**1) K-means clustering**

> set.seed(8953)

> iris2 <- iris

> iris2$Species <- NULL

> (kmeans.result <- kmeans(iris2, 3))

> table(iris$Species, kmeans.result$cluster)

> plot(iris2[c("Sepal.Length", "Sepal.Width")], col = kmeans.result$cluster)

> points(kmeans.result$centers[, c("Sepal.Length", "Sepal.Width")],col = 1:3, pch = 8, cex = 2) # plot cluster centers

**2) Hierarchical clustering**

> set.seed(2835)

> idx <- sample(1:dim(iris)[1], 40)

> irisSample <- iris[idx, ]

> irisSample$Species <- NULL

> hc <- hclust(dist(irisSample), method = "ave")

> plot(hc, hang = -1, labels = iris$Species[idx])

> rect.hclust(hc, k = 3)

> groups <- cutree(hc, k = 3)

**3) DENSITY BASED CLUSTERING**

install.packages("fpc")

library(fpc)

iris2 <- iris[-5]

ds <- dbscan(iris2, eps = 0.42, MinPts = 5)

table(ds$cluster, iris$Species)

plot(ds, iris2)

plotcluster(iris2, ds$cluster)

plot(ds, iris2[c(1, 4)])

**4) BAGGING**

install.packages("ipred")

install.packages("e1071")

install.packages("caret")

library(ipred)

library(rpart)

library(e1071)

library(caret)

url="http://freakonometrics.free.fr/german\_credit.csv"

GermanCredit=read.csv(url, header = TRUE, sep = ",")

names(GermanCredit)

class(GermanCredit$Creditability)

# Convert target variable to factor

GermanCredit$Creditability <- factor(GermanCredit$Creditability)

German.bagging <- bagging(Creditability ~., data=GermanCredit,

control=rpart.control(maxdepth=5, minsplit=15))

# store the dataset

GermanCredit$pred.class <- predict(German.bagging,

GermanCredit)

# Create Confusion Matrix

confusionMatrix(data=factor(GermanCredit$pred.class),

reference=factor(GermanCredit$Creditability),

positive='1')

**5)time series**

a <- ts(1:20, frequency = 12, start = c(2011, 3))

print(a)

str(a)

attributes(a)

plot(AirPassengers)

apts <- ts(AirPassengers, frequency = 12)

f <- decompose(apts)

plot(f$figure, type = "b")

plot(f)

# TIMEFORECASTING

fit <- arima(AirPassengers, order = c(1, 0, 0), list(order = c(2, 1, 0), period = 12))

fore <- predict(fit, n.ahead = 24)

# error bounds at 95% confidence level

U <- fore$pred + 2 \* fore$se

L <- fore$pred - 2 \* fore$se

ts.plot(AirPassengers, fore$pred, U, L, lty = c(1, 1, 2, 2))

legend("topleft",lty = c(1, 1, 2), c("Actual", "Forecast", "Error Bounds (95%Confidence)"))

#TIME CLUSTERING

install.packages("dtw")

library(dtw)

idx <- seq(0, 2 \* pi, len = 100)

a <- sin(idx) + runif(100)/10

b <- cos(idx)

align <- dtw(a, b, keep = T)

dtwPlotTwoWay(align)

sc <- read.table("C:/Program Files/R/synthetic\_control.data", header = F, sep = "")

idx <- c(1, 101, 201, 301, 401, 501)

sample1 <- t(sc[idx, ])

plot.ts(sample1, main = "six classes")

#HIERARCHICAL

n <- 10

s <- sample(1:100, n)

idx <- c(s, 100 + s, 200 + s, 300 + s, 400 + s, 500 + s)

sample2 <- sc[idx, ]

observedLabels <- rep(1:6, each = n)

# hierarchical clustering with Euclidean distance

hc <- hclust(dist(sample2), method = "ave")

plot(hc, labels = observedLabels, main = "")

memb <- cutree(hc, k = 8)

table(observedLabels, memb)

myDist <- dist(sample2, method = "DTW")

hc <- hclust(myDist, method = "average")

plot(hc, labels = observedLabels, main = "hierarchical")

