1.

- `fit.knn = knn.cv(X.train, Y.train, k=i)`
- `fit.glm <- cv.glmnet(x=as.matrix(X.train),y=Y.train, family="multinomial")`
- `fit.lda = lda(X.train, Y.train)`
- `fit.log.nnet = multinom(Y.train ~ ., data = cbind(X.train, Y.train))`
- `fit.rf = randomForest(data=cbind(X.train, Y.train), Y.train~.,   mtry=mm, nodesize=ns, importance=TRUE, keep.forest=TRUE,  ntree=sz)`
- `fit.reg.tree = rpart(data=cbind(Y.train, X.train), method="class", Y.train ~   . , cp=0)`
- `fit.nnet = nnet(X.train, Y.train.num, size = sz, decay = dc, maxit =   2000,softmax = T, trace = F)`
- `fit.svm.0 = svm(Y.train ~ ., data = cbind(Y.train, X.train), kernel =    "radial", cost = cst, gamma = gma)`

2.

I used 10-fold repeated cross validation against all models. I selected the model which would produce the lowest misclassifcation error rate to produce the final predictions. I trained this final model on the full training set once determining the best one and then ran it against the test set.

3. Inside of the repeated 10-fold CV I ran all of my tuning for:
   1. **KNN** where k = (1:40)
   2. **Random forest** where mtry = (1:7), nodesize=(1,3,5,6,7,10), ntree=(500,1000,1500)
   3. **Neural net** where decay = (0, 0.01, 0.1, 1), size = (1, 3, 6, 10)
   4. **SVM** where cost = (5, 10), gamma = (0.1, 1)

   To compare these models, I used the validation misclassification error rate on each fold and then plotted the matrix of resultant values on a box and whiskers plot. I evaluated the best model to be the one with the lowest mean misclassification error rate and also considered the 25th and 75th percentiles to ensure they were also at a minimum. That is, if two models had a very similar means, I chose the one with the lower percentiles but slightly wider spread, not the one with very consistent values.
4. I looked at the importance of variables to find if there were any that significantly helped in reducing the misclassification error rate for validation during the multiple 10-fold CV. I found that many different models selected a few different variables each, with only a few being common throughout all the models. This helped my analysis as I found the B class was the most difficult to decipher of all classes. There were only a select few variables used in separating B from other classes so I tried to target these variables to improve the B classification aside from models like Random Forest trees and found only marginal improvements.
5. Random Forest.

```
data = na.omit(read.csv("P2Data2020.csv"))
test = na.omit(read.csv("P2Data2020testX.csv"))
data$Y = factor(data$Y, labels=c("A", "B", "C", "D", "E"))
fit.rf = randomForest(data=data, Y~., mtry=5, nodesize=5, importance=TRUE,
keep.forest=TRUE, ntree=500)
Y.hat = predict(fit.rf, newdata = test)
write.table(Y.hat, 'output.csv', sep=',', row.names = F, col.names = F)
```

6. Misclassification Error Rate: 0.176

Pred Obs	A	B	C	D	E
A	31	17	0	1	0
B	3	32	0	15	0
C	0	0	38	0	2
D	1	5	0	58	0
E	0	0	0	0	47

7. X7, X8, X9 X13, X16