**Wilson Regularization code**

library(foreign)

library(Hmisc)

library(psych)

library(magrittr)

library(dplyr)

library(ggpubr)

library(corrplot)

library(aod)

library(ggplot2)

library(tidyverse)

library(caret)

setwd("/Users/laytonwoohbill/Desktop/Depaul/2019-summer/Final Project")

MyData <- read.csv(file="data.csv", header=TRUE, sep=",")

MyData

head(MyData)

describe(MyData)

plot(MyData)

summary(MyData)

table(MyData$diagnosis)

levels(MyData$diagnosis)

factor2 <- factor(MyData$diagnosis)

#dummy

library(dummies)

MyData <- cbind(MyData, dummy(MyData$diagnosis, sep = "\_"))

#plot

library(GGally)

library(ggplot2)

library(corrplot)

library(glmnet)

# Split the data into training and test set

set.seed(123)

smp\_size<-floor(0.70\*nrow(MyData))

train\_ind <- sample(seq\_len(nrow(MyData)), size = smp\_size)

train <- MyData[train\_ind, ]

test <- MyData[-train\_ind, ]

#split data in to three group

train.all<- train[,c(3:32)]

train.b<- train[,c(34)]

train.mean<- train[,c(3:12)]

train.se<- train[,c(34,13:22)]

train.worst<- train[,c(34,23:32)]

summary(train.all)

test.all<- test[,c(3:32)]

test.b<- test[,c(34)]

test.mean<- test[,c(3:12)]

test.se<- test[,c(34,13:22)]

test.worst<- test[,c(34,23:32)]

M<-cor(train.all, method="spearman")

M

corrplot(M, method = "square")

ggcorr(train.all)

fit.lasso = glmnet(as.matrix(train.all), train.b,alpha=1)

plot(fit.lasso)

plot(fit.lasso,xvar="lambda",label=TRUE)

fit.ridge = glmnet(as.matrix(train.all), train.b, alpha=0)

plot(fit.ridge)

plot(fit.ridge,xvar="lambda",label=TRUE)

# Find the best lambda using cross-validation

cv.lasso <- cv.glmnet(as.matrix(train.all),train.b, alpha=1,

family = "binomial",type.measure="class",nfolds=10)

plot(cv.lasso)

# Display regression coefficients

coef(cv.lasso )

cv.lasso$lambda.min

cv.lasso$lambda.1se

coef(cv.lasso, cv.lasso$lambda.min)

coef(cv.lasso, cv.lasso$lambda.1se)

# Final model with lambda.min

lasso.model <- glmnet(as.matrix(train.all),train.b,alpha = 1, family = "binomial",

lambda = cv.lasso$lambda.min)

lasso.model

coef(lasso.model)

# Make prediction on test data

x.test <- model.matrix(radius\_mean ~., test.all)

x.test

probabilities <- predict(lasso.model,newx = x.test,s=NULL)

probabilities

predicted.classes <- ifelse(probabilities > 0.5, "B", "M")

predicted.classes

# Model accuracy

observed.classes <- test$diagnosis

table(predicted.classes,observed.classes)

mean(predicted.classes == observed.classes)

# Fit the final model on the training data

model <- glmnet(as.matrix(train.all), train.b, alpha = 1, family = "binomial",

lambda = cv.lasso$lambda.1se)

model

coef(model)

# Make prediction on test data

y.test <- model.matrix(radius\_mean ~., test.all)

y.test

probabilities\_y <- predict(model,newx = y.test,s=NULL)

predict.classes <- ifelse(probabilities\_y > 0.5, "B", "M")

predict.classes

# Model accuracy

observed.classes <- test$diagnosis

table(predict.classes,observed.classes)

mean(predict.classes == observed.classes)

# Find the best lambda using cross-validation

cv.ridge <- cv.glmnet(as.matrix(train.all),train.b,family = "binomial",

alpha=0,type.measure="class",nfolds=10)

coef(cv.ridge)

cv.ridge$lambda.min

cv.ridge$lambda.1se

coef(cv.ridge, cv.ridge$lambda.min)

coef(cv.ridge, cv.ridge$lambda.1se)

