Homework 4

Tanner Huck

2023-05-26

Problem 1

Intro) Using the College.csv data set, our goal is to predict the out of state tuition for each university with the use of other variables.

a. First we will split the data into 90% training and 10% testing.

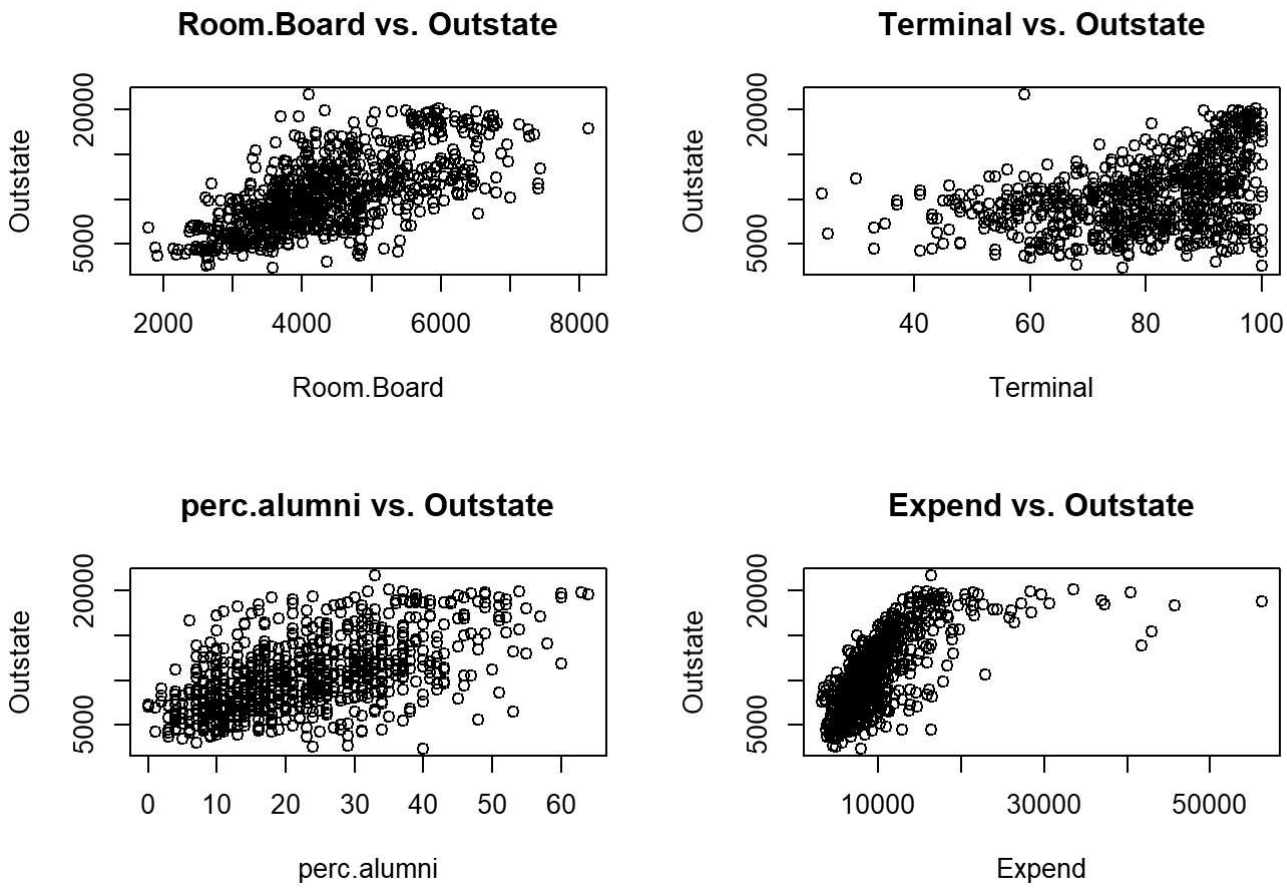
b. Now using forward selection, we want to identify the subset of predictors that satisfactorily predict the out-of-state tuition (not including the Name).

```
##   (Intercept)    PrivateYes    Room.Board      Terminal   perc.alumni
## -3475.7449964  2846.5146484     1.0449424    46.2444031    61.8400157
##        Expend
##     0.2288625
```

```
## [1] 3730958
```

From R I found that the predictors Private, Room.Boar, Termina, perc.alumni, and Expend did a good job of predicting out of state tuition. The MSE of our model with these factors was about 3.7309576^{6}. This is very large, thus we can see that our model does not do a good job of predicting tuition, thus there might be a more complex relationship going on.

c. Now we will perform some exploratory analysis to study the dependencies of out of state tuition and our 5 predictors.



