rm(list = ls())

Data = read.csv("Data.csv", stringsAsFactors = FALSE)

n = dim(Data)[1] # The number of rows of the data.frame

### ID is the 43th column in the dataset

# The code below generates a spread sheet with 101 predictors (X1 - X101)

# and the word corresponding to the names of predictors

# iters = c(5:7, 13:42, 44:dim(Data)[2])

# k = integer(1)

# for (i in iters) {

# k = k + 1

# names(Data)[i] = paste("X", k, sep = "")

# }

#write.csv(predictor.table, "predictor\_table.csv", row.names = FALSE)

predictors = Data[,-c(1:4,8:13,43)]

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

#Check the randomness of NA rows

# data\_na = Data[is.na(Data$X5),]

#

# boxplot(data\_na$stars)

# hist(data\_na$stars)

#

# freqs = table(data\_na$stars)

# null.probs = table(Data$stars)

#

# barplot(freqs)

# barplot(null.probs)

#

# chisq.test(freqs, null.probs) #Goodness of fit test

# if(!require(MissMech)) {

# install.packages("MissMech");

# require(MissMech)

# }

# MCAR\_test = TestMCARNormality(predictors, del.lesscases = 6, imputation.number = 1, method = "Auto",

# imputation.method = "Dist.Free", nrep = 100, n.min = 30, seed = 110, alpha = 0.05, imputed.data = NA)

#

# if(!require(BaylorEdPsych)) {

# install.packages("BaylorEdPsych");

# require(BaylorEdPsych)

# }

# if(!require(mvnmle)) {

# install.packages("mvnmle");

# require(mvnmle)

# }

# LittleMCAR(predictors)

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

#imputation

# #mice package########################################

# if(!require(mice)) {

# install.packages("mice");

# require(mice)

# }

# imputed\_mice\_ = mice(predictors, m = 3, maxit = 100)

# imputed\_mice1 = complete(imputed\_mice\_,1)

# imputed\_mice2 = complete(imputed\_mice\_,2)

# imputed\_mice3 = complete(imputed\_mice\_,3)

# sen\_score=rowMeans(cbind(imputed\_mice1$X5,imputed\_mice2$X5,imputed\_mice3$X5))

# imputed\_mice1$X5 = sen\_score

# imputed\_mice = imputed\_mice1

# write.csv(imputed\_mice, "imputed\_mice.csv", row.names=F)

# #imputed\_mice=read.csv("imputed\_mice.csv", stringsAsFactors = FALSE)

#missFOrest package################################################

# if(!require(missForest)) {

# install.packages("missForest");

# require(missForest)

# }

# imputed\_forest = missForest(predictors,maxiter = 10, ntree = 100, verbose = T)$ximp

# write.csv(imputed\_forest, "imputed\_missForest.csv", row.names=F)

imputed\_missForest=read.csv("imputed\_missForest.csv", stringsAsFactors = FALSE)

#mi package#######################################################

# if(!require(mi)) {

# install.packages("mi");

# require(mi)

# }

# imputed\_mi = mi(predictors, verbose = T)

# mi2stata(imputed\_mi, m=3, "imputed\_mi.csv")

# imputed\_mi = read.csv("imputed\_mi.csv", stringsAsFactors = FALSE)

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

#superLearner

# if(!require(SuperLearner)) {

# install.packages("SuperLearner"); require(SuperLearner)}

# if(!require(gam)) {

# install.packages("gam"); require(gam)}

# if(!require(gbm)) {

# install.packages("gbm"); require(gbm)}

# if(!require(randomForest)) {

# install.packages("randomForest"); require(randomForest)}

# if(!require(nnet)) {

# install.packages("nnet"); require(nnet)}

#

# if(!require(glmnet)) {

# install.packages("glmnet"); require(glmnet)}

# if(!require(polspline)) {

# install.packages("polspline"); require(polspline)}

#

# SL.library <- c("SL.glm", "SL.mean",

# "SL.randomForest", "SL.glmnet","SL.gam","SL.gbm","SL.nnet","SL.polymars","SL.step")

# set.seed(100)

# predictors=imputed\_missForest

# #here Gaussian() fits a continuous Y and binomial() fits a binary Y

# Yelpstars.SL <-SuperLearner(Y=Data$stars,X=predictors, SL.library=SL.library,

# family=gaussian(), method="method.NNLS", verbose=TRUE)

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

if(!require(leaps)) {

install.packages("leaps"); require(leaps)}

stars=Data$stars

predictors$X5 = imputed\_missForest$X5

Data\_use = cbind(stars,predictors)

