows:

Graphical user interface, text, application, table, Excel

Description automatically generated

Click on output\_dir and check the file content as the output:



Alternatively, you may type the following command on CMD window as:**C:\>hadoopdfs -cat /output\_dir/\***

Text

Description automatically generated

**PRACTICAL NO.10: Implement an application that stores Big Data in MongoDB and manipulate it usingPython**

**STEP 1: INSTALLATION:**

**MongoDB, Python &PyMongo**

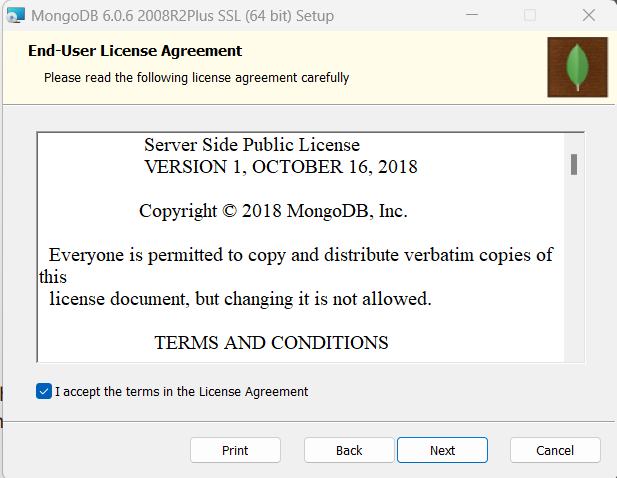
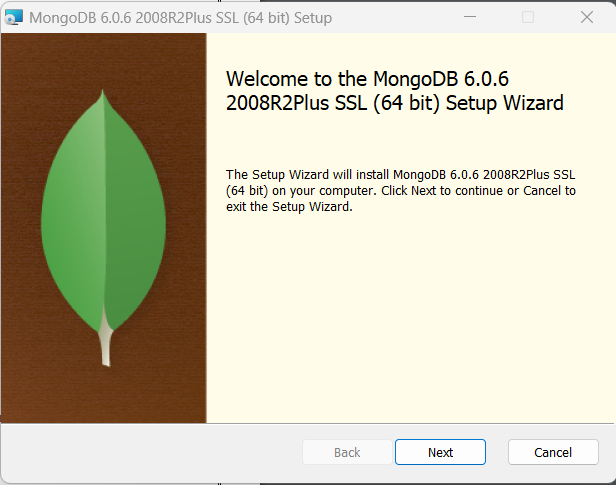
**STEP 2: MANIPULATION USING PYTHON:**

1. **Create a Database**
2. **Create a Collection**
3. **Insert into Collection**
4. **Insert Multiple data into Collection**

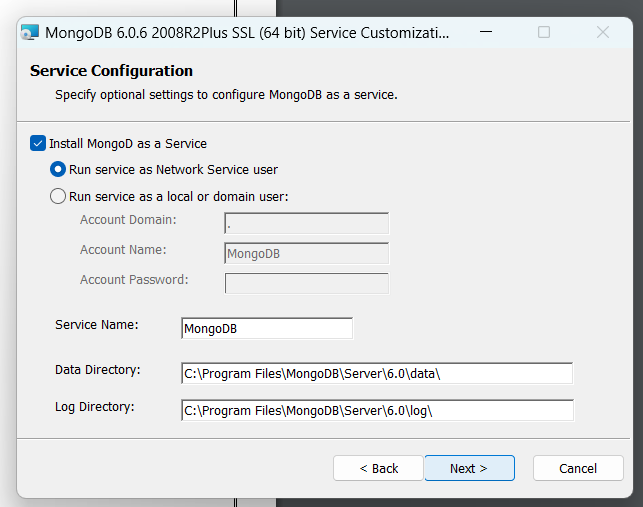
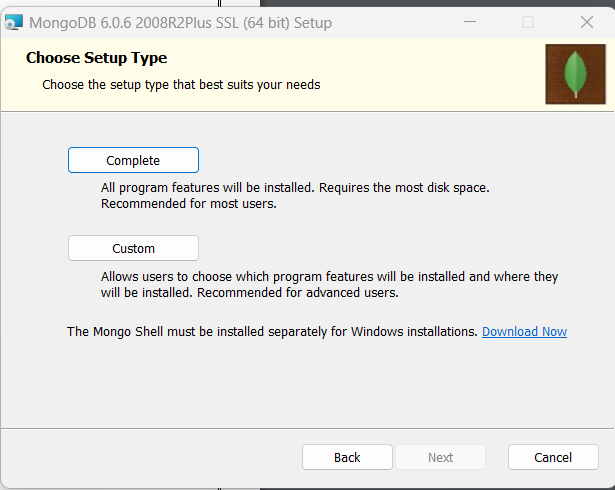
**STEP 1:**