Here I made scatter plots to display the relationship between our predictors and tuition. From these plots we can see that none of the predictors have any clear linear relationship with tuition. Notably, is seems that the terminal and expend predictors have an especially poor linear relationship. Now the MSE we calculated in part a) makes more sense, sense a linear model might not be appropriate to predict tuition.

d. Now instead of linear, we want to fit a generalized additive model (GAM) on the chosen subset of predictors to mode the tuition. From part c) we saw that all of our predictors had a relatively non-linear relationship with tuition, so we will use these in our model.

```
GAM_model <- gam(Outstate ~ Private + s(Room.Board) + s(Terminal) + s(perc.alumni) + s(Expend),
data = train)


GAM_predict <- predict(GAM_model, test)
MSE_gam <- mean((test$Outstate - GAM_predict)^2)
MSE_gam
```

```
## [1] 4037881
```

From fitting the GAM model we found an MSE of about 4.0378814^{6} which is higher than the MSE from the linear model at about 3.7309576^{6}. Thus the linear model is still better at predicting out of state tuition.

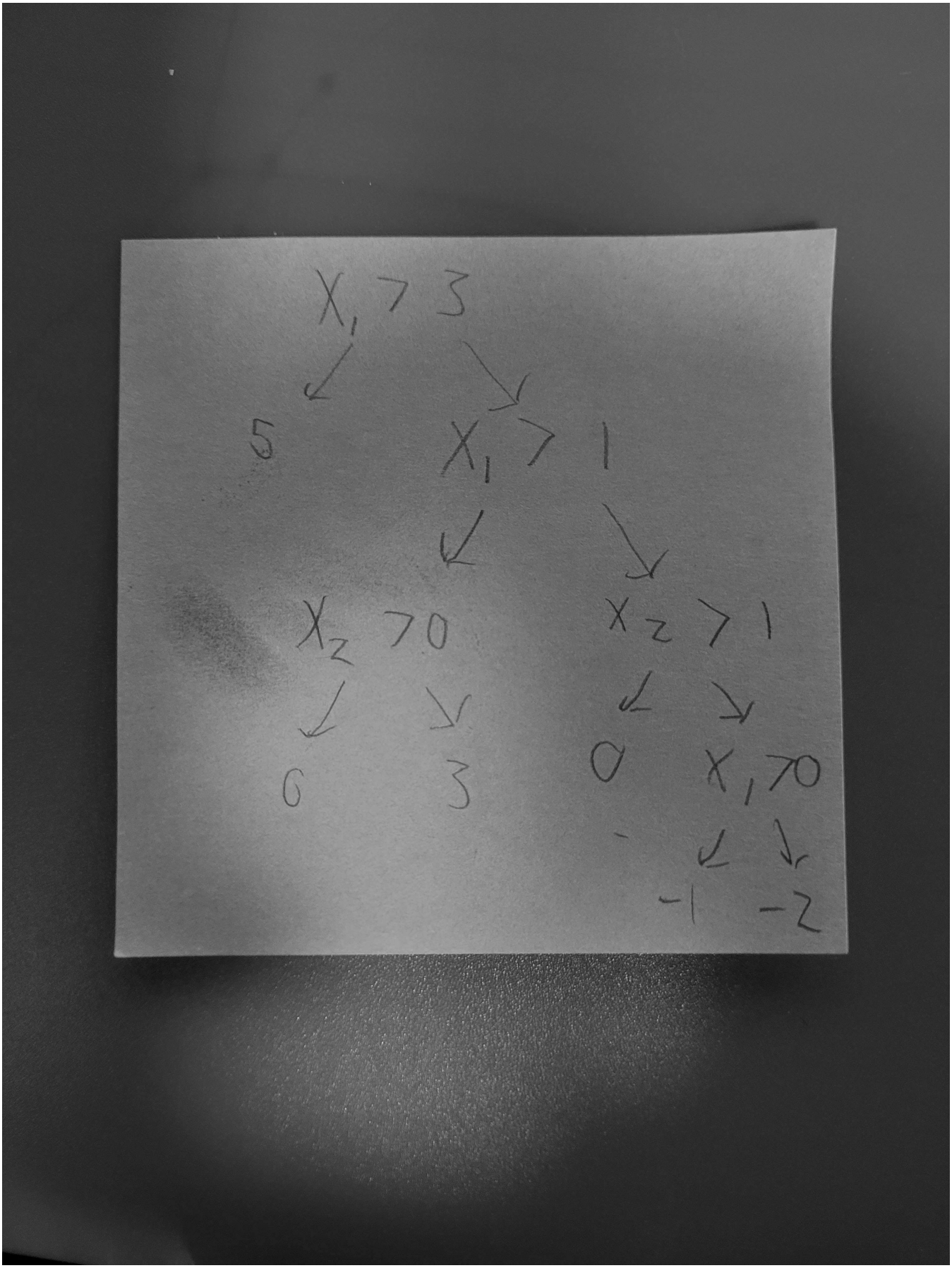
e. From our GAM model, we want to check if any of our predictors have significant evidence of a non-linear relationship.

