***Data Manipulation***

oj <- read.csv("C:/Users/Ankit/Downloads/Data set for R/oj.csv",header=TRUE)

View(oj)

names(oj)

str(oj)

head(oj) #top 6

tail(oj) #last 6

summary(oj)

oj$brand <- as.character(oj$brand)

oj$brand <- as.factor(oj$brand)

table(oj$week) #count per observation

oj[1,3] # data.frame[row , column]

#oj[ row , column ]

oj[c(1,12,38,256) ,c(1,3,6) ] #combination = c()

oj[1:5,c("brand","feat")] # BRAND!=brand

dat <- oj[oj$brand=="dominicks", ] #select (column)\* from oj where (row) brand = "dominiks"

summary(dat)

dim(dat)

table(dat$brand)

dat1 <- oj[oj$brand=="dominicks" | oj$brand=="tropicana", ]

# select \* from oj where brand in ("dominicks","tropicana")

head(dat1,4) #top 6 by default

table(dat1$brand)

dim(dat1)

dat2 <- oj[oj$brand=="tropicana" & oj$feat==1, ]

dim(dat2)

d1 <- oj[oj$brand == "dominicks" | oj$brand == "minute.maid" & oj$feat == 0,]

dim(d1)

d2 <- oj[(oj$brand == "dominicks" | oj$brand == "minute.maid") & oj$feat == 0,]

#(A or B) & c ->

dim(d2)

x <- 17:33

which(x > 26)

index <- which(oj$brand=="dominicks")

head(index)

dat2 <- oj[which(oj$brand=="dominicks") , ]

dat31 <- subset(oj,oj$brand=="crest")

dat31 <- subset(oj,oj$brand=="crest",select=c(logmove,INCOME)) #keep keyword in SAS

dat32 <- subset(oj,oj$brand=="crest",select=c(-week,-brand)) #drop keyword in SAS

dat4 <- oj[ , c("brand","feat")] # 2 columns

dat4.1 <- oj[,c(4:6)] # filtering of columns based on index of column

which(colnames(oj) =="feat" )

dat4.2 <- oj[,-c(3:5)] #drop of column numbers from 3 to 5 (3 4 5)

dat5 <- oj[oj$brand=="dominicks" & oj$feat==0,c("week","store")]

head(dat5)

# mysplit <- split(oj, oj$brand)

# summary(mysplit)

oj$logInc <- log(oj$INCOME)# creating a new variable

oj$flag <- ifelse(oj$brand=="tropicana", 1,0)

dim(oj)

table(oj$flag)

#sorting

#1 2 3 4 5

numbers <- c(8,100,-4,5,10)

order(numbers)

numbers[order(numbers,decreasing = TRUE)]

dat5.1 <- oj[order(oj$week,decreasing = FALSE), ]

numbers[order(numbers)]

# aggregate function in sql min, max, sum, avg, count, SD

agg <- aggregate(oj$price,

by=list(oj$brand),

FUN=mean)

# aggregate(numeric variable, by what?, by which function)

#select dept, avg(age) from file group by dept

agg

class(agg)

# family of apply function ...apply sapply lapply tapply

#tapply(numeric variable, on what basis(class), function)

tap <- tapply(oj$price,oj$brand, mean) #pivot equivalent in R

tap

tap1 <- tapply(oj$price, INDEX = list(oj$brand,oj$store), mean)

tap1

#Merging and Join

df1 = data.frame(CustomerId=c(1,2,3,4,5,6, 7,8,9,10,11,12), Product=c("Toaster","Radio","TV","Toaster","Radio","TV", "Toaster","Radio","TV","Toaster","Radio","TV"))

df2 = data.frame(CustomerId=c(1,9,3,18,5,6,7,8),

State=c("Alabama","Ohio","California",

"Nebraska","Alabama","Ohio",

"New Jersey","Iowa"))

merge(x = df1, y = df2, by = "CustomerId", all = TRUE) #full join

merge(x = df1, y = df2, by = "CustomerId", all.x=TRUE) #left join

merge(x = df1, y = df2, by = "CustomerId", all.y=TRUE) #right join

merge(x = df1, y = df2, by = "CustomerId") #inner join

#sqldf package

install.packages("sqldf")

library(sqldf)

sqldf("select \* from oj limit 10")

#dplyr package

install.packages("dplyr")

library(dplyr)

str(oj)

glimpse(oj) #dplyr

dat1 <- filter(oj,brand=="dominicks")

head(dat1)

dat2 <- filter(oj,brand=="dominicks" | brand=="tropicana")

dat2

#filter(dataset name, row condition, column condition)

dat3 <- filter(oj,brand=="tropicana",feat==0)

head(dat3)

dat4 <- filter(oj,brand=="minute.maid" & feat==1)

head(dat4)

# %in% operator

filter(oj,brand %in% c("minute.maid","tropicana"))

filter(oj,brand=="minute.maid"|brand=="tropicana")

filter(oj,between(ETHNIC, 0.1, 0.2)) # where ethnic between .1 and .2

filter(oj,ETHNIC >= 0.1 & ETHNIC <= 0.2)

ojnew <- filter(oj,!is.na(WORKWOM)) #is.na !is.na

slice(oj,1213:1218) #filtering of row oj[1213:1218,]

sample\_n(oj,12)

set.seed(133)

sample(1:10, 1)

sample\_frac(oj,0.1, replace=TRUE) #fraction of 10%

a <- top\_n(oj,5,week)

top\_n(oj,5,-logmove)

head(select(oj,brand,INCOME,feat)) #keep

head(select(oj,-brand,-INCOME,-feat)) #drop

head(select(oj,week:price))

head(select(oj,contains("SSTR"))) #filtering on the basis of contains or like

headselect(oj,starts\_with("E",ignore.case=TRUE))) #find(i)

select(oj,ends\_with("0",ignore.case = FALSE))

select(oj,matches("[cw]",ignore.case=TRUE)) #which contains c or w or both

head(select(oj,matches("[0-9]")))

colnames(oj)[2] <- "brand"

colnames(oj)

#rename(oj, new value = previous value)

dat1 <- rename(oj,zone=flag)

head(dat1)

dat12 <- mutate(oj,logIncome=log(INCOME,base = 10))

head(dat12)

glimpse(dat12)

mutate\_each(oj,funs(mean),

INCmean = INCOME,

AGE60mean = AGE60,

ETHNmean = ETHNIC)

head(mutate\_each(oj,funs(sqrt,log), INCOME, AGE60, ETHNIC))

head(transmute(oj,units=exp(logmove)))

#sorting of data dplyr package

dat13 <- arrange(oj,INCOME)

head(dat13)

dat13 <- arrange(oj,-INCOME)

head(dat13)

dat14 <- arrange(oj,desc(INCOME))

head(dat14)

dat14 <- arrange(oj,-INCOME, -week) #proc sql order by income desc

View(dat14)

#treating missing values

a <- c(1,2,3,4,5,6,NA,NA,NA,7,8,9)

is.na(a)

sum(is.na(a))

mean(a)

m <- mean(a,na.rm=TRUE)

a[is.na(a)] <- m

a

# Some basic string manipulation

a <- "Batman"

substr(a,start=2,stop=6) # same in SAS as well

length(b)

nchar(a)

length(a)

tolower(a)

toupper(a)

b <- "Bat-Man-man-cat"

strsplit(b,split="-")

c <- "Bat/Man"

strsplit(c,split="/")

paste(b,c) #concate equivalent in SAS CAT family cat cats catx

paste(b,c,sep="??")

names <- c("Anil","Paul","Ajit","Ram","John")

#searching for patterns

grep("a",names,ignore.case = TRUE)

#Returns the indices of all elements in names which contain the search string

#present or not

grepl("/",c)

grepl(b,c)

b <- "Bat-Man-man-cat"

sub("-","/",b)

#sub replaces the first occurrence of the search pattern with the replacement pattern

d <- "Bat-Ma-n"

sub("-","/",d) # only first occurrence

gsub("-","/",d) # all occurrences

#gsub replaces all occurrences

e <- " Ankit "

trimws(e,which = c("right"))

trimws(e, which = c("left"))

trimws(e, which = c("both"))

#dealing with dates

fd1 <- read.csv("C:/Users/44088714/Desktop/Machine learning and R training/flightsDelay.csv")

glimpse(fd1)

head(fd1$date ,10)

class(fd1$date)

fd1$FlightDate <- as.Date(fd1$date,"%d-%b-%y")

class(fd1$FlightDate)

glimpse(fd1)

str(fd1)

# Problems with using as.Date

# 1. We need to look up the input format specifiers - more a pain than a problem

# 2. If input format is inconsistent - e.g. if some rows have day, month and year delimited by hyphens and other rows have slashes, or if some rows specify month with 3 letters and others with 2 digits, etc.

# The package lubridate makes dealing with dates a lot easier

data()

library(lubridate)

fd1 <- read.csv("C:/Users/44088714/Desktop/Machine learning and R training/flightsDelay.csv")

glimpse(fd1)

# We only need to check whether the input format is

# day-month-year or

# month-day-year

# year-month-day

# Accordingly, we pick one of the three lubridatefunctions dmy, mdy or ymd

# In this case, we need ymd\_hms

glimpse(fd1)

str(fd1)

# 2017-11-04 12:08:08

fd1$FlightDate <- ymd\_hms(fd1$date) #(substr(fd1$date, 1,nchar(fd1$date)-9 ))

class(fd1$FlightDate)

months(fd1$FlightDate[1])

weekdays(fd1$FlightDate[1])

fd1$FlightDate[60]-fd1$FlightDate[30]

difftime(fd1$FlightDate[30],fd1$FlightDate[800],units = "weeks")

difftime(fd1$FlightDate[230],fd1$FlightDate[764],units = "days")

difftime(fd1$FlightDate[1213],fd1$FlightDate[1048],units = "hours")

# d <- duplicated(claim$Policy.Number)

# head(d,20)

# #Returns a vector of TRUE and FALSE, depending on whether each item is unique or is a duplicate of any previous item (naturally, the first element in this vector always will be FALSE)

# sum(duplicated(claim$Policy.Number))

# claim1 <- claim[!duplicated(claim$Policy.Number),]

# sum(duplicated(claim1$Policy.Number))

# levels(claim$Mfr.Model)

# length(levels(claim$Mfr.Model))

# audit <- read.csv("D:/Training/Datasets/audit.csv")

# names(audit)

# t <- table(audit$Employment)

# t

# names(t)

# which.max(t)

# most\_freq\_level <- names(t[which.max(t)])

***Visualization and data exploration***

#Alumni Donation

library(lattice)