**INSTALLATION of MongoDB:**

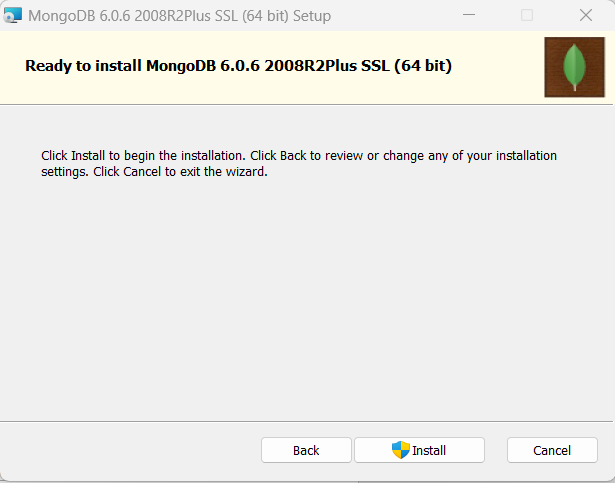
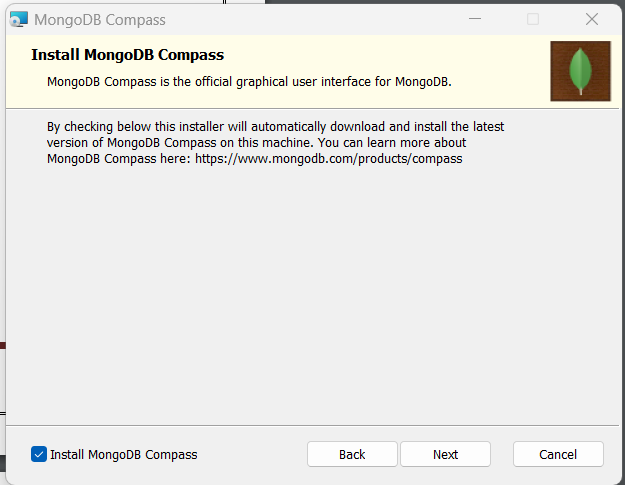
* **Go to the Link:**<https://www.mongodb.com/try/download/community->> Download MongoDB Community Server.
* Once download is complete open the msi file. Click Next in the startup screen.
* Accept the End-User License Agreement->Click Next.