# TIMECLASSIFICATION

classId <- rep(as.character(1:6), each = 100)

newSc <- data.frame(cbind(classId, sc))

install.packages("party")

library(party)

ct <- ctree(classId ~ ., data = newSc, controls = ctree\_control(minsplit = 20, minbucket = 5, maxdepth = 5))

pClassId <- predict(ct)

table(classId, pClassId)

(sum(classId == pClassId))/nrow(sc)

install.packages("wavelets")

library(wavelets)

wtData <- NULL

for (i in 1:nrow(sc)) {

a <- t(sc[i, ])

wt <- dwt(a, filter = "haar", boundary = "periodic")

wtData <- rbind(wtData, unlist(c(wt@W, wt@V[[wt@level]])))

}

wtData <- as.data.frame(wtData)

wtSc <- data.frame(cbind(classId, wtData))

ct <- ctree(classId ~ ., data = wtSc, controls = ctree\_control(minsplit=20, minbucket=5, maxdepth=5))

pClassId <- predict(ct)

table(classId, pClassId)

(sum(classId==pClassId)) / nrow(wtSc)

plot(ct, ip\_args = list(pval = F), ep\_args = list(digits = 0))

k <- 20

newTS <- sc[501, ] + runif(100) \* 15

distances <- dist(newTS, sc, method = "DTW")

s <- sort(as.vector(distances), index.return = TRUE)

# class IDs of k nearest neighbours

table(classId[s$ix[1:k]])

**NAIVEBAYES**   
  
install.packages("caret")  
library(caTools)  
library(caret)  
library(e1071)  
  
f<-data.freeame(iris)  
ind<-sample(nrow(iris),floor(nrow(iris)\*0.7))  
train<-f[ind,]  
test<-f[-ind,]  
x\_train<-train[,1:4]  
y\_train<-train$Species  
x\_test<-test[,1:4]  
y\_test<-test$Species  
  
classifier=naiveBayes(x\_train,y\_train,laplace=1)  
predictions<-predict(classifier,x\_test)  
confu<-confusionMatrix(predictions,y\_test)  
print(confu)  
  
**FPGrowth**  
  
library(rCBA)  
data("iris")  
classifier<-rCBA::buildFPGrowth(iris[sample(nrow(iris),20),],"Species",parallel=FALSE)  
model<-classifier$model  
predictions<-rCBA::classification(iris,model)  
table(predictions)  
sum(as.character(iris$Species)==as.character(predictions),na.rm=TRUE)/length(predictions)  
  
**APRIORI**   
  
library(arules)  
library(arulesViz)  
library(datasets)  
  
data("Groceries")  
itemFrequencyPlot(Groceries,topN=20,type="absolute")  
  
rules<-apriori(Groceries,parameter=list(supp=0.001,conf=0.6))  
options(digits=2)  
inspect(rules[1:5])  
plot(rules,method="graph",interactive=TRUE,shading="confidence")  
  
#sorting rules  
rules<-sort(rules,by="confidence",decreasing=TRUE)  
  
#using appearance  
rules<-apriori(Groceries,parameter=list(supp=0.001,conf=0.6),  
 appearance=list(default="lhs",rhs="whole milk"),  
 control=list(verbose=F))  
rules<-sort(rules,by="confidence",decreasing=TRUE)  
options(digits=2)  
inspect(rules[1:5])  
  
#plotting  
plot(rules,method="graph",interactive=TRUE,shading=NA)  
  
  
**ECLAT**  
  
library(arules)  
library(arulesViz)  
library(datasets)  
  
data("Groceries")  
itemFrequencyPlot(Groceries,topN=20,type="absolute")  
itemsets<-eclat(Groceries,parameter=list(supp=0.001,maxlen=3))  
  
rules<-ruleInduction(itemsets,Groceries,confidence=0.6)  
inspect(rules[1:5])  
plot(rules,method="graph",interactive=TRUE,shading=NA)  
  
