Supervised VS Unsupervised theory :

Data mining methods are categorized as either Unsupervisedor Supervised:

Most supervised data mining methods apply the following methodology for building and evaluating a model.

1- First, the algorithm is provided with a training set of data,

which includes the preclassified values of the target variable in addition to the predictor variables. For example, if we are interested in classifying income bracket, based on age, gender, and occupation, our classification algorithm would need a large pool of records, containing complete (as complete as possible) information about every field, including the target field, income bracket. In other words, the records in the training set need to be preclassified.

2-However, the training set is necessarily incomplete; that is, it does not include the “new” or future data that the data modelers are really interested in classifying. Therefore, the algorithm needs to guard against “memorizing” the training set and blindly applying all patterns found in the training set to the future data.

3-Therefore, the next step in supervised data mining methodology is to examine how the provisional data mining model performs on a test set of data. In the test set, a holdout data set, the values of the target variable are hidden temporarily from the provisional model, which then performs classification according to the patterns and structure it learned from the training set. The efficacy of the classifications are then evaluated by comparing them against the true values of the target variable. The provisional data mining model is then adjusted to minimize the error rate on the test set.

4- The adjusted data mining model is then applied to a validation data set, another holdout data set, where the values of the target variable are again hidden temporarily from the model. The adjusted model is itself then adjusted, to minimize the error rate on the validation set.

Overfittting and uderfitting: pg no : 112

Usually, the accuracy of the provisional model is not as high on the test or validation

sets as it is on the training set, often because the provisional model is overfitting

on the training set. Overfitting results when the provisional model tries to account for

every possible trend or structure in the training set, even idiosyncratic ones

Increasing the complexity of the model in order to increase

the accuracy on the training set eventually and inevitably leads to a degradation in

the generalizability of the provisional model to the test and validation setsHowever, as the model complexity

increases, the validation set error rate soon begins to flatten out and increase because

the provisional model has memorized the training set rather than leaving room for

generalizing to unseen data. The point where the minimal error rate on the validation

set is encountered is the optimal level of model complexity, as indicated in Figure 5.2.

Complexity greater than this is considered to be overfitting; complexity less than this

is considered to be underfitting.

Unsupervised Methods–

A target variable is not specified–Instead, the algorithm searches for patterns and structure among the variables

–Clustering is the most common unsupervised method–For example, political consultants analyze voter clusters in congressional districts that may be responsive to their particular candidate

–Important variables such as gender, age, income, and race are input to the clustering algorithm

–Voter profiles for fund-raising and advertising are created

**KNN :**

**Theory :**

k-Nearest Neighbor Algorithm–The k-Nearest Neighbor algorithm is an example of instance-based learningwhere training set records are first stored–Next, the classification of a new unclassifiedrecord is performed by comparing it to records in the training set it is most similar to–k-Nearest Neighbor is used most often for classification, although it is also applicable to estimation and prediction tasks

Example:

Which drug should Patient 1 be prescribed?–Since Patient 1’s profile places them in the scatter plot near patients prescribed drug Y, we classify Patient 1 as drug Y–All points near Patient 1 are prescribed drug Y, making this a straightforward classification

**Considerations when using k-Nearest Neighbor–**

How many neighbors should be used? k= ?

–How is the distance between points measured?

–How is the information from two or more neighbors combined when making a classification decision?

–Should all points be weighted equally, or should some points have more influence?

#purpose : Explain distance formula and calculate distance.

########################################################

#1 (10 Points)

Is the following function a proper distance function? Why? Explain

your answer. Measure the distance between (0, 0, 0), (0, 1, 0), (0, 1,

1), and (1, 1, 1)

d(x, y) = Ʃ (|x!" $!| %)

ANSWER :

Yes It is a proper distance function. This can be proven with given

below explanation.