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Outstate ~ Private + s(Room.Board) + s(Terminal) + s(perc.alumni) +
##     s(Expend)
##
## Parametric Terms:
##         df     F p-value
## Private  1 150.8  <2e-16
##
## Approximate significance of smooth terms:
##                  edf Ref.df      F p-value
## s(Room.Board)  2.323  2.971 33.850 < 2e-16
## s(Terminal)    1.517  1.886  8.384 0.00043
## s(perc.alumni) 2.459  3.118 17.067 < 2e-16
## s(Expend)      6.490  7.631 38.514 < 2e-16
```

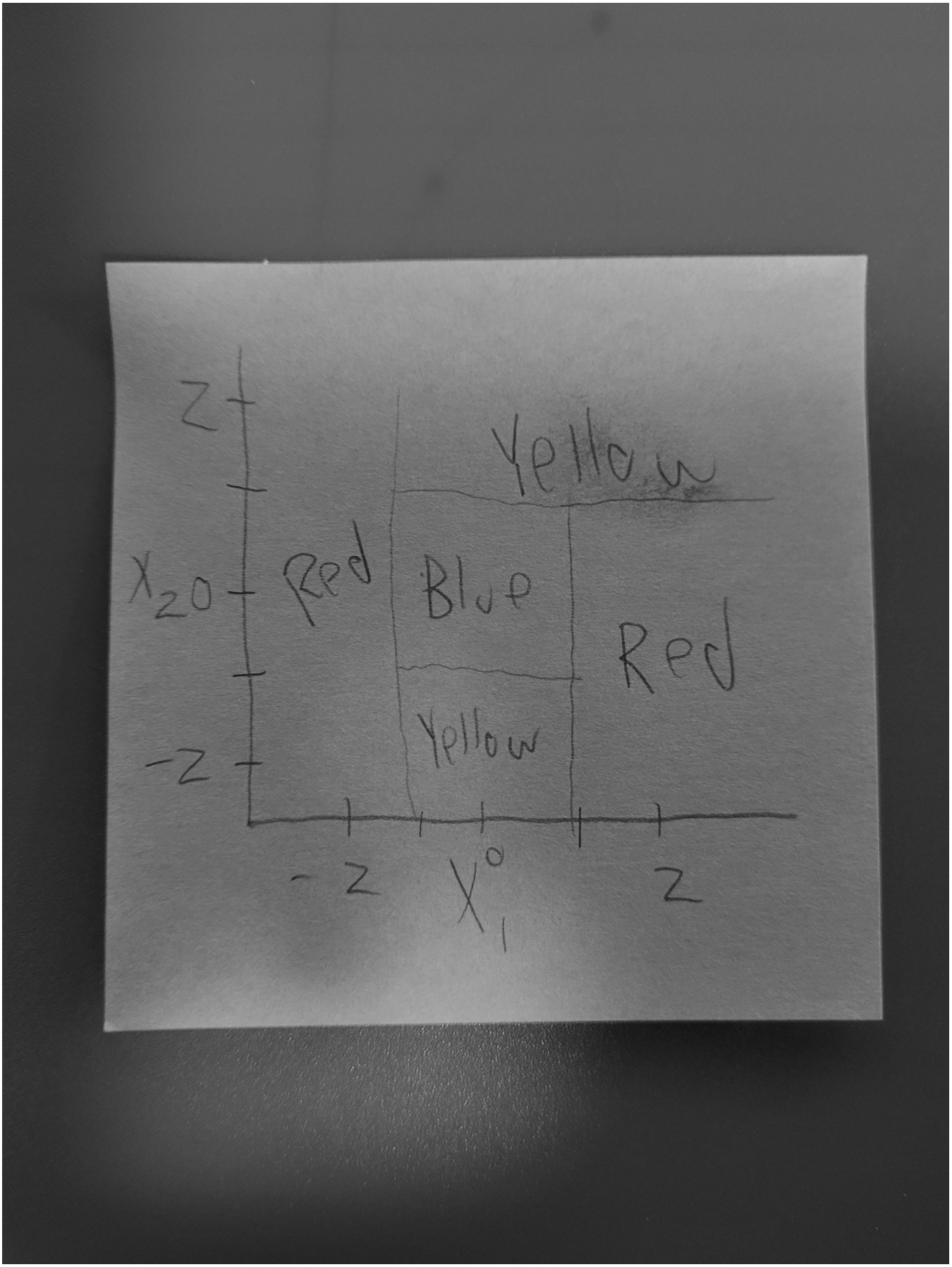
From looking at `anova(GAM_model)` in R, we can see that all Room.Board, Terminal, perc.alumni, and Expend had very low p-values, which signifies that each variable had evidence of a non-linear relationship wth tuition.

Problem 2

a. First we will draw the decision tree corresponding to the partition of the variable space illustrated in Figure 1(a), where the numbers inside each box represent the average value of the response in that particular region.



b. Now, consider a classification problem with three classes: yellow, red, and blue. Assuming that we are predicting the class based on X_1, X_2 where $-2 \leq X_1, X_2, \leq 2$. We want to draw the partition of the variable space for this tree, asusuming tree is given by Fig 1(b).


Z
Yellow
X_2 0
Red
Blue
Red
Yellow
-Z
-Z
X^0_1
Z

Problem 3

Intro) Now we will build a ensemble classifier. We will train several classifiers on the training data and use them together to make the prediction. We will use the `seeds` data set.

a. First we will download the data and split into training and testing. We will have 150 training points.