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

don[1:5,]

table(don$Class.Year)

barchart(table(don$Class.Year),horizontal=FALSE,xlab="Class Year",col="black")

don$TGiving=don$FY00Giving+don$FY01Giving+don$FY02Giving+don$FY03Giving+don$FY04Giving

mean(don$TGiving)

sum(don$TGiving)

sd(don$TGiving)

seq(0,1,.05)

quantile(don$TGiving,probs=seq(0,1,0.05))

quantile(don$TGiving,probs=seq(0.95,1,0.01))

hist(don$TGiving)

histogram(don$TGiving)

plot(density(don$TGiving))

densityplot(don$TGiving)

hist(don$TGiving[don$TGiving!=0][don$TGiving[don$TGiving!=0]<=1000])

hist(don$TGiving[don$TGiving!=0 & don$TGiving <= 1000] ) # easier equivalent of previous command

par(mfrow=c(1,1))

boxplot(don$TGiving,horizontal=TRUE,xlab="Total Contribution")

boxplot(don$TGiving,outline=FALSE,horizontal=TRUE,xlab="Total Contribution") #remove outlier

ddd=don[don$TGiving>=30000,]

ddd

ddd1=ddd[,c(1:5,12)]

ddd1

ddd1[order(ddd1$TGiving,decreasing=TRUE),]

boxplot(TGiving~Class.Year,data=don,outline=FALSE)

boxplot(TGiving~Gender,data=don,outline=FALSE)

boxplot(TGiving~Marital.Status,data=don,outline=FALSE)

boxplot(TGiving~AttendenceEvent,data=don,outline=FALSE)

t4=tapply(don$TGiving,don$Major,mean,na.rm=TRUE)

t4

t5=table(don$Major)

t5

t6=cbind(t4,t5)

t7=t6[t6[,2]>10,]

t7[order(t7[,1],decreasing=TRUE),]

barchart(t7[,1],col="black")

t4=tapply(don$TGiving,don$Next.Degree,mean,na.rm=TRUE)

t4

t5=table(don$Next.Degree)

t5

t6=cbind(t4,t5)

t7=t6[t6[,2]>10,]

t7[order(t7[,1],decreasing=TRUE),]

t7

barchart(t7[,1],col="black")

densityplot(~TGiving|factor(Class.Year),

data=don[don$TGiving<=1000,][don[don$TGiving<=1000,]$TGiving>0,],

plot.points=FALSE,col="black")

t11=tapply(don$TGiving,don$Class.Year,FUN=sum,na.rm=TRUE)

t11

barplot(t11,ylab="Average Donation")

barchart(tapply(don$FY04Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal=FALSE,ylim=c(0,225000),col="black")

barchart(tapply(don$FY03Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal= FALSE,ylim=c(0,225000),col="black")

barchart(tapply(don$FY02Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal= FALSE,ylim=c(0,225000),col="black")

barchart(tapply(don$FY01Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal= FALSE,ylim=c(0,225000),col="black")

barchart(tapply(don$FY00Giving,don$Class.Year,FUN=sum,na.rm=TRUE),horizontal= FALSE,ylim=c(0,225000),col="black")

don$TGivingIND=cut(don$TGiving,c(-1,0.5,10000000),labels=FALSE)-1#ifelse condition

mean(don$TGivingIND)

t5=table(don$TGivingIND,don$Class.Year)

t5

barplot(t5,beside=TRUE)

?mosaicplot

mosaicplot(t5)

mosaicplot(factor(don$Class.Year) ~ factor(don$TGiving))

t50=tapply(don$TGivingIND,INDEX = list(don$Class.Year),FUN=mean,na.rm=TRUE)

t50

barchart(t50,horizontal=FALSE,col="black")

don$FY04GivingIND=cut(don$FY04Giving,c(-1,0.5,10000000),labels=FALSE)-1

t51=tapply(don$FY04GivingIND,don$Class.Year,FUN=mean,na.rm=TRUE)

t51

barchart(t51,horizontal=FALSE,col="black")

Data=data.frame(don$FY04Giving,don$FY03Giving,don$FY02Giving,don$FY01Giving,don$FY00Giving)

correlation=cor(Data)

correlation

plot(Data)

install.packages("ellipse")

library(ellipse)

plotcorr(correlation)

mosaicplot(factor(don$Gender)~factor(don$TGivingIND))

mosaicplot(factor(don$Marital.Status)~factor(don$TGivingIND))

?mosaicplot

t2=table(factor(don$Marital.Status),factor(don$TGivingIND))

mosaicplot(t2)

mosaicplot(factor(don$AttendenceEvent)~factor(don$TGivingIND))

t2=table(factor(don$Marital.Status),factor(don$TGivingIND),factor(don$AttendenceEvent))

t2

mosaicplot(t2[,,1])

mosaicplot(t2[,,2])

#Orange Juice

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

oj$store <- as.factor(oj$store)

oj[1:2,]

t1=tapply(oj$logmove,oj$brand,FUN=mean,na.rm=TRUE)

t1

t2=tapply(oj$logmove,INDEX=list(oj$brand,oj$week),FUN=mean,na.rm=TRUE)

t2

par(mfrow=c(1,1))

plot(t2[1,],type= "l",xlab="week",ylab="dominicks",ylim=c(7,12))

plot(t2[2,],type= "l",xlab="week",ylab="minute.maid",ylim=c(7,12))

plot(t2[3,],type= "l",xlab="week",ylab="tropicana",ylim=c(7,12))

logmove=c(t2[1,],t2[2,],t2[3,])

week1=c(40:160)

week=c(week1,week1,week1)

brand1=rep(1,121)

brand2=rep(2,121)

brand3=rep(3,121)

brand=c(brand1,brand2,brand3)

#as part of lattice package

xyplot(logmove~week|factor(brand),type= "l",layout=c(1,3),col="black")

boxplot(logmove~brand,data=oj)

histogram(~logmove|brand,data=oj,layout=c(1,3)) # layout can be changed according to situation

densityplot(~logmove|brand,data=oj,layout=c(1,3),plot.points=FALSE) #| different graphs on the basis of classification

densityplot(~logmove,groups=brand,data=oj,plot.points=FALSE) # all at the same time in the same graph

xyplot(logmove~week,data=oj,col="black")

xyplot(logmove~week|brand,data=oj,layout=c(1,3),col="black")

xyplot(logmove~price,data=oj,col="black")

xyplot(logmove~price|brand,data=oj,layout=c(1,3),col="black")

smoothScatter(oj$price,oj$logmove)

densityplot(~logmove,groups=feat, data=oj, plot.points=FALSE)

xyplot(logmove~price,groups=feat, data=oj) #groups part of lattce package

oj1=oj[oj$store == 5,]

xyplot(logmove~week|brand,data=oj1,type="l",layout=c(1,3),col="black")

xyplot(logmove~price,data=oj1,col="black")

xyplot(logmove~price|brand,data=oj1,layout=c(1,3),col="black")

densityplot(~logmove|brand,groups=feat,data=oj1,plot.points=FALSE)

xyplot(logmove~price|brand,groups=feat,data=oj1)

t21=tapply(oj$INCOME,oj$store,FUN=mean,na.rm=TRUE)

t21

t21[t21==max(t21)]

t21[t21==min(t21)]

oj1=oj[oj$store == 62,] # store no 62

oj2=oj[oj$store == 75,] # store no 75

oj3=rbind(oj1,oj2) #like apend in sas join horizontally

xyplot(logmove~price|store,data=oj3)

xyplot(logmove~price|store,groups=feat,data=oj3)

## store in the wealthiest neighborhood

## store in the poorest neighborhood

#Toyota Car

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

toyota[1:3,]

summary(toyota)

hist(toyota$Price)

## next we create indicator variables for the categorical variable

## FuelType with its three nominal outcomes: CNG, Diesel, and Petrol

#factor variable with n levels n-1

# factor gender 2 levels flag variable 1 (1/0)

toyota$FuelType1=ifelse(toyota$FuelType=="CNG",1,0)

toyota$FuelType2=ifelse(toyota$FuelType=="Diesel",1,0)

auto=toyota[-4]

auto[1:3,]

plot(Price~Age,data=auto)

smoothScatter(auto$Price ~ auto$Age, data = auto)

plot(Price~KM,data=auto)

smoothScatter(auto$Price ~ auto$KM, data = auto)

plot(Price~HP,data=auto)

smoothScatter(auto$Price ~ auto$HP, data = auto)

plot(Price~MetColor,data=auto)

table(auto$Price,auto$MetColor)

densityplot(~auto$Price|factor(auto$MetColor))

plot(Price~Automatic,data=auto)

plot(Price~CC,data=auto)

densityplot(~auto$Price|factor(auto$CC))

plot(Price~Doors,data=auto)

plot(Price~Weight,data=auto)

?densityplot

??densityplot

?lattice

#ggplot2 cheat sheet

***Basic Statistics***

#basic statistics code

proc print data = sashelp.sale(obs=20);

run;

proc contents data=sashelp.sale varnum;

run;

data prod\_sample;

set sashelp.dsale;

where country='CANADA';

run;

#Take a random sample of size 30.

proc surveyselect data = sashelp.sale

method = SRS

rep = 1

sampsize = 30 seed = 12345 out = prod\_sample\_30;

id \_all\_;

run;

# for getting N mean STD Min and Max

proc means data=sashelp.prdsale ;

var actual;

run;

#/\* Simple Random Sample; Size is 100 \*/

proc surveyselect data = sashelp.prdsale

method = SRS

rep = 1

sampsize = 100 seed = 12345 out = prod\_sample\_100;

id \_all\_;

run;

proc means data=prod\_sample\_100 ;

var actual;

run;

***Linear Regression***

############Linear Regression###########################

#

# lm(formula, data, subset, weights, na.action,

# method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,

# singular.ok = TRUE, contrasts = NULL, offset,)

library(MASS)

# Example 1: mtcars dataset #

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

data(mtcars)

boxplot(mpg~gear, data=mtcars)

mtcars <- within(mtcars, {

cyl <- factor(cyl, ordered=T)

am <- factor(am)

gear <- factor(gear, ordered=T)

vs <- factor(vs)

carb <- factor(carb,ordered=T)

