Project: Resting Heartrate & Myocardial infarction prediction

After importing the data (HR.csv) into R, the following code was used to fit a linear regression predicting RestingHR with predictor variables, postExerciseHR and Exercise.

Using 10-fold CV to find cross-validated mean squared error (MSE):

Linear Regression

```
library(caret)
set.seed(1510)

q1a.trC <- trainControl(

  method = "cv",     #just 1 CV, 10-fold

  number = 10)

model1 <- train(RestingHR ~ postExerciseHR + Exercise,

                data = Heartratedata,

                method = "lm",

                trControl = q1a.trC)

model1

#RMSE = 7.17

model2 <- train(RestingHR ~ postExerciseHR + as.factor(Exercise),

                data = Heartratedata,

                method = "lm",

                trControl = q1a.trC)

model2

#RMSE = 7.22
```

From the MSE shown, I would choose model 1. The difference in MSE is due to Exercise being a qualitative variable therefore, when coding it as a factor variable, the results differ as R treats the variable with either a linear effect or a separate effect depending on its category.

‘Exercise’ is also an ordinal categorical variable so a linear effect is applicable and recoding it into a factor would not cause the MSE to differ too much.

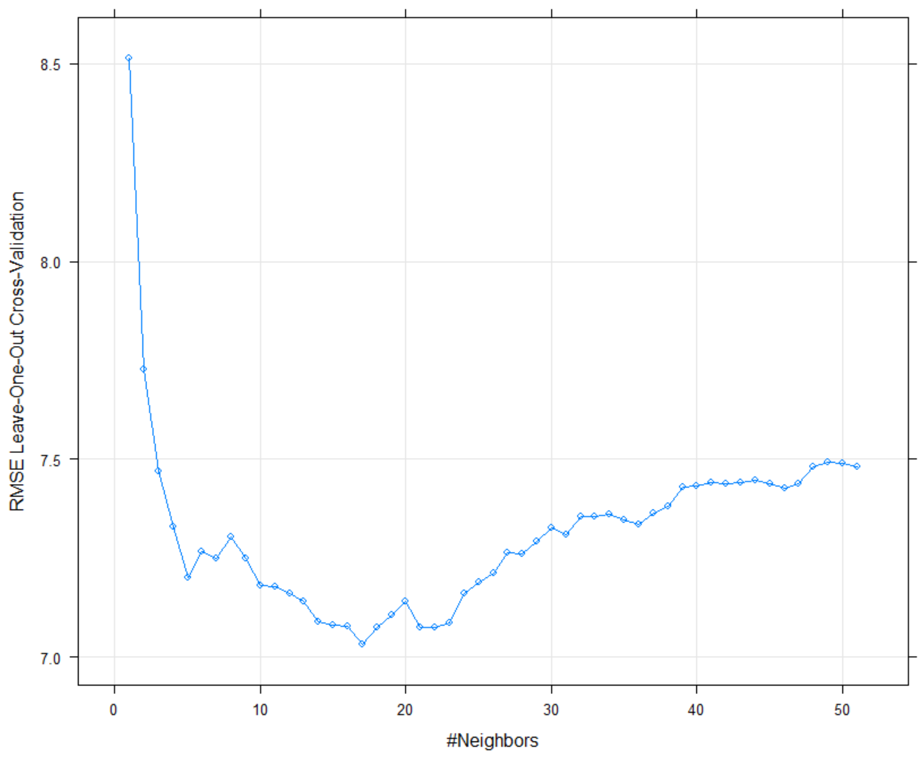
K-Nearest Neighbour Regression

To compute k-nearest neighbour regression with a leave-one-out MSE computed for each neighbour, from k = 1:51, we used the following code:

```
set.seed(1510)

HR.knn.loocv <- trainControl(

                method = "LOOCV")

model.knn <- train(RestingHR ~ postExerciseHR + Exercise,
                    data = Heartratedata, method= "knn",
                    trControl = HR.knn.loocv,
                    preProcess = c("center","scale"),
                    tuneGrid = expand.grid(k=1:51))
model.knn$results
model.knn$bestTune
#17
```

It provided the number 17 and this was confirmed by plotting the MSE for the entire sequence.