# data\_mice = cbind(stars,imputed\_mice)

# data\_mi=cbind(stars,imputed\_mi)

#cross validation

subsets\_reg = regsubsets(stars~.+ X38\*X94\*X98 + X71:X75 + X94\*X52\*X36\*X75 + X94\*X33\*X77\*X95,

data=Data\_use, weights=NULL, nbest=3, nvmax=100, method = "seqrep", really.big = T)

parameters = subsets\_reg$xnames

s=summary(subsets\_reg)

order\_cp = order(s$cp)

order\_bic = order(s$bic)

order\_adjr2 = order(s$adjr2, decreasing = T)

if(!require(boot)) {

install.packages("boot"); require(boot)}

parameters[s$which[order\_adjr2[1]]]

parameters[s$which[order\_cp[1]]]

parameters[s$which[order\_bic[1]]]

cat(parameters[s$which[order\_adjr2[1]]], sep = " + ")

#cross validate#################################

if(!require(boot)) {

install.packages("boot"); require(boot)}

data\_inter = predictors

data\_inter$X38X94 = data\_inter$X38\*data\_inter$X94

data\_inter$X38X98 = data\_inter$X38\*data\_inter$X98

data\_inter$X94X98 = data\_inter$X94\*data\_inter$X98

data\_inter$X71X75 = data\_inter$X71\*data\_inter$X75

data\_inter$X52X94 = data\_inter$X52\*data\_inter$X94

data\_inter$X36X94 = data\_inter$X36\*data\_inter$X94

data\_inter$X36X52 = data\_inter$X36\*data\_inter$X52

data\_inter$X75X94 = data\_inter$X75\*data\_inter$X94

data\_inter$X52X75 = data\_inter$X52\*data\_inter$X75

data\_inter$X36X75 = data\_inter$X36\*data\_inter$X75

data\_inter$X33X94 = data\_inter$X33\*data\_inter$X94

data\_inter$X77X94 = data\_inter$X77\*data\_inter$X94

data\_inter$X33X77 = data\_inter$X33\*data\_inter$X77

data\_inter$X94X95 = data\_inter$X94\*data\_inter$X95

data\_inter$X33X95 = data\_inter$X33\*data\_inter$X95

data\_inter$X77X95 = data\_inter$X77\*data\_inter$X95

data\_inter$X38X94X98 = data\_inter$X38\*data\_inter$X94\*data\_inter$X98

data\_inter$X36X52X94 = data\_inter$X36\*data\_inter$X52\*data\_inter$X94

data\_inter$X52X75X94 = data\_inter$X52\*data\_inter$X75\*data\_inter$X94

data\_inter$X36X75X94 = data\_inter$X36\*data\_inter$X75\*data\_inter$X94

data\_inter$X36X52X75 = data\_inter$X36\*data\_inter$X52\*data\_inter$X75

data\_inter$X33X77X94 = data\_inter$X33\*data\_inter$X77\*data\_inter$X94

data\_inter$X33X94X95 = data\_inter$X33\*data\_inter$X94\*data\_inter$X95

data\_inter$X77X94X95 = data\_inter$X77\*data\_inter$X94\*data\_inter$X95

data\_inter$X33X77X95 = data\_inter$X33\*data\_inter$X77\*data\_inter$X95

data\_inter$X36X52X75X94= data\_inter$X36\*data\_inter$X52\*data\_inter$X75\*data\_inter$X94

data\_inter$X33X77X94X95= data\_inter$X33\*data\_inter$X77\*data\_inter$X94\*data\_inter$X95

#write.csv(data\_inter, "data\_inter.csv", row.names = F)

#data\_inter = read.csv("data\_inter.csv", stringsAsFactors = F)

for (i in 1:nrow(s$which)) {

model = s$which[i,][-1]

model[1]=T

subset = c(1:ncol(data\_inter))[model]

if (length(subset) > 1) {

Xw = data\_inter[,subset]

wlm = glm(stars ~ ., data = Xw)

}

else {

Xw = data\_inter[,subset[1]]

wlm = glm(stars ~ ., data = Xw)

}

s$CV[i] = cv.glm(model.frame(wlm), wlm, K=5)$delta[1]

}

order\_cv = order(s$CV)

parameters[s$which[order\_cv[1]]]

#penalized regression###################################################

if(!require(glmnet)) {

install.packages("glmnet"); require(glmnet)}

predictors = data\_inter[-1]

lasso.cv = cv.glmnet(predictors,stars,nfold=5)

plot(lasso.cv)

lasso.cv$lambda.min #minimizing lambda

coef(lasso.cv, s = "lambda.min") #coefficient estimate

predict(lasso.cv, newx = predictors,s = "lambda.min") #prediction