})

str(mtcars)

plot(mtcars)

plot(mpg~disp, data=mtcars)

cor(mtcars$mpg,mtcars$wt) # with(mtcars,cor(mpg,wt))

m1 <- lm(mpg ~ wt, data=mtcars)

summary(m1)

plot(mtcars$wt,mtcars$mpg)

abline(m1)

names(m1)

m1$coefficients

m1$residuals

m1$fitted

summary((mtcars$mpg - m1$fitted)-m1$residuals) # difference is not EXACTLY zero

all.equal((mtcars$mpg - m1$fitted),m1$residuals) #

m1$model

# Null model

m0 <- mean(mtcars$mpg)

m0residuals <- mtcars$mpg - m0

par(mfrow=c(2,1))

hist(m0residuals, xlim=c(-20,20))

hist(m1$residuals,xlim=c(-20,20))

m2 <- lm(mpg ~ wt+disp, data=mtcars)

summary(m2)

m3 <- lm(mpg ~ wt+disp+hp, data=mtcars)

summary(m3)

m4 <- lm(mpg ~ wt+hp, data=mtcars)

summary(m4)

m5 <- lm(mpg ~ wt + disp + I(disp^2),data=mtcars)

# this uses disp^2 as a term

# not giving the I() would treat

# disp^2 as the interaction term of disp with itself

cor(mtcars[,c("mpg","wt","disp","hp")])

# mpg is strongly correlated with wt, disp and hp

plot(mtcars[,c('mpg','wt','disp','hp')])

# check if mpg is correlated with disp and hp after

# taking out the effect of wt

df <- data.frame(r1=m1$residuals,

mtcars[,c("wt","disp","hp")])

cor(df)

plot(df)

# comparing two models

# The F statistic is a formal comparison of these two models

# using the variances of the residuals

anova(m2,m1)

# tells us significance of the difference

# the one with lower RSS is better

# if difference not significant,

# use model with fewer parameters (parsimony principle)

anova(m4,m1)

anova(m4,m2) # won't work for two models with same DF

AIC(m4)

AIC(m2)

# gives us an idea but is not a TEST

m6 <- lm(mpg ~ wt + am, data=mtcars)

summary(m4)

mtcars$cyl <- factor(as.character(mtcars$cyl))

m7 <- lm(mpg ~ wt+cyl, data=mtcars)

summary(m7)

anova(m7)

# In this case tells us that cyl is significant

par(mfrow=c(2,2))

plot(m1)

# 1,1: checking if residuals are larger (smaller) as

# fitted values get larger (smaller)

### non-linear relationship here can indicate heteroscedasticity

# non-constant residuals may be addressed with transformation of dependent var

# 1,2: checking if residuals are normally distributed

# 1,3: basically similar to 1,1 but uses standardized residuals

# (to have mean 0 & SD 1) and sqrt(abs())

### 1,4: influence of extreme points (leverage points)

# might be addressed with transformation of independent variable

par(mfrow=c(2,2))

plot(m4)

m5 <- lm(mpg ~ wt+cyl+am, data= mtcars)

summary(m5)

par(mfrow=c(2,2))

plot(m5)

m6 <- lm(mpg ~ wt + cyl \* am, data= mtcars)

summary(m6)

# \* gives each individual effect and the interaction effect

# : gives ONLY the interaction (rarely used)

install.packages("car")

library(car)

vif(m6)

# Cutoff variously 3,4,6,10

#------------------ Akaike information criterion-------------------

ma <- lm(mpg~.,data=mtcars)

summary(ma)

s.ma <- stepAIC(ma)

summary(s.ma)

plot(s.ma)

######2nd example##################

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

FuelEff

par(mfrow=c(1,1))

plot(GPM~WT,data=FuelEff)

plot(GPM~DIS,data=FuelEff)

FuelEff=FuelEff[-1]

## regression on all data

m1=lm(GPM~.,data=FuelEff)

summary(m1)

cor(FuelEff)

install.packages("leaps")

library(leaps)

X=FuelEff[,2:7]

y=FuelEff[,1]

out=summary(regsubsets(X,y,nbest=2,nvmax=ncol(X)))

tab=cbind(out$which,out$rsq,out$adjr2,out$cp)

tab

m2=lm(GPM~WT,data=FuelEff)

summary(m2)

## cross-validation (leave one out) for the model on all six regressors

n=length(FuelEff$GPM)

diff=dim(n)

percdiff=dim(n)

for (k in 1:n) {

train1=c(1:n)

train=train1[train1!=k]

## the R expression "train1[train1!=k]" picks from train1 those

## elements that are different from k and stores those elements in the

## object train.

## For k=1, train consists of elements that are different from 1; that

## is 2, 3, ., n.

m1=lm(GPM~.,data=FuelEff[train,])

pred=predict(m1,newdat=FuelEff[-train,])

obs=FuelEff$GPM[-train]

diff[k]=obs-pred

percdiff[k]=abs(diff[k])/obs

}

me=mean(diff)

rmse=sqrt(mean(diff\*\*2))

mape=100\*(mean(percdiff))

me # mean error

rmse # root mean square error

mape # mean absolute percent error

## cross-validation (leave one out) for the model on weight only

n=length(FuelEff$GPM)

diff=dim(n)

percdiff=dim(n)

for (k in 1:n) {

train1=c(1:n)

train=train1[train1!=k]

m2=lm(GPM~WT,data=FuelEff[train,])

pred=predict(m2,newdat=FuelEff[-train,])

obs=FuelEff$GPM[-train]

diff[k]=obs-pred

percdiff[k]=abs(diff[k])/obs

}

me=mean(diff)

rmse=sqrt(mean(diff\*\*2))

mape=100\*(mean(percdiff))

me # mean error

rmse # root mean square error

mape # mean absolute percent error

#############3rd example###############

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

toyota[1:3,]

summary(toyota)

hist(toyota$Price)

## next we create indicator variables for the categorical variable

## FuelType with its three nominal outcomes: CNG, Diesel, and Petrol

v1=rep(1,length(toyota$FuelType))

v2=rep(0,length(toyota$FuelType))

toyota$FuelType1=ifelse(toyota$FuelType=="CNG",v1,v2)

toyota$FuelType2=ifelse(toyota$FuelType=="Diesel",v1,v2)

auto=toyota[-4]

auto[1:3,]

plot(Price~Age,data=auto)

plot(Price~KM,data=auto)

plot(Price~HP,data=auto)

plot(Price~MetColor,data=auto)

plot(Price~Automatic,data=auto)

plot(Price~CC,data=auto)

plot(Price~Doors,data=auto)

plot(Price~Weight,data=auto)

## regression on all data

m1=lm(Price~.,data=auto)

summary(m1)

set.seed(1)

## fixing the seed value for the random selection guarantees the

## same results in repeated runs

n=length(auto$Price)

n1=1000

n2=n-n1

train=sample(1:n,n1)

## regression on training set

m1=lm(Price~.,data=auto[train,])

summary(m1)

pred=predict(m1,newdat=auto[-train,])

obs=auto$Price[-train]

diff=obs-pred

percdiff=abs(diff)/obs

me=mean(diff)

rmse=sqrt(sum(diff\*\*2)/n2)

mape=100\*(mean(percdiff))

me # mean error

rmse # root mean square error

mape # mean absolute percent error

## cross-validation (leave one out)

n=length(auto$Price)

diff=dim(n)

percdiff=dim(n)

for (k in 1:n) {

train1=c(1:n)

train=train1[train1!=k]

13

m1=lm(Price~.,data=auto[train,])

pred=predict(m1,newdat=auto[-train,])

obs=auto$Price[-train]

diff[k]=obs-pred

percdiff[k]=abs(diff[k])/obs

}

me=mean(diff)

rmse=sqrt(mean(diff\*\*2))

mape=100\*(mean(percdiff))

me # mean error

rmse # root mean square error

mape # mean absolute percent error

## cross-validation (leave one out): Model with just Age

n=length(auto$Price)

diff=dim(n)

percdiff=dim(n)

for (k in 1:n) {

train1=c(1:n)

train=train1[train1!=k]

m1=lm(Price~Age,data=auto[train,])

pred=predict(m1,newdat=auto[-train,])

obs=auto$Price[-train]

diff[k]=obs-pred

percdiff[k]=abs(diff[k])/obs

}

me=mean(diff)

rmse=sqrt(mean(diff\*\*2))

mape=100\*(mean(percdiff))

me # mean error

rmse # root mean square error

mape # mean absolute percent error

## Adding the squares of Age and KM to the model

auto$Age2=auto$Age^2

auto$KM2=auto$KM^2

m11=lm(Price~Age+KM,data=auto)

summary(m11)

m12=lm(Price~Age+Age2+KM+KM2,data=auto)

summary(m12)

m13=lm(Price~Age+Age2+KM,data=auto)

summary(m13)

plot(m11$res~m11$fitted)

hist(m11$res)

plot(m12$res~m12$fitted)

***Logistic Regression***

## analyzing individual observations

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

dpen[1:4,]

dpen[359:362,]

m1=glm(Death~VRace+Agg,family=binomial,data=dpen)

m1

summary(m1)

## calculating logits

exp(m1$coef[2])

exp(m1$coef[3])

## plotting probability of getting death penalty as a function of

## aggravation separately for black (in black) and white (in red)

## victim

fitBlack=dim(501)

fitWhite=dim(501)

ag=dim(501)

for (i in 1:501) {

ag[i]=(99+i)/100

fitBlack[i]=exp(m1$coef[1]+ag[i]\*m1$coef[3])/(1+exp(m1$coef[1]+

+ ag[i]\*m1$coef[3]))

fitWhite[i]=exp(m1$coef[1]+m1$coef[2]+ag[i]\*m1$coef[3])/

+ (1+exp(m1$coef[1]+m1$coef[2]+ag[i]\*m1$coef[3]))

}

plot(fitBlack~ag,type="l",col="black",ylab="Prob[Death]",

xlab="Aggravation",ylim=c(0,1),

main="red line for white victim; black line for black victim")

points(fitWhite~ag,type="l",col="red")

## analyzing summarized data

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

dpenother

m1=glm(Death~VRace+Agg,family=binomial,weights=Freq,data=dpenother)

m1

summary(m1)

exp(m1$coef[2])

exp(m1$coef[3])

#####Example 2: Loan Acceptance

library(car) ## needed to recode variables

set.seed(1)

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

loan[1:3,]

## familiarize yourself with the data

hist(loan$Age)

hist(loan$Experience)

hist(loan$Income)

hist(loan$Family) ## below we treat loan$Family as categorical

hist(loan$CCAvg)

hist(loan$Mortgage)

hist(loan$SecuritiesAccount)

hist(loan$CDAccount)

hist(loan$Online)

hist(loan$CreditCard)

hist(loan$Education) ## below we treat loan$Education as categorical

response=loan$PersonalLoan

hist(response)

MeanRes=mean(response)

MeanRes

## creating indicator variables for loan$Family and loan$Education

v1=rep(1,dim(loan)[1])

v2=rep(0,dim(loan)[1])