######

ridge.model <- glmnet(as.matrix(train.all),train.b,alpha = 0, family = "binomial",

lambda = cv.ridge$lambda.min)

ridge.model

coef(ridge.model)

# Make prediction on test data

a.test <- model.matrix(radius\_mean ~., test.all)

a.test

probabilities\_a <- predict(ridge.model,newx = a.test,s=NULL)

a\_predict.classes <- ifelse(probabilities\_a > 0.5, "B", "M")

a\_predict.classes

# Model accuracy

observed.classes <- test$diagnosis

table(a\_predict.classes,observed.classes)

mean(a\_predict.classes == observed.classes)

######

######

######

se\_ridge.model <- glmnet(as.matrix(train.all),train.b,alpha = 0, family = "binomial",

lambda = cv.ridge$lambda.1se)

se\_ridge.model

coef(se\_ridge.model)

# Make prediction on test data

b.test <- model.matrix(radius\_mean ~., test.all)

b.test

probabilities\_b <- predict(se\_ridge.model,newx = b.test,s=NULL)

b\_predict.classes <- ifelse(probabilities\_b > 0.5, "B", "M")

b\_predict.classes

# Model accuracy

observed.classes <- test$diagnosis

table(b\_predict.classes,observed.classes)

mean(b\_predict.classes == observed.classes)

**Elizabeth PCA Code**

#Principial Component Analysis (PCA) and Factor Analysis in R

#Libraries

library(Hmisc) #Describe Function

library(psych) #Multiple Functions for Statistics and Multivariate Analysis

library(GGally) #ggpairs Function

library(ggplot2) #ggplot2 Functions

library(corrplot) #Plot Correlations

library(REdaS) #Bartlett's Test of Sphericity

library(psych) #PCA/FA functions

library(factoextra) #PCA Visualizations

library("FactoMineR") #PCA functions

library(ade4) #PCA Visualizations

#set working directory

setwd("C:\\Users\\eniet\\Documents\\School\\Current\\DSC424\\Project\\")

PCA\_Plot = function(pcaData)

{

library(ggplot2)

theta = seq(0,2\*pi,length.out = 100)

circle = data.frame(x = cos(theta), y = sin(theta))

p = ggplot(circle,aes(x,y)) + geom\_path()

loadings = data.frame(pcaData$rotation, .names = row.names(pcaData$rotation))

p + geom\_text(data=loadings, mapping=aes(x = PC1, y = PC2, label = .names, colour = .names, fontface="bold")) +

coord\_fixed(ratio=1) + labs(x = "PC1", y = "PC2")

}

PCA\_Plot\_Secondary = function(pcaData)

{

library(ggplot2)

theta = seq(0,2\*pi,length.out = 100)

circle = data.frame(x = cos(theta), y = sin(theta))

p = ggplot(circle,aes(x,y)) + geom\_path()

loadings = data.frame(pcaData$rotation, .names = row.names(pcaData$rotation))

p + geom\_text(data=loadings, mapping=aes(x = PC3, y = PC4, label = .names, colour = .names, fontface="bold")) +

coord\_fixed(ratio=1) + labs(x = "PC3", y = "PC4")

}

PCA\_Plot\_Psyc = function(pcaData)

{

library(ggplot2)

theta = seq(0,2\*pi,length.out = 100)

circle = data.frame(x = cos(theta), y = sin(theta))

p = ggplot(circle,aes(x,y)) + geom\_path()

loadings = as.data.frame(unclass(pcaData$loadings))

s = rep(0, ncol(loadings))

for (i in 1:ncol(loadings))

{

s[i] = 0

for (j in 1:nrow(loadings))

s[i] = s[i] + loadings[j, i]^2

s[i] = sqrt(s[i])

}

for (i in 1:ncol(loadings))

loadings[, i] = loadings[, i] / s[i]

loadings$.names = row.names(loadings)

p + geom\_text(data=loadings, mapping=aes(x = PC1, y = PC2, label = .names, colour = .names, fontface="bold")) +

coord\_fixed(ratio=1) + labs(x = "PC1", y = "PC2")

