**STAT 4360 (Introduction to Statistical Learning, Spring 2023)**

**Mini Project 4  
Name: Ann Biju**

1g.

A close-up of black text

Description automatically generated

Linear Regression seems to have the smallest test MSE rate.

2g.

A group of black text

Description automatically generated

The Best Subset Selection model has one of the smallest test error rates, so I would recommend using it over the other classification models.

**Python Code (or R Code)**

# Question 1a

setwd("C:/ann/fall 2023/stat 4360/project 4")

# Load required libraries

library(caret)

library(boot)

library(MASS)

library(leaps)

library(glmnet)

# Read the wine dataset

wine <- read.delim("wine.txt", sep = "\t", header = TRUE)

linear\_regression\_model <- function(data) {

lm(Quality ~ ., data = data)

}

# Create a function to perform LOOCV

loocv <- function(data, model\_func) {

n <- nrow(data)

mse\_values <- numeric(n)

for (i in 1:n) {

# Split the data into training and test subsets

train\_data <- data[-i, ]

test\_data <- data[i, ]

# Fit the model on the training data

model <- model\_func(train\_data)

model

# Predict on the test data

predicted <- predict(model, test\_data)

# Calculate the MSE for the test point

mse\_values[i] <- mean((test\_data$Quality - predicted)ˆ2)

}

return(mean(mse\_values))

}

linear\_regression\_model

# Calculate LOOCV MSE for the linear regression model

loocv\_mse <- loocv(wine, linear\_regression\_model)

# Print the LOOCV MSE for linear regression

cat("LOOCV MSE for Linear Regression: ", loocv\_mse, "\n")

#QUESTION 1B

best\_subset\_selection <- function(data) {

pred <- colnames(data)[-1]

best\_model <- NULL

best\_adj\_r2 <- -Inf

for (k in 1:length(pred)) {

subsets <- regsubsets(Quality ~ ., data = data, nvmax = k)

summary\_subsets <- summary(subsets)

adj\_r2\_values <- summary\_subsets$adjr2

if (max(adj\_r2\_values) > best\_adj\_r2) {

best\_adj\_r2 <- max(adj\_r2\_values)

best\_model <- which(adj\_r2\_values == best\_adj\_r2)

}

}

return(best\_model)

}

# Find the best subset of predictors

best\_model\_indices <- best\_subset\_selection(wine)

# Extract the names of the best predictors

best\_predictors <- colnames(wine)[-1][best\_model\_indices]

# Fit the best model using the selected predictors

best\_model <- lm(Quality ~ ., data = wine[, c("Quality", best\_predictors)])

# Calculate LOOCV MSE for the best model using the selected predictors

loocv\_mse\_best\_model <- loocv(wine[, c("Quality", best\_predictors)], linear\_regression\_model)

loocv\_mse\_best\_model

#QUESTION 1C

forward\_stepwise\_selection <- function(data) {

all\_predictors <- colnames(data)[-1] # Exclude the response variable (Quality)

n <- nrow(data)

best\_predictors <- character(0)

best\_adj\_r2 <- -Inf

current\_model <- NULL

for (k in 1:length(all\_predictors)) {

remaining\_predictors <- setdiff(all\_predictors, best\_predictors)

adj\_r2\_values <- numeric(length(remaining\_predictors))

for (i in 1:length(remaining\_predictors)) {

candidate\_predictors <- c(best\_predictors, remaining\_predictors[i])

model <- lm(Quality ~ ., data = data[, c("Quality", candidate\_predictors)])

adj\_r2\_values[i] <- summary(model)$adj.r.squared

}

best\_candidate <- which.max(adj\_r2\_values)

if (adj\_r2\_values[best\_candidate] <= best\_adj\_r2) {

break

}

best\_adj\_r2 <- adj\_r2\_values[best\_candidate]

best\_predictors <- c(best\_predictors, remaining\_predictors[best\_candidate])

current\_model <- lm(Quality ~ ., data = data[, c("Quality", best\_predictors)])

}

return(current\_model)

}

# Find the best model using forward stepwise selection

best\_forward\_stepwise\_model <- forward\_stepwise\_selection(wine)

best\_forward\_stepwise\_model

loocv\_mse\_best\_forward\_stepwise\_model <- loocv(wine[, c("Quality", best\_predictors)], linear\_regression)

loocv\_mse\_best\_forward\_stepwise\_model

#QUESTION 1D

backward\_stepwise\_selection <- function(data) {

all\_predictors <- colnames(data)[-1] # Exclude the response variable (Quality)

best\_predictors <- all\_predictors

current\_model <- lm(Quality ~ ., data = data)

best\_adj\_r2 <- summary(current\_model)$adj.r.squared

while (length(best\_predictors) > 1) {

adj\_r2\_values <- numeric(length(best\_predictors))

for (i in 1:length(best\_predictors)) {

predictors\_to\_remove <- setdiff(best\_predictors, best\_predictors[i])