## creating indicator variables for family size (4 groups: 1, 2, 3, 4)

loan$FamSize2=ifelse(loan$Family==2,1,0)

loan$FamSize3=ifelse(loan$Family==3,1,0)

loan$FamSize4=ifelse(loan$Family==4,1,0)

## creating indicator variables for education level (3 groups: 1, 2, 3)

loan$Educ2=ifelse(loan$Education==2,1,0)

loan$Educ3=ifelse(loan$Education==3,1,0)

xx=cbind(response,Age=loan$Age,Exp=loan$Experience,Inc=loan$Income,Fam2=loan$FamSize2,Fam3=loan$FamSize3,Fam4=loan$FamSize4,CCAve=loan$CCAvg,Mort=loan$Mortgage,SecAcc=loan$SecuritiesAccount,CD=loan$CDAccount,Online=loan$Online,CreditCard=loan$CreditCard,Educ2=loan$Educ2,Educ3=loan$Educ3)

xx[1:3,]

## split the data set into training and test (evaluation) set

n=dim(loan)[1]

n

n1=floor(n\*(0.6))

n1

n2=n-n1

n2

train=sample(1:n,n1)

## model fitted on all data

m1=glm(response~.,family=binomial,data=data.frame(xx))

summary(m1)

xx=xx[,-1]

xtrain <- xx[train,]

xnew <- xx[-train,]

ytrain <- response[train]

ynew <- response[-train]

## model fitted on the training data set

m2=glm(response~.,family=binomial,data=data.frame(response=ytrain,xtrain))

summary(m2)

## create predictions for the test (evaluation) data set

ptest=predict(m2,newdata=data.frame(xnew),type="response")

## predicted probabilities

hist(ptest)

plot(ynew~ptest)

## coding as 1 if probability 0.5 or larger

gg1=floor(ptest+0.5)

ttt=table(ynew,gg1)

ttt

error=(ttt[1,2]+ttt[2,1])/n2

error

## coding as 1 if probability 0.3 or larger

gg2=floor(ptest+0.7)

ttt=table(ynew,gg2)

ttt

error=(ttt[1,2]+ttt[2,1])/n2

error

bb=cbind(ptest,ynew)

bb

bb1=bb[order(ptest,decreasing=TRUE),]

bb1

## order cases in test set according to their success prob

## actual outcome shown next to it

## overall success probability in evaluation (test) data set

xbar=mean(ynew)

xbar

## calculating the lift

## cumulative 1's sorted by predicted values

## cumulative 1's using the average success prob from evaluation set

axis=dim(n2)

ax=dim(n2)

ay=dim(n2)

axis[1]=1

ax[1]=xbar

ay[1]=bb1[1,2]

for (i in 2:n2) {

axis[i]=i

ax[i]=xbar\*i

ay[i]=ay[i-1]+bb1[i,2]

}

aaa=cbind(bb1[,1],bb1[,2],ay,ax)

aaa[1:20,]

plot(axis,ay,xlab="number of cases",ylab="number of successes",main="Lift: Cum successes sorted by pred val/success prob")

points(axis,ax,type="l")

###Example 3: German Credit Data

#### \*\*\*\*\*\*\* German Credit Data \*\*\*\*\*\*\* ####

#### \*\*\*\*\*\*\* data on 1000 loans \*\*\*\*\*\*\* ####

## read data and create relevant variables

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

credit

credit$Default <- factor(credit$Default)

## re-level the credit history and a few other variables

credit$history = factor(credit$history, levels=c("A30","A31","A32","A33","A34"))

levels(credit$history) = c("good","good","poor","poor","terrible")

credit$foreign <- factor(credit$foreign, levels=c("A201","A202"), labels=c("foreign","german"))

credit$rent <- factor(credit$housing=="A151")

credit$purpose <- factor(credit$purpose, levels=c("A40","A41","A42","A43","A44","A45","A46","A47","A48","A49","A410"))

levels(credit$purpose) <- c("newcar","usedcar",rep("goods/repair",4),"edu",NA,"edu","biz","biz")

## for demonstration, cut the dataset to these variables

credit <- credit[,c("Default","duration","amount","installment","age", "history", "purpose","foreign","rent")]

credit[1:3,]

summary(credit) # check out the data

## create a design matrix

## factor variables are turned into indicator variables

## the first column of ones is omitted

Xcred <- model.matrix(Default~.,data=credit)[,-1]

Xcred[1:3,]

## creating training and prediction datasets

## select 900 rows for estimation and 100 for testing

set.seed(1)

train <- sample(1:1000,900)

xtrain <- Xcred[train,]

xnew <- Xcred[-train,]

ytrain <- credit$Default[train]

ynew <- credit$Default[-train]

credglm=glm(Default~.,family=binomial,data=data.frame(Default=ytrain,xtrain))

summary(credglm)

## prediction: predicted default probabilities for cases in test set

ptest <- predict(credglm,newdata=data.frame(xnew),type="response")

data.frame(ynew,ptest)

## What are our misclassification rates on that training set?

## We use probability cutoff 1/6

## coding as 1 (predicting default) if probability 1/6 or larger

gg1=floor(ptest+(5/6))

ttt=table(ynew,gg1)

ttt

error=(ttt[1,2]+ttt[2,1])/100

error

#######Logistic regression case study#########

#---------initial data exploration and manipulation------------------

traindata\_R <- read.csv(file.choose())

str(traindata\_R)

traindata\_R$Wife\_education <- as.factor(traindata\_R$Wife\_education)

traindata\_R$Husband\_education <- as.factor(traindata\_R$Husband\_education)

traindata\_R$Wife\_religion <- as.factor(traindata\_R$Wife\_religion)

traindata\_R$Wife\_working <- as.factor(traindata\_R$Wife\_working)

traindata\_R$Husband\_occupation <- as.factor(traindata\_R$Husband\_occupation)

traindata\_R$Standard\_of\_living\_index <- as.factor(traindata\_R$Standard\_of\_living\_index)

traindata\_R$Media\_exposure <- as.factor(traindata\_R$Media\_exposure)

traindata\_R$Party\_voted\_for <- as.factor(traindata\_R$Party\_voted\_for)

str(traindata\_R)

levels(as.factor(traindata\_R$Number\_of\_children\_ever\_born))

table(traindata\_R$Number\_of\_children\_ever\_born)

plot(density(traindata\_R$Number\_of\_children\_ever\_born))

#higher vale of children per women can be treated as outlier

table(traindata\_R$Number\_of\_children\_ever\_born, traindata\_R$Party\_voted\_for)

#women with more children prefer to vote for party 1 /B

table(traindata\_R$Party\_voted\_for)

table(traindata\_R$Wife\_age)

plot(density(traindata\_R$Wife\_age))

# ------------- can be done using following methods as well -----------------

prop.table(table(traindata\_R$Wife\_education,traindata\_R$Standard\_of\_living\_index),1)

prop.table(table(traindata\_R$Wife\_education,traindata\_R$Wife\_working),1)

tapply(traindata\_R$Number\_of\_children\_ever\_born ,FUN = mean,INDEX = traindata\_R$Wife\_education )

#-------------------------- Logistic regresison ---------------

model\_log <- glm(Party\_voted\_for~. , data = traindata\_R, family = "binomial")

summary(model\_log)

library(MASS)

model\_log.step <- stepAIC(model\_log)

#----removing some variables does improve the model as AIC in not improved significantly

#---- so as mentioned in the problem statement, we will keep all the variables.

model\_log.pred.prob <- predict(model\_log,type="response")

plot(model\_log.pred.prob,traindata\_R$Party\_voted\_for)

# ----- changing to categorical variable in test data set ------------

testdata\_R <- read.csv(file.choose())

testdata\_R$Wife\_education <- as.factor(testdata\_R$Wife\_education)

testdata\_R$Husband\_education <- as.factor(testdata\_R$Husband\_education)

testdata\_R$Wife\_religion <- as.factor(testdata\_R$Wife\_religion)

testdata\_R$Wife\_working <- as.factor(testdata\_R$Wife\_working)

testdata\_R$Husband\_occupation <- as.factor(testdata\_R$Husband\_occupation)

testdata\_R$Standard\_of\_living\_index <- as.factor(testdata\_R$Standard\_of\_living\_index)

testdata\_R$Media\_exposure <- as.factor(testdata\_R$Media\_exposure)

testdata\_R$Party\_voted\_for <- as.factor(testdata\_R$Party\_voted\_for)

traindata\_R$pred.prob <- predict(model\_log,type="response")

testdata\_R$pred.prob <- predict(model\_log,type="response",newdata= testdata\_R)

# plotting ROC curve ---------

install.packages("ROCR")

library(ROCR)

train\_pred <- prediction(traindata\_R$pred.prob,

traindata\_R$Party\_voted\_for)

train\_perf <- performance(train\_pred, measure = "tpr", x.measure = "fpr")

par(mfrow=c(2,1))

plot(train\_perf,main="Training data")

abline(0,1)

val\_pred <- prediction(testdata\_R$pred.prob,

testdata\_R$Party\_voted\_for)

val\_perf <- performance(val\_pred, measure = "tpr", x.measure = "fpr")

plot(val\_perf,main="Testing data")

abline(0,1)

#The two curves are almost identical, and indicate that

#the model performs much better than a random classifier.

#-------- now finding optimum cut off for logistic regresison-------

opt.cut = function(perf, pred)

{

cut.ind = mapply(FUN=function(x, y, p)

{

d = (x - 0)^2 + (y-1)^2

ind = which(d == min(d))

c(sensitivity = y[[ind]], specificity = 1-x[[ind]],

cutoff = p[[ind]])

}, perf@x.values, perf@y.values, pred@cutoffs)

}

print(opt.cut(train\_perf, train\_pred))

c <- opt.cut(train\_perf, train\_pred)

cutoff <- c[3]

cutoff

traindata\_R$class <- ifelse(traindata\_R$pred.prob>cutoff,1,0)

t1 <- table("Actual"=traindata\_R$Party\_voted\_for,

"Predicted"=traindata\_R$class)

t1

round(prop.table(t1,1),2)

(t1[1,1] + t1[2,2])/sum(t1)

#We have achieved an accuracy of 67% with which is little poor

#We can then do the same thing for the validation data:

testdata\_R$class <- ifelse(testdata\_R$pred.prob>cutoff, 1 ,0)

t2 <- table("Actual"= testdata\_R$Party\_voted\_for, "Predicted"=testdata\_R$class)

t2

round(prop.table(t2,1),2)

(t2[1,1] + t2[2,2])/sum(t2)

#in the test dataset, the accuracy is little better than compared to train dataset.