 **DECISION** **TREE-DPLYR**  
  
library(readr)  
library(dplyr)  
library(party)  
library(rpart)  
library(rpart.plot)  
library(ROCR)  
library(magrittr)  
  
titanic3<-"https://goo.gl/At238b"%>%  
 read.csv%>%  
 select(survived,embarked,sex,sibsp,parch,fare)%>%  
 mutate(embarked=factor(embarked),sex=factor(sex))  
  
#splitting data  
.data<-c("training","test")%>%  
 sample(nrow(titanic3),replace=TRUE)%>%  
 split(titanic3,.)  
  
#implementation using rpart  
rtree\_fit<-rpart(survived~.,.data$training)  
rpart.plot(rtree\_fit)  
  
#implementation using ctree  
tree\_fit<-ctree(survived~.,data=.data$training)  
tree\_roc<-tree\_fit%>%  
 predict(newdata=.data$test)%>%  
 prediction(.data$test$survived)%>%  
 performance("tpr","fpr")  
tree\_roc  
  
  
 **DECISION** **TREE-CARET(information** **gain**)   
  
library(caret)  
library(rpart.plot)  
  
data\_url<-c("https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data")  
download.file(url=data\_url,destfile="car.data")  
  
car\_df<-read.csv("car.data",sep=",",header=FALSE)  
  
#data slicing  
  
set.seed(3033)  
intrain<-createDataPartition(y=car\_df, p=0.7, list=FALSE)  
training<-car\_df[intrain,]  
testing<-car\_df[-intrain,]  
  
#training the decision tree  
  
trctr<-trainControl(method="repeatedcv", number=10, repeats=3)  
set.seed(3333)  
dtree\_fit<-train(V7 ~., data=training, method="rpart",  
 parms=list(split="information"),  
 trControl=trctr,  
 tuneLength=10)  
  
dtree\_fit  
  
#plotting  
prp(dtree\_fit$finalModel, box.palette="Reds",tweak=1.2)  
  
#prediction  
test\_pred<-predict(dtree\_fit,newdata=testing)  
confusionMatrix(test\_pred,testing$V7)  
  
  
**DECISION** **TREE-CARET(Gini** **Index**)  
  
library(caret)  
library(rpart.plot)  
  
data\_url<-c("https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data")  
download.file(url=data\_url,destfile="car.data")  
  
car\_df<-read.csv("car.data",sep=",",header=FALSE)  
  
#data slicing  
  
set.seed(3033)  
intrain<-createDataPartition(y=car\_df, p=0.7, list=FALSE)  
training<-car\_df[intrain,]  
testing<-car\_df[-intrain,]  
  
#training the decision tree  
  
trctr<-trainControl(method="repeatedcv", number=10, repeats=3)  
set.seed(3333)  
dtree\_fit\_gini<-train(V7 ~., data=training, method="rpart",  
 parms=list(split="gini"),  
 trControl=trctr,  
 tuneLength=10)  
  
dtree\_fit  
  
#plotting  
prp(dtree\_fit\_gini$finalModel, box.palette="Blues",tweak=1.2)  
  
#prediction  
test\_pred\_gini<-predict(dtree\_fit\_gini,newdata=testing)  
confusionMatrix(test\_pred\_gini,testing$V7)  
  