Data analysts define distance metrics to measure similarity. A

distance metric

or distance function is a real-valued function d, such that for any

coordinates x, y,

and z:

1. d(x, y) ≥ 0, and d(x, y) = 0 if and only if x = y

2. d(x, y) = d(y, x)

3. d(x, z) ≤ d(x, y) + d(y, z)

Property 1: assures us that distance is always nonnegative, and the

only way for distance to be zero is for the coordinates (e.g., in the

scatter plot) to be the same.

If both points are same then only given distance formula compute

result as zero.

For eg : say point A = (0,0) and point B = (0,0)

To calculate distance between A and B using given distance formula:

D(A, B) = (x1 - x2) ^2 + (y1 – y2) ^ 2

= (0 - 0) ^2 + (0 - 0) ^ 2

= 0 + 0 = 0

Hence given distance formula d(x, y) = Ʃ (|x!" $!| %) satisfy

property 1.

Property 2: indicates commutativity. In our example distance

between x to y or y to x yields same result because at end their

difference will be squared.

While calculating distance using given formula, distance can never

be negative because we compute square of differences.

For eg : say point A = (0,0) and point B = (1,1)

To calculate distance between A and B using given distance formula:

D(A, B) = (x1 - x2) ^2 + (y1 – y2) ^ 2

= (0 - 1) ^2 + (0 - 1) ^ 2

= 1 + 1 = 2

Hence distance between two point is positive

Similarly while calculating distance between B and A using given

distance formula , result be will same as above :

D(B, A) = (x1 - x2) ^2 + (y1 – y2) ^ 2

= (1 - 0) ^2 + (1 - 0) ^ 2

= 1 + 1 = 2

Hence distance between two points are same.

Hence given distance formula d(x, y) = Ʃ (|x!" $!| %) satisfy

property 2.

Property 3 is the triangle inequality, which states that introducing a

third point can never shorten the distance between two other

points.

In our case, if new point introduce between A and B say C with

points A = (0,0) B = (1,1) and C = (0.5,0.5)

Then distance between A and C

= (0-0.5)^2 + (0 – 0.5) ^ 2 = 0.5

Distance between C and B

= (1 – 0.5) ^ 2 + (1 – 0.5) ^ 2 = 0.5

Therefore distance between A and B = 0.5 + 0.5 = 1

From property 2 we get distance between A and B is 1

Hence given distance formula d(x, y) = Ʃ (|x!" $!| %) satisfy

property 3.

Therefore given formula is valid distance formula as it satisfy

distance property 1 , 2 and 3.

Let say point A = (0, 0, 0), B= (0, 1, 0), C= (0, 1, 1), and D= (1, 1, 1).

So calculating distance between them using Euclidean distance

formula :

dEuclidean(x, y) = sqrt (Σ (xi − yi) ^ 2)

a) Distance between A(0,0,0) and B(0,1,0)

= Sqrt( (x1- x2) ^ 2 + (y1 – y2) ^ 2 + (z1 – z2) ^ 2)

= sqrt( (0-0)^2 + (0-1)^2 + (0-0)^2)

= sqrt( 0 + 1 + 0)

= 1.

b) Distance between B(0,1,0) and A(0,0,0)

According to distance property 2 , D(A, B) = D(B, A)

Therefore D(B, A) = 1

c) Distance between A(0,0,0) and C(0,1,1)

= Sqrt( (x1- x2) ^ 2 + (y1 – y2) ^ 2 + (z1 – z2) ^ 2)

= sqrt( (0-0)^2 + (0-1)^2 + (0-1)^2)

= sqrt( 0 + 1 + 1)

= 1.41.

d) Distance between C(0,1,1)and A(0,0,0)

According to distance property 2 , D(A, C) = D(C, A)

Therefore D(C, A) = 1.41

e) Distance between A(0,0,0) and D(1,1,1)

= Sqrt( (x1- x2) ^ 2 + (y1 – y2) ^ 2 + (z1 – z2) ^ 2)

= sqrt( (0-1)^2 + (0-1)^2 + (0-1)^2)

= sqrt( 1 + 1 + 1)

= 1.73.

f) Distance between D(1,1,1)and A(0,0,0)