4

model <- lm(Quality ~ ., data = data[, c("Quality", predictors\_to\_remove)])

adj\_r2\_values[i] <- summary(model)$adj.r.squared

}

best\_candidate <- which.max(adj\_r2\_values)

if (adj\_r2\_values[best\_candidate] <= best\_adj\_r2) {

break

}

best\_adj\_r2 <- adj\_r2\_values[best\_candidate]

best\_predictors <- setdiff(best\_predictors, best\_predictors[best\_candidate])

current\_model <- lm(Quality ~ ., data = data[, c("Quality", best\_predictors)])

}

return(current\_model)

}

# Compute the test MSE for the best model using the selected predictors

loocv\_mse\_best\_backward\_stepwise\_model <- loocv(wine[, c("Quality", best\_predictors)], linear\_regression)

#QUESTION 1E

X <- as.matrix(wine[, -1]) # Predictors

Y <- wine$Quality # Response variable

# Create a function to perform LOOCV with Ridge regression

ridge\_regression\_loocv <- function(X, Y) {

n <- length(Y)

mse\_values <- numeric(n)

for (i in 1:n) {

# Create training and test data for LOOCV

X\_train <- X[-i, ]

Y\_train <- Y[-i]

X\_test <- X[i, , drop = FALSE]

Y\_test <- Y[i]

# Fit Ridge regression with different lambda values

lambdas <- 10ˆseq(-6, 6, length = 100)

ridge\_fit <- cv.glmnet(X\_train, Y\_train, alpha = 0, lambda = lambdas)

# Find the lambda with the minimum cross-validated MSE

best\_lambda <- ridge\_fit$lambda.min

# Fit the Ridge regression model with the best lambda on the entire training set

ridge\_model <- glmnet(X\_train, Y\_train, alpha = 0, lambda = best\_lambda)

# Predict on the test data

Y\_pred <- predict(ridge\_model, s = best\_lambda, newx = X\_test)

# Calculate the MSE for the test point

mse\_values[i] <- mean((Y\_test - Y\_pred)ˆ2)

}

return(mean(mse\_values))

}

# Calculate LOOCV MSE with Ridge regression

loocv\_mse\_ridge <- ridge\_regression\_loocv(X, Y)

loocv\_mse\_ridge

#QUESTION 1f

lasso\_regression\_loocv <- function(X, Y) {

n <- length(Y)

mse\_values <- numeric(n)

for (i in 1:n) {

# Create training and test data for LOOCV

X\_train <- X[-i, ]

Y\_train <- Y[-i]

X\_test <- X[i, , drop = FALSE]

Y\_test <- Y[i]

# Fit Lasso regression with different lambda values

lambdas <- 10ˆseq(-6, 6, length = 100)

lasso\_fit <- cv.glmnet(X\_train, Y\_train, alpha = 1, lambda = lambdas)

# Find the lambda with the minimum cross-validated MSE

best\_lambda <- lasso\_fit$lambda.min

# Fit the Lasso regression model with the best lambda on the entire training set

lasso\_model <- glmnet(X\_train, Y\_train, alpha = 1, lambda = best\_lambda)

# Predict on the test data

Y\_pred <- predict(lasso\_model, s = best\_lambda, newx = X\_test)

# Calculate the MSE for the test point

mse\_values[i] <- mean((Y\_test - Y\_pred)ˆ2)

}

return(mean(mse\_values))

}

# Calculate LOOCV MSE with Lasso regression

loocv\_mse\_lasso <- lasso\_regression\_loocv(X, Y)

#QUESTION 1g

# Create a data frame to store the test MSEs

test\_mse\_df <- data.frame(

Model = c("Linear Regression", "Best Subset Selection", "Forward Stepwise Selection"),

Test\_MSE = c(loocv\_mse, loocv\_mse\_best\_model, loocv\_mse\_best\_forward\_stepwise\_model)

)

# Print the summary

print(test\_mse\_df)

# QUESTION 2a

library(readr)

library(caret)

library(bestglm)

data <- read\_csv('diabetes.csv')

splitIndex <- createDataPartition(data$Outcome, p = 0.7, list = FALSE, times = 1)

train\_data <- data[splitIndex, ]

test\_data <- data[-splitIndex, ]

# Now, create a data frame 'train\_data' with both X and y

train\_data <- (train\_data)[,-8]

test\_data <- (test\_data)[,-8]

logistic\_model\_a <- glm(train\_data$Outcome ~ ., data = train\_data, family = binomial)

y\_pred\_a <- predict(logistic\_model\_a, newdata = test\_data, type = "response")

test\_error\_a <- 1 - sum((y\_pred\_a >= 0.5) == test\_data$Outcome) / nrow(test\_data)

cat("Test error (Log - All Predictors):", test\_error\_a, "\n")