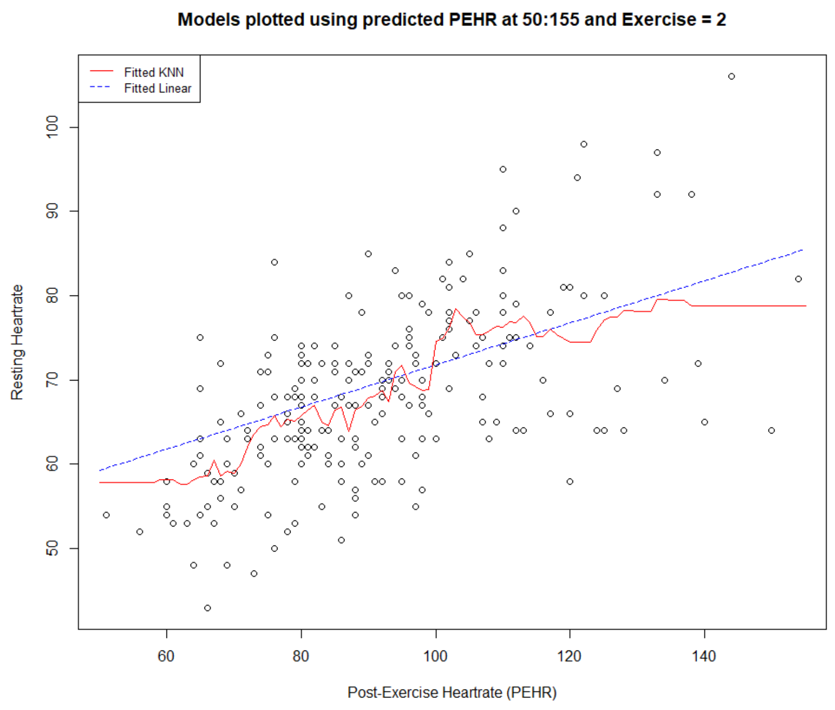
```
plot(model.knn)
```



Model Comparison

Now plotting a fitted knn where k = 17 and linear fitted model 1, using the variable values of postExerciseHR having a range from 50 to 155 and Exercise at category 2.

```
library(FNN)
HR.P <- seq(50,155)
knn17.datapred2 <- knn.reg(
  train = Heartratedata[c("postExerciseHR", "Exercise")],
  y = Heartratedata$RestingHR,

  test = data.frame(postExerciseHR=HR.P, Exercise=2),

  k = 17

)

model1.E2 <- predict(model1, newdata = data.frame(Exercise=2,
postExerciseHR=seq(50,155)))


plot(Heartratedata$postExerciseHR, Heartratedata$RestingHR, xlab = "Post-
Exercise Heartrate (PEHR)", ylab = "Resting Heartrate", main = "Models plotted
using predicted PEHR at 50:155 and Exercise = 2")

lines(HR.P,model1.E2, col="blue", lty=2, lwd=1.5)

lines(HR.P,knn17.datapred2$pred, col="red", lwd=1.5)

legend("topleft", legend = c("Fitted KNN", "Fitted Linear"),

       col = c("red", "blue"), lty = 1:2,lwd=1.5, cex = 0.8)
```



Imputation of missing entries

After importing the data (HR2.csv) into R, we compute the mean:

```
mean.PEHR <- round(mean(HRdatacomplete1$postExerciseHR, na.rm=TRUE),
2)

#91.27
```

Checking for missing data:

```
colSums(is.na(HRdatacomplete1[1:8]))
```

```
#postExercise is missing 37 entries.
```

Mean imputation;

```
HRdatacomplete1[is.na(HRdatacomplete1$postExerciseHR),
"postExerciseHR"] <- mean.PEHR

q2.model1 <- lm(RestingHR ~ postExerciseHR + Exercise, data =
HRdatacomplete1)

summary(q2.model1)
```

Using the mean imputation, we compute an adjusted-R2 of **0.4152**.

