1.

   -
     ```
     fit.ls = lm(Y.train ~ ., data=cbind(Y.train, X.train))
     ```

   -
     ```
     fit.start = lm(Y.train ~ 1, data = cbind(Y.train, X.train))
     fit.end = lm(Y.train ~ ., data = cbind(Y.train, X.train))
     step.def = step(fit.start, list(upper = fit.end), trace=0)
     ```

   -
     ```
     lambda.vals = seq(from = 0, to = 100, by = 0.05)
     fit.ridge = lm.ridge(Y.train ~ ., lambda = lambda.vals, data
     = cbind(Y.train, X.train))
     ```

   -
     ```
     matrix.train.raw = model.matrix(Y.train ~ ., data =
     cbind(Y.train, X.train))
     matrix.train = matrix.train.raw[,-1]
     all.LASSOs = cv.glmnet(x = matrix.train, y = Y.train)
     ```

   -
     ```
     fit.pls = plsr(Y.train ~ ., data = cbind(Y.train, X.train),
     validation="CV", segments=5)
     ```

   -
     ```
     fit.gam = gam(data=cbind(Y.train, X.train),
           formula=Y.train ~ s(X2, sp=0.4) + s(X4,sp=0.2) + s(X12,
                 sp=0.2) + s(X15, sp=0.2),
           family=gaussian(link=identity))
     ```

   -
     ```
     nn.params = tune.neural.net(data[-1],data[1])
     nn.hidden = as.integer(str_split(nn.params,',')[[1]])[1]
     nn.shrink = as.integer(str_split(nn.params,',')[[1]])[2]
     fit.nnet = nnet(X.train, Y.train, linout=T, size=nn.hidden,
     decay=nn.shrink, maxit=500, trace=F)
     ```

   -
     ```
     fit.ppr = ppr(Y.train ~ ., data = train.ppr,
      max.terms = max.terms, nterms = 1, sm.method = "gcvspline")
     ```

   -
     ```
     fit.tree = rpart(Y.train ~ X2 + X4 + X12 + X15, data =
     cbind(Y.train, X.train), cp = 0.05)
     ```

2. I used 10-fold cross-validation and took relative MSPE over all models. I selected the model which would produce the lowest RMSPE on average and decided to use that one to create the final predictions.

3. The neural network was tuned on a separate round of CV to find the optimal parameters. For the tuning the hidden layers and shrinkage value needed to be determined. The parameters were found by running through all combinations of hidden layers(1, 3, 5, 7) and shrinkage(0.001, 0.1, 0.5, 1, 2, 3) for minimum average MSPE. For the GAM, I found the smoothing parameters that were found automatically with CV were not optimal and I manually viewed the smoothing plots for each variable and adjusted the smoothing parameters accordingly. I tested the range of smoothing spline sp parameters as follows, permutations of (0,0.1, 0.2, 0.4, 0.6, 0.8, 1) on each variable, testing for minimum MSPE average. Thankfully, GAMs are relatively easy to tune, by viewing the smoothing plots for the model, and adjusting based on the appearance of the smooth.

4.
```
#Seed default
seed = 10

#Helper functions
get.folds = function(n, K) {
  set.seed(seed)
  n.fold = ceiling(n / K)
  fold.ids.raw = rep(1:K, times = n.fold)
  fold.ids = fold.ids.raw[1:n]
  folds.rand = fold.ids[sample.int(n)]
  return(folds.rand)
}
get.MSPE = function(Y, Y.hat){
  return(mean((Y - Y.hat)^2))
}

#Modelling functions, non-optimal models were excluded from report
gam.model = function(X.train, Y.train, X.valid , Y.valid){
  fit.gam = gam(data=cbind(Y.train, X.train),
              formula=Y.train ~ s(X2, sp=0.1) + s(X4,sp=0.2) + s(X12,
sp=0.2) + s(X15, sp=0.2),
              family=gaussian(link=identity))
  pred.gam = predict(fit.gam ,newdata=X.valid)
  MSPE.gam = get.MSPE(Y.valid, pred.gam)
  return(list(MSPE.gam, pred.gam))
}
#Load Data
data = na.omit(read.csv("Data2020.csv"))
#data = data[,c(1,3,4,5,13)]
test = na.omit(read.csv("Data2020testX.csv"))
#test = test[,c(2,4,6,12)]

#Get num rows as n
n = nrow(data)

#Split into CV folds
K=10
folds = get.folds(n, K)

#CV Comparison of Diff Models
all.models = c("LS", "STEPWISE", "RIDGE", "LASSO-MIN", "LASSO-1SE",
"PLS", "GAM", "PPR", "NNET", "FULL TREE", "MIN TREE", "1SE TREE")
all.MSPEs = array(0, dim = c(K,length(all.models)))
colnames(all.MSPEs) = all.models

# CV method
for(i in 1:K){
 X.train = data[folds != i,-1]
 X.valid = data[folds == i,-1]
 Y.train = data[folds != i,1]
 Y.valid = data[folds == i,1]

 #GAM
 all.MSPEs[i, "GAM"] = gam.model(X.train, Y.train, X.valid, Y.valid)
[[1]]
 #All other models that were not optimal have been excluded from the
report.

}
```

```
par(mfrow=c(1,1))
boxplot(all.MSPEs, main = paste0("CV MSPEs over ", K, " folds"))
all.RMSPEs = apply(all.MSPEs, 2, function(W) W/min(W))
boxplot(t(all.RMSPEs))

Y.hat = gam.model(data[,-1], data[,1], test, test[,1])[[2]]

write.table(Y.hat, 'output.csv', sep = ",", row.names = F, col.names =
F)
```

5. X2, X4, X12, X15