Practical 1

AIM: Practical of Principal Component Analysis(PCA).

Code:

```
data_iris <- iris[1:4]
Cov_data <- cov(data_iris )
Eigen_data <- eigen(Cov_data)
PCA_data <- princomp(data_iris ,cor="False")
Eigen_data$values
PCA_data$sdev^2
PCA_data$loadings[,1:4]
Eigen_data$vectors
summary(PCA_data)
biplot (PCA_data)

screeplot(PCA_data, type="lines")
model2 = PCA_data$loadings[,1]
model2_scores <- as.matrix(data_iris) %*% model2

library(class)
install.packages("e1071")
library(e1071)
mod1<-naiveBayes(iris[,1:4], iris[,5])
mod2<-naiveBayes(model2_scores, iris[,5])
table(predict(mod1, iris[,1:4]), iris[,5])
table(predict(mod2, model2_scores), iris[,5])
```

Practical 2

AIM: Practical of Clustering.

Code:

```
install.packages("ggplot2")

library(ggplot2)

All Command:

scatter <- ggplot(data=iris, aes(x = Sepal.Length, y = Sepal.Width))

scatter + geom_point(aes(color=Species, shape=Species)) +

theme_bw()+

xlab("Sepal Length") +  ylab("Sepal Width") +

ggtitle("Sepal Length-Width")


ggplot(data=iris, aes(Sepal.Length, fill = Species))+

theme_bw()+

geom_density(alpha=0.25)+

labs(x = "Sepal.Length", title="Species vs Sepal Length")


vol <- ggplot(data=iris, aes(x = Sepal.Length))

vol + stat_density(aes(ymax = ..density..,  ymin = -..density..,

fill = Species, color = Species),

geom = "ribbon", position = "identity") +

facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Length",

title="Species vs Sepal Length")


vol <- ggplot(data=iris, aes(x = Sepal.Width))

vol + stat_density(aes(ymax = ..density..,  ymin = -..density..,

fill = Species, color = Species),

geom = "ribbon", position = "identity") +

facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Width",

title="Species vs Sepal Width")
```

```
irisData <- iris[,1:4]

totalwSS<-c()

for (i in 1:15)

{

clusterIRIS <- kmeans(irisData, centers=i)

totalwSS[i]<-clusterIRIS$tot.withinss

}

plot(x=1:15,

y=totalwSS,

type="b",

xlab="Number of Clusters",

ylab="Within groups sum-of-squares")


install.packages("NbClust")

library(NbClust)

par(mar = c(2,2,2,2))

nb <- NbClust(irisData, method = "kmeans")


hist(nb$Best.nc[1,], breaks = 15, main="Histogram for Number of Clusters")


install.packages("vegan")

library(vegan)

modelData <- cascadeKM(irisData, 1, 10, iter = 100)

plot(modelData, sortg = TRUE)

modelData$results[2,]

which.max(modelData$results[2,])




library(cluster)

cl <- kmeans(iris[,-5], 2)

dis <- dist(iris[,-5])^2

sil = silhouette (cl$cluster, dis)

plot(sil, main = "Clustering Data with Silhoutte plot using 2 Clusters", col = c("cyan", "blue"))
```

```
library(cluster)

cl <- kmeans(iris[,-5], 8)

dis <- dist(iris[,-5])^2

sil = silhouette (cl$cluster, dis)

plot(sil, main = "Clustering Data with Silhoutte plot using 8 Clusters", col = c("cyan",

"blue", "orange", "yellow", "red", "gray", "green", "maroon"))


install.packages("factoextra")

library(factoextra)

install.packages("clustertend")

library(clustertend)

genx<-function(x){

runif(length(x), min(x), (max(x)))

}

random_df <- apply(iris[,-5], 2, genx)

random_df <- as.data.frame(random_df)

iris[,-5] <- scale(iris[,-5])

random_df <- scale(random_df)

res <- get_clust_tendency(iris[,-5],

n = nrow(iris) -1 ,

graph = FALSE)

res$hopkins_stat

hopkins(iris[,-5], n = nrow(iris) -1)

res <- get_clust_tendency(random_df, n = nrow(random_df)-1,

graph = FALSE)

res$hopkins_stat
```

Practical 3

AIM: Practical of Time-series forecasting.