After reimporting the data (HR2.csv) into R as a separate dataset, we fit a linear model using the following code and form predictions using the model. (for **prediction imputation**)

```
p.model <- lm(postExerciseHR ~ RestingHR + Smoke + Sex + Exercise +
Hgt + Wgt, data = HRdatacomplete2)

PredictPEHR <- round(predict(p.model),2)
```

These predictions are then used to replace missing data in the “postExerciseHR” variable.

```
HRdatacomplete2$postExerciseHR <-
ifelse(is.na(HRdatacomplete2$postExerciseHR), PredictPEHR,
HRdatacomplete2$postExerciseHR)
```

Now using the “**complete model**”,

```
q2.model2 <- lm(RestingHR ~ postExerciseHR + Exercise, data =
HRdatacomplete2)

summary(q2.model2)
```

An adjusted-R2 value of **0.395** is obtained.

It can be noted that model 1 (mean imputation) has a higher adjusted-R2 than model 2 (prediction imputation). These differences can likely be attributed to model 2 missing observations being replaced by predictions that are far too precise, since the predictions come from a fitted model that uses all variables in the dataset, resulting in an overfitting of predicted values. Therefore, when considering this and how the adj-R2 for model 1 larger than model 2, it is likely that model 1 is closer to the truth.

Factor conversion to predict myocardial infarction.

When creating a logistic regression model to predict AMI, all other variables were used. Therefore before fitting a glm, all categorical predictor variables were converted from int to Factor variables.

```
#factor conversion

MIData$sex <- factor(MIData$sex, levels = c(0,1),

                      labels = c("Female", "Male"))

MIData$cpta <- factor(MIData$cpta, levels = c(0,1),

                     labels = c("No", "Yes"))
```

```
MIData$cpaa <- factor(MIData$cpaa, levels = c(0,1),

                      labels = c("No", "Yes"))

MIData$cpanp <- factor(MIData$cpanp, levels = c(0,1),

                      labels = c("No", "Yes"))

MIData$sugar <- factor(MIData$sugar, levels = c(0,1),

                       labels = c("No", "Yes"))

MIData$STT <- factor(MIData$STT, levels = c(0,1),

                       labels = c("No", "Yes"))

MIData$LVH <- factor(MIData$LVH, levels = c(0,1),

                     labels = c("No", "Yes"))

MIData$angina <- factor(MIData$angina, levels = c(0,1),

                     labels = c("No", "Yes"))

#vessels goes up to 4 not 3

MIData$vessels <- factor(MIData$vessels)

MIData$stress <- factor(MIData$stress)
```

Confusion Matrix

Running a logistic regression and using it to compute a confusion matrix using the following code,

```
q3.glm.model1.pred <- predict(q3.glm.model1,

                         type = "response")

glm.model1.AMI.class <- factor(ifelse(q3.glm.model1.pred >.5, "yes",
"no"))

ConfusionMat <- table(glm.model1.AMI.class, MIData$AMI)

ConfusionMat
```

This produces the following confusion matrix.

glm.model1.AMI.class	No	Yes
No	111	22
Yes	27	143

Accuracy computation using Confusion Matrix

Accuracy was computed using Caret,

```
#using Caret Package

Caret.ConfusionMat<- confusionMatrix(glm.model1.AMI.class, MIData$AMI)

Caret.ConfusionMat$overall[1]
```

```
#Accuracy computes to 0.84%
```

The accuracy was found to be **84%** and the respective 95% confidence interval was then generated for accuracy using the following code,

```
n.boot <- 1000

#1000 bootstraps

sample.size <- dim(MIData)[1]

accuracy.boot <- NULL


set.seed(1510)

for (i in 1:n.boot) {

  id.bs <- sample.int(sample.size,

                      sample.size,

                      replace = TRUE)

  model.bs <- glm(formula = AMI == "yes" ~ age + sex + cpta + cpaa + 
cpanp + bp + chol + sugar + STT + LVH + maxHR + angina + peak

                  + vessels + stress, family = binomial, data = 
MIData[id.bs, ])

  pred.model.bs <- predict(model.bs, type = "response")

  pred.death.bs <- factor(ifelse(pred.model.bs >.5, "yes", "no"))

  confusion.matrix.bs <- confusionMatrix(pred.death.bs, 
MIData[id.bs, ]$AMI)

  accuracy.boot[i] <- confusion.matrix.bs$overall[1] #Kappa bootstrap

}

quantile(accuracy.boot, c(0.025, 0.975))

#Accuracy 95% CI

2.5%     97.5%

0.8203795 0.9042904
```

The respective accuracy confidence intervals compute to **95% CI [0.82, 0.90]**.