According to distance property 2 , D(A, D) = D(D, A)

Therefore D(D, A) = 1.73

g) Distance between B(0,1,0) and C(0,1,1)

= Sqrt( (x1- x2) ^ 2 + (y1 – y2) ^ 2 + (z1 – z2) ^ 2)

= sqrt( (0-0)^2 + (1-1)^2 + (0-1)^2)

= sqrt( 0 + 0 + 1)

= 1.

h) Distance between C(0,1,1)and B(0,1,0)

According to distance property 2 , D(C, B) = D(B, C)

Therefore D(C, B) = 1

i) Distance between B(0,1,0) and D(1,1,1)

= Sqrt( (x1- x2) ^ 2 + (y1 – y2) ^ 2 + (z1 – z2) ^ 2)

= sqrt( (0-1)^2 + (1-1)^2 + (0-1)^2)

= sqrt( 1 + 0 + 1)

= 1.41.

j) Distance between D(1,1,1)and B(0,1,0)

According to distance property 2 , D(D, B) = D(B, D)

Therefore D(D, B) = 1.41

k) Distance between C(0,1,1) and D(1,1,1)

= Sqrt( (x1- x2) ^ 2 + (y1 – y2) ^ 2 + (z1 – z2) ^ 2)

= sqrt( (0-1)^2 + (1-1)^2 + (1-1)^2)

= sqrt( 1 + 0 + 0)

= 1.

l) Distance between D(1,1,1)and C(0,1,1)

According to distance property 2 , D(D, C) = D(C, D)

Therefore D(D, C) = 1.

Read file:

file<-filename<-file.choose()

bc<- read.csv(file, na.strings = "?" )

I.Summarizing each column (e.g. min, max, mean )

II.Identifying missing values

#summary

summary(bc)

#check missing value

is.na(bc)

#identify all missing values

missing<-bc[is.na(bc$F6),]

OR

na\_values\_covidData <- is.na(covidData)

View(na\_values\_covidData)

III.Displaying the frequency table of “Infected” vs. “MaritalStatus”

table(Infected = covidData$Infected, Martial\_status = covidData$MaritalStatus)

IV.Displaying the scatter plot of “Age”, “MaritalStatus” and “MonthAtHospital”, one pair at a time

pairs(bc[c(2:5,11)], main = "Breast Cancer Graph",

pch = 21, bg = c("red", "green")[factor(bc$Class)])

plot(covidData$Age , covidData$MaritalStatus , main = 'Age vs MaritalStatus', xlab = "Age", ylab = "MaritalStatus", pch = 21, bg = c("red", "green3"))

plot(covidData$Age , covidData$MonthAtHospital, main = 'Age vs MonthAtHospital', xlab = "Age", ylab = "MonthAtHospital", pch = 21, bg = c("red", "green3"))

plot(covidData$MaritalStatus, covidData$MonthAtHospital, main = 'MonthAtHospital vs MaritalStatus', xlab = "MaritalStatus", ylab = "MonthAtHospital", pch = 21, bg = c("red", "green3"))

V.Show box plots for columns: “Age”, “MaritalStatus” and “MonthAtHospital”

boxplot(bc[2:5])

boxplot(bc[6:9])

OR

boxplot(covidData$Age, covidData$MaritalStatus , covidData$MonthAtHospital)

histogram :

for(i in 2:3){

hist(bc[[i]],main=paste("Breast Cancer column= ", clnms[i]))

}

VI.Replacing the missing values of “Cases” with the “mean” of “Cases”.

covidData[is.na(covidData$Cases) , 5] <- mean(covidData$Cases, na.rm = TRUE)

View(covidData)

With mode :

mfv <- function(x) {

unique.x <- unique(x)

tab<-tabulate(match(x, unique.x))

unique.x[tab == max(tab)]

}

mfv\_value<-mfv(bc$F6)

bc[is.na(bc$F6),"F6"]<-mean(bc$F6,na.rm=TRUE)

bc[is.na(bc$F6),"F6"]<-mfv\_value[1]

summary(bc)

**KNN :**

With distance formula:

Given :

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | output |
| 100 | 200 | 300 | yes |

Firstly normalize data:

(Data – minimum ) /(maximum - minimum)

If k = 3 then classify 2,3,4

Then = Sqrt ((100-2)^2 + (200 - 3)^2 + (300 - 4)^2)

Then check lowest distance and output

If output matches, right calssify else wrong classify.