```
set.seed(7979)

data <- read.csv("seeds.txt")
seeds_split <- initial_split(data, prop=0.76)
train <- training(seeds_split)
test <- testing(seeds_split)
```

```
dim(train)
```

```
## [1] 150   8
```

After splitting the seeds data in R we can verify that the training data has 150 points.

b. Next we will start building our model using LDA. We will sample 150 points with replacement from the training data set, train an LDA classifier on this bootstrap sample, and then report the error and confusion matrix on test data.

```
## [1] "LDA Classification Error: 0.0416666666666667"
```

```
## [1] "LDA Confusion Matrix:"
```

```
##    LDA_predicts_list
##      1  2  3
##   1 15  1  1
##   2  0 18  0
##   3  0  0 13
```

From our R code, we obtain an error of about 0.0417 and the above confusion matrix. We can learn from the confusion matrix that we incorectly predicted a 3 when it should have been a 1.

c. The next member of our model will be a k-nearest neighbor classifier. We will obtain a new bootstrap sample the same way as in part b. Then for $k = 3, 4, \ldots, 30$ we will find a k that minimizes the error on the bootstrap sample and report its error on test data.

```
## [1] "Errors:"
```

```
##  [1] "0.0533333333333333" "0.06"               "0.0533333333333333"
##  [4] "0.0733333333333333" "0.06"               "0.08"
##  [7] "0.0733333333333333" "0.08"               "0.0733333333333333"
## [10] "0.0666666666666667" "0.06"               "0.08"
## [13] "0.08"               "0.08"               "0.0666666666666667"
## [16] "0.0933333333333333" "0.08"               "0.08"
## [19] "0.08"               "0.08"               "0.08"
## [22] "0.08"               "0.0866666666666667" "0.0866666666666667"
## [25] "0.0666666666666667" "0.08"               "0.0866666666666667"
## [28] "0.1"
```

```
## [1] "Optimal k value (k0): 3"
```

```
## [1] "Best Training Error: 0.0533333333333333"
```

```
## [1] "KNN Confusion Matrix:"
```

```
##
## knn_pred  1  2  3
##        1 40  0  0
##        2  3 49  0
##        3  5  0 53
```

```
## [1] "Test error 0.0533333333333333"
```

Thus we find the optimal k to be 3 that has a error of about 0.05. From the confusion matrix we see that we predict 3 1's that should have been 2's and 5 1's that should have been 3's, a total of 8 errors; giving us a test error of about 0.05.

d. Finally, we will add a random Forrest. We will train the random Forrest classifier on the training data and report the error and confusion matrix.

```
## [1] "Random Forest Classification Error: 0.0416666666666667"
```

```
## [1] "Random Forest Confusion Matrix:"
```

```
##
##      1  2  3
##   1 15  0  2
##   2  0 18  0
##   3  0  0 13
```

Hence the error is about 0.042 and the confusion matrix is shown above. We can see that we predict 3 2'2, that should have been 1's.

e. Then we will use our combine model. For each test data point, we will predict using each classifier made in b-d and choose which class has the most votes out of the predictions. Then find the error and confusion

matrix.