}

PCA\_Plot\_Psyc\_Secondary = function(pcaData)

{

library(ggplot2)

theta = seq(0,2\*pi,length.out = 100)

circle = data.frame(x = cos(theta), y = sin(theta))

p = ggplot(circle,aes(x,y)) + geom\_path()

loadings = as.data.frame(unclass(pcaData$loadings))

s = rep(0, ncol(loadings))

for (i in 1:ncol(loadings))

{

s[i] = 0

for (j in 1:nrow(loadings))

s[i] = s[i] + loadings[j, i]^2

s[i] = sqrt(s[i])

}

for (i in 1:ncol(loadings))

loadings[, i] = loadings[, i] / s[i]

loadings$.names = row.names(loadings)

print(loadings)

p + geom\_text(data=loadings, mapping=aes(x = PC3, y = PC4, label = .names, colour = .names, fontface="bold")) +

coord\_fixed(ratio=1) + labs(x = "PC3", y = "PC4")

}

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

########### Import Data & EDA ##########

data <- read.csv('data.csv', header = TRUE, sep = ',')

#Check Sample Size and Number of Variables

dim(data)

#569-Sample Size and 33 variables

#Show Structure of Dataset; shows all variables in dataset

str(data, list.len=ncol(data))

#column names

names(data)

#Preview Data

head(data)

#remove id and assignment and preview

data2 <- data[,3:32]

head(data2)

#Show descriptive statistics

library(Hmisc)

describe(data2)

#Check for missing values

sum(is.na(data))

#Exploratory Analysis Graphing: Histograms, Correlations

#histogram

library(ggplot2)

ggplot2

ggplot(data = data2,aes(data2$radius\_mean)) + geom\_histogram()

# To save the ggplot as png

ggsave("p1.png")

#Check Correlation Plot

library(corrplot)

c = cor(data2)

corrplot(c, method = 'ellipse', order = 'AOE')

# Run a correlation test to see how correlated the variables are. Which correlations are significant

options("scipen"=100, "digits"=5)

round(cor(data2), 2)

MCorrTest = corr.test(data2, adjust="none")

MCorrTest

M = MCorrTest$p

M

# Now, for each element, see if it is < .01 (or whatever significance) and set the entry to

# true = significant or false

MTest = ifelse(M < .01, T, F)

MTest

# Now lets see how many significant correlations there are for each variable. We can do

# this by summing the columns of the matrix

colSums(MTest) - 1 # We have to subtract 1 for the diagonal elements (self-correlation)

#all variables are signigicatly correlated to 17-29 other variables

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

#PCA

#test for factorability

#KMO sampling adequacy

library(psych)

KMO(data2)

#Overall MSA = 0.83 this is goood

#Test Bartlett's Test of Sphericity

library(REdaS)

bart\_spher(data2)

#p-value < 2.22e-16 (Very Small Number) meaning,

#Test for Reliability Analysis using Cronbach's Alpha # check last 10 min of lecture. thatis where she changed this code

library(psych)

alpha(data2,check.keys=TRUE)

#raw\_alpha = 0.59

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

#create PCA and normalize

dataPCA = prcomp(data2, scale = T)

summary(dataPCA)

#scree plot

plot(dataPCA, main= 'Scree Plot - Kaiser Meyer Method')

abline(1, 0)

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

#Check PCA visualizations

plot(dataPCA) #Scree Plot

PCA\_Plot(dataPCA) #PCA\_plot1

PCA\_Plot\_Secondary(dataPCA) #PCA\_Plot2

biplot(p2) #Biplot

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

#Best Way to Conduct PCA Analysis

p2 = psych::principal(data2, rotate="varimax", nfactors=3, scores=TRUE)

p2

print(p2$loadings, cutoff=.653, sort=T)