If 2 yes and 1 no then yes(Majority wins)

If same randomly choosen.

R Program for KNN:

#install kknn

#install.packages("kknn")

library(kknn)

#load data

fileName <- file.choose()

termination <- read.csv(fileName)

#View

View(termination)

nrow(termination)

# Index choosen for testing n training

index <- sample(nrow (termination), as.integer(.70 \* nrow(termination)))

training<-termination[index,c(2,3,21)]

nrow(training)

test<- termination[-index,c(2,3,21)]

nrow(test)

#factor

termination <- as.data.frame(sapply(termination, as.numeric))

#KNN formula

predict\_k <- kknn(formula=STATUS~., training, test[,-19], k=5, kernel ="rectangular")

fit <- fitted(predict\_k)

output\_k <- table(Actual=test$STATUS, Fitted=fit)

wrong<- ( test$STATUS != fit)

rate\_k<-sum(wrong)/length(wrong)

**Naïve bayes:** <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>

If question asked abt solving can use link and for theory purpose

**Conditional Probability:**

P(A/B) = P(A ^ B) / P(B)

But Bayes and Naïve Bayes=

P(s / 1st) = (P(1st / S) \* P(S)) / P(1st)

P(~s / 1st) = (P(1st / ~s) \* P(~s)) / P(1st)

Covid\_nomissing<-na.omit(Covid\_raw)

#Discretize the "MonthAtHospital" into "less than 6 months"and "6 or more months".

MH\_cat<- ifelse(Covid\_nomissing$MonthAtHospital<6,"<6",">=6")

#Also discretize the age into "less than 35", "35 to 50" and "51 or over".

age\_cat<- ifelse(Covid\_nomissing$Age<35,"L35",ifelse(Covid\_nomissing$Age>50,"M50","B35\_50"))

#clean data Convert to factor only if not number and other like string or boolean or character

Covid\_clean<-data.frame(

cbind(age\_cat=factor(age\_cat)

,Exposure=factor(Covid\_nomissing$Exposure)

,MaritalStatus=factor(Covid\_nomissing$MaritalStatus)

,Cases=Covid\_nomissing$Cases

,MH\_cat=factor(MH\_cat)

,Infected= factor(Covid\_nomissing$Infected)

)

)

#create test and training

index<-sort(sample(nrow( Covid\_clean),round(.30\*nrow(Covid\_clean ))))

training<- Covid\_clean[-index,]

test<- Covid\_clean[index,]

library(e1071)

# naïve bayes algo

nBayes <- naiveBayes( factor(Infected)~., data =training )

## Naive Bayes prediction using all variables

category\_all<-predict(nBayes,test )

#confusion matrix and error

table(NBayes=category\_all,Survived=test$Infected )

NB\_wrong<-sum(category\_all!=test$Infected )

NB\_error\_rate<-NB\_wrong/length(category\_all)

NB\_error\_rate

**Decision tree:**

Theory:

The target variable : good or bad credit risks

The predictor variables : are savings (low, medium, and high), assets (low or not low), and income (≤$50,000 or >$50,000).

Problem solving:

1- split into two : PL : needed , PR : not needed

2- PL : good PL , bad PL

3- PR : good PR , bad PR

4- 2 \* PL \* PR

5- (good PL – good PR) + (bad PL – bad PR)

6- 4 \* 5 i.e : 2 \* PL \* PR \* (good PL – good PR) + (bad PL – bad PR)

A screenshot of text

Description automatically generated

A screenshot of a cell phone

Description automatically generated

R program CART :

rm(list=ls())

#read

dev.off()

file<-file.choose()

Covid\_raw<- read.csv(file)

summary(Covid\_raw)

Covid\_nomissing<-na.omit(Covid\_raw)

#Discretize the "MonthAtHospital" into "less than 6 months"and "6 or more months".