**VISUALIZATION** id outlook temperature humidity windy play  
1 1 sunny 85 85 FALSE no  
2 2 sunny 80 90 TRUE no  
3 3 overcast 83 86 FALSE yes  
4 4 rainy 70 96 FALSE yes  
5 5 rainy 68 80 FALSE yes  
6 6 rainy 65 70 TRUE no  
7 7 overcast 64 65 TRUE yes  
8 8 sunny 72 95 FALSE no  
9 9 sunny 69 70 FALSE yes  
10 10 rainy 75 80 FALSE yes  
11 11 sunny 75 70 TRUE yes  
12 12 overcast 72 90 TRUE yes  
13 13 overcast 81 75 FALSE yes  
14 14 rainy 71 91 TRUE no  
> data=read.csv(file.choose())  
Dimensionality  
> dim(data)  
[1] 14 6  
Variable names  
> names(data)  
[1] "id" "outlook" "temperature" "humidity" "windy" "play"   
Structure  
> str(data)  
'data.frame': 14 obs. of 6 variables:  
 $ id : int 1 2 3 4 5 6 7 8 9 10 ...  
 $ outlook : Factor w/ 3 levels "overcast","rainy",..: 3 3 1 2 2 2 1 3 3 2 ...  
 $ temperature: int 85 80 83 70 68 65 64 72 69 75 ...  
 $ humidity : int 85 90 86 96 80 70 65 95 70 80 ...  
 $ windy : logi FALSE TRUE FALSE FALSE FALSE TRUE ...  
 $ play : Factor w/ 2 levels "no","yes": 1 1 2 2 2 1 2 1 2 2 ...  
Slicing of data  
> data[1:6,1:5]  
 id outlook temperature humidity windy  
1 1 sunny 85 85 FALSE  
2 2 sunny 80 90 TRUE  
3 3 overcast 83 86 FALSE  
4 4 rainy 70 96 FALSE  
5 5 rainy 68 80 FALSE  
6 6 rainy 65 70 TRUE  
First six rows  
> data[1:6,]  
 id outlook temperature humidity windy play  
1 1 sunny 85 85 FALSE no  
2 2 sunny 80 90 TRUE no  
3 3 overcast 83 86 FALSE yes  
4 4 rainy 70 96 FALSE yes  
5 5 rainy 68 80 FALSE yes  
6 6 rainy 65 70 TRUE no  
> head(data,5)  
 id outlook temperature humidity windy play  
1 1 sunny 85 85 FALSE no  
2 2 sunny 80 90 TRUE no  
3 3 overcast 83 86 FALSE yes  
4 4 rainy 70 96 FALSE yes  
5 5 rainy 68 80 FALSE yes  
Last 5 rows  
> n=nrow(data)  
> data[(n-4):n,]  
 id outlook temperature humidity windy play  
10 10 rainy 75 80 FALSE yes  
11 11 sunny 75 70 TRUE yes  
12 12 overcast 72 90 TRUE yes  
13 13 overcast 81 75 FALSE yes  
14 14 rainy 71 91 TRUE no  
> tail(data,5)  
 id outlook temperature humidity windy play  
10 10 rainy 75 80 FALSE yes  
11 11 sunny 75 70 TRUE yes  
12 12 overcast 72 90 TRUE yes  
13 13 overcast 81 75 FALSE yes  
14 14 rainy 71 91 TRUE no  
Second attribute of first 10 rows  
> data[1:10,2]  
 [1] sunny sunny overcast rainy rainy rainy overcast sunny sunny   
[10] rainy   
Levels: overcast rainy sunny  
Distribution of every dimension  
> summary(data)  
 id outlook temperature humidity windy play   
 Min. : 1.00 overcast:4 Min. :64.00 Min. :65.00 Mode :logical no :5   
 1st Qu.: 4.25 rainy :5 1st Qu.:69.25 1st Qu.:71.25 FALSE:8 yes:9   
 Median : 7.50 sunny :5 Median :72.00 Median :82.50 TRUE :6   
 Mean : 7.50 Mean :73.57 Mean :81.64   
 3rd Qu.:10.75 3rd Qu.:78.75 3rd Qu.:90.00   
 Max. :14.00 Max. :85.00 Max. :96.00   
 Frequency of each class   
> table(data$id)  
  
 1 2 3 4 5 6 7 8 9 10 11 12 13 14   
 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
> table(data$outlook)  
  
overcast rainy sunny   
 4 5 5   
> table(data$temperature)  
  
