REGRESSION

Simple Linear Regression

1. Install the package **'lattice'**

2. Make the **scatter-Plot** by using function as **"plot(x,y)"** where x&y are the variables

3. Find the **co-releation coefficient** between the variables **"cor(x,y) or cor(filename)"**

4. Now to build Linear Regression as **"temp <- lm(o/p ~ i/p ,data=filename)"** and later find **summary(temp)**

5. Find the Prediction at confidence intervals as show in code

**Source Code**

wc.at <- read.csv(“file.choose()”)

install.packages("lattice")

library("lattice")

??lattice

View(wc\_at)

summary(wc\_at)

dotchart(wc\_at$Waist)

dotchart(wc\_at$Waist, main="Dot Plot of Waist Circumferences")

plot(Waist,AT)

dotchart(wc\_at$AT, main="Dot Plot of Adipose Tissue Areas")

boxplot(wc\_at$Waist,col="dodgerblue4")

boxplot(wc\_at$AT,col="red", horizontal = T)

attach(wc\_at)

plot(Waist,AT,main = "scatter plot for SLR")

?plot

attach(wc\_at)

# Correlation coefficient

cor(Waist, AT)

#Build Linear regression

reg <- lm(AT~Waist, data=wc\_at)

summary(reg)

confint(reg,level=0.95)

predict(reg,interval="predict")

reg\_log <- lm(AT ~ log(Waist))

summary(reg\_log)

confint(reg\_log, level=0.95)

predict(reg\_log, interval="predict")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Multiple Linear Regression

1. Initially find the output variable and input variable in the given dataset

2. Then find the scatter-Plot by using function as **"plot(x,y) or pairs(filename)"**

3. Later find the **co-releation coefficient matrix** by using **"cor(filename) or cor(x,y)"**

4. Now compare the releation between the o/p to i/p variables and i/p to i/p variables which the releation must be greater than **0.8 or 0.85**

5. Now to build the Multi Linear Regression Model as **"temp <- (o/p ~ i/p1+ i/p2+ i/p3+.......+i/pn)"** and find **summary(temp)**

6. check the significance between the input variables which having with the output variables in which the releation is less than **0.8/0.85** **"temp <- (o/p ~ i/p+i/p+......)"**

7. Now it is Better to delete influential observations rather than deleting entire column as **"influence.measures(filename)"**

8. We then delete the influential observations by **"temp<-lm(o/p ~ i/p1+ i/p2+ i/p3+.......+i/pn,data=filename[-c(70,71..)])"** continue the process until the influence observations are removed from the data

9. Variance Inflation factor to check collinearity b/n variables as "vif(filename)" and **vif>10** then there exists collinearity among all the variables

10. Finally build the model as **"temp <- (o/p ~ i/p1+ i/p2+ i/p3+.......+i/pn)"**

**Source Code**

Cars <- read.csv(file.choose())

View(Cars)

attach(Cars)

cor(HP,VOL)

cor(HP,SP)

pairs(Cars)

cor(Cars)

install.packages("corpcor")

library(corpcor)

cor2pcor(cor(Cars))

model.car <- lm(MPG~VOL+HP+SP+WT)

summary(model.car)

model.carV<-lm(MPG~VOL)

summary(model.carV)

model.carW<-lm(MPG~WT)

summary(model.carW)

model.carVW<-lm(MPG~VOL+WT)

summary(model.carVW)

pairs(Cars,upper.panel = panel.cor,main="Scatter plot matrix with Correlation coefficients")

influence.measures(model.car)

library(car)

influenceIndexPlot(model.car,id.n=5)

influencePlot(model.car,id.n=5)

model.car1<-lm(MPG~VOL+SP+HP+WT,data=Cars[-77,])

summary(model.car1)

model.car2<-lm(MPG~VOL+SP+HP+WT,data=Cars[-c(71,77),])

summary(model.car2)

model.car3<-lm(MPG~VOL+SP+HP+WT,data=Cars[-c(71,77,79),])

summary(model.car3)

vif(model.car)

finalmodel<-lm(MPG~VOL+SP+HP)

summary(finalmodel)

finalmodel1 <- lm(MPG~SP+HP+WT)

summary(finalmodel1)