MH\_cat<- ifelse(Covid\_nomissing$MonthAtHospital<6,"<6",">=6")

#Also discretize the age into "less than 35", "35 to 50" and "51 or over".

age\_cat<- ifelse(Covid\_nomissing$Age<35,"L35",ifelse(Covid\_nomissing$Age>50,"M50","B35\_50"))

#ID Age Exposure MaritalStatus Cases MonthAtHospital Infected

Covid\_clean<-data.frame(

cbind(age\_cat=factor(age\_cat)

,Exposure=factor(Covid\_nomissing$Exposure)

,MaritalStatus=factor(Covid\_nomissing$MaritalStatus)

,Cases=Covid\_nomissing$Cases

,MH\_cat=factor(MH\_cat)

,Infected= factor(Covid\_nomissing$Infected)

)

)

#testing and training data separate

index<-sort(sample(nrow( Covid\_clean),round(.30\*nrow(Covid\_clean ))))

training<- Covid\_clean[-index,]

test<- Covid\_clean[index,]

#develop plot library

library(rpart)

library(rpart.plot) # Enhanced tree plots

library(rattle) # Fancy tree plot

library(RColorBrewer) # colors needed for rattle

#perform Rpart algo

CART\_class<-rpart( factor(Infected) ~.,data=training )

#plot Rpart

rpart.plot(CART\_class)

dev.off()

#predict

CART\_predict<-predict(CART\_class,test,,type="class")

df<-as.data.frame(cbind(test,CART\_predict))

#create table and find error rate

table(Actual=test[,"Infected"],CART=CART\_predict)

CART\_wrong<-sum(test[,"Infected"]!=CART\_predict)

error\_rate=CART\_wrong/length(test$Infected )

Entrophy: Refer Excel C4.5

In engineering applications, information is analogous to signal, and entropy is

analogous to noise, so it makes sense that the entropy for medium savings is zero, since

the signal is crystal clear and there is no noise: If the customer has medium savings,

he or she is a good credit risk, with 100% confidence. The amount of information

required to transmit the credit rating of these customers is zero, as long as we know

that they have medium savings.

1- GET DATA

STEP 2: TARGET ATTRIBUTE (SALARY)

2- OUTPUT :

LEVEL 1 : TOTAL IN LEVEL 1 - A

LEVEL 2 : TOTAL IN LEVEL 2 AND SO ON - B

3- PJ IS CALCULATED AS : A / TOTAL , B/ TOTAL ETC.

4- -LOG2(PJ) : LOG(A / TOTAL ,2)

5- 3 \* 4

STEP 3: INTERMEDIATE ATTRIBUTE (LIKE OCCU, AGE) Candidate split:

Level 1 and same for level 2,3,4

Level 1 2 3 4 total -PJ log2(Pj) -Pj \*log2(Pj)

Service 2 3 5 6 16 2/16 log(2/16) -2/16 \*log(2/16)

∑-pj\*log2(pj) = U = Then perform summation of level 1,2,3,4

H(T) = (total(level1) / total) \* (summation of level 1,2,3,4) or U

Entropy = - summation of(H(T)) : for INTERMEDIATE ATTRIBUTE(LIKE occ, Age)

Information gain = (5- 3 \* 4) - Entropy

Those having high entropy is best attribute for decision tree.