#initial cutoff value of .568 and n = 6 comps with 2 vars

#PCAs Other Available Information

ls(p2)

p2$values

p2$communality

p2$rot.mat

#scores of PCA

#these are the new values representing the original data

#but instead of having a avlue for each feature it has a value for each set of

#correlated features aka each component

scores <- p2$scores

dim(scores) #569 3

head(scores)

##### merge eigenvalues with class ######

#get diagnosis cols to merge w/ scores

diagnosis <- data[,2]

head(diagnosis)

#M = 2 and B = 1

#bind scores with diagnosis and review

all<- cbind(scores, diagnosis)

dim(all)

head(all)

**Matt’s CCA Code**

library(foreign)

library(CCA)

ccaWilks = function(set1, set2, cca)

{

ev = ((1 - cca$cor^2))

ev

n = dim(set1)[1]

p = length(set1)

q = length(set2)

k = min(p, q)

m = n - 3/2 - (p + q)/2

m

w = rev(cumprod(rev(ev)))

# initialize

d1 = d2 = f = vector("numeric", k)

for (i in 1:k)

{

s = sqrt((p^2 \* q^2 - 4)/(p^2 + q^2 - 5))

si = 1/s

d1[i] = p \* q

d2[i] = m \* s - p \* q/2 + 1

r = (1 - w[i]^si)/w[i]^si

f[i] = r \* d2[i]/d1[i]

p = p - 1

q = q - 1

}

pv = pf(f, d1, d2, lower.tail = FALSE)

dmat = cbind(WilksL = w, F = f, df1 = d1, df2 = d2, p = pv)

}

## Project Data CCA import

setwd("C:/Users/mtimm/Downloads")

projectData = read.csv("data.csv", header = TRUE, sep = ",")

## split data into two sets

standardMeasurements = projectData[, 3:6]

characteristicMeasurements = projectData[, 7:12]

## cca and correlation table

ccProject <- cc(standardMeasurements, characteristicMeasurements)

matcor(standardMeasurements, characteristicMeasurements)

##wilks measurement to run p-test

wilksProject = ccaWilks(standardMeasurements, characteristicMeasurements, ccProject)

round(wilksProject, 6)

wilksProject

## all scores

ccProject$scores

loadingsProject = comput(standardMeasurements, characteristicMeasurements, ccProject)

loadingsProject$corr.X.xscores

loadingsProject$corr.Y.yscores

## CCA values plot

ccProject$cor

par(mfrow = c(1,2))

barplot(ccProject$cor, main = "Canonical correlations", col = "pink")

##helioplot

plt.cc(ccProject, var.label = TRUE, ind.names = projectData[,1])

##nonstandardized coef

ccProject[3:4]

##standardized coefficients

s1 <- diag(sqrt(diag(cov(standardMeasurements))))

s1 %\*% ccProject$xcoef

s2 <- diag(sqrt(diag(cov(characteristicMeasurements))))

s2 %\*% ccProject$ycoef

**Ishita Mehta Linear Discriminant Analysis**

setwd('C:/Users/ishita.mehta/Downloads/breast-cancer-wisconsin-data')

data <- read.csv('data.csv',header = TRUE,sep = ',')

dim(data)

head(data)

names(data)

sum(is.na(data))

# NO missing values

library(psych)

describe(data)

data <- data[,2:32]

head(data)

str(data, list.len=ncol(data))

# Frequency

table <- table(data$diagnosis)

barplot(table,xlab="diagnosis")

# Assumptions: Equal variance

heplots::covEllipses(data[,2:31],

data$diagnosis,

fill = TRUE,

pooled = FALSE,

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

variables = c(1:30),

fill.alpha = 0.05)

boxm <- heplots::boxM(data[, c(2:31)], data$diagnosis)

boxm

plot(boxm)

# LDA

library(MASS)

tumorLDA = lda(data$diagnosis ~ ., data=data)

tumorLDA

plot(tumorLDA)

tumorLDA2 = lda(data$diagnosis ~ ., data=data, CV=T)

tumorLDA2

tumorLDA2$class

table(tumorLDA2$class, data$diagnosis)

coef(tumorLDA)