It is expected that the accuracy will be lower than the result found from Q3b.

Computing accuracy,

```
set.seed(1510)

MItrc <- trainControl(

              method = "cv",

              number = 10,
              classProbs = TRUE,

              summaryFunction = twoClassSummary,

              savePred = TRUE)


model.10CV <- train(AMI ~ age + sex + cpta + cpaa + cpanp + bp + chol
+ sugar + STT + LVH + maxHR + angina + peak

                  + vessels + stress,

                  data = MIData, method = "glm",

                  family = "binomial",

                  trControl = MItrc,

                  metric = "ROC")

confusionMatrix(model.10CV)
```

10-fold accuracy computes to **81%**, which is lower than the non-cross validated accuracy (84%). The is likely due to the model overfitting on the original training data due to the number of variables in the model, thus when used on testing data via cross validation, it provides a lower accuracy.

Ridge Regression

Using the following code to fit a ridge regression:

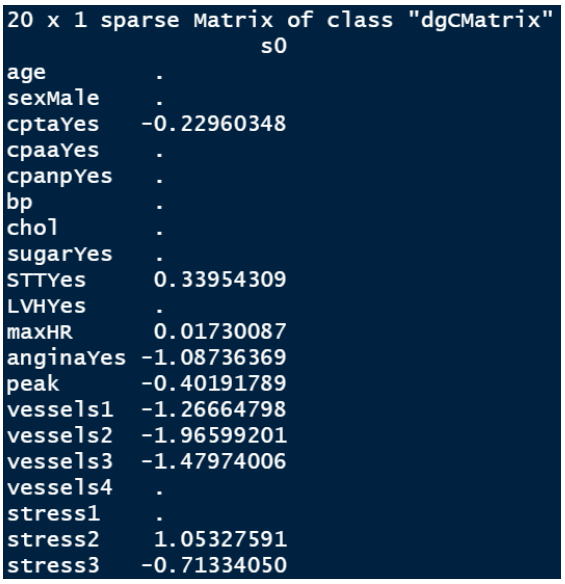
```
MItrc <- trainControl(

  method = "cv",

  number = 10,

  classProbs = TRUE,

  summaryFunction = twoClassSummary,

  savePred = TRUE)

MI.ridge.modelCV <- train(AMI ~ age + sex + cpta + cpaa + cpanp + bp +
chol + sugar + STT + LVH + maxHR + angina + peak

                          + vessels + stress,

                          data = MIData,

                          method = "glmnet",

                          family = "binomial",

                          trControl = MItrc,

                          metric = "ROC",

                          tuneGrid = MI.grid)

coef(MI.ridge.modelCV$finalModel,
     MI.ridge.modelCV$bestTune$lambda)

confusionMatrix(MI.ridge.modelCV)

Accuracy (average): 0.8218
```

The accuracy found (82%) is higher than the accuracy found in c) (81%). Since ridge regression is a method that reduces variance and works well with models with many predictors, this was expected.

Fitting the same logistic regression model with LASSO

```
MI.X <- model.matrix(AMI ~ age + sex + cpta + cpaa + cpanp + bp + chol
+ sugar + STT + LVH + maxHR + angina + peak

                     + vessels + stress, data=MIData)[,-1]

MI.Y <- MIData[,"AMI"]

#using alpha 1 since Lasso

set.seed(1510)

cv.lambdaL <- cv.glmnet(x=MI.X, y=MI.Y,

                       alpha = 1,

                       family = binomial)

plot(cv.lambdaL)

l.minL <- cv.lambdaL$lambda.min

l.minL

#l minL is 0.0146

set.seed(1510)

MI.lasso.model <- glmnet(x=MI.X, y=MI.Y,

                         alpha = 1,

                         family = binomial,

                         lambda = l.minL)

MI.lasso.model$beta
```