In short: pgno :137

Total : 8

Occupation 4 , service : 1 : good : 0 , bad 1, management : 2 : good 1, Bad 1 : Other : 1

1-first find overall occupation : -4/8 \* log(4/8) = 0.56

2- service : -1/4 log(1/4) -0/4 log 0/4 = 0.4, management = 0.5 other = 0

3- add service + management +other = 0.3 – entropy

4- subtract occupation – 3 : 0.56 – 0.3 = 0.26 – information gain

**Random Forest methodology**

## categories are represented by the “factor” data

breastCancer$Class<-factor(breastCancer$Class, levels = c(2,4), labels = c("benign","malignant"))

breastCancer <- breastCancer

# get tarining and test data

index <- sort(sample(nrow(breastCancer),round(.30\*nrow(breastCancer))))

training <- breastCancer[-index,]

test <- breastCancer[index,]

#perform classification using randomForest.

library(randomForest)

result <- randomForest(Class~. , data = training , importance = TRUE, ntree = 1000)

#identify important features

importance(result)

varImpPlot(result)

#perform prediction

randomForest\_predict <- predict(result , test)

table(randomForest\_predict, test$Class)

#error rate for randomForest.

wrong <- (test[,11] != randomForest\_predict)

error\_rate= sum(wrong)/length(wrong)

error\_rate

**C5.0 methodology**

rm(list=ls())

filename <- file.choose()

breastCancer <- read.csv(filename )

View(breastCancer)

dev.off()

## categories are represented by the “factor” data

breastCancer$Class<-factor(breastCancer$Class, levels = c(2,4), labels = c("benign","malignant"))

breastCancer <- breastCancer

# get tarining and test data

index <- sort(sample(nrow(breastCancer),round(.30\*nrow(breastCancer))))

training <- breastCancer[-index,]

test <- breastCancer[index,]

#load library

library(C50)

#classification and prediction using C5.0 method

C5.0\_result <- C5.0(x = training[,2:10], y = training$Class)

summary(C5.0\_result)

C5.0\_predict <- predict(C5.0\_result , test[,2:10])

table(C5.0\_predict, test$Class)

#error rate for C5.0\_predict

wrong <- (test[,11] != C5.0\_predict)

error\_rate= sum(wrong)/length(wrong)

error\_rate

#classification and prediction using C5.0 method with 10 trials.

C5.0\_trail <- C5.0(x = training[,2:10], y = training$Class, trials = 10)

C5.0\_trail\_predict <- predict(C5.0\_trail , test[,2:10])

table(C5.0\_trail\_predict, test$Class)

wrong <- (test[,11] != C5.0\_trail\_predict)

error\_rate= sum(wrong)/length(wrong)

error\_rate

#classification and prediction using C5.0 method with rules option as true.

C5.0\_rule <- C5.0(x = training[,2:10], y = training$Class, rules = TRUE)

C5.0\_rule\_predict <- predict(C5.0\_rule , test[,2:10])

table(C5.0\_rule\_predict, test$Class)

wrong <- (test[,11] != C5.0\_rule\_predict)

error\_rate= sum(wrong)/length(wrong)

error\_rate

#classification and prediction using C5.0 method with 10 trials and rules option as true..

C5.0\_rule\_trial <- C5.0(x = training[,2:10], y = training$Class,trials = 10, rules = TRUE)

C5.0\_rule\_trial\_predict <- predict(C5.0\_rule\_trial , test[,2:10])

table(C5.0\_rule\_trial\_predict, test$Class)

wrong <- (test[,11] != C5.0\_rule\_trial\_predict)

error\_rate= sum(wrong)/length(wrong)

error\_rate

**Artificial Neural Networks (ANN)**

he weighted sum is converted into an output value using a mathematical function called transfer function,–transfer function normalizes the output into the range of [0,1],

= 1/ (1+ e ^ -x)

Working:

1- Calculate node \* weight for each hidden and output node.

