**Practical No. 01**

**Aim: A) Clustering algorithms for unsupervised classification. Read a datafile all\_Customers.csv and apply k-means clustering. Plot the cluster data using R visualizations.**

**Code:**

# K-Means Clustering

# Importing the dataset

dataset = read.csv("D:\\bda prac\\Mall\_Customers.csv")

head(dataset)

dataset = dataset[4:5]

head(dataset)

wcss = vector()

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

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)

y\_kmeans = kmeans$cluster

# Visualising the clusters

library(cluster)

clusplot(dataset,

y\_kmeans,

lines = 0,

shade = TRUE,

color = TRUE,

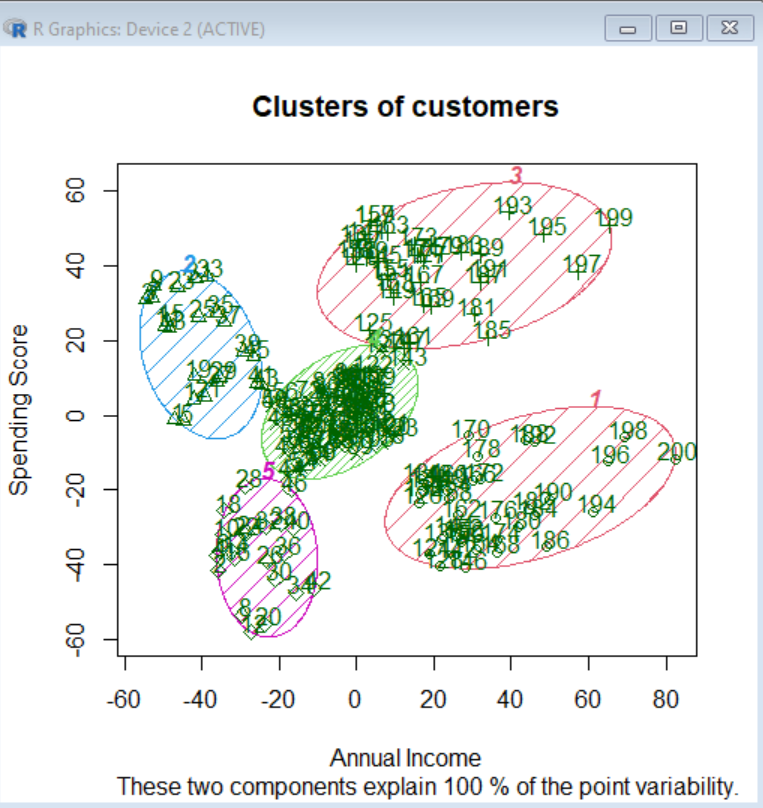
labels = 2,

main = paste('Clusters of customers'),

xlab = 'Annual Income',

ylab = 'Spending Score')

**Output:**

****

**Aim: B) Implement Apriori Algorithm Recommending grocery items to a customer that is most frequently bought together, given a data set of transactions by customers of a store, using built-in Groceries file.**

**Code:**

install.packages("arules")

install.packages("arulesViz")

install.packages("RColorBrewer")

library(arules)

library(arulesViz)

library(RColorBrewer)

data("Groceries")

Groceries

summary(Groceries)

class(Groceries)

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

summary(rules)

inspect(rules[1:10])

arules::itemFrequencyPlot(Groceries, topN = 20,

                          col = brewer.pal(8, 'Pastel2'),

                          main = 'Relative Item Frequency Plot',

                          type = "relative",

                          ylab = "Item Frequency(Relative)")

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

summary(itemset)

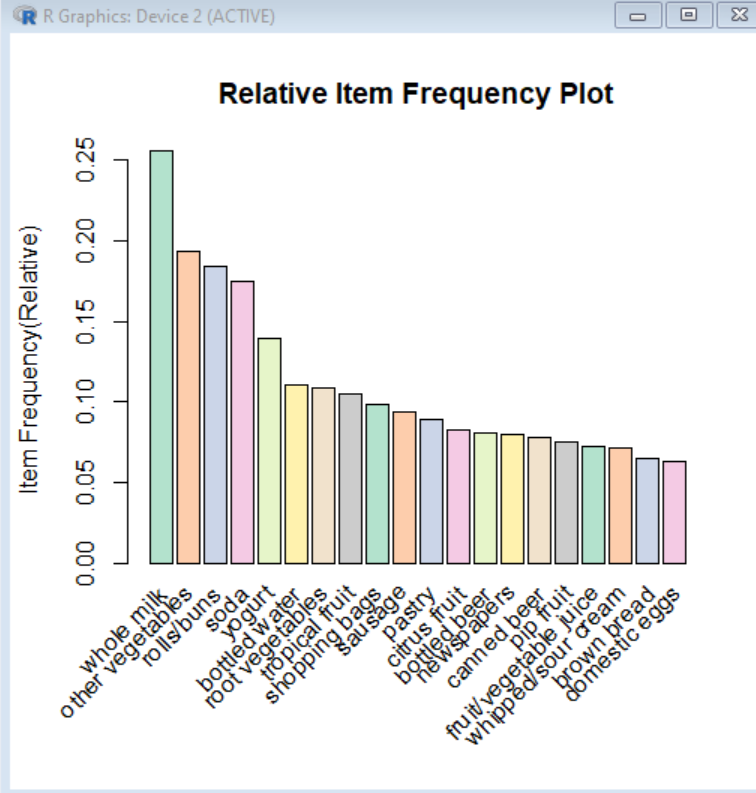
inspect(itemset[1:10])