Logistic Regression

1.Find the Regression for the data set by using syntax as **“temp <- glm(o/p ~factor(i/p)+ factor(i/p)+ factor(i/p)……..+i/p+i/p,family=binary/ordinal/nominal,data=filename)”**where ‘factor’ is term used for the binary columns or variables

2.Find the odd Ratio as **“exp(coef(temp))”**

3.Make Predection as **"prob <- predict(temp, original-filename)"**

4.Confusion Matrix Table as **"confusion <- table(prob,orginalfilename$o/p variable)"**

5.Find the Accuracy as **"Accuracy <-sum(diag(confusion)/sum(confusion))"**

**Sourece Code**

claimants <- read.csv(file.choose())

View(claimants)

attach(claimants)

fit<-lm(ATTORNEY~factor(CLMSEX)+factor(CLMINSUR)+factor(SEATBELT)+CLMAGE+LOSS)

summary(fit)

logit<- glm(ATTORNEY~factor(CLMSEX)+factor(CLMINSUR)+factor(SEATBELT)+CLMAGE+LOSS,family=binomial,data = claimants)

summary(logit)

exp(coef(logit))

prob <- predict(logit,claimants)

prob

confusion<-table(prob>0.5,claimants)

confusion

Accuracy<-sum(diag(confusion)/sum(confusion))

Accuracy

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

UNSUPERVISED-LEARNING

H-CLUSTERING

1. Remove the useless coloumn’s(like S.NO,S.ID,) and install the package **'readxl'**

2. Normolized the data using scalling function **" temp <- scale(filename)"**

3. Finding the distance with **euclidean** method by using of function **"temp1 <- dist(temp ,method="euclidean")"**

4. Grouping/Clustering the data using simple/complete method as **"temp2 <-hclust(temp1,method="simple/complete")"**

5. Plot the clusters using **Dendogram** as **"plot(temp2,hang=-1)"**

6. Now to decide how many clusters so that cut the **dendogram** tree we need as k=2/3/4/.......... **" temp 3<- cutree(temp2,k=2/3/4.....)"**

7. After cutting the tree we can view the clusters in console tab as 1,1,1,2,2,1,3,2,1,3,2,1... **" temp4 <- as.matrix(temp3)"**

8. Later combine the series of data to the intital-dataset as **" temp5 <- data.frame(old\_filename,temp4)"**

9. Finally save the dataset in the desktop by installing the package **'xlsx'** then using **"write.xlsx(temp5 ,file=”varun.xlsx”)"**

10. we can observe the dataset with clusters in XL-SHEET.

**Source-Code**

library(readxl)

input <- read\_excel("G:\\Hierarchical Clustering\\University\_Clustering.xlsx")

View(input)

attach(input)

mydata<-input[,c(3,4:8)]

View(mydata)

normalized\_data<-scale(mydata)

View(normalized\_data)

d<-dist(normalized\_data,method="euclidean")

summary(d)

d

?hclust

fit<-hclust(d,method="complete")

fit

?hclust

plot(fit)

plot(fit,hang=-1)

groups<-cutree(fit,k=3)

groups

rect.hclust(fit,k=3,border="red")

membership<-as.matrix(groups)

View(membership)

final1<-data.frame(input,membership)

View(final1)

library(xlsx)

write.xlsx(final1,file="jan1st.xlsx")

getwd()

­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

KMeans-Clustering

1. Remove the useless coloumns and install the packages **'readxl, plyr, and animation'**

2. Normolized the data using scalling function **" temp <- scale(filename)"**

3. When compared to the H-Clustering it was different why because here we need to decide initially how many Cluster’s we required so make the clusters as **" temp1 <- kmeans(temp,4)"** where 4 is the clusters

4. Now compared the clusters by using the concept of the centroid of the data variables so install the package **‘animation’** which is similar as **" temp2 <- kmeans.ani(temp,4)"** where 4 is the clusters untill it is equal to the orginal cluster if it was not equal the animation is rotates untill it was equal

5. Later combine the series of data to the intital-dataset as **" temp3 <- data.frame(old\_filename ,temp1)"**

6. Finally save the dataset in the desktop by installing the package **'xlsx'** then using **"write.xlsx(temp3 ,file=”varun.xlsx”)"**

7. we can observe the dataset with clusters in XL-SHEET.