# Confusion matrix

library(caret)

cm = confusionMatrix(tumorLDA2$class,data$diagnosis)

cm

fourfoldplot(cm$table)

**Dayana Linear and Logistic Regression**

if (!require(car)) install.packages("car")

if (!require(MASS)) install.packages("MASS")

if (!require(corrplot)) install.packages("corrplot")

if (!require(DescTools)) install.packages("DescTools") #VIF Function

if (!require(Hmisc)) install.packages("Hmisc")

library(readxl)

breastc <- read\_excel("C:/Users/dayan/Desktop/breastc.xlsx")

str(breastc)

#cheking the dimensions

dim(breastc)

#convertin target variable to numeric:

breastc <- breastc[,-1]

breastc$diagnosis <- factor(ifelse(breastc$diagnosis=="B","Benign","Malignant"))

str(breastc)

breastc$diagnosis= as.numeric(breastc$diagnosis)-1

head(breastc)

View(breastc)

b=table(breastc$diagnosis)

barplot(b)

#cheking missing values

sum(is.na(breastc))

#distribution

hist(breastc$diagnosis)

hist(breastc$radius\_mean)

hist(breastc$texture\_mean)

hist(breastc$perimeter\_mean)

hist(breastc$area\_mean)

hist(breastc$smoothness\_mean)

hist(breastc$compactness\_mean)

hist(breastc$concavity\_mean)

#Boxplots

boxplot(breastc$concavity\_mean, col = "blue", main="Diagnosis ",

ylab="concavity\_mean" ) #\*mean extream outlier

#Boxplot by Grouping Variable

boxplot(radius\_mean~diagnosis,data=breastc, main="Diagnosis by radius\_mean", xlab="Diagnosis",

ylab="radius\_mean", col = c("blue","red"))

boxplot(texture\_mean~diagnosis,data=breastc, main="Diagnosis by texture\_mean", xlab="Diagnosis",

ylab="texture\_mean", col = c("blue","red"))

boxplot(fractal\_dimension\_mean~diagnosis,data=breastc, main="Diagnosis by fractal\_dimension\_mean", xlab="Diagnosis",

ylab="fractal\_dimension\_mean", col = c("blue","red"))

boxplot(perimeter\_mean ~diagnosis,data=breastc, main="Diagnosis by perimeter\_mean ", xlab="Diagnosis",

ylab="perimeter\_mean ", col = c("blue","red"))

boxplot(symmetry\_mean~diagnosis,data=breastc, main="Diagnosis by symmetry\_mean", xlab="Diagnosis",

ylab="symmetry\_mean", col = c("blue","red"))

boxplot(compactness\_se~diagnosis,data=breastc, main="Diagnosis by compactness\_se", xlab="Diagnosis",

ylab="compactness\_se", col = c("blue","red"))

boxplot(concavity\_mean~diagnosis,data=breastc, main="Diagnosis by concavity\_mean", xlab="Diagnosis",

ylab="concavity\_mean", col = c("blue","red"))

#correlation between the variables

my\_data <- breastc[,c(1:31)]

head(my\_data,6)

res <-cor(my\_data)

round(res,3)

cor(my\_data, use="complete.obs")

#plot

corrplot(res)

#different correletion

sp.cor<-cor(my\_data, method="spearman")

sp.cor

#train and test data I

nobs=nrow(breastc)

nosample = round(nobs\*0.7)

train\_ind <-sample(nobs,size=nosample)

train <-breastc[train\_ind,]

test <- breastc[-train\_ind,]

#linear probability model I

reg1 <-lm(diagnosis~., data=train)

summary(reg1)

library(DescTools)

VIF(reg1)

str(breastc)

#many variables higth correlate to each other

#applying stepwise method data I

reg3 = step(lm(diagnosis ~ ., data=train))

summary(reg3)

VIF(reg3)