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

summary(itemsets\_3)

inspect(itemsets\_3)

**Output:**

****

**Practical No. 02**

**Aim: A) Import data from web storage – binary.csv. Name the dataset and do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not.**

**Code:**

#fetch the data

college <- read.csv("D:\\bda prac\\binary.csv")

head(college)

nrow(college)

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

library(caTools)

split <- sample.split(college, SplitRatio = 0.75)

split

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

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

# Training model

fit\_logistic\_model <- glm(admit ~ .,

data = training\_reg,

family = "binomial")

# Predict test data based on model

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

test\_reg, type = "response")

predict\_reg

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

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

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

# Changing probabilities

predict\_reg <- ifelse(predict\_reg >0.5, 1, 0)

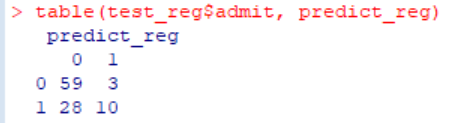
predict\_reg

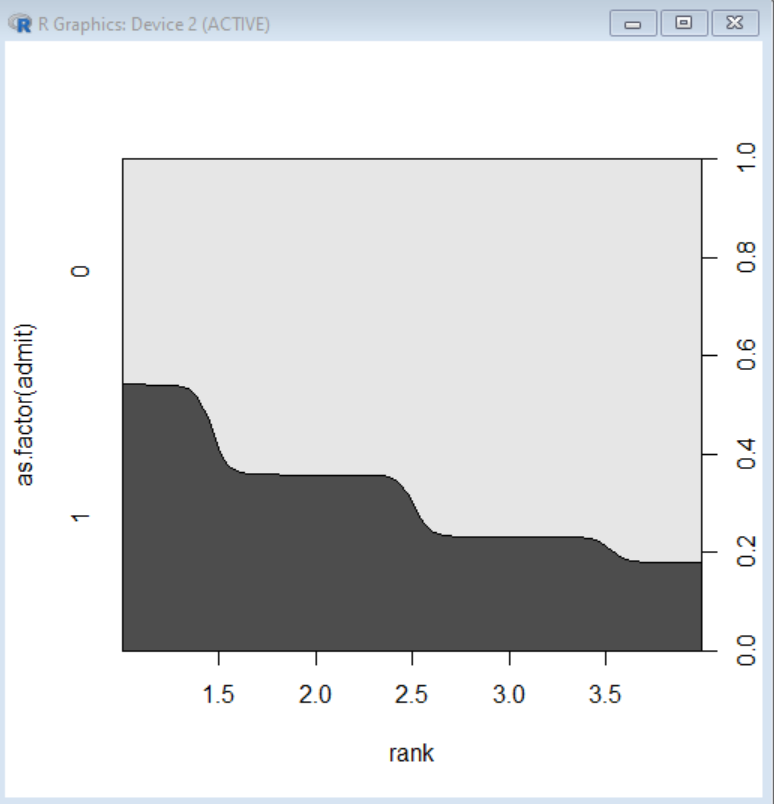
# Evaluating model accuracy

# using confusion matrix

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

**Output:**

****



**Aim: B) Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset – binary.csv.**

**Code:**

#fetch the data

college <- read.csv("D:\\bda prac\\binary.csv")

head(college)

nrow(college)

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

library(caTools)

split <- sample.split(college, SplitRatio = 0.75)

split

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

test\_reg <- subset(college, 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)

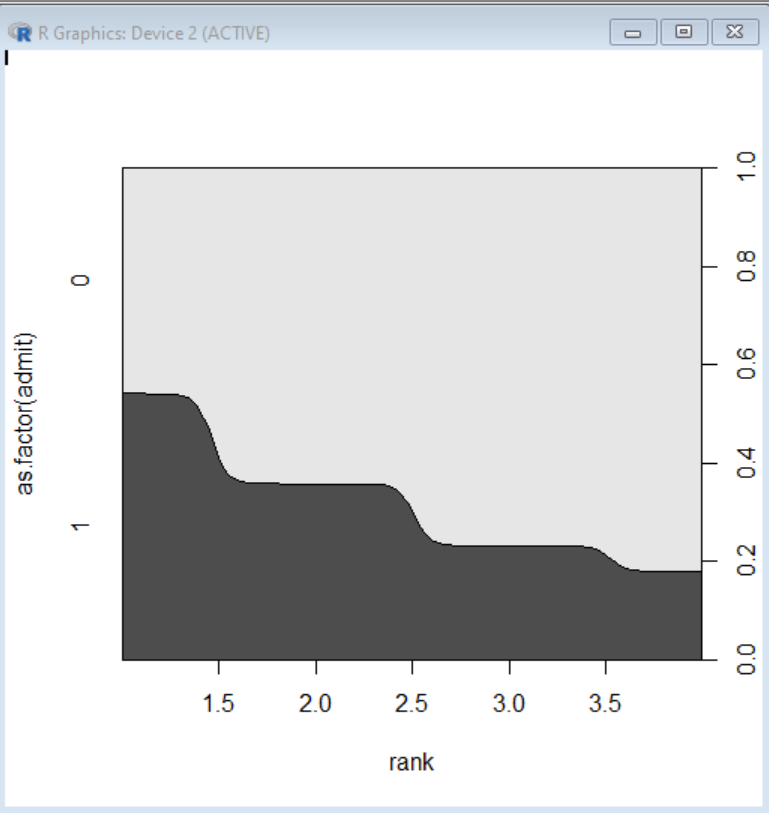
predict\_reg

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

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

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

**Output:**



**Practical No. 03**

**Aim: A) Implement Decision Tree classification technique using Social\_Network\_Ads.csv dataset.**

**Code:**

# Decision Tree Classification

# Importing the dataset

dataset = read.csv("D:\\bda prac\\Social\_Network\_Ads.csv")