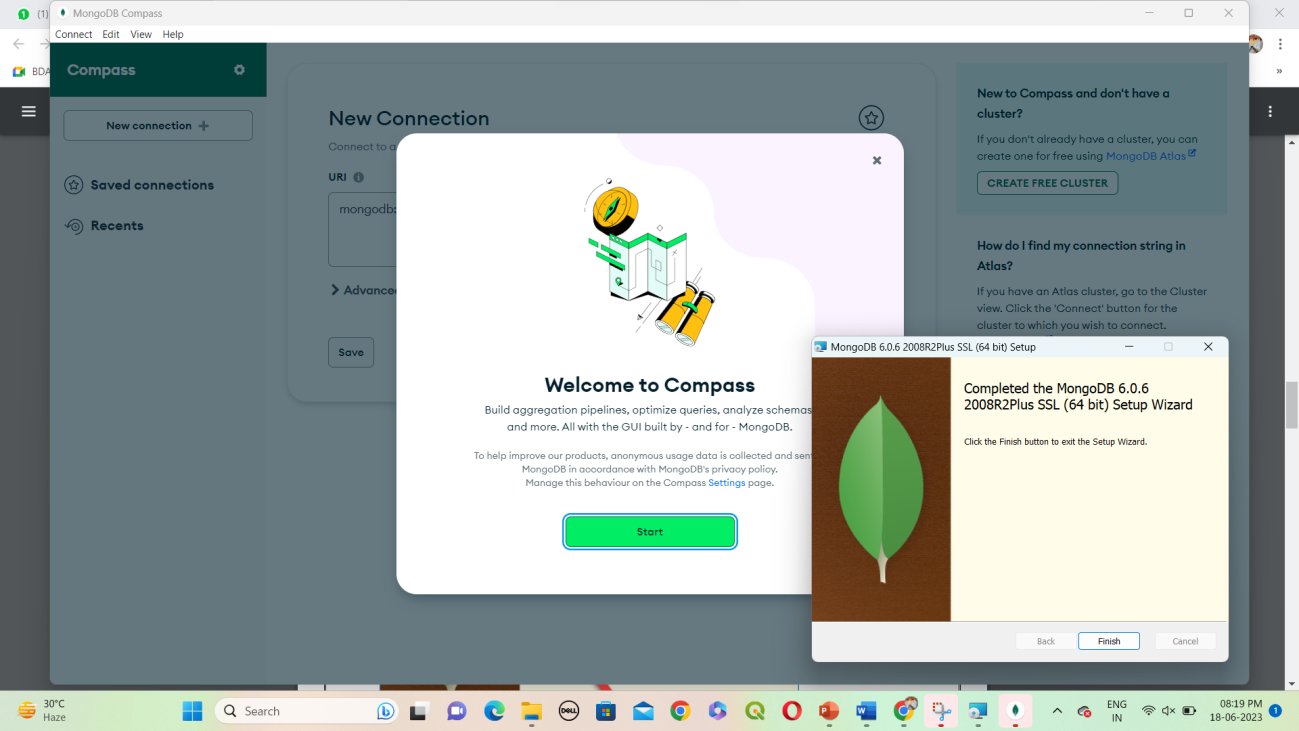
* Click on the "Complete" button to install all of the components.
* Select “Run service as Network Service user”-> Click Next.



* Select the Install MongoDB Compass-> Click Next.
* Click on the Install button to start the installation.

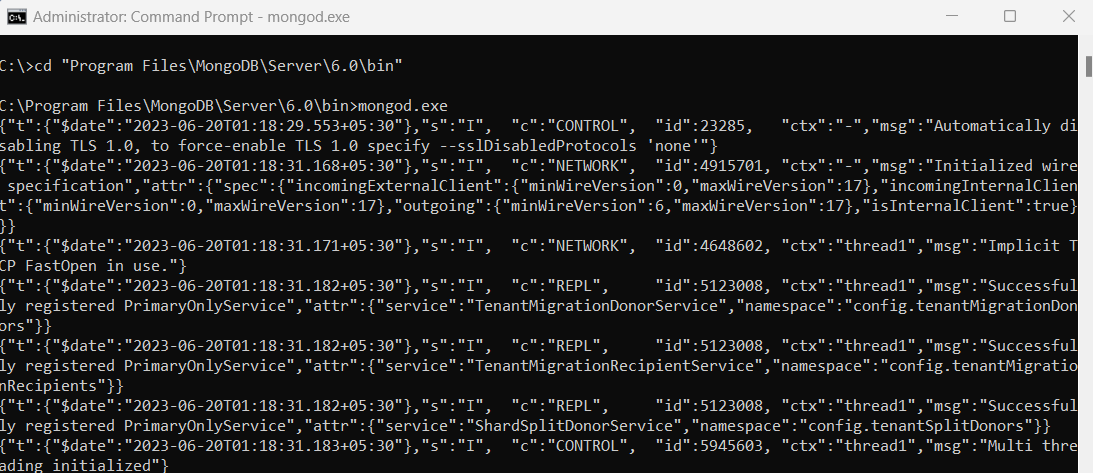
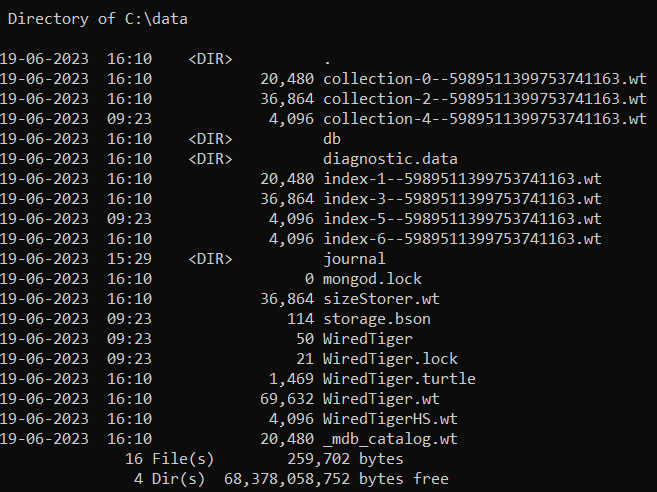