Code:

```
data(AirPassengers)
class(AirPassengers)
start(AirPassengers)
end(AirPassengers)
frequency(AirPassengers)
summary(AirPassengers)

plot(AirPassengers)

abline(reg=lm(AirPassengers~time(AirPassengers)))

cycle(AirPassengers)

plot(aggregate(AirPassengers,FUN=mean))

boxplot(AirPassengers~cycle(AirPassengers))

acf(log(AirPassengers))

acf(diff(log(AirPassengers)))

(fit <- arima(log(AirPassengers), c(0, 1, 1),seasonal = list(order = c(0, 1, 1), period = 12)))

pred <- predict(fit, n.ahead = 10*12)
ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))
```

Practical 4

AIM: Practical of Simple/Multiple Linear Regression

Code:

```
lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients

plot(iris$Petal.Length, iris$Petal.Width, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",
ylab="Petal width")

abline(lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients, col="black")

lm(Petal.Width ~ Petal.Length, data=iris)$coefficients

plot(iris$Petal.Length, iris$Petal.Width, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",
ylab="Petal width")

abline(lm(Petal.Width ~ Petal.Length, data=iris)$coefficients, col="black")

summary(lm(Petal.Width ~ Petal.Length, data=iris))

plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Sepal Width",
ylab="Sepal Length")

abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")

summary(lm(Sepal.Length ~ Sepal.Width, data=iris))

plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,
bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",
ylab="Sepal length")
abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")
abline(lm(Sepal.Length ~ Sepal.Width,
data=iris[which(iris$Species=="setosa"),])$coefficients, col="red")
abline(lm(Sepal.Length ~ Sepal.Width,
data=iris[which(iris$Species=="versicolor"),])$coefficients, col="green3")
abline(lm(Sepal.Length ~ Sepal.Width,
data=iris[which(iris$Species=="virginica"),])$coefficients, col="blue")

lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients

lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="versicolor"),])$coefficients

lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficients

lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients


summary(lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris))

summary(step(lm(Sepal.Length ~ Sepal.Width * Species, data=iris)))

lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients

lm(Sepal.Length ~ Sepal.Width:Species + Species, data=iris)$coefficients
```

Practical 5

AIM: Practical of Logistics Regression.

Code:

```
library(datasets)
ir_data<- iris
head(ir_data)

str(ir_data)

levels(ir_data$Species)

sum(is.na(ir_data))

ir_data<-ir_data[1:100,]

set.seed(100)
samp<-sample(1:100,80)
ir_test<-ir_data[samp,]
ir_ctrl<-ir_data[-samp,]

install.packages("ggplot2")
library(ggplot2)
install.packages("GGally")
library(GGally)
ggpairs(ir_test)

y<-ir_test$Species; x<-ir_test$Sepal.Length
glfit<-glm(y~x, family = 'binomial')

summary(glfit)

newdata<- data.frame(x=ir_ctrl$Sepal.Length)
predicted_val<-predict(glfit, newdata, type="response")
prediction<-data.frame(ir_ctrl$Sepal.Length, ir_ctrl$Species,predicted_val)
prediction
```

```
qplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab
= 'Prediction using Logistic Reg.')
```

Practical 6

AIM: Practical of Decision Tree.

Code:

```
mydata<-data.frame(iris)

attach(mydata)

install.packages("rpart")

library(rpart)

model<-rpart(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=mydata, method="class")

plot(model)

text(model,use.n=TRUE,all=TRUE,cex=0.8)


install.packages("tree")

library(tree)

model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=mydata, method="class",
split="gini")

plot(model1)

text(model1,all=TRUE,cex=0.6)


install.packages("party")

library(party)


model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=mydata)

plot(model2)


library(tree)

mydata<-data.frame(iris)

attach(mydata)


model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=mydata,

method="class",

control =  tree.control(nobs =  150, mincut = 10))

plot(model1)

text(model1,all=TRUE,cex=0.6)


predict(model1,iris)
```

```
model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,

data = mydata, controls = ctree_control(maxdepth=2))

plot(model2)
```