#print(dataset)

dataset = dataset[3:5] # columns 3 4 ad 5

print(dataset)

# Encoding the target feature as factor(just like a vector having levels

# levels to convey that only two possible values for purchased - 0 & 1

dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

print (dataset$Purchased)

# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

set.seed(123)

#split = sample.split(dataset$Purchased, SplitRatio = 0.75)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Feature Scaling - scale() method centers and/or scales the columns of a numeric matrix.

training\_set[-3] = scale(training\_set[-3]) # scaling first 2 columns, don't consider 3rd column

test\_set[-3] = scale(test\_set[-3])

#print(test\_set[-3])

# Fitting Decision Tree Classification to the Training set

install.packages('rpart')

library(rpart) # for partitioning tree

install.packages('rpart.plot')

library(rpart.plot)

classifier = rpart(formula = Purchased ~ .,data = training\_set)

# Predicting the Test set results

y\_pred = predict(classifier, newdata = test\_set[-3], type = 'class')

print(y\_pred)

# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

print(cm)

y\_grid = predict(classifier, newdata = grid\_set, type = 'class')

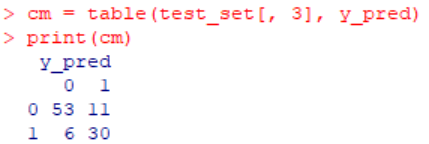
# Plotting the tree

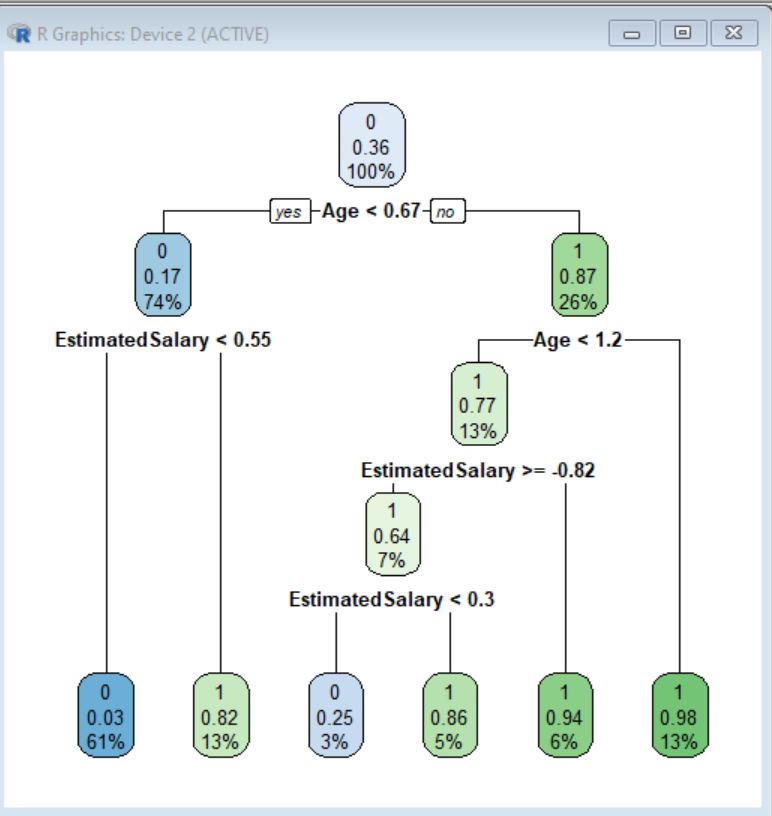
#extra=106 class model with a binary response

#extra=104 class model with a response having more than two levels

rpart.plot(classifier, extra = 106)

**Output:**

****

****

**Aim: B) Implement SVM Classification technique using Social\_Network\_Ads.csv dataset. Evaluate the performance of classifier.**

**Code:**

# Support Vector Machine (SVM)

# Importing the dataset

dataset = read.csv("D:\\bda prac\\Social\_Network\_Ads.csv")

dataset = dataset[3:5]

print(dataset)

print(dataset$Purchased)

# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training\_set = subset(dataset, split == TRUE)

print(training\_set)

test\_set = subset(dataset, split == FALSE)

print(test\_set)

# Feature Scaling

training\_set[-3] = scale(training\_set[-3]) # [-3] means 3rd index will be dropped

test\_set[-3] = scale(test\_set[-3])

print(training\_set[-3])

print (test\_set[-3])

# Fitting SVM to the Training set

install.packages('e1071')

library(e1071)

classifier = svm(formula = Purchased ~ .,

data = training\_set,

type = 'C-classification',

kernel = 'linear')

print (classifier)

# Predicting the Test set results

y\_pred = predict(classifier, newdata = test\_set[-3])

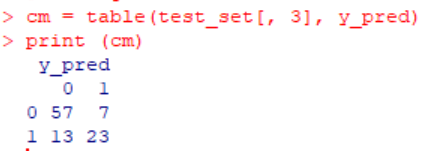
print(y\_pred)

# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

print (cm)

**Output:**

****

**Practical No. 04**

**Aim: A) Implement Naïve Bayes Classification technique using Social\_Network\_Ads.csv dataset. Evaluate the performance of classifier.**