```
## [1] "Classification Error: 0.0416666666666667"
```

```
## [1] "Confusion Matrix:"
```

```
##
##      1  2  3
##   1 15  0  2
##   2  0 18  0
##   3  0  0 13
```

Hence using the full committee to test, we get an error of about 0.042 and the above confusion matrix. Comparatively, this is about the same as the random tree model. It only mislabels 2 points.

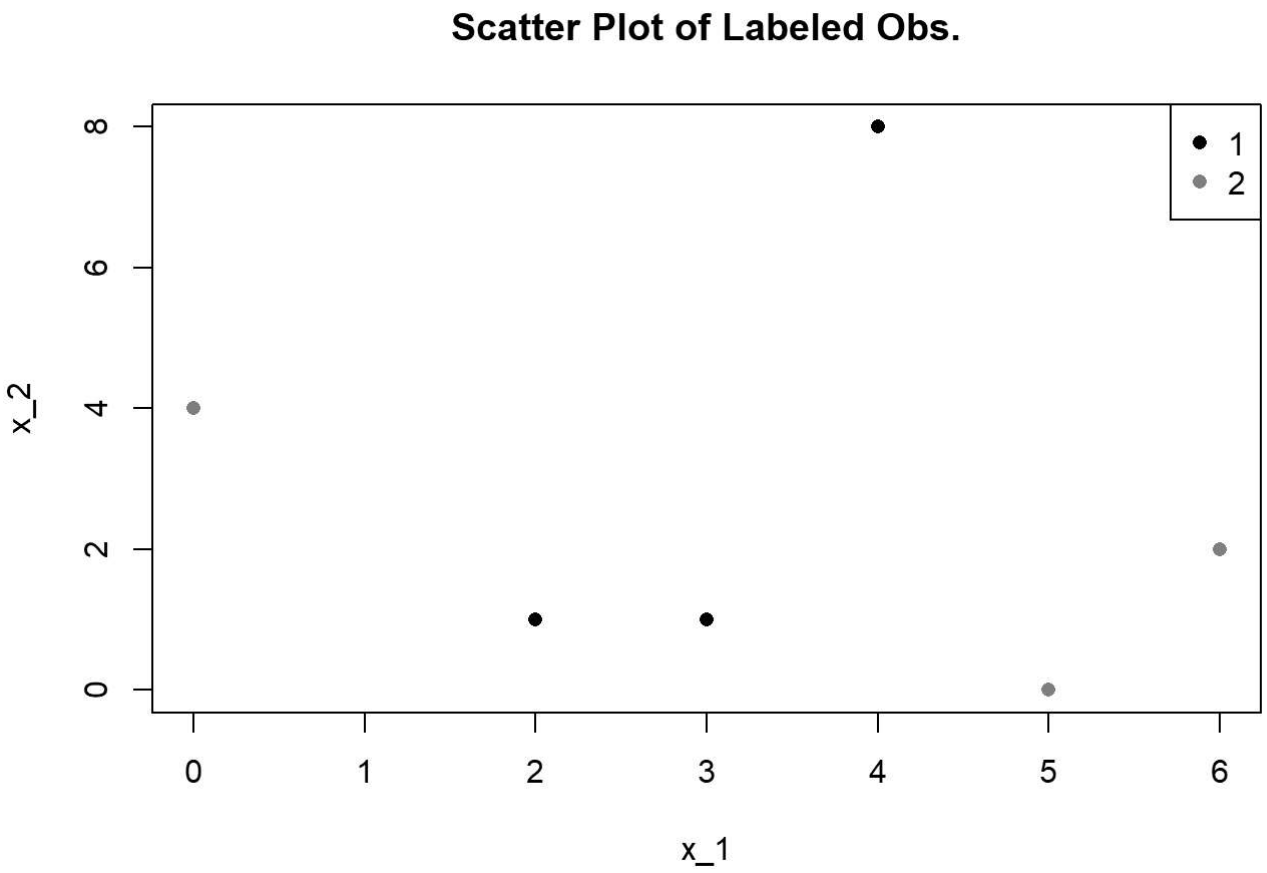
f. Another way we could make predictions is instead of choosing the most common vote, we could randomly selected a label from the three different predicted labels. An additional way we could do this is by examining the error of each model. We could see which model gives us the least amount of error and use it's label predictions.

Problem 4

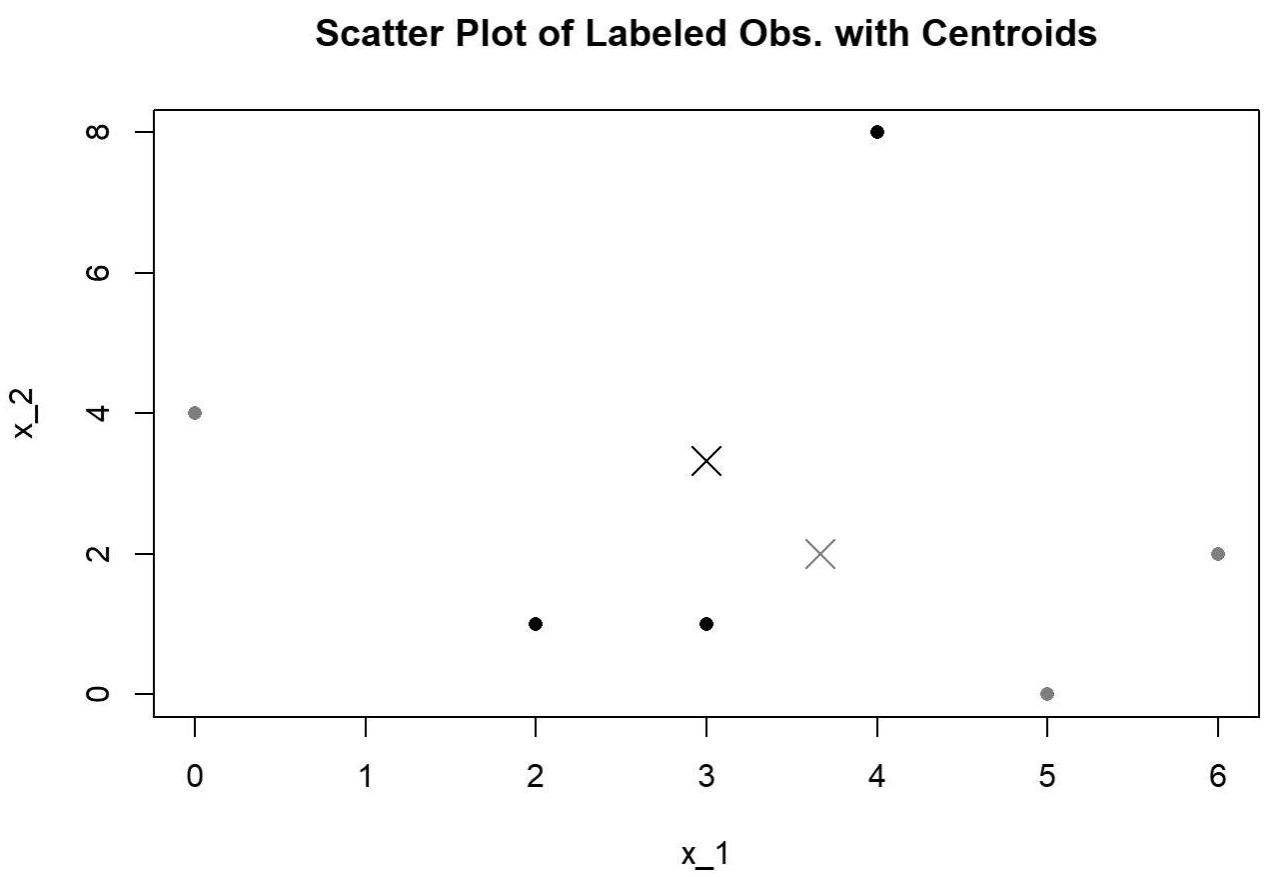
Intro) Now we will perform K means clustering manually, with $K = 2$, on a small example with $n = 6$ observations and $p = 2$ features. We will stat with the randomly assigned cluster labels $1, 2, 2, 1, 1, 2$ for the observations respectivily.

a. First, plotting our observations,