#also the performance is in line with the results in train dataset

#which assures that model performs good for out of bag samples as well.

install.packages("caret")

library(caret)

library(e1071)

confusionMatrix(traindata\_R$Party\_voted\_for,traindata\_R$class)

confusionMatrix(testdata\_R$Party\_voted\_for,testdata\_R$class)

#----- collective ROC curve---------

par(mfrow=c(2,1))

plot(train\_perf,main="Training data", col = "red")

abline(0,1)

legend("bottomright", legend=c("Logistic Regression"), col=c("red"), lty=1, lwd=2, cex = .3)

plot(val\_perf,main="Testing data", col = "red")

abline(0,1)

legend("bottomright", legend=c("Logistic Regression"), col=c("red"), lty=1, lwd=2, cex = .3)

#------Accuracy of the model-------------

t2 #Confusion matrix logistic regression

(t2[1,1] + t2[2,2])/sum(t2) #Logistic Regression accuracy on test data

#Somer's D test

install.packages("Hmisc")

library(Hmisc)

Hmisc::somersd(y=traindata\_R$Party\_voted\_for,x=traindata\_R$pred.prob)

Hmisc::rcorr.cens(y=traindata\_R$Party\_voted\_for,x=traindata\_R$pred.prob)

#Hosmer Lemeshow test as Home work

***KNN***

#### \*\*\*\*\*\*\* Forensic Glass \*\*\*\*\*\* ####

library(textir) ## needed to standardize the data

library(MASS) ## a library of example datasets

data(fgl) ## loads the data into R; see help(fgl)

fgl

levels(fgl$type)

## data consists of 214 cases

## here are illustrative box plots of the features stratified by

## glass type

par(mfrow=c(3,3), mai=c(.3,.6,.1,.1))

plot(RI ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Al ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Na ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Mg ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Ba ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Si ~ type, data=fgl, col=c(grey(.2),2:6))

plot(K ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Ca ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Fe ~ type, data=fgl, col=c(grey(.2),2:6))

## for illustration, consider the RIxAl plane

## use nt=200 training cases to find the nearest neighbors for

## the remaining 14 cases. These 14 cases become the evaluation

## (test, hold-out) cases

n=length(fgl$type)

nt=200

set.seed(1) ## to make the calculations reproducible in repeated runs

train <- sample(1:n,nt)

## Standardization of the data is preferable, especially if

## units of the features are quite different

## could do this from scratch by calculating the mean and

## standard deviation of each feature, and use those to

## standardize.

## Even simpler, use the normalize function in the R-package textir;

## it converts data frame columns to mean-zero sd-one

x <- scale(fgl[,c(4,1)],center = T,scale = T)

plot(density(x[,2]))

plot(density(x[,1]))

View(x)

x[1:3,]

library(class)

nearest1 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=1)

nearest5 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=5)

data.frame(fgl$type[-train],nearest1,nearest5)

## plot them to see how it worked

par(mfrow=c(1,2))

## plot for k=1 (single) nearest neighbor

plot(x[train,],col=fgl$type[train],cex=.8,main="1-nearest neighbor")

points(x[-train,],bg=nearest1,pch=21,col=grey(.9),cex=1.25)

## plot for k=5 nearest neighbors

plot(x[train,],col=fgl$type[train],cex=.8,main="5-nearest neighbors")

points(x[-train,],bg=nearest5,pch=21,col=grey(.9),cex=1.25)

legend("topright",legend=levels(fgl$type),fill=1:6,bty="n",cex=.75)

## calculate the proportion of correct classifications on this one

## training set

pcorrn1=100\*sum(fgl$type[-train]==nearest1)/(n-nt)

pcorrn5=100\*sum(fgl$type[-train]==nearest5)/(n-nt)

pcorrn1

pcorrn5

## cross-validation (leave one out)

pcorr=dim(10)

for (k in 1:10) {

pred=knn.cv(x,fgl$type,k)

pcorr[k]=100\*sum(fgl$type==pred)/n

}

pcorr

par(mfrow=c(1,1))

plot(1:10,pcorr,type = "b",pch = 16, lwd =3, col = "red")

## Note: Different runs may give you slightly different results as ties

## are broken at random

## using all nine dimensions (RI plus 8 chemical concentrations)

x <- scale(fgl[,c(1:9)],center = TRUE,scale = T)

nearest1 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=1)

nearest5 <- knn(train=x[train,],test=x[-train,],cl=fgl$type[train],k=5)

data.frame(fgl$type[-train],nearest1,nearest5)

## calculate the proportion of correct classifications

pcorrn1=100\*sum(fgl$type[-train]==nearest1)/(n-nt)

pcorrn5=100\*sum(fgl$type[-train]==nearest5)/(n-nt)

pcorrn1

pcorrn5

## cross-validation (leave one out)

pcorr=dim(10)

for (k in 1:10) {

pred=knn.cv(x,fgl$type,k)

pcorr[k]=100\*sum(fgl$type==pred)/n

}

pcorr

#### \*\*\*\*\*\*\* German Credit Data \*\*\*\*\*\*\* ####

#### \*\*\*\*\*\*\* data on 1000 loans \*\*\*\*\*\*\* ####

library(textir) ## needed to standardize the data

library(class) ## needed for knn

## read data and create some `interesting' variables

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

credit

credit$Default <- factor(credit$Default)

## re-level the credit history and a few other variables

credit$history = factor(credit$history, levels=c("A30","A31","A32","A33","A34"))

levels(credit$history) = c("good","good","poor","poor","terrible")

credit$foreign <- factor(credit$foreign, levels=c("A201","A202"), labels=c("foreign","german"))

credit$rent <- factor(credit$housing=="A151")

credit$purpose <- factor(credit$purpose, levels=c("A40","A41","A42","A43","A44","A45","A46","A47","A48","A49","A410"))

levels(credit$purpose) <- c("newcar","usedcar",rep("goods/repair",4),"edu",NA,"edu","biz","biz")

## for demonstration, cut the dataset to these variables

credit <- credit[,c("Default","duration","amount","installment","age", "history", "purpose","foreign","rent")]

credit[1:3,]

summary(credit) # check out the data

## for illustration we consider just 3 loan characteristics:

## amount,duration,installment

## Standardization of the data is preferable, especially if

## units of the features are quite different

## We use the normalize function in the R-package textir;

## it converts data frame columns to mean-zero sd-one

x <- scale(credit[,c(2,3,4)],center = TRUE,scale = T)

x[1:3,]

## training and prediction datasets

## training set of 900 borrowers; want to classify 100 new ones

set.seed(1)

train <- sample(1:1000,900) ## this is training set of 900 borrowers

xtrain <- x[train,]

xnew <- x[-train,]

ytrain <- credit$Default[train]

ynew <- credit$Default[-train]

## k-nearest neighbor method

library(class)

nearest1 <- knn(train=xtrain, test=xnew, cl=ytrain, k=1)

nearest3 <- knn(train=xtrain, test=xnew, cl=ytrain, k=3)

data.frame(ynew,nearest1,nearest3)[1:10,]

## calculate the proportion of correct classifications

pcorrn1=100\*sum(ynew==nearest1)/100

pcorrn3=100\*sum(ynew==nearest3)/100

pcorrn1

pcorrn3

## plot for 3nn

plot(xtrain[,c("amount","duration")],col=c(4,3,6,2)[credit[train,"installment"]],pch=c(1,2)[as.numeric(ytrain)],main="Predicted default, by 3 nearest neighbors",cex.main=.95)

points(xnew[,c("amount","duration")],bg=c(4,3,6,2)[credit[train,"installment"]],pch=c(21,24)[as.numeric(nearest3)],cex=1.2,col=grey(.7))

legend("bottomright",pch=c(1,16,2,17),bg=c(1,1,1,1),legend=c("data 0","pred 0","data 1","pred 1"),title="default",bty="n",cex=.8)

legend("topleft",fill=c(4,3,6,2),legend=c(1,2,3,4),title="installment %",horiz=TRUE,bty="n",col=grey(.7),cex=.8)

## above was for just one training set

## cross-validation (leave one out)

pcorr=dim(10)

for (k in 1:10) {

pred=knn.cv(x,cl=credit$Default,k)

pcorr[k]=100\*sum(credit$Default==pred)/1000

}

pcorr

# Weight neighbor KNN############

library(kknn)

data(iris)

m <- dim(iris)[1]

val <- sample(1:m, size = round(m/3), replace = FALSE,

prob = rep(1/m, m))

iris.learn <- iris[-val,]

iris.valid <- iris[val,]

iris.kknn <- kknn(Species~., iris.learn, iris.valid, distance = 1,

kernel = "triangular")

summary(iris.kknn)

fit <- fitted(iris.kknn)

table(iris.valid$Species, fit)

pcol <- as.character(as.numeric(iris.valid$Species))

pairs(iris.valid[1:4], pch = pcol, col = c("green3", "red")

[(iris.valid$Species != fit)+1])

data(ionosphere)

ionosphere.learn <- ionosphere[1:200,]

ionosphere.valid <- ionosphere[-c(1:200),]

fit.kknn <- kknn(class ~ ., ionosphere.learn, ionosphere.valid)

table(ionosphere.valid$class, fit.kknn$fit)

(fit.train1 <- train.kknn(class ~ ., ionosphere.learn, kmax = 15,

kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 1))

table(predict(fit.train1, ionosphere.valid), ionosphere.valid$class)

(fit.train2 <- train.kknn(class ~ ., ionosphere.learn, kmax = 15,

kernel = c("triangular", "rectangular", "epanechnikov", "optimal"), distance = 2))

table(predict(fit.train2, ionosphere.valid), ionosphere.valid$class)

***Naive Bayes***

# Load required libraries

library(tm)

install.packages("RTextTools")

library(RTextTools)

library(e1071)

library(dplyr)

library(caret)

# Library for parallel processing

df<- read.csv(file.choose(), stringsAsFactors = FALSE)

glimpse(df)

set.seed(1)

df <- df[sample(nrow(df)), ]

df <- df[sample(nrow(df)), ]

glimpse(df)

df$class <- as.factor(df$class)

corpus <- Corpus(VectorSource(df$text))

corpus

inspect(corpus[1:3])

# Use dplyr's %>% (pipe) utility to do this neatly.

corpus.clean <- corpus %>%

tm\_map(content\_transformer(tolower)) %>%

tm\_map(removePunctuation) %>%

tm\_map(removeNumbers) %>%

tm\_map(removeWords, stopwords(kind="en")) %>%

tm\_map(stripWhitespace)

dtm <- DocumentTermMatrix(corpus.clean)