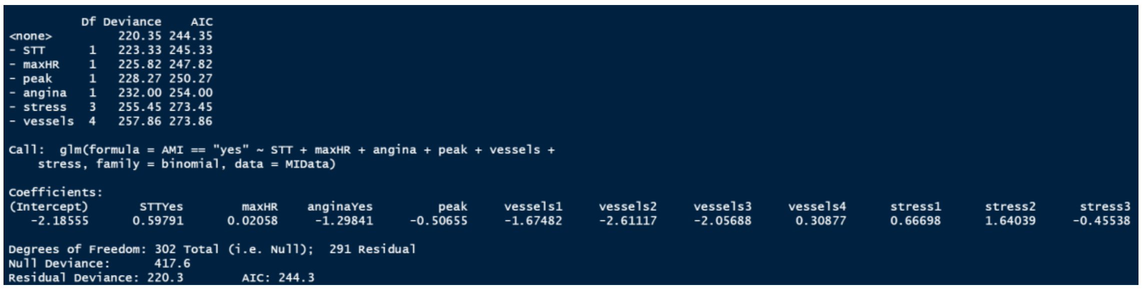
The LASSO model shows that cpta, STT, maxHR, angina, peak, vessels and stress were selected for the final model.



```
20 x 1 sparse Matrix of class "dgCMatrix"
                    s0
age          .
sexMale      .
cptaYes     -0.22960348
cpaaYes      .
cpanpYes     .
bp           .
chol         .
sugarYes     .
STTYes       0.33954309
LVHYes       .
maxHR        0.01730087
anginaYes   -1.08736369
peak        -0.40191789
vessels1    -1.26664798
vessels2    -1.96599201
vessels3    -1.47974006
vessels4     .
stress1      .
stress2      1.05327591
stress3     -0.71334050
```

When choosing variables using a **stepwise backward selection**,

```
library(bestglm)

library(leaps)

MI.bkwrd <- glm(AMI == "yes" ~ age + sex + cpta + cpaa + cpanp + bp +
chol + sugar + STT + LVH + maxHR + angina + peak

    + vessels + stress, family = binomial, data = MIData)

step(MI.bkwrd)
```



```
          Df Deviance    AIC
<none>         220.35 244.35
- STT      1   223.33 245.33
- maxHR    1   225.82 247.82
- peak     1   228.27 250.27
- angina   1   232.00 254.00
- stress   3   255.45 273.45
- vessels  4   257.86 273.86

Call:  glm(formula = AMI == "yes" ~ STT + maxHR + angina + peak + vessels + 
    stress, family = binomial, data = MIData)

Coefficients:
(Intercept)       STTYes        maxHR    anginaYes         peak     vessels1     vessels2     vessels3     vessels4      stress1      stress2      stress3  
   -2.18555      0.59791      0.02058     -1.29841     -0.50655     -1.67482     -2.61117     -2.05688      0.30877      0.66698      1.64039     -0.45538  

Degrees of Freedom: 302 Total (i.e. Null);  291 Residual
Null Deviance:	    417.6 
Residual Deviance: 220.3 	AIC: 244.3
```

The final model provided only has the variables: STT, maxHR, peak, angina, stress and vessels. When compared to the model provided by LASSO, the stepwise backward selected model excludes cpta.