**Source Code**

install.packages("plyr")

library(plyr)

View(input)

mydata<-input[1:25,c(2,3:8)]

View(mydata)

View(normalized\_data)

km <- kmeans(normalized\_data,4)

str(km)

install.packages("animation")

library(animation)

km <- kmeans.ani(normalized\_data, 4)

km$centers

final2<- data.frame(mydata, km$cluster) # append cluster membership

final2

final3 <- final2[,c(ncol(final2),1:(ncol(final2)-1))]

final3

library(xlsx)

write.xlsx(final1,file="final1.xlsx")

getwd()

­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Principal Component Analysis**

1. Initially find the Principal Component Analysis for the given data set as

**"temp <-princomp(filename)"**

2. Later use the function **"loadings(temp)"** that it will display the best columns for the Clustering analysis

3. Now bind the best coloumns as **"temp1 <- cbind(orginalfile,temp)"**

4. finally use the H-Clustering or K-Means clustering

**Source Code**

mydata<-read\_csv("E:/Confidential/R Codes/PCA/Universities.csv")

View(mydata)

help(princomp)

data <- mydata[-1]

attach(data)

cor(data)

pcaObj<-princomp(mydata[-1], cor = TRUE, scores = TRUE, covmat = NULL)

summary(pcaObj)

loadings(pcaObj)

plot(pcaObj)

biplot(pcaObj)

pcaObj$scores[,1:3]

mydata<-cbind(mydata,pcaObj$scores[,1:3])

View(mydata)

**SUPERVISED-LEARNING**

**KNN-** **K Nearest Neighbors**

1. Initially if there are any English variables instead of numbers we need to replace them as **"filename$colname<-factor(filename$colname,levels = c("B", "M"),labels = c("Benign", "Malignant"))"**

2. Later to normilize the dataset using the 'scale' but scale cannot be used because coloumns are having different variables like binomial values and numeric values so we need the below method

**normalize <- function(x) {**

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

**}**

3. Simply Normilized the data by using the formulea as

**"temp<- as.data.frame(lapply(filename, normalize))"**

4. Now divide the data into training and testing data as

**train <- temp[1:469, ]**

**test <- temp[470:569, ]**

**train-labels <- filename[1:469, 1]**

**test-labels <- filename[470:569, 1]**

5. Later build the model as

**" pred <- knn(train = train, test = test, cl = train-labels, k=23)"** replace the k value for best model accuracy value and k must be always odd number

6.Now build the cross table as

**"CrossTable(x = test-labels, y = pred ,prop.chisq=FALSE)"**

Source-Code

View(wbcd)

wbcd <- wbcd[-1]

View(wbcd)

table(wbcd$diagnosis)

wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"),

labels = c("Benign", "Malignant"))

View(wbcd)

round(prop.table(table(wbcd$diagnosis)) \* 100, digits = 1)

View(wbcd)

summary(wbcd)

summary(wbcd[,c("radius\_mean", "area\_mean", "smoothness\_mean")])

str(wbcd)

normalize <- function(x) {

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

wbcd\_n <- as.data.frame(lapply(wbcd[2:31], normalize))

View(wbcd\_n)

summary(wbcd\_n)

wbcd\_train <- wbcd\_n[1:469, ]

View(wbcd\_test)

wbcd\_test <- wbcd\_n[470:569, ]

wbcd\_train\_labels <- wbcd[1:469, 1]

View(wbcd\_train\_labels)

wbcd\_test\_labels <- wbcd[470:569, 1]

library(class)

wbcd\_train\_labels <- wbcd\_train\_labels[["diagnosis"]]

wbcd\_test\_labels <- wbcd\_test\_labels[["diagnosis"]]

wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test,

cl = wbcd\_train\_labels, k=23)

library(gmodels)

CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred,

prop.chisq=FALSE)

wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test, cl = wbcd\_train\_labels, k=2)

CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq=FALSE)

wbcd\_test\_pred <- knn(train = wbcd\_train, test = wbcd\_test, cl = wbcd\_train\_labels, k=21)

CrossTable(x = wbcd\_test\_labels, y = wbcd\_test\_pred, prop.chisq=FALSE)