# Inspect the dtm

inspect(dtm[40:50, 10:15])

df.train <- df[1:1500,]

df.test <- df[1501:2000,]

dtm.train <- dtm[1:1500,]

dtm.test <- dtm[1501:2000,]

corpus.clean.train <- corpus.clean[1:1500]

corpus.clean.test <- corpus.clean[1501:2000]

dim(dtm.train)

fivefreq <- findFreqTerms(dtm.train, 5)

length((fivefreq))

dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary = fivefreq))

dim(dtm.train.nb)

dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary = fivefreq))

dim(dtm.train.nb)

# Function to convert the word frequencies to yes (presence) and no (absence) labels

convert\_count <- function(x) {

y <- ifelse(x > 0, 1,0)

y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))

y

}

trainNB <- apply(dtm.train.nb, 2, convert\_count)

testNB <- apply(dtm.test.nb, 2, convert\_count)

classifier <- naiveBayes(trainNB, df.train$class, laplace = 1)

pred <- predict(classifier, newdata=testNB)

table("Predictions"= pred, "Actual" = df.test$class )

conf.mat <- confusionMatrix(pred, df.test$class)

conf.mat

conf.mat$byClass

conf.mat$overall

conf.mat$overall['Accuracy']

library(wordcloud)

wordcloud(corpus.clean.train,

min.freq=40,max.words = 300,

random.order=F,

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

Pos <- subset(df.train,class=="Pos")

Neg <- subset(df.train,class=="Neg")

wordcloud(Pos$text,min.freq=20,random.order=F,max.words = 300,scale=c(3,0.5),

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

text(0.5,1,"Positive")

wordcloud(Neg$text,min.freq=20,random.order=F,max.words = 300,scale=c(3,0.5),

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

text(0.5,1,"Negative")

############Iris dataset##############

mydata <- iris

head(mydata)

set.seed(1234)

sam <- sample(nrow(mydata),floor(nrow(mydata)\*0.67))

train <- mydata[sam,]

test <- mydata[-sam,]

m <- naiveBayes(Species ~ ., data = train)

m

p <- predict(m, train)

tp <- table("Actual"=train[,5],"Predicted"=p)

round(100\*prop.table(tp,1),2)

q <- predict(m,test)

tq <- table("Actual"=test[,5],"Predicted"=q)

round(100\*prop.table(tq,1),2)

***Decision Tree***

#########Prostate Cancer#############

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

prostate

library(tree)

## Construct the tree

pstree <- tree(lcavol ~., data=prostate, mindev=0.1, mincut=1)

pstree <- tree(lcavol ~., data=prostate, mincut=1)

pstree

par(mfrow=c(1,1))

plot(pstree, col=8)

text(pstree, digits=1)

#Pruning of tree

pstcut <- prune.tree(pstree,k=1.7)

plot(pstcut)

text(pstcut)

pstcut

pstcut <- prune.tree(pstree,k=2.05)

plot(pstcut)

text(pstcut)

pstcut

pstcut <- prune.tree(pstree,k=3)

plot(pstcut)

text(pstcut)

pstcut

#pruning without any parameter

pstcut <- prune.tree(pstree)

pstcut

plot(pstcut)

pstcut <- prune.tree(pstree,best=3)

pstcut

plot(pstcut)

## Use cross-validation to prune the tree

set.seed(2)

cvpst <- cv.tree(pstree, K=10)

cvpst$size

cvpst$dev

plot(cvpst, pch=21, bg=8, type="p", cex=1.5, ylim=c(65,100))

pstcut <- prune.tree(pstree, best=3)

pstcut

plot(pstcut, col=8)

text(pstcut)

## Plot what we end up with

plot(prostate[,c("lcp","lpsa")],cex=0.2\*exp(prostate$lcavol))

abline(v=.261624, col=4, lwd=2)

lines(x=c(-2,.261624), y=c(2.30257,2.30257), col=4, lwd=2)

###############Example 2: Motorcycle Acceleration

library(MASS)

library(tree)

data(mcycle)

mcycle

plot(accel~times,data=mcycle)

mct <- tree(accel ~ times, data=mcycle)

mct

plot(mct, col=8)

text(mct, cex=.75) ## we use different font size to avoid print overlap

## scatter plot of data with overlay of fitted function

x=c(1:6000)

x=x/100

y1=seq(-4.357,-4.357,length.out=1510)

y2=seq(-39.120,-39.120,length.out=140)

y3=seq(-86.31,-86.31,length.out=300)

y4=seq(-114.7,-114.7,length.out=490)

y5=seq(-42.49,-42.49,length.out=300)

y6=seq(10.25,10.25,length.out=240)

y7=seq(40.72,40.72,length.out=520)

y8=seq(3.291,3.291,length.out=2500)

y=c(y1,y2,y3,y4,y5,y6,y7,y8)

plot(accel~times,data=mcycle)

lines(y~x)

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

library(MASS)

library(tree)

## read in the iris data

iris

iristree <- tree(Species~.,data=iris)

iristree

plot(iristree)

plot(iristree,col=8)

text(iristree,digits=2)

summary(iristree)

irissnip=snip.tree(iristree,nodes=c(7,12))

irissnip

plot(irissnip)

text(irissnip)

***Random Forest***

#### \*\*\*\*\*\*\* Forensic Glass \*\*\*\*\*\* ####

#library(textir) ## needed to standardize the data

library(MASS) ## a library of example datasets

data(fgl) ## loads the data into R; see help(fgl)

fgl

levels(fgl$type)

## data consists of 214 cases

## here are illustrative box plots of the features stratified by

## glass type

par(mfrow=c(3,3), mai=c(.3,.6,.1,.1))

plot(RI ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Al ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Na ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Mg ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Ba ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Si ~ type, data=fgl, col=c(grey(.2),2:6))

plot(K ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Ca ~ type, data=fgl, col=c(grey(.2),2:6))

plot(Fe ~ type, data=fgl, col=c(grey(.2),2:6))

fgl$type <- as.factor(fgl$type)

table(fgl$type)

set.seed(1234)

sam <- sample(nrow(fgl),floor(nrow(fgl)\*0.85))

train <- fgl[sam,]

test <- fgl[-sam,]

library(randomForest)

fit <- randomForest(type ~ ., data = train,ntree=500 ,mtry = 3)

importance(fit)

p <- predict(fit,train)

t1 <- table("Actual fgl type"=train$type,"Predicted type"=p)

100\*prop.table(t1,1)

q <- predict(fit,newdata=test)

t2 <- table("Actual fgl type"=test$type,"Predicted type"=q)

round(100\*prop.table(t2,1),1)

pcorr=100\*sum(test$type==q)/(nrow(test))

pcorr

***Clustering***

###################K Means Clustering########################

head(iris)

data <- iris[,-5]

class <- iris[,5]

wss <- 0

for (i in 1:15) wss[i] <- kmeans(data,centers=i)$tot.withinss

plot(1:15, wss, type="b",

xlab="Number of Clusters",

ylab="Within groups sum of squares",col="blue",pch=16,lwd=3)

# There is inflection point or "elbow of the graph" at k = 3;

# some knowledge of the data (namely the number of species) also tells us

# that it might be logical to look for three clusters in our data.

results <- kmeans(data,centers=3)

class(results)

results$size

results$cluster

table(class,results$cluster)

#install.packages("gmodels")

library(gmodels)

CrossTable(class,results$cluster)

par(mfrow=c(1,2))

plot(data$Petal.Length,data$Petal.Width,col=results$cluster,pch=19,

xlab="Petal Length",ylab="Petal Width",main="By cluster")

plot(data$Petal.Length,data$Petal.Width,col=class,pch=19,

xlab="Petal Length",ylab="Petal Width",main="By species")

par(mfrow=c(1,2))

plot(data$Sepal.Length, data$Sepal.Width,col=results$cluster,pch=19,

xlab="Sepal Length",ylab="Sepal Width",main="By cluster")

plot(data$Sepal.Length, data$Sepal.Width,col=class,pch=19,

xlab="Sepal Length",ylab="Sepal Width",main="By cluster")

p <- princomp(data)

par(mfrow=c(1,2))

plot(p$scores[,1],p$scores[,2],col=results$cluster,

pch=16,

xlab="Principal Component 1",

ylab="Principal Component 2",

main="By cluster")

plot(p$scores[,1],p$scores[,2],col=class,

pch=16,

xlab="Principal Component 1",

ylab="Principal Component 2",

main="By species")

library(cluster)

par(mfrow=c(1,1))

clusplot(data,results$cluster,color=TRUE,shade=F,labels=0,

main="Results for k = 3 clusters",xlab="Principal Component 1",

ylab="Principal Component 2")

#Example 2: European Protein Consumption

### \*\*\* European Protein Consumption, in grams/person-day \*\*\* ###

## read in the data

food <- read.csv("F:/Data Science class by me/Data set for R/protein.csv")

food[1:3,]

## first, clustering on just Red and White meat (p=2) and k=3 clusters

set.seed(1) ## to fix the random starting clusters

grpMeat <- kmeans(food[,c("WhiteMeat","RedMeat")], centers=3, nstart=10)

grpMeat

## list of cluster assignments

o=order(grpMeat$cluster)

data.frame(food$Country[o],grpMeat$cluster[o])

## plotting cluster assignments on Red and White meat scatter plot

plot(food$Red, food$White, type="n", xlim=c(3,19), xlab="Red Meat", ylab="White Meat")

text(x=food$Red, y=food$White, labels=food$Country, col=grpMeat$cluster+1)

## same analysis, but now with clustering on all protein groups

## change the number of clusters to 7

wss <- 0

for (i in 1:15) wss[i] <- kmeans(food[,-1],centers=i, nstart = 10)$tot.withinss

plot(1:15, wss, type="b",

xlab="Number of Clusters",

ylab="Within groups sum of squares",col="blue",pch=16,lwd=3)

set.seed(1)

grpProtein <- kmeans(food[,-1], centers=5, nstart=10)

o=order(grpProtein$cluster)

data.frame(food$Country[o],grpProtein$cluster[o])

plot(food$Red, food$Cereals, type="n", xlim=c(3,19), xlab="Red Meat", ylab="Cereals")

text(x=food$Red, y=food$Cereals, labels=food$Country, col=grpProtein$cluster)

#Example 3: Monthly US Unemployment Rates

## read the data; series are stored column-wise with labels in first row

raw <- read.csv("F:/Data Science class by me/Data set for R/unempstates.csv")

View(raw)

raw[1:3,]

## time sequence plots of three series

plot(raw[,5],type="l",ylim=c(0,12),lwd=3,xlab="month",ylab="unemployment rate") ## CA

points(raw[,32],type="l", cex = .5, lwd=3,col = "dark red") ## New York

points(raw[,15],type="l", cex = .5, lwd=3,col = "dark green") ## Iowa

## transpose the data

## then we have 50 rows (states) and 416 columns (time periods)

rawt=matrix(nrow=50,ncol=416)

rawt=t(raw)

rawt[1:3,]