64 65 68 69 70 71 72 75 80 81 83 85   
 1 1 1 1 1 1 2 2 1 1 1 1   
> table(data$humidity)  
  
65 70 75 80 85 86 90 91 95 96   
 1 3 1 2 1 1 2 1 1 1   
> table(data$windy)  
  
FALSE TRUE   
 8 6   
> table(data$play)  
  
 no yes   
 5 9   
Pie chart  
> pie(data1, col=c('red','blue'), labels= NULL, main="Windy piechart")  
> legend("topright",c('True','False'),fill = c('blue', red''))  
Variance  
> var(data$temperature)  
[1] 43.18681  
> var(data$humidity)  
[1] 105.7857  
Co-Variance  
> cov(data$temparature,data$humidity)  
Error: is.numeric(x) || is.logical(x) is not TRUE  
> cov(data$temperature,data$humidity)  
[1] 21.2967  
Correlation  
> cor(data$temperature,data$humidity)  
[1] 0.3150818  
Histogram  
hist(data$humidity,main="Temperature",xlab="X",ylab="Y",col=c("red","green","blue"))  
> hist(data$temperature,main="Temperature",xlab="X",ylab="Y",col=c("red","green"))  
Density  
> plot(density(data$temperature))  
> plot(density(data$humidity))  
Scatter plot  
> plot(data$temperature,data$humidity, main="Scatter plot", xlab = "Temperature",ylab = "Humidity",pch=19)  
Pairs plot  
> pairs(data)  
Box Whisker plot  
> boxplot(data$temperature,main="Box Whisker plot", xlab="Temperature", horizontal = TRUE)  
Line charts  
> plot(data$temperature,main="Numeric line chart", xlab = "Index", ylab = "Temperature", type= "o")  
> plot(data$outlook,main="Cateygorical line chart", xlab = "Index", ylab = "Outlook", type= "l")  
Cleveland dot chart  
> dotchart(data$temperature,color = "red", pch=19, main='Cleveland Dot Chart', xlab="Temperature")  
Barplot  
> barplot(data$temperature, main="Barplot", xlab = "Index", ylab = "Temperature", col = "blue", border = "red"**)**

**RANDOM FOREST**

ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))

train.data <- iris[ind==1,]

test.data <- iris[ind==2,]

install.packages("randomForest")

library(randomForest)

rf <- randomForest(Species ~ ., data=train.data, ntree=100, proximity=T)

table(predict(rf), train.data$Species)

print(rf)

plot(rf, main = "randomForest")

importance(rf)

irisPred <- predict(rf, newdata = test.data)

table(irisPred, test.data$Species)

varImpPlot(rf)

plot(margin(rf, test.data$Species))

**WORD CLOUD**

install.packages('Scale', dependencies=TRUE, repos='<http://cran.rstudio.com/>')

install.packages('tm', dependencies=TRUE, repos='<http://cran.rstudio.com/>')

options(repos='<http://cran.rstudio.com/>')

install.packages("SnowballC")

install.packages("wordcloud")

install.packages("RColorBrewer")

install.packages("stringr")install.packages("twitteR")

install.packages("Scale")

library(Scale)

library(tm)

library(SnowballC)

library(wordcloud)

library(RColorBrewer)

library(stringr)

library(twitteR)

url <- "<http://www.rdatamining.com/data/rdmTweets-201306.RData>"

download.file(url, destfile = "C:\\Users\\Admin\\Downloads\\rdmTweets-201306.RData")

load("C:\\Users\\Admin\\Downloads\\rdmTweets-201306.RData")

## load tweets into R

load(file = "C:\\Users\\Admin\\Downloads\\rdmTweets-201306.RData")

text <- readLines("C:\\Users\\Admin\\Downloads\\rdmTweets-201306.RData")

text

docs <- Corpus(VectorSource(text))