APPENDIX (Full R Code)

```
library(psych)

library(FNN)

library(caret)




#Q1a

Heartratedata <- read.csv("C:/Users/Kieran/Desktop/Stats
Datasets/Semester 2 2022/Machine Learning/HR.csv",
stringsAsFactors=TRUE)


set.seed(1510)

q1a.trC <- trainControl(

  method = "cv",     #just 1 CV, 10-fold

  number = 10)


model1 <- train(RestingHR ~ postExerciseHR + Exercise,

                data = Heartratedata,

                method = "lm",

                trControl = q1a.trC)

model1


#RSME = 7.17




model2 <- train(RestingHR ~ postExerciseHR + as.factor(Exercise),

                data = Heartratedata,

                method = "lm",

                trControl = q1a.trC)

model2

#RSME = 7.22


summary(model1$finalModel)
```

```
#Better predictive ability from model2.

#This is due to Exercise being a qualitative variable, but the reason
why

#the difference between the MSE's is due to how there


Heartratedata$Exercise2 <- as.factor(Heartratedata$Exercise)


#Q1b

set.seed(1510)

HR.knn.loocv <- trainControl(

                method = "LOOCV")

model.knn <- train(RestingHR ~ postExerciseHR + Exercise,

                   data = Heartratedata, method= "knn",

                   trControl = HR.knn.loocv,

                   preProcess = c("center","scale"),

                   tuneGrid = expand.grid(k=1:51))

model.knn$results

model.knn$bestTune

#17

plot(model.knn)

#17 has the lowest RMSE


#Q1c do this lol


HR.P <- seq(50,155)

knn17.datapred1 <- knn.reg(

  train = Heartratedata[c("postExerciseHR", "Exercise")],

  y = Heartratedata$RestingHR,

  test = data.frame(postExerciseHR=HR.P, Exercise=1),

  k = 17

)
```

```
knn17.datapred2 <- knn.reg(
  train = Heartratedata[c("postExerciseHR", "Exercise")],
  y = Heartratedata$RestingHR,
  test = data.frame(postExerciseHR=HR.P, Exercise=2),
  k = 17
)

knn17.datapred3 <- knn.reg(
  train = Heartratedata[c("postExerciseHR", "Exercise")],
  y = Heartratedata$RestingHR,
  test = data.frame(postExerciseHR=HR.P, Exercise=3),
  k = 17
)

summary(Heartratedata$postExerciseHR)

model1.E1 <- predict(model1, newdata = data.frame(Exercise=1,
postExerciseHR=seq(50,155)))
model1.E2 <- predict(model1, newdata = data.frame(Exercise=2,
postExerciseHR=seq(50,155)))
model1.E3 <- predict(model1, newdata = data.frame(Exercise=3,
postExerciseHR=seq(50,155)))

#Only using Exercise = 2, range 50-155
plot(Heartratedata$postExerciseHR, Heartratedata$RestingHR, xlab =
"Post-Exercise Heartrate (PEHR)", ylab = "Resting Heartrate", main =
"Models plotted using predicted PEHR at 50:155 and Exercise = 2")
lines(HR.P,model1.E2, col="blue", lty=2, lwd=1.5)
lines(HR.P,knn17.datapred2$pred, col="red", lwd=1.5)
legend("topleft", legend = c("Fitted KNN", "Fitted Linear"),
       col = c("red", "blue"), lty = 1:2,lwd=1.5, cex = 0.8)


#Using All
```

```r
plot(Heartratedata$postExerciseHR, Heartratedata$RestingHR)

lines(HR.P,model1.E1, col="blue", lty=2)

lines(HR.P,model1.E2, col="green", lty=2)

lines(HR.P,model1.E3, col="red", lty=2)

lines(HR.P,knn17.datapred1$pred, col="blue")

lines(HR.P,knn17.datapred2$pred, col="green")

lines(HR.P,knn17.datapred3$pred, col="red")


#Q2 ----------------------------------------------------------------

#Q2a

HRdatacomplete1 <- read.csv("C:/Users/Kieran/Desktop/Stats
Datasets/Semester 2 2022/Machine Learning/HR2.csv",
stringsAsFactors=TRUE)


mean.PEHR <- round(mean(HRdatacomplete1$postExerciseHR, na.rm=TRUE),
2)

#91.27

colSums(is.na(HRdatacomplete1[1:8]))


HRdatacomplete1[is.na(HRdatacomplete1$postExerciseHR),
"postExerciseHR"] <- mean.PEHR


q2.model1 <- lm(RestingHR ~ postExerciseHR + Exercise, data =
HRdatacomplete1)

summary(q2.model1)

#0.415 adj R


#Q2b

HRdatacomplete2 <- read.csv("C:/Users/Kieran/Desktop/Stats
Datasets/Semester 2 2022/Machine Learning/HR2.csv",