```
##   number x_1 x_2 label
## 1      1   3   1     1
## 2      2   6   2     2
## 3      3   0   4     2
## 4      4   2   1     1
## 5      5   4   8     1
## 6      6   5   0     2
```


Scatter Plot of Labeled Obs.
x_2
x_1
0
2
4
6
8
0
1
2
3
4
5
6
1
2

b. Then computing the centroid for each cluster based on the labels and plotting them.



c. Then we want to assign each observation to the centroid it is closest to and report the cluster labels for each observation and its euclidean distance to the centroid.

```
## Observation: 1 Cluster: 2 Distance: 1.20185
## Observation: 2 Cluster: 2 Distance: 2.333333
## Observation: 3 Cluster: 1 Distance: 3.073181
## Observation: 4 Cluster: 2 Distance: 1.943651
## Observation: 5 Cluster: 1 Distance: 4.772607
## Observation: 6 Cluster: 2 Distance: 2.403701
```

Thus after assigning each point a new label based on which centroid it is closed to, we assign the labels $2, 2, 1, 2, 1, 2$ for the six observations respectively.

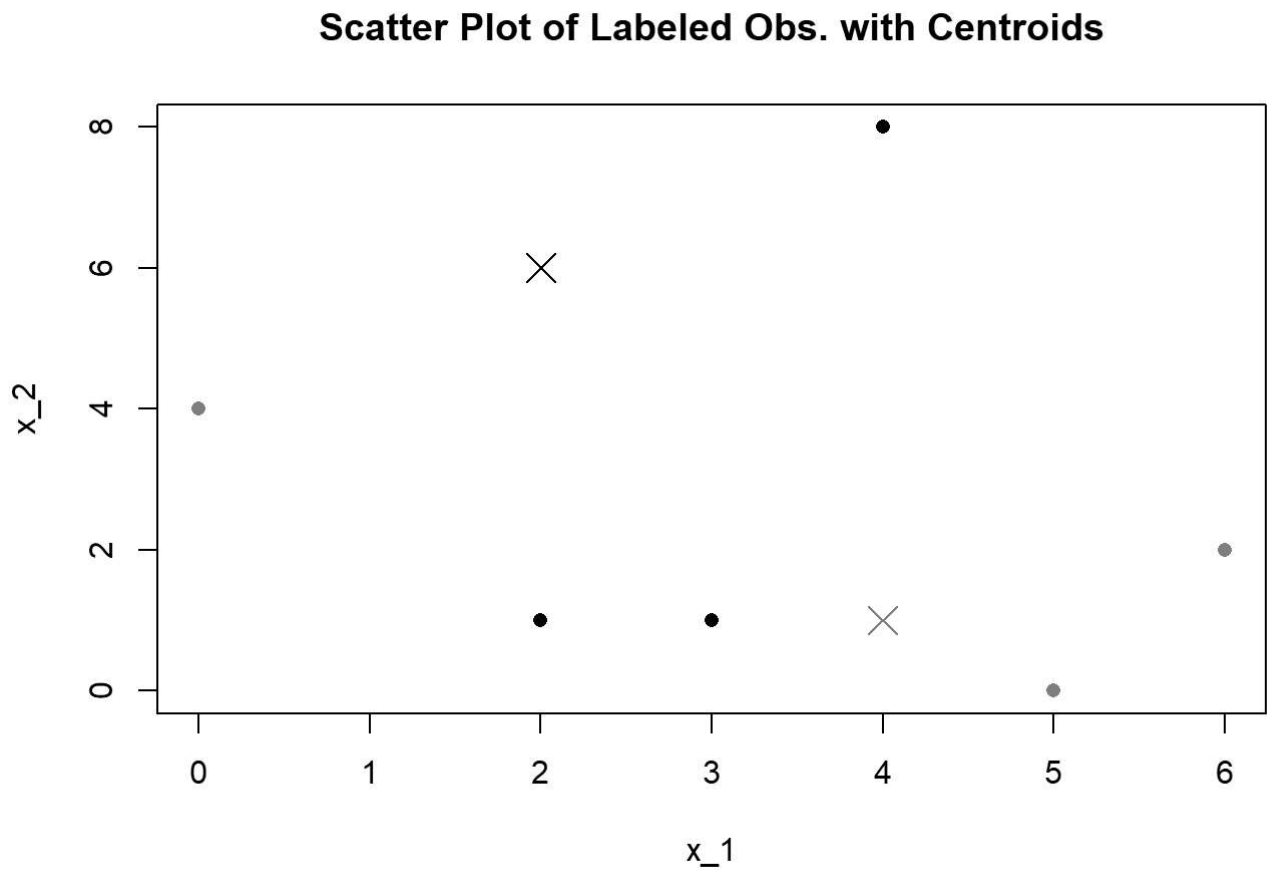
d. Now we want to repeat parts b and c until the answers stop changing and report the final labels and distances.