## k-means clustering in 416 dimensions

set.seed(1)

grpunemp2 <- kmeans(rawt, centers=2, nstart=10)

table(sort(grpunemp2$cluster))

grpunemp3 <- kmeans(rawt, centers=3, nstart=10)

sort(grpunemp3$cluster)

grpunemp4 <- kmeans(rawt, centers=4, nstart=10)

sort(grpunemp4$cluster)

grpunemp5 <- kmeans(rawt, centers=5, nstart=10)

sort(grpunemp5$cluster)

## another analysis

## data set unemp.csv with means and standard deviations for each state

## k-means clustering on 2 dimensions (mean, stddev)

unemp <- read.csv("F:/Data Science class by me/Data set for R/unemp.csv")

unemp[1:3,]

set.seed(1)

grpunemp <- kmeans(unemp[,c("mean","stddev")], centers=5, nstart=10)

## list of cluster assignments

o=order(grpunemp$cluster)

data.frame(unemp$state[o],grpunemp$cluster[o])

plot(unemp$mean,unemp$stddev,type="n",xlab="mean", ylab="stddev")

text(x=unemp$mean,y=unemp$stddev,labels=unemp$state, col=grpunemp$cluster+1)

############################Heirarchical Clustering################

head(mtcars)

View(mtcars)

str(mtcars)

d <- dist(mtcars)

ward <- hclust(d,method="ward.D")

plot(ward,cex=0.8)

rect.hclust(ward, k=3)

#protein dataset

# library(cluster)

# ## Protein Data

# food <- read.csv("F:/Data Science class by me/Data set for R/protein.csv")

# food[1:3,]

# ## we use the program agnes in the package cluster

# ## argument diss=FALSE indicates that we use the dissimilarity

# ## matrix that is being calculated from raw data.

# ## argument metric="euclidian" indicates that we use Euclidian distance

# ## no standardization is used as the default

# ## the default is "average" linkage

# ## first we consider just Red and White meat clusters

# food2=food[,c("WhiteMeat","RedMeat")]

# food2agg=agnes(food2,diss=FALSE,metric="euclidian")

# food2agg

# plot(food2agg) ## dendrogram

# food2agg$merge ## describes the sequential merge steps

# ## identical result obtained by first computing the distance matrix

# food2aggv=agnes(daisy(food2),metric="euclidian")

# plot(food2aggv)

# ## Using data on all nine variables (features)

# ## Euclidean distance and average linkage

# foodagg=agnes(food[,-1],diss=FALSE,metric="euclidian")

# plot(foodagg) ## dendrogram

# foodagg$merge ## describes the sequential merge steps

# ## Using data on all nine variables (features)

# ## Euclidean distance and single linkage

# foodaggsin=agnes(food[,-1],diss=FALSE,metric="euclidian",method="single")

# plot(foodaggsin) ## dendrogram

# foodaggsin$merge ## describes the sequential merge steps

# ## Euclidean distance and complete linkage

# foodaggcomp=agnes(food[,-1],diss=FALSE,metric="euclidian",method="single")

# plot(foodaggcomp) ## dendrogram

# foodaggcomp$merge ## describes the sequential merge steps

#####Unemployment rate #####################################

#

# library(cluster)

# raw <- read.csv("F:/Data Science class by me/Data set for R/unempstates.csv")

# raw[1:3,]

# ## Correlation on levels

# corlevel=cor(data.frame(raw))

# disslevel=1-corlevel

# outcorlevel=agnes(disslevel,diss=TRUE,metric="euclidian",method="single")

# plot(outcorlevel) ## dendrogram; single linkage

# outcorlevel=agnes(disslevel,diss=TRUE,metric="euclidian",method="complete")

# plot(outcorlevel) ## dendrogram; complete linkage

# outcorlevel=agnes(disslevel,diss=TRUE,metric="euclidian")

# plot(outcorlevel) ## dendrogram; average linkage

# ## Correlation on differences

# X=matrix(nrow=415,ncol=50)

# for (j in 1:50) {

# for (i in 1:415) {

# X[i,j]=raw[i+1,j]-raw[i,j]

# }

# }

# colnames(X)=colnames(raw)

# cordiff=cor(data.frame(X))

# dissdiff=1-cordiff

# outcordiff=agnes(dissdiff,diss=TRUE,metric="euclidian",method="single")

# plot(outcordiff) ## dendrogram; single linkage

# outcordiff=agnes(dissdiff,diss=TRUE,metric="euclidian",method="complete")

# plot(outcordiff) ## dendrograml; complete linkage

# outcordiff=agnes(dissdiff,diss=TRUE,metric="euclidian")

# plot(outcordiff) ## dendrogram; average linkage

**Principal Component Analysis**

food <- read.csv("F:/Data Science class by me/Data set for R/protein.csv")

food[1:3,]

## correlation matrix

cor(food[,-1])

pcafood <- prcomp(food[,-1], scale=TRUE)

## we strip the first column (country labels) from the data set

## scale = TRUE: variables are first standardized. Default is FALSE

pcafood

names(pcafood)

biplot(pcafood, scale = 0)

foodpc <- predict(pcafood)

foodpc

## how many principal components do we need?

plot(pcafood, main="")

mtext(side=1, "European Protein Principal Components", line=1, font=2)

par(mfrow=c(1,1))

std\_dev <- pcafood$sdev

pr\_var <- std\_dev^2

prop\_varex <- pr\_var/sum(pr\_var)

plot(prop\_varex, xlab = "Principal Component",

ylab = "Proportion of Variance Explained",

type = "b")

## how do the PCs look?

par(mfrow=c(1,2))

plot(foodpc[,1:2], type="n", xlim=c(-4,5))

text(x=foodpc[,1], y=foodpc[,2], labels=food$Country)

plot(foodpc[,3:4], type="n", xlim=c(-3,3))

text(x=foodpc[,3], y=foodpc[,4], labels=food$Country)

pcafood$rotation[,2]

par(mfrow=c(1,1))

biplot(pcafood, scale = 0)

#Example 2: Monthly US Unemployment Rates

## read the data; series are stored column-wise with labels in first row

raw <- read.csv("F:/Data Science class by me/Data set for R/unempstates.csv")

## transpose so that we have 50 rows (states) and 416 columns

raw

rawt=matrix(nrow=50,ncol=416)

rawt=t(raw)

rawt[1:3,]

pcaunemp <- prcomp(rawt,scale=FALSE)

pcaunemp

plot(pcaunemp, main="")

mtext(side=1,"Unemployment: 50 states",line=1,font=2)

pcaunemp$rotation[,1]

pcaunemp$rotation[1:10,1] ## just the first 10 values

ave=dim(416)

for (j in 1:416) {

ave[j]=mean(rawt[,j])

}

par(mfrow=c(1,2))

## plot negative loadings for first principal component

plot(-pcaunemp$rotation[,3])

## plot monthly averages of unemployment rates

plot(ave,type="l",ylim=c(3,10),xlab="month",ylab="ave unemployment rate")

abs(cor(ave,pcaunemp$rotation[,1]))

pcaunemp$rotation[,2]

pcaunemp$rotation[,3]

## below we obtain the scores of the principal components

## the first 2-3 principal components do a good job

unemppc <- predict(pcaunemp)

unemppc

## below we construct a scatter plot of the first two

## principal components

## we assess whether an informal clustering on the first two

## principal components would have lead to a similar

## clustering than the clustering results of the k-means

## clustering approach applied on all 416 components

## the graph indicates that it does

set.seed(1)

grpunemp3 <- kmeans(rawt,centers=3,nstart=10)

par(mfrow=c(1,1))

plot(unemppc[,1:2],type="n")

text(x=unemppc[,1],y=unemppc[,2],labels= row.names(rawt) ,col=rainbow(7)[grpunemp3$cluster])

**Support Vector Machine (SVM)**

library("e1071")

head(iris,5)

attach(iris)

x <- subset(iris, select=-Species)

y <- Species

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

summary(svm\_model)

svm\_model1 <- svm(x,y)

summary(svm\_model1)

pred <- predict(svm\_model1,x)

system.time(pred <- predict(svm\_model1,x))

table(pred,y)

svm\_tune <- tune(svm, train.x=x, train.y=y,

kernel="polynomial", ranges=list(cost=10^(-1:2), gamma=c(.5,1,2)))

print(svm\_tune)

svm\_model\_after\_tune <- svm(Species ~ ., data=iris, kernel="polynomial", cost=.1, gamma=1)

summary(svm\_model\_after\_tune)

pred <- predict(svm\_model\_after\_tune,x)

system.time(predict(svm\_model\_after\_tune,x))

table(pred,y)

# visualize (classes by color, SV by crosses):

plot(cmdscale(dist(iris[,-5])),

col = as.integer(iris[,5]),

pch = c("o","+")[1:150 %in% svm\_model\_after\_tune$index + 1])

**Market Basket Analysis**

### \*\*\* Play counts \*\*\* ###

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

View(lastfm)

lastfm[1:19,]

length(lastfm$user) ## 289,955 records in the file

lastfm$user <- factor(lastfm$user)

levels(lastfm$user) ## 15,000 users

levels(lastfm$artist) ## 1,004 artists

library(arules) ## a-rules package for association rules

## Computational environment for mining association rules and

## frequent item sets

## we need to manipulate the data a bit before using arules

## we split the data in the vector x into groups defined in vector f

## in supermarket terminology, think of users as shoppers and artists

## as items bought

playlist <- split(x=lastfm[,"artist"],f=lastfm$user)

## split into a list of users

playlist[1:2]

## the first two listeners (1 and 3) listen to the following bands

## an artist may be mentioned by the same user more than once

## it is important to remove artist duplicates before creating

## the incidence matrix

playlist <- lapply(playlist,unique) ## remove artist duplicates

playlist <- as(playlist,"transactions")

## view this as a list of "transactions"

## transactions is a data class defined in arules

itemFrequency(playlist)

## lists the support of the 1,004 bands

## number of times band is listed to on the playlist of 15,000 users

## computes relative frequency of artist mentioned by the 15,000 users

itemFrequencyPlot(playlist,support=.08, topN = 20)

## plots the item frequencies (only bands with > % support)

## Finally, we build the association rules

## only associations with support > 0.01 and confidence > .50

## this rules out rare bands

musicrules <- apriori(playlist,parameter=list(support=.01,confidence=.5))

inspect(musicrules)

