**Practical 2:**

**Aim**: Classification using SVM

**Requirement:**

R tool

**Code:**

getwd()

read.csv()

ds=read.csv("E:/Rajdeep/bigdata pract/dataset/social.csv",TRUE,",")

ds

ds=ds[3:5]

ds

install("catools")

library(caTools)

set.seed(123)

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

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

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

ds

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

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

test\_set[-3]

training\_set[-3]

install.packages('e1071')

library('e1071')

classifier=svm(formula=Purchased ~ ., data= training\_set , type='C-classification',kernal='linear')

classifier

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

y\_pred

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

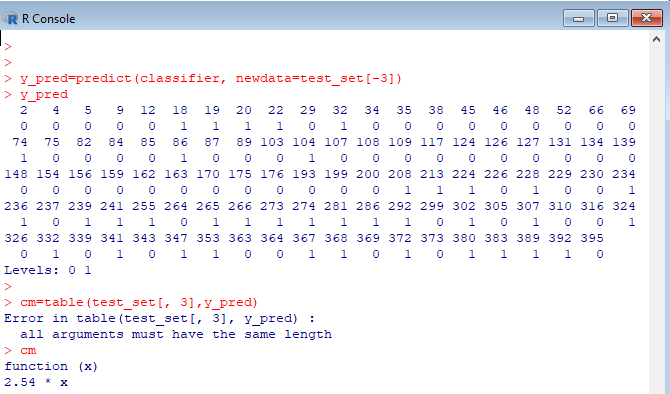
cm

****

****

****

**Output:**



Practical 3:

**Aim**: write program in R of Naive baye's theorem

**Requirement:**

R tool

**Code:**

data(iris)

str(iris)

install packages("e1071")

install packages("caTools")

install packages("caret")

library(e1071)

library(caTools)

library(caret)

split <- sample.split(iris,SplitRatio=0.7)

train\_c1 <-subset(iris,split=="TRUE")

test\_c1 <- subset(iris,split == "FALSE")

train\_scale <- scale(train\_c1[, 1:4])

test\_scale <- scale(test\_c1[,1:4])

set.seed(120)

classifier\_c1 <- naiveBayes(Species ~ ., data = train\_c1)

classifier\_c1

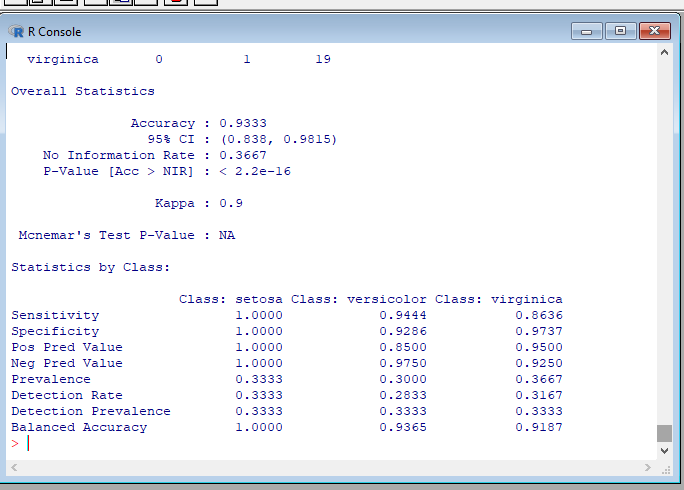
y\_pred <- predict(classifier\_c1, newdata= test\_c1)

cm <- table(test\_c1$Species, y\_pred)

cm

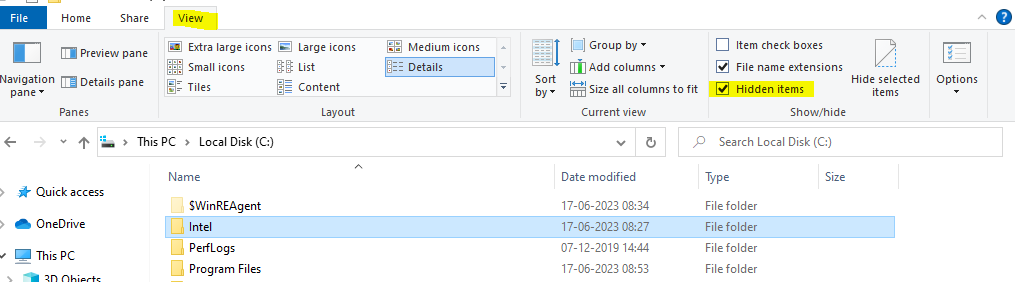
confusionMatrix(cm)