```
## Observation: 1 Cluster: 2 Distance: 1
## Observation: 2 Cluster: 2 Distance: 2.236068
## Observation: 3 Cluster: 1 Distance: 2.828427
## Observation: 4 Cluster: 2 Distance: 2
## Observation: 5 Cluster: 1 Distance: 2.828427
## Observation: 6 Cluster: 2 Distance: 1.414214
```

After repeating steps b though c just one more time, we can see that we receive the same labels we had before repeating. Hence the final labels and euclidean distances can be seen in the table above.

e. Finally, plotting the results form part d.

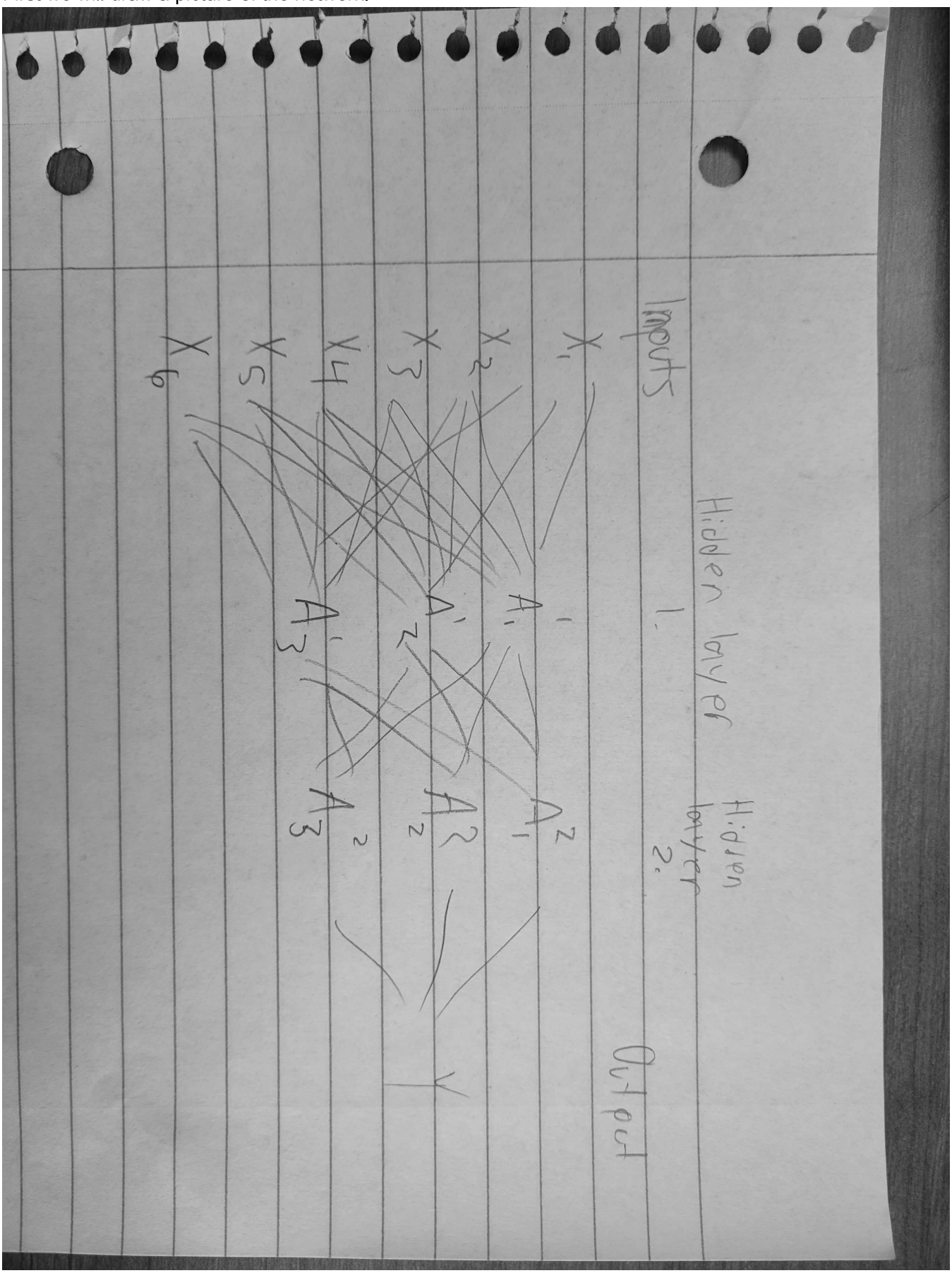
```
centroids <- aggregate(cbind(x_1, x_2) ~ label, data = data_new, FUN = mean)