**SVM-Support Vector Machine**

1. Implement the All EDA (Exploatiory Data Analysis) on the given data set

2. Now build the model with Linear and Non-Linear Technique

**Temp-linear <- svm(Filename$o/p-variable ~ ., data=Filename ,kernel = "linear")**

**summary(final\_svm)**

3. Implement the Non Linear model as install the library(kenrlab)

**Temp-nonlinear <- ksvm(Filename$o/p-variable ~ ., data=Filename,kernel = "rbfdot/polydot/tanhdot/vanilladot/laplacedot/besseldot/anovadot/splinedot/matrix")**

**summary(svm\_linear)**

4. Find the prediction value and confusion matrix/table to find the better accuracy

**pred <- predict(svm-linear ,filename)**

**tab <-table(predicted = pred ,Actual = filename$o/pvariable)**

**sum(diag(tab))/sum(tab)**

5. Other wise divide the data set as training and testing data

Source Code

data()

data("iris")

View(iris)

attach(iris)

install.packages('e1071', dependencies=TRUE)

library(e1071)

final\_svm <- svm(Species ~ ., data=iris)

summary(final\_svm)

pred <- predict(final\_svm,iris)

tab <-table(predicted = pred,Actual = iris$Species)

sum(diag(tab))/sum(tab)

View(iris)

svm\_linear <- svm(Species ~ ., data=iris,kernel = "linear")

summary(svm\_linear)

pred <- predict(svm\_linear,iris)

tab <-table(predicted = pred,Actual = iris$Species)

sum(diag(tab))/sum(tab)

library(kernlab)

svm\_nonlinear <- ksvm(Species ~ ., data = iris, kernel = "rbfdot1")

prediction <- predict(svm\_nonlinear,iris)

table(prediction)

prop.table(table(prediction))

tab <-table(prediction = prediction,Actual = iris$Species)

tab

sum(diag(tab))/sum(tab)

**NB-Classifier**

1. Initiall apply the EDA for the given data set

2. If we need to plot that data as2.

**plot(as.factor(HouseVotes84[HouseVotes84$Class=="republican",2]))**

**plot(as.factor(HouseVotes84[HouseVotes84$Class=="democrat",2]))**

3. Divide the dataset into training and testing datasets

**set.seed(3)**

**train<-order(runif(100/300/400))**

**test<--train**

**View(train)**

**View(test)**

**# Divide the data into Training and Testing Data**

**training<-HouseVotes84[train,]**

**View(training)**

**testing<-HouseVotes84[test,]**

**View(testing)**

4. Now build the Naive Baye Model

**library(e1071)**

**model<-naiveBayes(training$Class~.,data=training)**

**model**

**# Predicting the model on Test data**

**pred<-predict(model,newdata = testing[,-1])**

**mean(pred==testing[,1])**

**Source Code**

library(mlbench)

data("HouseVotes84")

?? Housevotes84

View(HouseVotes84)

help("HouseVotes84")

summary(HouseVotes84)

barplot(table(as.factor(HouseVotes84[,1]),as.factor(HouseVotes84[,2])),legend=c("democrat","republic"))

plot(as.factor(HouseVotes84[HouseVotes84$Class=="republican",2]))

plot(as.factor(HouseVotes84[HouseVotes84$Class=="democrat",2]))

str(HouseVotes84)

set.seed(3)

train<-order(runif(290))

test<--train

View(train)

View(test)

training<-HouseVotes84[train,]

View(training)

testing<-HouseVotes84[test,]

View(training)

View(testing)

library(e1071)

model<-naiveBayes(training$Class~.,data=training)

model

pred<-predict(model,newdata = testing[,-1])

mean(pred==testing[,1])

­­­­­­­­­­­­­ **ASSOCIATION RULES**

1. Plot the graph as **"itemFrequencyPlot(Filename)"**

2. Simply apply the APRIORI algorithim as

**"temp<- apriori(filename,parameter=list(support=0.001, confidence = 1))"**

3. Visualzing the rules by using lift as

**"inspect(sort(temp, by="lift"))"**

4. Now save the csv file to the desktop as **"write(temp,file="varun123.csv")"** and use **getwd()**

**Source Code**

install.packages("arules")

library(arules)

install.packages("arulesViz") # extension to arules

library(arulesViz)