**Code:**

# Naive Bayes

# Importing the dataset

dataset = read.csv('C:\\2022-23\\BDA practical 2023\\Social\_Network\_Ads.csv')

dataset = dataset[3:5]

# Encoding the target feature as factor

dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))

# Splitting the dataset into the Training set and Test set

#install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

# Feature Scaling

training\_set[-3] = scale(training\_set[-3])

test\_set[-3] = scale(test\_set[-3])

# Fitting Naive Bayes to the Training set

install.packages('e1071')

library(e1071)

classifier = naiveBayes(x = training\_set[-3],

y = training\_set$Purchased)

# Predicting the Test set results

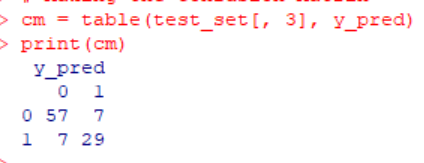
y\_pred = predict(classifier, newdata = test\_set[-3])

# Making the Confusion Matrix

cm = table(test\_set[, 3], y\_pred)

print(cm)

**Output:**

****

**Aim: B) Find the confusion matrix to find restaurant review based of sentiment analysis of Natural Language processing. Use Resaurentreviews.tsv file for your study.**

**Code:**

dataset\_original = read.delim("D:\\bda prac\\Restaurant\_Reviews.txt", quote = '', stringsAsFactors = FALSE)

install.packages('tm')

install.packages('SnowballC')

library(tm)

library(SnowballC)

corpus = VCorpus(VectorSource(dataset\_original$Review))

corpus = tm\_map(corpus, content\_transformer(tolower))

corpus = tm\_map(corpus, removeNumbers)

corpus = tm\_map(corpus, removePunctuation)

corpus = tm\_map(corpus, removeWords, stopwords())

corpus = tm\_map(corpus, stemDocument)

corpus = tm\_map(corpus, stripWhitespace)

dtm = DocumentTermMatrix(corpus)

dtm = removeSparseTerms(dtm, 0.999)

dataset = as.data.frame(as.matrix(dtm))

dataset$Liked = dataset\_original$Liked

print(dataset$Liked)

dataset$Liked = factor(dataset$Liked, levels = c(0,1))

install.packages(caTools)

library(caTools)

set.seed(123)

split = sample.split(dataset$Liked, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

test\_set = subset(dataset, split == FALSE)

install.packages('randomForest')

library(randomForest)

classifier = randomForest(x = training\_set[-692],

y = training\_set$Liked,

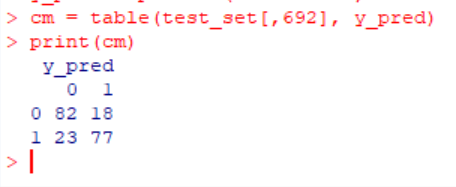
ntree = 10)

y\_pred = predict(classifier, newdata = test\_set[-692])

cm = table(test\_set[,692], y\_pred)

print(cm)

**Output:**

****

**Practical No. 05**

**Aim: Take the inbuilt data file: iris and perform classification on that data using various classification models – Decision Tree, K Nearest Neighbour and Support Vector Machine. Find the confusion matrix for all three models and evaluate them by finding their accuracy. Find the algorithm which performs best on the given data file, out of all these three models.**

**Code:**

#PBL

install.packages('rpart')

install.packages('rpart.plot')

install.packages('gmodels')

install.packages('e1071')

library(rpart)

library(rpart.plot)

library(gmodels)

library(e1071)

data(iris)

summary(iris)

#normalize the continuous variables before performing any analysis on the dataset

temp = as.data.frame(scale(iris[,1:4]))

temp$Species = iris$Species # levels: setosa versicolor virginica

summary(temp)

# Splitting the dataset into the Training set and Test set

install.packages('caTools')

library(caTools)

set.seed(123)

split = sample.split(temp$Species, SplitRatio = 0.75)

train = subset(temp, split == TRUE)

test = subset(temp, split == FALSE)

nrow(train)

nrow(test)

#1. Decision Trees

dt\_classifier = rpart(formula = Species ~ .,data = train)

# Predicting the Test set results

dt\_y\_pred = predict(dt\_classifier, newdata = test, type = 'class')

print(dt\_y\_pred)

# Making the Confusion Matrix for Decision Tree

cm = table(test$Species, dt\_y\_pred)

print(cm)

#accuracy of DT model

DTaccu = ((12+9+11)/nrow(test))\*100 #true positive nos of 3\*3 confusion matrix

DTaccu

#2. k-Nearest Neighbours

install.packages("class")

library(class)

cl = train$Species

set.seed(1234)

knn\_y\_pred = knn(train[,1:4],test[,1:4],cl,k=5)

# cm of k-Nearest Neighbours

cm = table(test$Species, knn\_y\_pred)

print(cm)

#accuracy of KNN model

KNNaccu = ((12+11+11)/nrow(test))\*100 #true positive nos of 3\*3 confusion matrix

KNNaccu

#3. Support Vector Machine(SVM)

svmclassifier = svm(Species ~ . ,data = train)

svm\_y\_pred = predict(svmclassifier,newdata = test)

cm = table(test$Species, svm\_y\_pred)

print(cm)

#accuracy of SVM model

SVMaccu = ((12+11+11)/nrow(test))\*100 #true positive nos of 3\*3 confusion matrix

SVMaccu

#Decision Tree vs kNN

which(dt\_y\_pred != knn\_y\_pred)

