1) Models used:
   - Least squares: fit.ls = lm(Y ~ ., data = data.train)
   - Stepwise
     ```
     fit.start = lm(Y ~ 1, data = data.train)
     fit.end = lm(Y ~ .^2, data = data.train)
     step.BIC = step(fit.start, list(upper = fit.end), k = log(n.train), trace = 0)
     ```
   - Ridge
     ```
     lambda.vals = seq(from = 0, to = 100, by = 0.05)
     fit.ridge = lm.ridge(Y ~ ., lambda = lambda.vals, data = data.train)
     ```
   - LASSO: all.LASSOs = cv.glmnet(x = matrix.train, y = Y.train)
   - GAM:
     ```
     fit.gam = gam(data=data.train,
                   formula = Y ~ s(X1) + s(X2) + s(X3) + X4 + s(X5) + s(X6) +
                     s(X7) + s(X8) + s(X9) + X10 + s(X11) + X12 + s(X13) + s(X14) + s(X15),
                   family = gaussian(link=identity))
     ```
   - PPR:
     ```
     fit.ppr.best = ppr(Y ~ ., data = data.train,
                        max.terms = 15, nterms = best.terms, sm.method = "gcvspline")
     ```
   - Regression trees:
     - fit.tree = rpart(Y ~., data = data.train, cp = 0, method = "anova")

2) I used a 10-fold Cross validation on the data. I first split the “Data2020.csv” into a training set and a validation set. For each process, I then calculated the MSPE for the 10 folds. I then chose the model with minimum MSPE and used that on the “Data2020testX.csv”.

3) Tuning:
   - Ridge&LASSO:
     - used lambda values: lambda.vals = seq(from = 0, to = 100, by = 0.05)
   - PLS:
     - For optimal number of components:
       ```
       CV.pls = fit.pls$validation # Info from training data
       PRESS.pls = CV.pls$PRESS    # Sum of squared CV residuals
       CV.MSPE.pls = PRESS.pls / nrow(data.train)  # MSPE for internal CV
        ind.best.pls = which.min(CV.MSPE.pls) # Optimal # of components
       ```
   - Regression trees:
     - For pruning min and se
       ```
       ind.min = which.min(info.tree[,"xerror"])
        CP.min.raw = info.tree[ind.min, "CP"]
        if(ind.min == 1){
              ### If minimum CP is in row 1, store this value
                      CP.min = CP.min.raw
       } else{
             CP.above = info.tree[ind.min-1, "CP"]

                ### (Geometric) average
         CP.min = sqrt(CP.min.raw * CP.above)
           }
       ```

4)

```
library(pls)
library(mgcv)
library(rpart)
data_raw <- read.csv("Data2020.csv")
test_raw <- read.csv("Data2020testX.csv")
data <- na.omit(data_raw[, c(1:16)])
test <- na.omit(test_raw, c(1:15))

source("Helper Functions (1).R")
set.seed(2928893)


get.folds = function(n, K) {
  ### Get the appropriate number of fold labels
  n.fold = ceiling(n / K) # Number of observations per fold (rounded up)
  fold.ids.raw = rep(1:K, times = n.fold) # Generate extra labels
  fold.ids = fold.ids.raw[1:n] # Keep only the correct number of labels

  ### Shuffle the fold labels
  folds.rand = fold.ids[sample.int(n)]

  return(folds.rand)
}
### Number of folds
K = 10

### Construct folds
n = nrow(data) # Sample size
folds = get.folds(n, K)

all.models = c("GAM")
all.MSPEs = array(0, dim = c(K, length(all.models)))
colnames(all.MSPEs) = all.models

for(i in 1:K){
  data.train = data[folds != i,]
  data.valid = data[folds == i,]
  n.train = nrow(data.train)
  X.train = data.train[, -1]
  X.valid = data.valid[, -1]
  ### Get response vectors
  Y.train = data.train$Y
  Y.valid = data.valid$Y
  fit.gam = gam(data=data.train,
          formula = Y ~ s(X1) + s(X2) + s(X3) + X4 + s(X5) + s(X6) +
               s(X7) + s(X8) + s(X9) + X10 + s(X11) + X12 + s(X13) + s(X14) + s(X15),
              family = gaussian(link=identity))
  summary(fit.gam)
  pred.gam = predict(fit.gam, data.valid)
  MSPE.gam = get.MSPE(Y.valid, pred.gam)
  all.MSPEs[i, "GAM"] = MSPE.gam
}

fit.gam = gam(data=data,
            formula = Y ~ s(X1) + s(X2) + s(X3) + X4 + s(X5) + s(X6) +
              s(X7) + s(X8) + s(X9) + X10 + s(X11) + X12 + s(X13) + s(X14) + s(X15),
            family = gaussian(link=identity))
summary(fit.gam)
pred.gam = predict(fit.gam, test)
#write.table(pred.gam, "/Users/dollina/Desktop/Submit.csv", sep = ",",row.names =
FALSE, col.names=FALSE)
```

5) Variables that are important: evaluated this used LASSO and BIC and then tested on PCA.
    - X2
    - X4
    - X10
    - X12