**Output:**



**Install python package:**

1. You will need to make the hidden folder visible: go to “C:” drive on top click on tab “view”
2. Select “hidden Items” option:

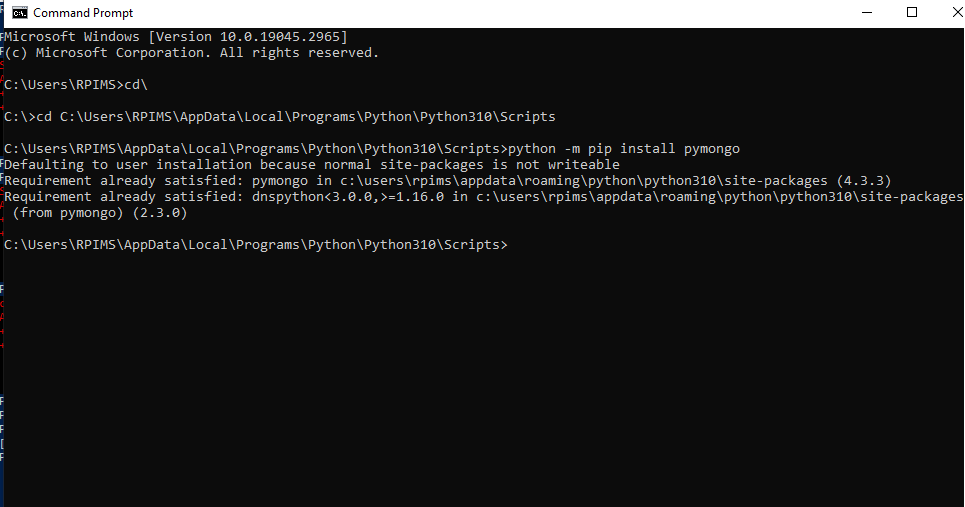


1. Go to the below path:

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts

1. Set the below path in command prompt and then use the below command:

python -m pip install pymongo



**Practical :4**

**Aim:** Implement an application that stores big data in Hbase / MongoDB and

manipulate it using R / Python

**Requirement:**

a. Python Package: 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.



**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.



**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.



**Step 7)** Installation begins. Click Next once completed.



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



**Program 1:** Displaying the database name:

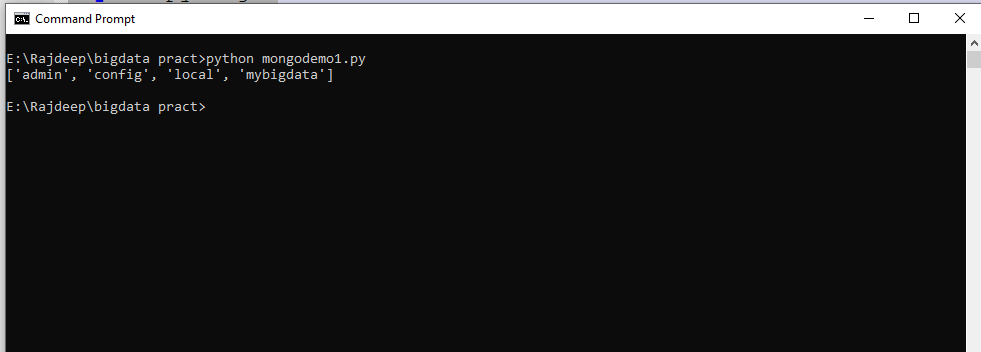
import pymongo

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

mydb = myclient["mybigdata"]

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

Output:



**Program 2:** Creating collection:

import pymongo

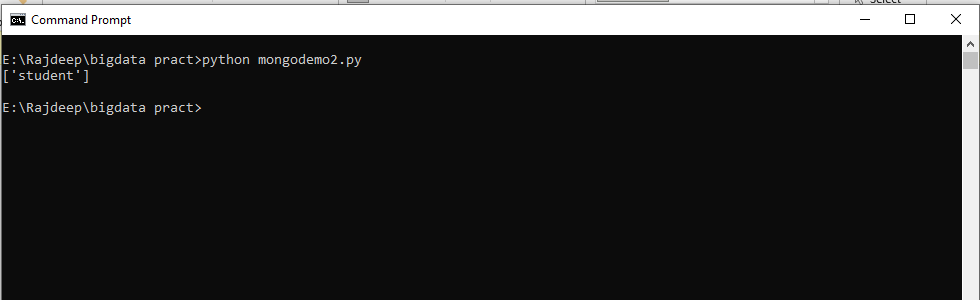
myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mybigdata"]

mycol=mydb["student"]