#Decision Tree vs SVM

which(dt\_y\_pred != svm\_y\_pred)

#svm vs kNN

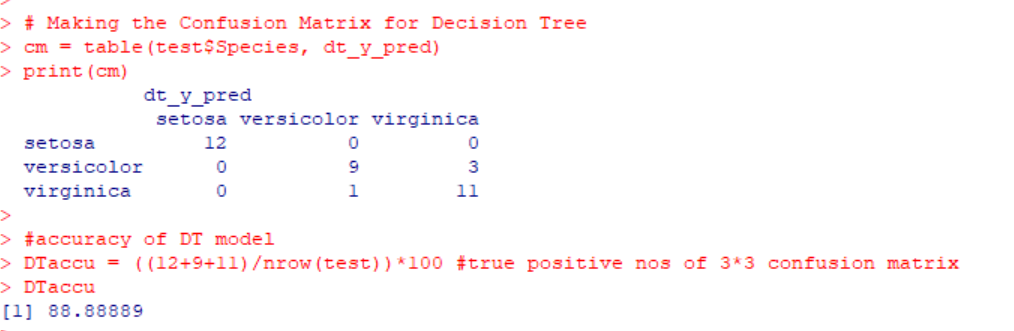
which(svm\_y\_pred != knn\_y\_pred) #both are equal

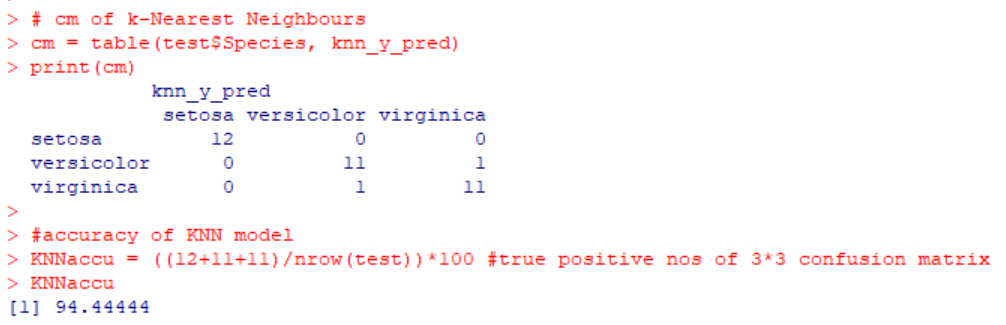
#Comparison of the accuracy of different models on testing dataset.

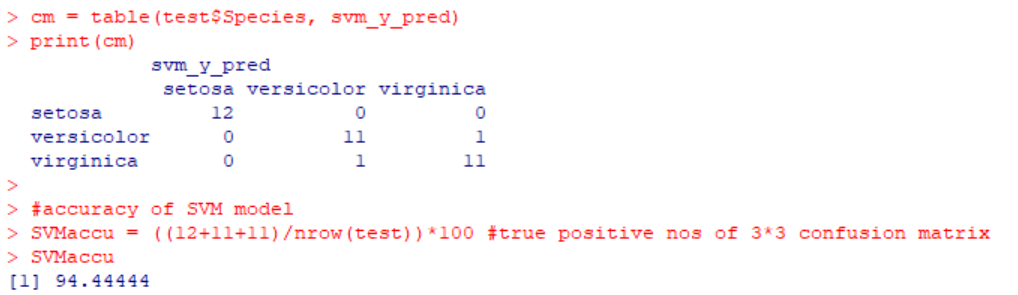
models = data.frame(Technique = c("Decision Tree","kNN","SVM"),Accuracy\_Percentage = c(88.88889,94.44444,94.44444))

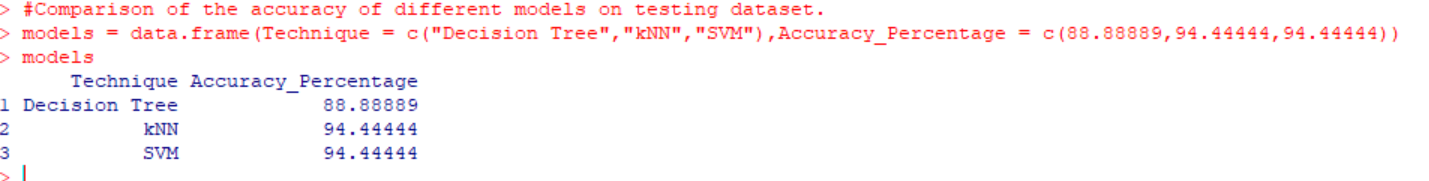
models

**Output:**

****

****

****

****

**Practical No. 06**

**Aim: Install, configure and run Hadoop and HDFS and explore HDFS on Windows**

**Steps to Install Hadoop**

1. Install Java JDK 1.8
2. Download Hadoop and extract and 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 and YARN files
7. Set Java Home environment in Hadoop environment
8. Setup Complete. Test by executing start-all.cmd

**There are two ways to install Hadoop, i.e.**

1. Single node
2. Multi node

Here, we use multi node cluster.