stringsAsFactors=TRUE)


p.model <- lm(postExerciseHR ~ RestingHR + Smoke + Sex + Exercise +
Hgt + Wgt, data = HRdatacomplete2)

summary(p.model)
```

```
PredictPEHR <- round(predict(p.model),2)

HRdatacomplete2$postExerciseHR <-
ifelse(is.na(HRdatacomplete2$postExerciseHR), PredictPEHR,
HRdatacomplete2$postExerciseHR)


#now using "Complete model"

q2.model2 <- lm(RestingHR ~ postExerciseHR + Exercise, data =
HRdatacomplete2)

summary(q2.model2)


#0.395 adj R


#Q2c

#Using mean imputation seems to have provided a higher adjusted R2
which is

#indicative of a better model. The predicted values may be too precise
in fitting the incomplete data and

#thus these fitted values may be overestimating or underestimating the
imputed values.  Thus, I believe that

# the former model is closer to the truth.

#----------------------------------------------------------------------
-------------------------------------------



#Q3a

MIData <- read.csv("C:/Users/Kieran/Desktop/Stats Datasets/Semester 2
2022/Machine Learning/heart_assignment1-1.csv", stringsAsFactors=TRUE)


#factor conversion

MIData$sex <- factor(MIData$sex, levels = c(0,1),

                      labels = c("Female", "Male"))


MIData$cpta <- factor(MIData$cpta, levels = c(0,1),

                     labels = c("No", "Yes"))


MIData$cpaa <- factor(MIData$cpaa, levels = c(0,1),
                      labels = c("No", "Yes"))

MIData$cpanp <- factor(MIData$cpanp, levels = c(0,1),
                      labels = c("No", "Yes"))

MIData$sugar <- factor(MIData$sugar, levels = c(0,1),
                       labels = c("No", "Yes"))

MIData$STT <- factor(MIData$STT, levels = c(0,1),
                       labels = c("No", "Yes"))

MIData$LVH <- factor(MIData$LVH, levels = c(0,1),
                     labels = c("No", "Yes"))

MIData$angina <- factor(MIData$angina, levels = c(0,1),
                     labels = c("No", "Yes"))

#vessels goes up to 4 not 3
MIData$vessels <- factor(MIData$vessels)

MIData$stress <- factor(MIData$stress)

#----------------------------------------------------------------------
--------------------------------------------


q3.glm.model1 <- glm(AMI == "yes" ~ age + sex + cpta + cpaa + cpanp +
bp + chol + sugar + STT + LVH + maxHR + angina + peak
                    + vessels + stress, family = binomial, data =
MIData)

summary(q3.glm.model1)
mean(q3.glm.model1$residuals^2)
```

```
q3.glm.model1.pred <- predict(q3.glm.model1,
                         type = "response")
glm.model1.AMI.class <- factor(ifelse(q3.glm.model1.pred >.5, "yes",
"no"))

ConfusionMat <- table(glm.model1.AMI.class, MIData$AMI)
ConfusionMat

#Q3b-----------

#using Caret Package
Caret.ConfusionMat<- confusionMatrix(glm.model1.AMI.class, MIData$AMI)
Caret.ConfusionMat$overall[1]

#Accuracy computes to 0.84% according to caret package

n.boot <- 1000
sample.size <- dim(MIData)[1]
accuracy.boot <- NULL

set.seed(1510)
for (i in 1:n.boot) {
  id.bs <- sample.int(sample.size,
                      sample.size,
                      replace = TRUE)
  model.bs <- glm(formula = AMI == "yes" ~ age + sex + cpta + cpaa +
cpanp + bp + chol + sugar + STT + LVH + maxHR + angina + peak
                  + vessels + stress, family = binomial, data =
MIData[id.bs, ])
  pred.model.bs <- predict(model.bs, type = "response")
  pred.death.bs <- factor(ifelse(pred.model.bs >.5, "yes", "no"))
  confusion.matrix.bs <- confusionMatrix(pred.death.bs,
MIData[id.bs, ]$AMI)
  accuracy.boot[i] <- confusion.matrix.bs$overall[1] #Kappa bootstrap
}

quantile(accuracy.boot, c(0.025, 0.975))


#using seed 1510,

#accuracy returns lower interval, 0.82, upper interval 0.90




#Q3c--------------------------------------------------------------------
-----------------------------------------------------------