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

Output:



**Program 3**: Inserting Data

import pymongo

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

mydb = myclient["mybigdata"]

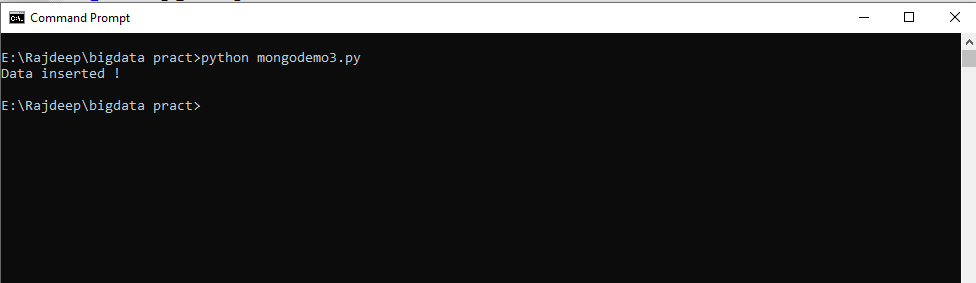
mycol=mydb["student"]

mydict={"name":"vai", "address":"bhy"}

x=mycol.insert\_one(mydict)

print("Data inserted !")

Output:



**Program 4**: Insert Multiple data into Collection

import pymongo

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

mydb = myclient["mybigdata"]

mycol=mydb["student"]

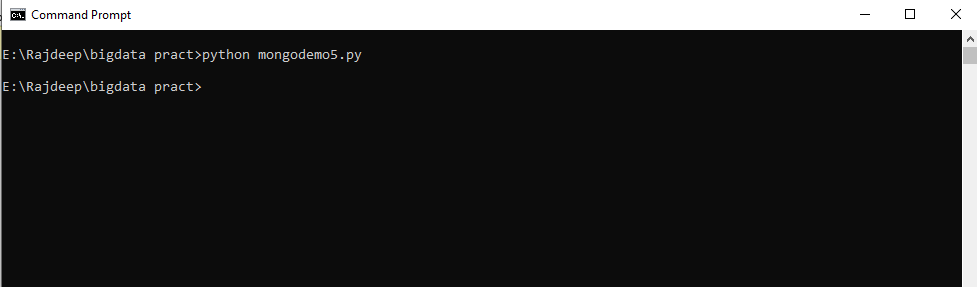
mylist=[{"name":"Ganesh", "address":"Mumbai"}, {"name":"Varun", "address":"Mumbai"},

{"name":"Prasoon", "address":"Pune"}, {"name":"Satish", "address":"Pune"},]

x=mycol.insert\_many(mylist)

print("Data inserted !")

Output:



**Program 5:** Displaying the collection data:

import pymongo

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

mydb = myclient["mybigdata"]

mycol = mydb["student"]

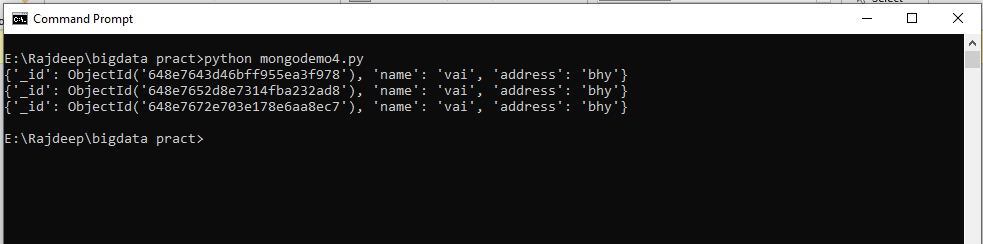
myquery = { "name": "Vai" }

mydoc = mycol.find(myquery)

for x in mydoc:

print(x)

Output:



**Practical 5:**

K means clustering.