1. **Install Java**
2. – Java JDK Link to download

<https://www.oracle.com/java/technologies/javase-jdk8-downloads.html>

1. – extract and install Java in C:\Java
2. – open cmd and type -> javac -version

Graphical user interface, text

Description automatically generated

1. **Download Hadoop**

<https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz>

* right click .rar.gz file -> show more options -> 7-zip->and extract to C:\Hadoop-3.3.0\

Graphical user interface, text, application

Description automatically generated

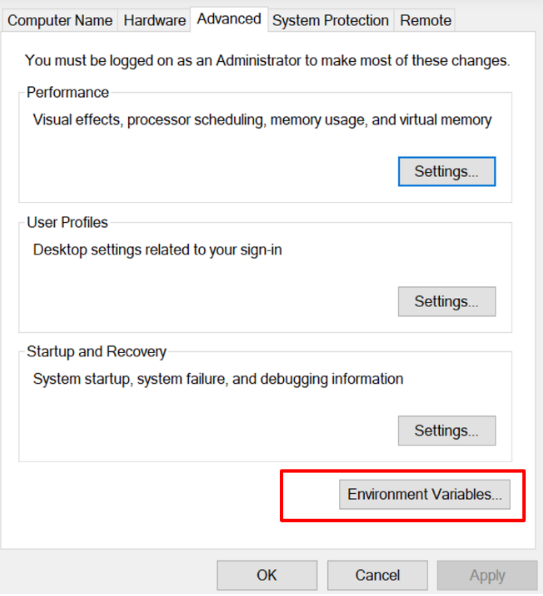
1. **Set the path JAVA\_HOME Environment variable**
2. **Set the path HADOOP\_HOME Environment variable**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated



Click on **New to both user variables and system variables.**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Click on user variable -> path -> edit-> add path for Hadoop and java upto ‘bin’** Graphical user interface, text, application

Description automatically generated

Click Ok, Ok, Ok.

1. **Configurations**

**Edit file C:/Hadoop-3.3.0/etc/hadoop/core-site.xml,**

paste the xml code in folder and save

**======================================================**

<configuration>

<property>

<name>fs.defaultFS</name>

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

</property>

</configuration>

**======================================================**

**Rename “mapred-site.xml.template” to “mapred-site.xml” and edit this file C:/Hadoop-3.3.0/etc/hadoop/mapred-site.xml, paste xml code and save this file.**

**======================================================**

<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>

**======================================================**

**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”**

**======================================================**

**Edit file C:\Hadoop-3.3.0/etc/hadoop/hdfs-site.xml,**

**paste xml code and save this file.**

<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 xml code and save this file.**

<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. **Edit file C:/Hadoop-3.3.0/etc/hadoop/hadoop-env.cmd**

Find “JAVA\_HOME=%JAVA\_HOME%” and replace it as

set JAVA\_HOME="C:\Java\jdk1.8.0\_361"

**======================================================**

1. **Download “redistributable” package**

**Download and run VC\_redist.x64.exe**

This is a “redistributable” package of the Visual C runtime code for 64-bit applications, from Microsoft. It contains certain shared code that every application written with Visual C expects to have available on the Windows computer it runs on.

1. **Hadoop Configurations**

**Download bin folder from**

[**https://github.com/s911415/apache-hadoop-3.1.0-winutils**](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.**

1. **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.**
2. **Format the NameNode**