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

docs <- tm\_map(docs, trans, "/")

docs <- tm\_map(docs, trans, "@")

docs <- tm\_map(docs, trans, "\\|")

docs <- tm\_map(docs, content\_transformer(tolower))

docs <- tm\_map(docs, removeNumbers)

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

docs <- tm\_map(docs, removeWords, c("sudharsan","friendName"))

docs <- tm\_map(docs, removePunctuation)

docs <- tm\_map(docs, stripWhitespace)

docs <- tm\_map(docs, stemDocument)

dtm <- TermDocumentMatrix(docs)

mat <- as.matrix(dtm)

v <- sort(rowSums(mat),decreasing=TRUE)

data <- data.frame(word = names(v),freq=v)

head(data, 10)

set.seed(1056)

wordcloud(words = data$word, freq = data$freq, min.freq = 1,

max.words=200, random.order=FALSE, rot.per=0.35,

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

**TWITTER DATA ANALYSIS**

# Install Requried Packages  
installed.packages("SnowballC")  
installed.packages("tm")  
installed.packages("twitteR")  
installed.packages("syuzhet")  
  
# Load Requried Packages  
library("SnowballC")  
library("tm")  
library("twitteR")  
library("syuzhet")  
  
  
# Authonitical keys  
consumer\_key <- 'lxUjweJ9KUUEZf7GFtE9XIvN7'  
consumer\_secret <- 'EbTdnIWAx20o5K0O3xFmOg0CdVgaI4oSsaFpynAireAHDQ9W8h'  
access\_token <- '938562404689911808-92Fkm6UgXEJxmXDCOi24Zr681G8DQZM'  
access\_secret <- 'V9Ri2kFlEiEoylQfJ6aUPLFiUW70S5VOlq0FT6vWQF2Xh'  
  
#setting up connection for twitter api   
setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)  
tweets <- userTimeline("realDonaldTrump", n=100)  
tweets.df <- twListToDF(tweets)   
  
#removing url in tweets as well as @   
tweets.df2 <- gsub("http.\*","",tweets.df$text)  
tweets.df2 <- gsub("https.\*","",tweets.df2)  
tweets.df2 <- gsub("#.\*","",tweets.df2)  
tweets.df2 <- gsub("@.\*","",tweets.df2)  
head(tweets.df2)  
  
  
word.df <- as.vector(tweets.df2)  
emotion.df <- get\_nrc\_sentiment(word.df)  
emotion.df2 <- cbind(tweets.df2, emotion.df)   
head(emotion.df2)  
  
  
sent.value <- get\_sentiment(word.df)  
most.positive <- word.df[sent.value == max(sent.value)]  
most.positive  
  
  
most.negative <- word.df[sent.value <= min(sent.value)]   
most.negative  
  
head(sent.value)  
  
#let us seperate positive tweets and negative tweets  
  
positive.tweets <- word.df[sent.value > 0]  
negative.tweets <- word.df[sent.value<0]  
neutral.tweets <- word.df[sent.value==0]  
  
head(positive.tweets)  
  
head(negative.tweets)  
  
head(neutral.tweets)

**KNN**

iris <- read.csv("./iris.csv")

head(iris)

set.seed(99) # required to reproduce the results

rnum<- sample(rep(1:150)) # randomly generate numbers from 1 to 150

rnum

iris<- iris[rnum,] #randomize "iris" dataset

head(iris)

normalize <- function(x){

return ( x - min(x) / max(x) - min(x))

}

iris.n <- as.data.frame(lapply((iris[,0:4]), normalize))

head(iris.n)

iris.train<- iris.n[1:130,]

iris.train.target<- iris[1:130,5]

iris.test<- iris.n[131:150,]

iris.test.target<- iris[131:150,5]

install.packages(class)

library(class)

result <- knn(train = iris.train,test = iris.test, cl=iris.train.target,k=16)

table(iris.test.target, result)

acc <- 100\*sum(iris.test.target == result) / NROW (iris.test.target)

acc

head(result)