**Aim:** Read a datafile grades\_km\_input.csv and apply k-means clustering.

**Requirement:**

R tool

**Code:**

install.packages("plyr")

install.packages("ggplot2")

install.packages("cluster")

install.packages("lattice")

install.packages("grid")

install.packages("gridExtra")

library(plyr)

library(ggplot2)

library(cluster)

library(lattice)

library(grid)

library(gridExtra)

grade\_input=as.data.frame(read.csv("E:/Rajdeep/bigdata pract/dataset/grades\_km\_input.csv"))

kmdata\_orig=as.matrix(grade\_input[, c ("Student","English","Math","Science")])

kmdata=kmdata\_orig[,2:4]

kmdata[1:10,]

wss=numeric(15)

for(k in 1:15)wss[k]=sum(kmeans(kmdata,centers=k,nstart=25)$withinss)

plot(1:15,wss,type="b",xlab="Number of Clusters",ylab="Within sum of square")

km = kmeans(kmdata,3,nstart=25)

km

c( wss[3] , sum(km$withinss))

df=as.data.frame(kmdata\_orig[,2:4])

df$cluster=factor(km$cluster)

centers=as.data.frame(km$centers)

g1=ggplot(data=df, aes(x=English, y=Math, color=cluster )) +

geom\_point() + theme(legend.position="right") +

geom\_point(data=centers,aes(x=English,y=Math, color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend =FALSE)

g2=ggplot(data=df, aes(x=English, y=Science, color=cluster )) +

geom\_point () +geom\_point(data=centers,aes(x=English,y=Science, color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)

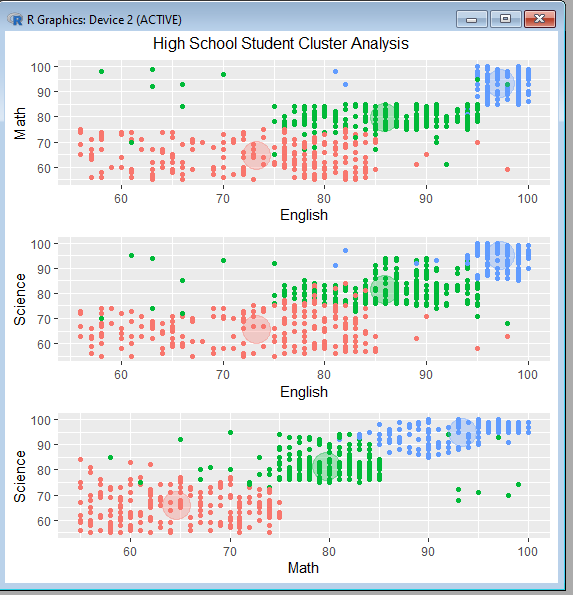
g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +

geom\_point () + geom\_point(data=centers,aes(x=Math,y=Science, color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)

tmp=ggplot\_gtable(ggplot\_build(g1))

grid.arrange(arrangeGrob(g1 + theme(legend.position="none"),g2 + theme(legend.position="none"),g3 + theme(legend.position="none"),top ="High School Student Cluster Analysis" ,ncol=1))

**Output:**



**Practical 6:**

1. Simple Linear regression

**Aim:** Create your own data for years of experience and salary in lakhs and apply linear regression model to predict the salary

**Requirement:**

R tool

Code:

years\_of\_exp = c(7,5,1,3)

salary\_in\_lakhs = c(21,13,6,8)

employee.data = data.frame(years\_of\_exp, salary\_in\_lakhs)

employee.data

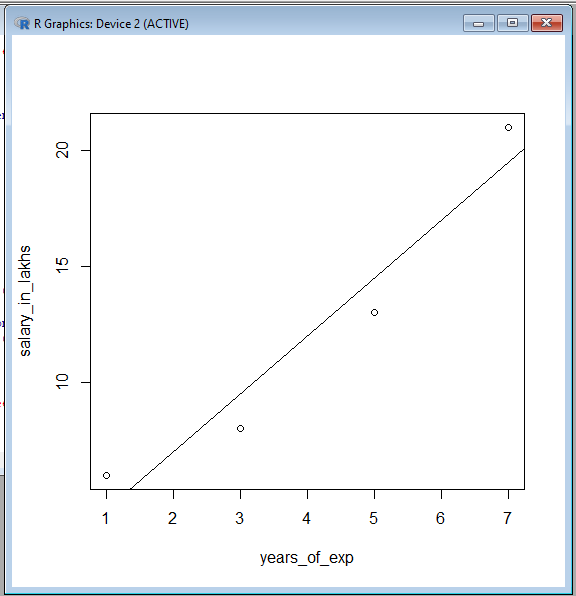
model <- lm(salary\_in\_lakhs ~ years\_of\_exp, data = employee.data)

summary(model)

plot(salary\_in\_lakhs ~ years\_of\_exp, data = employee.data)

abline(model)