#radius\_worst still very correlated to the res of the variables

goodmodel <-"diagnosis ~ radius\_mean + perimeter\_mean + compactness\_mean +

`concave points\_mean` + smoothness\_se + concavity\_se + `concave points\_se` +

symmetry\_se + radius\_worst + texture\_worst + area\_worst +

concavity\_worst + fractal\_dimension\_worst"

reg4<-lm(goodmodel, data=breastc)

summary(reg4)

summary(reg3)

lm.beta.fullfit <- lm.beta(reg4)

lm.beta.fullfit

VIF(reg4)

#Logistic Model

logit1 = glm(goodmodel, data=train, family=binomial(link = logit))

summary(logit1)

confint(logit1)

confint.default(logit1)

tidy(logit1, conf.int=TRUE)

VIF(logit1)

PseudoR2(logit1)

#lets try to clean the data taking out all the hight correlated variables

#drop

newdata <- breastc[,c(1,3,6,10:11,13,16:21,26,30)]

str(newdata)

newdata1 <- breastc[,c(1:12)]

str(newdata1)

#train and test data for the new data

nobs=nrow(newdata)

nosample = round(nobs\*0.7)

train\_in <-sample(nobs,size=nosample)

train2 <-newdata[train\_in,]

test2 <- newdata[-train\_in,]

#apply to the new dataset

reg2 <-lm(diagnosis~., data=train2)

summary(reg2)

VIF(reg2)

goodmodel2 <- "diagnosis ~ texture\_mean + smoothness\_mean + symmetry\_mean + fractal\_dimension\_mean+ texture\_se +

smoothness\_se + compactness\_se + concavity\_se +`concave points\_se` + symmetry\_se +

fractal\_dimension\_se + smoothness\_worst + symmetry\_worst"

#logistic for new data

logit2 = glm(goodmodel2, data=train2, family=binomial(link = logit))

summary(logit2)

VIF(logit2)

PseudoR2(logit2)

tidy(logit2, conf.int=TRUE)

yhat<-predict (logit2, type ="response")

data1 <- cbind(newdata$diagnosis,yhat)

head(data1)

#marginal

install.packages("mfx")

library(mfx)

logitmfx(goodmodel2 , data=train2)

#prediction using logit model to new test data(out of sample forecastin )

prob1 <-predict(logit2, newdata=test2,type="response")

#decision of the breast cancer

result1 <-ifelse(prob1 >0.5,1,0)

#true cancer

test2diagnosis <-test2$diagnosis

table(test2diagnosis,result1)

#calculate error rate

misclassification <- mean(result1 !=test2diagnosis)

misclassification

# calculate accuracy rate

accuracy <- 1-misclassification

accuracy

install.packages("ROCR")

library(ROCR)

pred <-prediction(prob1, test2$diagnosis)

pmf <-performance(pred,measure="tpr", x.measure="fpr")

auc <-performance(pred,measure="auc")

auc <- auc@y.values[[1]]

auc <- round(auc,digits=6)

auc

plot(pmf,colorize=TRUE,main=paste("Receiver Operation characteristics ROC", auc),

print.cutoffs.at=seq(0,1,by=0.1), tex.adj=c(-0.2,1.7))

#applying the model to the entire data set

#prediction using logit model to new test data(out of sample forecastin )

prob2 <-predict(logit2, newdata=breastc,type="response")

#decision of the breast cancer

result2 <-ifelse(prob2 >0.5,1,0)

#true cancer

testEdiagnosis <-breastc$diagnosis

table(testEdiagnosis,result2)

#calculate error rate

misclassification2 <- mean(result2 !=testEdiagnosis)

misclassification2

# calculate accuracy rate

accuracy2 <- 1-misclassification2

accuracy2

install.packages("ROCR")

library(ROCR)

pred2 <-prediction(prob2, breastc$diagnosis)

pmf2 <-performance(pred,measure="tpr", x.measure="fpr")

auc2 <-performance(pred,measure="auc")

auc2 <- auc@y.values[[1]]

auc2 <- round(auc2,digits=6)

auc2

plot(pmf,colorize=TRUE,main=paste("Receiver Operation characteristics ROC", auc2),

print.cutoffs.at=seq(0,1,by=0.1), tex.adj=c(-0.2,1.7))