**– Open cmd ‘Run as Administrator’ and type command “hdfs namenode –format”**

Text

Description automatically generated

Text

Description automatically generated

**11. Testing**

**– Open cmd ‘Run as Administrator’ and change directory to C:\Hadoop-3.3.0\sbin**

**– type start-all.cmd**

**OR**

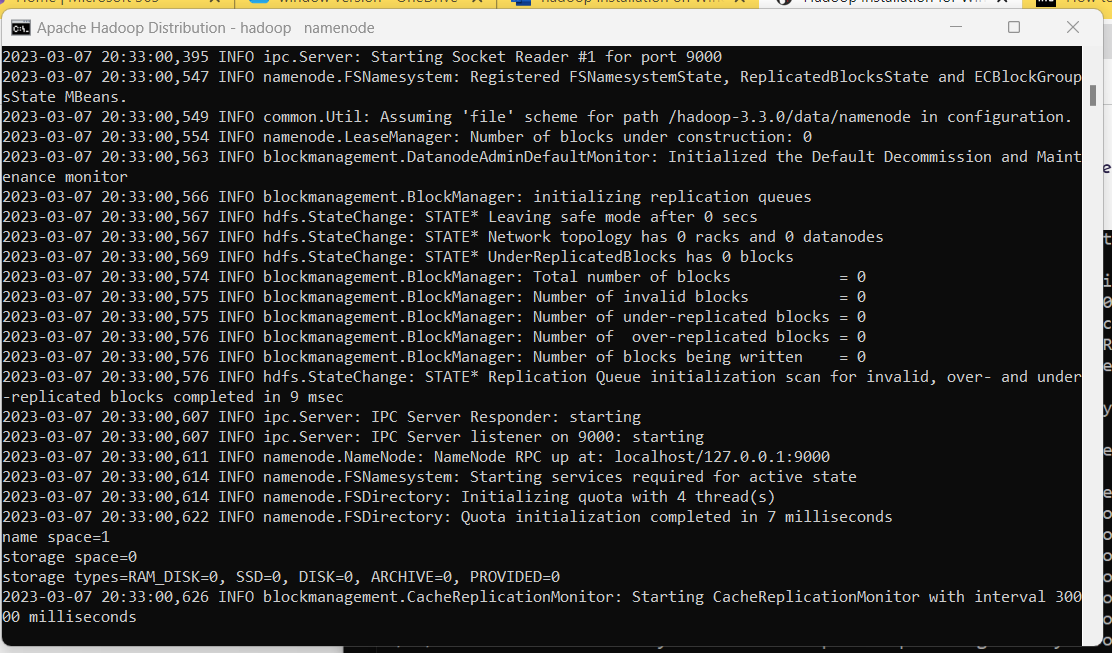
**- type start-dfs.cmd**

**– type start-yarn.cmd**

Text

Description automatically generated

**– You will get 4 more running threads for Datanode, namenode, resouce manager and node manager**



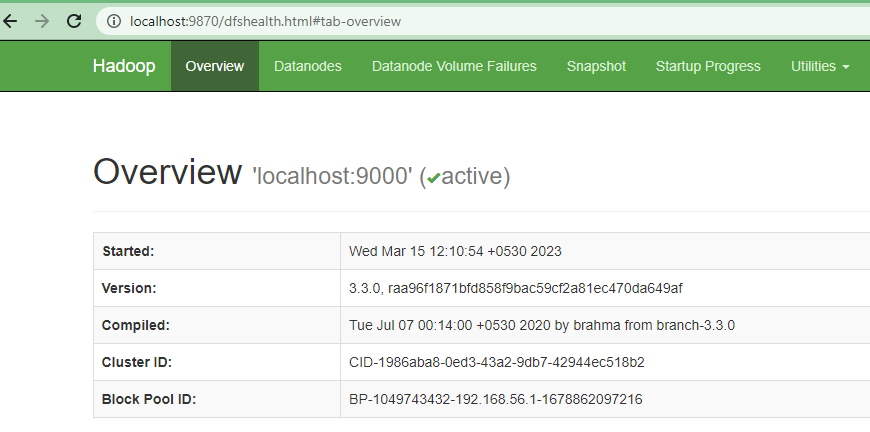
**Output:**

12. Type JPS command to start-all.cmd command prompt, you will get following output.

Text

Description automatically generated

13. Run <http://localhost:9870/> from any browser



Graphical user interface, text, application, table, email, Excel

Description automatically generated

**Practical No. 07**

**Aim: Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python**

**Requirements**

a. PyMongo

b. Mongo Database

**Step A: Install Mongo database**

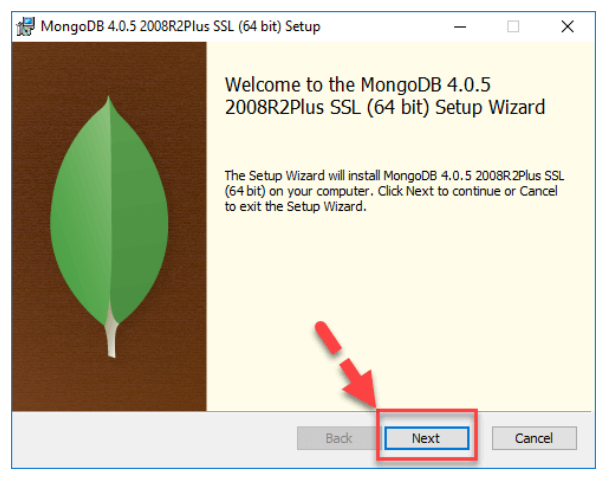
Step 1) Go to (https://www.mongodb.com/download-center/community) and Download

MongoDB Community Server. We will install the 64-bit version for Windows.

Graphical user interface, text, application

Description automatically generated

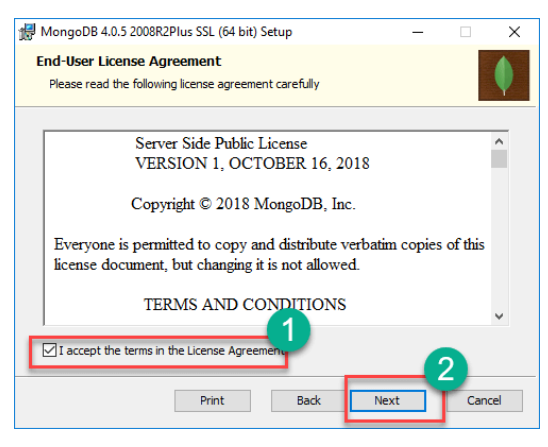
Step 2) Once download is complete open the msi file. Click Next in the start up screen



Step 3)

1. Accept the End-User License Agreement

2. Click Next



Step 4) Click on the "complete" button to install all of the components. The custom

option can be used to install selective components or if you want to change the location

of the installation.

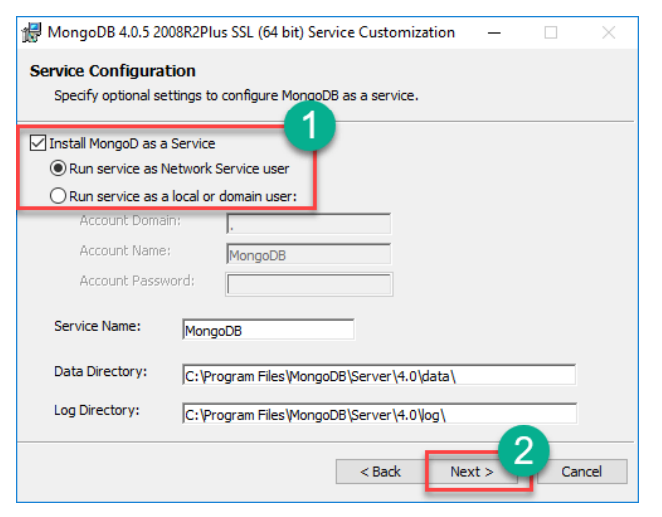
Graphical user interface, text, application, email

Description automatically generated

Step 5)

1. Select “Run service as Network Service user”. make a note of the data directory,

we”ll need this later.