## let's filter by lift > 5.

## Among those associations with support > 0.01 and confidence > .50,

## only show those with lift > 5

inspect(subset(musicrules, subset=lift > 5))

## lastly, order by confidence to make it easier to understand

inspect(sort(subset(musicrules, subset=lift > 5), by="confidence"))

#Example 2 Predicting Income#########################

library(arules)

data(AdultUCI)

View(AdultUCI)

dim(AdultUCI)

AdultUCI[1:3,]

AdultUCI[["fnlwgt"]] <- NULL

AdultUCI[["education-num"]] <- NULL

AdultUCI[["age"]] <- ordered(cut(AdultUCI[["age"]], c(15, 25, 45, 65,

+ 100)), labels = c("Young", "Middle-aged", "Senior", "Old"))

AdultUCI[["hours-per-week"]] <- ordered(cut(AdultUCI[["hours-per-week"]], c(0, 25, 40, 60, 168)), labels = c("Part-time",

"Full-time", "Over-time", "Workaholic"))

AdultUCI[["capital-gain"]] <- ordered(cut(AdultUCI[["capital-gain"]],

c(-Inf, 0, median(AdultUCI[["capital-gain"]][AdultUCI

[["capital-gain"]] > 0]), Inf)),

labels = c("None", "Low", "High"))

AdultUCI[["capital-loss"]] <- ordered(cut(AdultUCI[["capital-loss"]],

c(-Inf, 0, median(AdultUCI[["capital-loss"]][AdultUCI

[["capital-loss"]] > 0]), Inf)), labels = c("none", "low", "high"))

Adult <- as(AdultUCI, "transactions")

Adult

summary(Adult)

aa=as(Adult,"matrix")

## transforms transaction matrix into an incidence matrix

aa[1:2,] # print the first two rows of the incidence matrix

dim(aa)

itemFrequencyPlot(Adult[, itemFrequency(Adult) > 0.2], col=brewer.pal(8,'Pastel2'), topN = 20, cex.names = 1)

rules <- apriori(Adult, parameter = list(support = 0.01,confidence = 0.6))

rules

summary(rules)

rulesIncomeSmall <- subset(rules, subset = rhs %in% "income=small" & lift > 1.2)

inspect(sort(rulesIncomeSmall, by = "confidence")[1:10])

rulesIncomeLarge <- subset(rules, subset = rhs %in% "income=large" & lift > 1.2)

inspect(sort(rulesIncomeLarge, by = "confidence")[1:10])

**Time Series Analysis**

#########################TS ES####################

kings <- scan("http://robjhyndman.com/tsdldata/misc/kings.dat",skip=3)

kings

class(kings)

kingstimeseries <- ts(kings)

kingstimeseries

class(kingstimeseries)

births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")

birthstimeseries <- ts(births, frequency=12, start=c(1946,1))

birthstimeseries

plot.ts(kingstimeseries)

plot.ts(birthstimeseries)

par(mfrow=c(1,1))

d <- decompose(birthstimeseries)

plot(d)

souvenir <- scan("http://robjhyndman.com/tsdldata/data/fancy.dat")

souvenirtimeseries <- ts(souvenir, frequency=12, start=c(1987,1))

plot.ts(souvenirtimeseries)

logsouvenirtimeseries <- log(souvenirtimeseries)

plot.ts(logsouvenirtimeseries)

library("TTR") #technical trading rules

kingstimeseriesSMA3 <- SMA(kingstimeseries,n=3) #Simple moving Average

plot.ts(kingstimeseriesSMA3)

# SMA mainly works with smoothing of moving averages and to understand the trend

kingstimeseriesSMA8 <- SMA(kingstimeseries,n=8)

plot.ts(kingstimeseriesSMA8)

par(mfrow=c(3,1))

plot.ts(kingstimeseries,main="Original TS")

plot.ts(kingstimeseriesSMA3,main="SMA 3")

plot.ts(kingstimeseriesSMA8,main="SMA 8")

birthstimeseriescomponents <- decompose(birthstimeseries)

plot(birthstimeseriescomponents)

birthstimeseriescomponents$seasonal

plot(birthstimeseriescomponents)

birthstimeseriescomponents <- decompose(birthstimeseries)

plot(birthstimeseriescomponents)

birthstimeseriesseasonallyadjusted <-

birthstimeseries - birthstimeseriescomponents$seasonal

par(mfrow=c(1,1))

plot.ts(birthstimeseriesseasonallyadjusted)

birthstimeseriessacomps <- decompose(birthstimeseriesseasonallyadjusted)

plot(birthstimeseriessacomps)

par(mfrow=c(2,1))

plot(birthstimeseries)

plot(birthstimeseriesseasonallyadjusted)

plot(decompose(birthstimeseriesseasonallyadjusted))

######ES########

par(mfrow=c(1,1))

rain <- scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat",skip=1)

View(rain)

rainseries <- ts(rain,start=c(1813))

plot.ts(rainseries)

m <- mean(rainseries)

abline(h=m,lwd=3)

s <- sd(rainseries)

abline(h=m+s,lwd=2,lty=2)

abline(h=m-s,lwd=2,lty=2)

rainseriesforecasts <- HoltWinters(rainseries,

beta=FALSE,

gamma=FALSE)

rainseriesforecasts

plot(rainseriesforecasts)

rainseriesforecasts1 <- HoltWinters(rainseries,

beta=TRUE,

gamma=FALSE)

rainseriesforecasts1

plot(rainseriesforecasts1)

rainseriesforecasts$SSE

rainseriesforecasts1$SSE

rainseriesforecasts2 <- HoltWinters(rainseries,

beta=FALSE,

gamma=FALSE,

l.start=23.56)

plot(rainseriesforecasts2)

rainseriesforecasts2$SSE

library(forecast)

rainseriesforecasts2 <- forecast(rainseriesforecasts, h=8)

rainseriesforecasts2

plot(rainseriesforecasts2)

Box.test(rainseriesforecasts2$residuals, lag=20, type="Ljung-Box") #test for correlation testing

plot.ts(rainseriesforecasts2$residuals)

abline(h = mean(rainseriesforecasts2$residuals[-1]))

trainstart <- 1813

trainend <- 1900

teststart <- 1901

testend <- 1912

fyears <- testend - teststart + 1

train <- window(rainseries,

start=trainstart,

end=trainend)

test <- window(rainseries,

start=teststart,

end=testend)

rainseriesforecasts3 <-

HoltWinters(train,

beta=TRUE,

gamma=FALSE)

rainseriesforecaststest <-

forecast(rainseriesforecasts3,

h=fyears)

plot(rainseriesforecaststest,

xlim=c(trainstart,testend))

lines(test, col="darkgreen",lwd=3)

legend("bottomright",

legend=c("Actual", "HW forecast"),

col=c("darkgreen","blue"),

lty=1,

lwd=2,

cex=0.6)

acf(rainseriesforecasts2$residuals[-1], lag.max=20)

#Holt's Exponential Smoothing

skirts <- scan("http://robjhyndman.com/tsdldata/roberts/skirts.dat",skip=5)

skirtsseries <- ts(skirts,start=c(1866))

plot.ts(skirtsseries)

skirtsseriesforecasts <- HoltWinters(skirtsseries, gamma=FALSE)

skirtsseriesforecasts

plot(skirtsseriesforecasts)

HoltWinters(skirtsseries, gamma=FALSE, l.start=608, b.start=9)

skirtsseriesforecasts2 <- forecast(skirtsseriesforecasts, h=19)

plot(skirtsseriesforecasts2)

#acf(skirtsseriesforecasts2$residuals[2:45], lag.max=20)

Box.test(skirtsseriesforecasts2$residuals, lag=20, type="Ljung-Box")

plot.ts(skirtsseriesforecasts2$residuals)

abline(h = mean(skirtsseriesforecasts2$residuals,na.rm = TRUE))

#Holt-Winters Exponential Smoothing

logsouvenirtimeseries <- log(souvenirtimeseries)

souvenirtimeseriesforecasts <- HoltWinters(logsouvenirtimeseries)

souvenirtimeseriesforecasts

plot(souvenirtimeseriesforecasts)

souvenirtimeseriesforecasts2 <- forecast(souvenirtimeseriesforecasts, h=48)

plot(souvenirtimeseriesforecasts2)

#acf(souvenirtimeseriesforecasts2$residuals, lag.max=20)

Box.test(souvenirtimeseriesforecasts2$residuals, lag=20, type="Ljung-Box")

plot.ts(souvenirtimeseriesforecasts2$residuals)

############ARIMA############

data(AirPassengers)

View(AirPassengers)

class(AirPassengers)

#This tells you that the data series is in a time series format

start(AirPassengers)

end(AirPassengers)

frequency(AirPassengers)

summary(AirPassengers)

#The number of passengers are distributed across the spectrum

plot(AirPassengers)

#This will plot the time series

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

# This will fit in a line

cycle(AirPassengers)

#This will print the cycle across years.

plot(aggregate(AirPassengers,FUN=mean))

#This will aggregate the cycles and display a year on year trend

boxplot(AirPassengers~cycle(AirPassengers))

#Box plot across months will give us a sense on seasonal effect

library(tseries)

adf.test((AirPassengers), alternative="stationary", k=0)

acf(log(AirPassengers))

acf(diff(log(AirPassengers)))

pacf(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))

#########tractor sale############

data = read.csv(file.choose())

data = ts(data[,2],start = c(2003,1),frequency = 12)

plot(data, xlab='Years', ylab = 'Tractor Sales')

plot(diff(data),ylab='Differenced Tractor Sales')

plot(log10(data),ylab='Log (Tractor Sales)')

plot(diff(log10(data)),ylab='Differenced Log (Tractor Sales)') #stationary plot

par(mfrow = c(1,2))

acf(ts(diff(log10(data))),main='ACF Tractor Sales')

pacf(ts(diff(log10(data))),main='PACF Tractor Sales')

require(forecast)

library(tseries)

ARIMAfit = auto.arima(log10(data), approximation=FALSE,trace=FALSE)

summary(ARIMAfit)

par(mfrow = c(1,1))

pred = predict(ARIMAfit, n.ahead = 36)

pred

plot(data,type='l',xlim=c(2004,2018),ylim=c(1,1600),xlab = 'Year',ylab = 'Tractor Sales')

lines(10^(pred$pred),col='blue')

lines(10^(pred$pred+2\*pred$se),col='orange') #twice of stndard deviation

lines(10^(pred$pred-2\*pred$se),col='orange')

par(mfrow=c(1,2))

acf(ts(ARIMAfit$residuals),main='ACF Residual')

pacf(ts(ARIMAfit$residuals),main='PACF Residual')