?arules

data()

data(Groceries)

View(Groceries)

summary(Groceries)

itemFrequencyPlot(Groceries)

itemFrequencyPlot(Groceries, topN = 10)

inspect(head(Groceries, 2))

rules\_fileName <- apriori(Groceries,parameter=list(support=0.001, confidence = 1))

rules\_fileName

inspect(head(sort(rules\_fileName, by = "confidence"), 3))

inspect(sort(rules\_fileName, by="lift"))

inspect(sort(rules\_fileName, by="lift")[1:10])

library(xlsx)

write(rules\_fileName,file="newrules.csv",sep=",",row.names = FALSE)

getwd()

install.packages("arulesViz")

plot(rules\_fileName)

plot(head(rules\_fileName))

head(quality(rules\_fileName))

plot((rules\_fileName), method="grouped")

Decision tree

1. Install the package **"C50 and tree"**

2. Apply the decision tree on cleaned data set as

**"temp <- C5.0(trainingset$output ~. ,data=trainingset)"**

3. Plot the graph as **"plot(temp)"**

4. Find the predection as **"pred <- predict(model,testing[,-1])"**

5. Finally find the accuracy as **"mean(pred == testing$output)**"

Source Code

install.packages("C50")

install.packages("tree")

library(C50)

data()

data("iris")

View(iris)

?iris

iris\_setosa<-iris[iris$Species=="setosa",] # 50

iris\_versicolor <- iris[iris$Species=="versicolor",] # 50

iris\_virginica <- iris[iris$Species=="virginica",] # 50

iris\_train <- rbind(iris\_setosa[1:25,],iris\_versicolor[1:25,],iris\_virginica[1:25,])

iris\_test <- rbind(iris\_setosa[26:50,],iris\_versicolor[26:50,],iris\_virginica[26:50,])

View(iris\_train)

View(iris\_test)

irisc5.0\_train <- C5.0(iris\_train[,-5],iris\_train$Species)

plot(irisc5.0\_train) # Tree graph

plot(irisc5.0\_train,type="simple")

mean(iris\_train$Species==predict(irisc5.0\_train,iris\_train)) # 97.33% Accuracy

predc5.0\_test <- predict(irisc5.0\_train,newdata=iris\_test) # predicting on test data

mean(predc5.0\_test==iris\_test$Species) # 94.66% accuracy

library(gmodels)

CrossTable(iris\_test$Species,predc5.0\_test)

**Random Forest**

1. Install the package of **"randomforest"**

2. Apply the randomforest for cleaned dataset as

**"temp <-randomforest(trainingset$output ~. ,data=trainingsetn,tree = 100)"**

3. Plot the graph as **"varImpPlot(model2)"** so that the graph will gives the important coloumns for the tree building

4. Find the predection as **"pred <- predict(model,testing[,-1])"**

5. Finally find the accuracy as **"mean(pred == testing$output)"**

Source Code

library(randomForest)

data(iris)

View(iris)

iris\_setosa<-iris[iris$Species=="setosa",] # 50

iris\_versicolor <- iris[iris$Species=="versicolor",] # 50

iris\_virginica <- iris[iris$Species=="virginica",] # 50

iris\_train <- rbind(iris\_setosa[1:25,],iris\_versicolor[1:25,],iris\_virginica[1:25,])

iris\_test <- rbind(iris\_setosa[26:50,],iris\_versicolor[26:50,],iris\_virginica[26:50,])

model1 <- randomForest(Species~.,data=iris\_train)

model1

model2 <- randomForest(Species~.,data=iris\_train, ntree = 100, importance = TRUE)

model2

importance(model2)

varImpPlot(model2)

plot(round(importance(fit.forest)))

?randomeforest

table(predict(model1),iris\_train$Species)

pred\_test <- predict(model1,newdata=iris\_test)

table(pred\_test,iris\_test$Species)

CM <- table(pred\_test,iris\_test$Species)

CM

accuracy <- (sum(diag(CM)))/sum(CM)

accuracy

pred\_test <- predict(model2,newdata=iris\_test)

table(pred\_test,iris\_test$Species)

CM <- table(pred\_test,iris\_test$Species)

CM

accuracy <- (sum(diag(CM)))/sum(CM)

accuracy