2. Click Next

Step 6) Click on the Install button to start the installation.

Graphical user interface, text, application

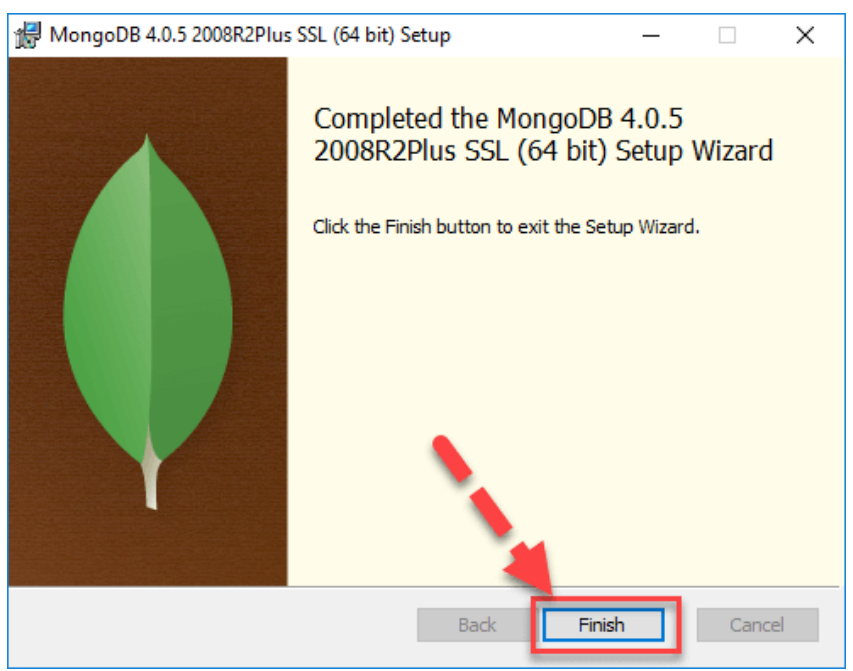
Description automatically generated

Step 7) Installation begins. Click Next once completed.

Graphical user interface, application

Description automatically generated

Step 8) Click on the Finish button to complete the installation.



**Test Mongodb**

**Step 1**) Go to " C:\Program Files\MongoDB\Server\4.0\bin" and double click on **mongo.exe.** Alternatively, you can also click on the MongoDB desktop icon.

* **Create the directory where MongoDB will store its files.**

Open command prompt window and apply following commands

C:\users\admin> cd\

C:\>md data\db

**Step 2) Execute mongodb**

Open another command prompt window.

C:\> cd C:\Program Files\MongoDB\Server\4.0\bin

C:\Program Files\MongoDB\Server\4.0\bin> mongod

*In case if it gives an error then run the following command:*

*C:\Program Files\MongoDB\Server\4.0\bin> mongod –repair*

Text

Description automatically generated

**Step 3) Connect to MongoDB using the Mongo shell**

Let the MongoDB daemon to run.

Open another command prompt window and run the following commands:

C:\users\admin> cd C:\Program Files\MongoDB\Server\4.0\bin

C:\Program Files\MongoDB\Server\4.0\bin>mongo

Text

Description automatically generated

**Step 4) Install PyMongo**

Open another command prompt window and run the following commands:

Check the python version on your desktop / laptop and copy that path from window explorer

C:\users\admin>cd C:\Program Files\Python311\Scripts

C:\Program Files\<Python38>\Scripts > python -m pip install pymongo

Text

Description automatically generated

Note: # **-m** option is for <module-name>

Now you have downloaded and installed a mongoDB driver.

**Step 5) Test PyMongo**

Run the following command from python command prompt

import pymongo

Now, either create a file in Python IDLE or run all commands one by one in sequence on Python cell

**Program 1: Creating a Database: create\_dp.py**

import pymongo

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

mydb = myclient["mybigdata"]

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



**Progam 2: Creating a Collection:  create\_collection.py**

import pymongo

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

mydb = myclient["mybigdata"]

mycol=mydb["student"]

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



**Progam 3: Insert into Collection:  insert\_into\_collection.py**

import pymongo

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

mydb = myclient["mybigdata"]

mycol=mydb["student"]

mydict={"name":"Beena", "address":"Mumbai"}

x=mycol.insert\_one(mydict) # insert\_one(containing the name(s) and value(s) of each field

**Program 4: Insert Multiple data into Collection: insert\_many.py**

import pymongo

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

mydb = myclient["mybigdata"]

mycol=mydb["student"]

mylist=[{"name":"Khyati", "address":"Mumbai"}, {"name":"Kruti", "address":"Mumbai"},

{"name":"Nidhi", "address":"Pune"}, {"name":"Komal", "address":"Pune"},]

x=mycol.insert\_many(mylist)

**Step 6) Test in Mongodb to check database and data inserted in collection**

a. If you want to check your database list, use the command show dbs in mongo

command prompt

> show dbs

Graphical user interface, text

Description automatically generated

b. If you want to use a database with name mybigdata, then use database

statement would be as follow:

> use mybigdata



c. If you want to check collection in mongodb use the command show collections

> show collections



d. If you want to display the first row from collection: db.collection\_name.find()

> db.student.findOne()

Text

Description automatically generated

e. If you want to display all the data from collection: db.collection\_name.find()

> db.student.find()

Text

Description automatically generated

f. count number of rows in a collection

> db.student.count()



**Site for R packages documentation:**

https://cran.r-project.org/web/packages/available\_packages\_by\_name.html