2- If > 1 use = 1/ (1+ e ^ -x) ==(1/(1+EXP(-I22)))

3- for final output : small delta

Predictoutput \* (1- Predictoutput) \* (Actual – Predict output)

4- Adjust calculate other delta : Outer layer

Big delta= Small delta \* node value \* adjustment

Adjust weight /new weight = big delta + old weight (if predict > actual else minus)

5- Adjust calculate other delta : Hidden layer

Small delta of A =

Value of hidden node \*(1- Value of hidden node) \* summation of (Weight from hidden to output \* adjust weight of output)

Big delta= Small delta of A \* node value \* adjustment

Adjust weight /new weight = big delta + old weight (if predict > actual else minus)

0.85320966

A close up of a map

Description automatically generated

A close up of text on a white background

Description automatically generated

# purpose : Use the ANN methodology to develop a classification model for the Diagnosis.

######################################################################

rm(list=ls())

filename <- file.choose()

breastCancer <- read.csv(filename, na.strings = "?" )

View(breastCancer)

dev.off()

#To factor the data set

breastCancer<-data.frame(lapply(na.omit(breastCancer),as.numeric))

# get tarining and test data

index <- sort(sample(nrow(breastCancer),round(.30\*nrow(breastCancer))))

training <- breastCancer[-index,]

test <- breastCancer[index,]

#install.packages("neuralnet")

library(neuralnet)

#perform classification using ANN

result <- neuralnet(diagnosis~. , data = training[-1] , hidden = 5, threshold = 0.01)

plot(result)

#Compute for ANN

predict <- compute(result, test)

predict$net.result

#Display output in actual and ANN output format.

clean\_output <- cbind(test$diagnosis , as.data.frame(predict$net.result))

colnames(clean\_output) <- c("Actual\_diagnosis" , "Neural\_Net\_output")

print(clean\_output)

#display confusion matrix and error rate for ANN

predict\_range <- ifelse(predict$net.result <1.5,1,2)

table(predict\_range , test$diagnosis)

wrong <- (test$diagnosis != predict\_range)

error\_rate= sum(wrong)/length(wrong)

error\_rate

**HCLUST:**

**Working:**

1- Arrange ascending order

2- find normal distance between all points i.e P1- P2 or Euclidean if said

3- if smallest distance group them in one cluster and repeat process, until have only one cluster.

Centroid should be as far as possible

Using centroid:

1- Say centroid m1, m2

2- calculate eulidean distance sqrt( (m1-x1)^2) and sqrt( (m2-x1)^2

And whichever is less x1 will go into th at if m1< m2 , x1 will go m1

3- do for all points.

4- recalculate centroid, for all points in m1 i.e x1,x2,x3 for m1, m2

M1 = x1 + x2 + x3 /3 , m2 = x1+x2+x3 /3

5- Update centroid and repear from step 2

6- do until no change in cluster

# purpose : Using kmean to develop a Cluster model.

######################################################################

rm(list=ls())

filename <- file.choose()

breastCancer <- read.csv(filename, na.strings = "?" )

View(breastCancer)

dev.off()

#To factor the data set

breastCancer<-data.frame(lapply(na.omit(breastCancer),as.numeric))

#obtain two cluster for all feature except ID and Diagnosis

breastCancer\_clust\_kmean <- kmeans(breastCancer[,-1,-2], 2 , nstart = 10)

#plot cluster

breastCancer\_clust\_kmean$cluster

breastCancer\_clust\_kmean$centers

#tabulate clustered rows against the “diagnosis” column

table(breastCancer\_clust\_kmean$cluster , breastCancer$diagnosis)

# purpose : Using hclust to develop a Cluster model.

######################################################################

rm(list=ls())

filename <- file.choose()

breastCancer <- read.csv(filename, na.strings = "?" )

View(breastCancer)

dev.off()

#To factor the data set

breastCancer<-data.frame(lapply(na.omit(breastCancer),as.numeric))

#obtain distance between two points of all feature except ID and Diagnosis

breastCancer\_dist <- dist(breastCancer[,-1,-2])

#obtain cluster for all feature except ID and Diagnosis

breastCancer\_clust <- hclust(breastCancer\_dist)

#plot cluster

plot(breastCancer\_clust)

#obtain two cluster

breastCancer\_clust\_two <- cutree(breastCancer\_clust , 2)

#tabulate clustered rows against the “diagnosis” column

table(breastCancer\_clust\_two , breastCancer$diagnosis)