#expected to be lower accuracy as the model may be overfitting due to
number of variables therefore when applied to other

#test sets of data, it may not be as accurate.

set.seed(1510)

MItrc <- trainControl(

              method = "cv",

              number = 10,

              classProbs = TRUE,

              summaryFunction = twoClassSummary,

              savePred = TRUE)


model.10CV <- train(AMI ~ age + sex + cpta + cpaa + cpanp + bp + chol
+ sugar + STT + LVH + maxHR + angina + peak

                    + vessels + stress,

                    data = MIData, method = "glm",

                    family = "binomial",

                    trControl = MItrc,

                    metric = "ROC")

confusionMatrix(model.10CV)


#accuracy 0.807

#expected outcome.
```

```
#Q3d---------------------------------------------------------------------
---------------------------------------------------------

#Using ridge regression

library(glmnet)

set.seed(1510)




MI.X <- model.matrix(AMI ~ age + sex + cpta + cpaa + cpanp + bp + chol
+ sugar + STT + LVH + maxHR + angina + peak

                     + vessels + stress, data=MIData)[,-1]

MI.Y <- MIData[,"AMI"]


set.seed(1510)

cv.lambda <- cv.glmnet(x=MI.X, y=MI.Y,

                       alpha = 0,

                       family = binomial)


plot(cv.lambda)

l.min <- cv.lambda$lambda.min

#lambda min = 0.04589


MI.ridge.model <- glmnet(x=MI.X, y=MI.Y,

                      alpha = 0,

                      family = binomial,

                      lambda = l.min)

MI.ridge.model$beta

summary(MI.ridge.model)


q3.rr.model.pred <- predict(MI.ridge.model,

                              type = "response")


#10 fold CV accuracy using alpha 0 and l min
```

```
set.seed(1510)

MI.grid <- expand.grid(alpha = 0,

                       lambda = seq(0,0.1,.0001))

#using MItrc again

MItrc <- trainControl(

  method = "cv",

  number = 10,

  classProbs = TRUE,

  summaryFunction = twoClassSummary,

  savePred = TRUE)


MI.ridge.modelCV <- train(AMI ~ age + sex + cpta + cpaa + cpanp + bp +
chol + sugar + STT + LVH + maxHR + angina + peak

                          + vessels + stress,

                          data = MIData,

                          method = "glmnet",

                          family = "binomial",

                          trControl = MItrc,

                          metric = "ROC",

                          tuneGrid = MI.grid)


coef(MI.ridge.modelCV$finalModel,

     MI.ridge.modelCV$bestTune$lambda)


confusionMatrix(MI.ridge.modelCV)


#Accuracy of 0.8152, higher than previous fit. This could be due to
overfitting as ridge regression trades variance for bias.


#Q3d Lasso fit

library(glmnet)


MI.X <- model.matrix(AMI ~ age + sex + cpta + cpaa + cpanp + bp + chol
+ sugar + STT + LVH + maxHR + angina + peak
```

```
                     + vessels + stress, data=MIData)[,-1]
MI.Y <- MIData[,"AMI"]

#using alpha 1 since Lasso
set.seed(1510)
cv.lambdaL <- cv.glmnet(x=MI.X, y=MI.Y,
                       alpha = 1,
                       family = binomial)
plot(cv.lambdaL)
l.minL <- cv.lambdaL$lambda.min
l.minL
#l minL is 0.0146
set.seed(1510)
MI.lasso.model <- glmnet(x=MI.X, y=MI.Y,
                         alpha = 1,
                         family = binomial,
                         lambda = l.minL)
MI.lasso.model$beta
#Variables chosen are cpta, STT, maxHR, Angina, peak, Vessels, stress

#stepwise backward selection
library(bestglm)
library(leaps)

MI.bkwrd <- glm(AMI == "yes" ~ age + sex + cpta + cpaa + cpanp + bp +
chol + sugar + STT + LVH + maxHR + angina + peak
    + vessels + stress, family = binomial, data = MIData)
step(MI.bkwrd)
#variables chosen are STT, maxHR, Angina, peak, Vessels, stress

#the variables selected are different. (cpta)
```