* Installation begins. Click Next once completed. Click on the Finish button to complete the installation.
* The following screen gets displayed:



**Test MongoDB:**

* Create the directory where MongoDB will store its files:
  + C:\>md data
  + C:\>cd data
  + C:\data>md db
* Setting the default path as C:\data:
  + C:\Program Files\MongoDB\Server\6.0\bin>mongod.exe --dbpath "C:\data"
  + C:\data>dir
* Start the mongodb daemon by running C:\Program Files\MongoDB\Server\6.0\bin>mongod.exe

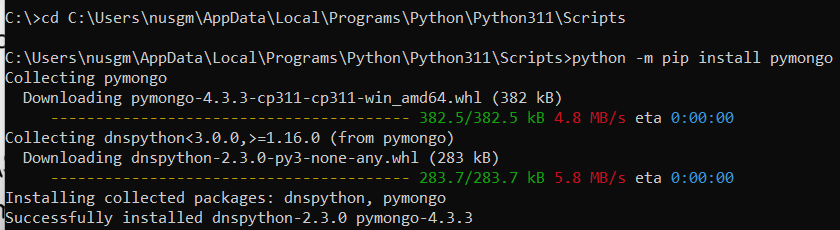


**INSTALLATIONofPython:**

* **Go to the Link:**https://www.python.org/downloads/->Download & Install Python.
* **Default Path of Python:**C:\Users\nusgm\AppData\Local\Programs\Python\Python311

**INSTALLATION of PyMongo:**

* Installing PyMongo by going to the path C:\Users\nusgm\AppData\Local\Programs\Python\Python311\Scripts>python -m pip install pymongo

****

**STEP 2: MANIPULATION USING PYTHON:**Open IDLE Shell 3.11.4 of Python & MongoDB Compass. Type the commands for the following operations:

1. **Create a Database**
2. **Create a Collection**
3. **Insert into Collection**
4. **Insert Multiple data into Collection**
5. **Create a Database:**

**CODE:**

import pymongo

myclient=pymongo.MongoClient("mongodb://localhost:27017/")

mydb=myclient["BDA\_DB"]

print(myclient.list\_database\_names())

**OUTPUT:**

['admin', 'config', 'local', 'mybigdata']

1. **Create a Collection:**

**CODE:**

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["BDA\_DB"]

mycol=mydb["IT"]

print(mydb.list\_collection\_names())

**OUTPUT:**

[]

1. **Insert into Collection:**

**CODE:**

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["BDA\_DB"]

mycol=mydb["IT"]

mydict={"studid": "1", "studname":"abc","studclass":"MScIT-I"}

x=mycol.insert\_one(mydict)

1. **Insert Multiple data into Collection:**

**CODE:**

import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["BDA\_DB"]

mycol=mydb["IT"]

mylist=[{"studid": "1", "studname":"abc","studclass":"MScIT-I"}, {"studid": "2", "studname":"pqr","studclass":"MScIT-I"},

{"studid": "3", "studname":"lmn","studclass":"MScIT-I"}, {"studid": "4", "studname":"xyz","studclass":"MScIT-I"}]

x=mycol.insert\_many(mylist)