# QUESTION 2B

best\_model\_b

logistic\_model\_b <- glm(train\_data$Outcome ~ ., data =

(train\_data)[,-train\_data$SkinThickness], family = binomial)

y\_pred\_b <- predict(logistic\_model\_b, newdata = test\_data, type = "response")

test\_error\_b <- 1 - sum((y\_pred\_b >= 0.5) == test\_data$Outcome) / nrow(test\_data)

cat("Test error (Best-Subset Selection - AIC):", test\_error\_b, "\n")

# QUESTION 2C

best\_model\_c <- bestglm(train\_data, IC = "AIC", family = binomial, method = "exhaustive")

best\_model\_c

logistic\_model\_c <- glm(train\_data$Outcome ~ ., data = train\_data[, -train\_data$SkinThickness], family =y\_pred\_c <- predict(logistic\_model\_c, newdata = test\_data, type = "response")

test\_error\_c <- 1 - sum((y\_pred\_c >= 0.5) == test\_data$Outcome) / nrow(test\_data)

cat("Test error (Forward Stepwise Selection - AIC):", test\_error\_c, "\n")

#QUESTION 2d

best\_model\_d <- bestglm(train\_data, IC = "AIC", family = binomial, method = "exhaustive")

logistic\_model\_d <- glm(train\_data$Outcome ~ ., train\_data[, -train\_data$SkinThickness], family = binomy\_pred\_d <- predict(logistic\_model\_d, newdata = test\_data, type = "response")

test\_error\_d <- 1 - sum((y\_pred\_d >= 0.5) == test\_data$Outcome) / nrow(test\_data)

cat("Test error (Backward Stepwise Selection - AIC):", test\_error\_d, "\n")

# QUESTION 2e

alpha\_values <- 0.1

lambda\_values <- 10ˆseq(10, -2, by = -1)

# Prepare the data

X\_train <- as.matrix(train\_data[, -8]) # Exclude the Outcome column

Y\_train <- train\_data$Outcome

# Fit a ridge classification model with cross-validated lambda

ridge\_model\_e <- cv.glmnet(X\_train, Y\_train, alpha = alpha\_values, lambda = lambda\_values)

# Find the best lambda from cross-validation

best\_lambda <- ridge\_model\_e$lambda.min

# Fit the final ridge classification model with the best lambda

final\_ridge\_model <- glmnet(X\_train, Y\_train, alpha = alpha\_values, lambda = best\_lambda)

# Predict on the test data

X\_test <- as.matrix(test\_data[, -8]) # Exclude the Outcome column

Y\_test <- test\_data$Outcome

y\_pred\_e <- predict(final\_ridge\_model, s = best\_lambda, newx = X\_test, type = "response")

# Calculate the test error rate

test\_error\_e <- 1 - sum((y\_pred\_e >= 0.5) == Y\_test) / length(Y\_test)

cat("Test error (Ridge classification):", test\_error\_e, "\n")

# QUESTION 2f

alpha\_values <- 0.1

lambda\_values <- 10ˆseq(10, -2, by = -1)

# Prepare the data

X\_train <- as.matrix(train\_data[, -8]) # Exclude the Outcome column

Y\_train <- train\_data$Outcome

# Fit a Lasso classification model with cross-validated lambda

lasso\_model\_f <- cv.glmnet(X\_train, Y\_train, alpha = alpha\_values, lambda = lambda\_values, family = "binomial")

# Find the best lambda from cross-validation

best\_lambda <- lasso\_model\_f$lambda

# Fit the final Lasso classification model with the best lambda

final\_lasso\_model <- glmnet(X\_train, Y\_train, alpha = alpha\_values, lambda = best\_lambda, family = "binomial")

# Predict on the test data

X\_test <- as.matrix(test\_data[, -8]) # Exclude the Outcome column

Y\_test <- test\_data$Outcome

y\_pred\_f <- predict(final\_lasso\_model, s = best\_lambda, newx = X\_test, type = "response")

# Calculate the test error rate

test\_error\_f <- 1 - sum((y\_pred\_f >= 0.5) == Y\_test) / length(Y\_test)

cat("Test error (Lasso classification):", test\_error\_f, "\n")

# QUESTION 2g

summary\_data <- data.frame(

Method = c('Best-Subset Selection (AIC)', 'Forward Stepwise Selection (AIC)',

'Backward Stepwise Selection (AIC)'),

Test\_Error\_Rate = c(test\_error\_b, test\_error\_c, test\_error\_d

)

)

print(summary\_data)