**Output:**



b.: Logistic regression:

**Aim:** Take the in-built data from ISLR package and apply generalized logistic regression to find whether a person would be defaulter or not; considering input as student, income and balance.

Code:

install.packages("ISLR")

library(ISLR)

data <- ISLR::Default

print (head(ISLR::Default))

summary(data)

nrow(data)

set.seed(1)

sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.7,0.3))

print (sample)

train <- data[sample, ]

test <- data[!sample, ]

nrow(train)

nrow(test)

model <- glm(default~student+balance+income, family="binomial", data=train)

summary(model)

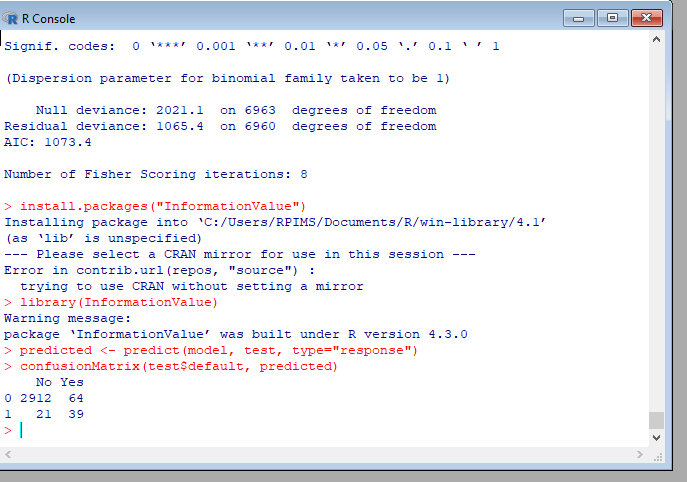
install.packages("InformationValue")

library(InformationValue)

predicted <- predict(model, test, type="response")

confusionMatrix(test$default, predicted)

Output:



**Practical 7:**

**Aim:** Implement Decision tree classification techniques

**Requirement:**

R tool

**Code:**

library("party")

print(head(readingSkills))

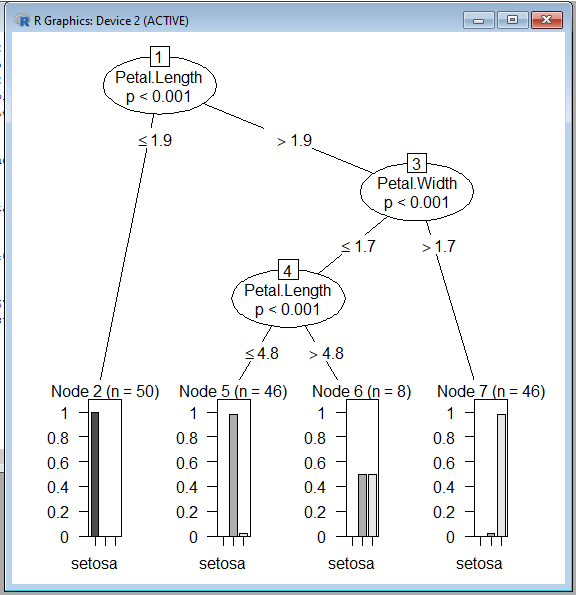
str(iris)

iris\_ctree <- ctree(Species ~ Sepal.Width + Sepal.Length + Petal.Length + Petal.Width, data=iris

print (iris\_ctree)

plot(iris\_ctree)

**Output:**



**Practical 8:**

Apriori algorithm

**Aim**: Perform Apriori algorithm using Groceries dataset from the R arules package.

**Requirement:**

R tool

**Code:**

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)")

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

summary(itemsets)

inspect(itemsets)

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

summary(itemsets\_3)

inspect(itemsets\_3)

**Output:**