plot(data$x_1, data$x_2, col = as.numeric(data$label), pch = 16,
     xlab = "x_1", ylab = "x_2", main = "Scatter Plot of Labeled Obs. with Centroids")
points(centroids$x_1, centroids$x_2, col = as.numeric(centroids$label), pch = 4, cex = 2)
```



Problem 5

Intro) Now we will consider a neural network with hidden layers: $p = 5$ input units and 3 units in the first hideen layer, 3 units in the second hidden layer, with a single output.

a. First we will draw a picture of the network.



Appendix

Question 1

```
set.seed(7979)

data <- read.csv("College.csv")

train_index <- sample(1:nrow(data), 0.9 * nrow(data))
train <- data[train_index, ]
test<- data[-train_index, ]

forward <- regsubsets(Outstate ~ ., data = train, nbest = 1, nvmax = 5, method = "forward",
really.big = T)
coef(forward, 5)

model <- lm(Outstate ~ Private + Room.Board  + Terminal + perc.alumni + Expend, data=train)
y_hat <- predict(model, test)
MSE <- mean((test$Outstate - y_hat)^2)
MSE

par(mfrow = c(2, 2))
plot(data$Room.Board, data$Outstate, xlab = "Room.Board", ylab = "Outstate", main = "Room.Board
vs. Outstate")
plot(data$Terminal, data$Outstate, xlab = "Terminal", ylab = "Outstate", main = "Terminal vs. Ou
tstate")
plot(data$perc.alumni, data$Outstate, xlab = "perc.alumni", ylab = "Outstate", main = "perc.alum
ni vs. Outstate")
plot(data$Expend, data$Outstate, xlab = "Expend", ylab = "Outstate", main = "Expend vs. Outstat
e")

GAM_model <- gam(Outstate ~ Private + s(Room.Board) + s(Terminal) + s(perc.alumni) + s(Expend),
data = train)


GAM_predict <- predict(GAM_model, test)
MSE_gam <- mean((test$Outstate - GAM_predict)^2)
MSE_gam

anova(GAM_model)
```

Question 3

```
set.seed(7979)

data <- read.csv("seeds.txt")
seeds_split <- initial_split(data, prop=0.76)
train <- training(seeds_split)
test <- testing(seeds_split)


boot_LDA_samp <- sample_n(train, 150, replace = TRUE)
LDA_model <- lda(Y ~ ., data=boot_LDA_samp)

LDA_predicts <- predict(LDA_model, test)
LDA_predicts_list <- as.character(LDA_predicts$class)

LDA_error <- mean(LDA_predicts_list != test$Y)

LDA_confusion_matrix <- table(test$Y, LDA_predicts_list)

print(paste("LDA Classification Error:", LDA_error))
print("LDA Confusion Matrix:")
print(LDA_confusion_matrix)



set.seed(7979)
boot_knear_samp <- sample_n(train, 150, replace = TRUE)
train_knear <- boot_knear_samp %>% dplyr::select(-Y)
test_knear <- boot_knear_samp$Y

train_errors <- rep(NA, 28)
highest_error <- 1
best_k <- NA

for (i in 3:30) {
  predictions <- knn(train_knear, train_knear, test_knear, k = i)
  errors <- mean(predictions != test_knear)
  train_errors[i - 2] <- errors
  if (train_errors[i - 2] < highest_error) {
    highest_error <- train_errors[i - 2]
    best_k <- i
  }
}

knn_pred <- knn(boot_knear_samp[-8], boot_knear_samp[-8], boot_knear_samp$Y, best_k)
knn_error <- mean(knn_pred != test_knear)
knn_confusion_matrix <- table(knn_pred, boot_knear_samp$Y)

print("Errors:")
print(paste(train_errors))
print(paste("Optimal k value (k0):", best_k))
print(paste("Best Training Error:", train_errors[best_k - 2]))
```

```
print("KNN Confusion Matrix:")
print(knn_confusion_matrix)
print(paste("Test error", knn_error))


set.seed(7979)

train$Y <- as.factor(train$Y)
tree_model <- randomForest(Y ~ ., data = train, ntree = 100, importance = TRUE, type="classifica
tion")
tree_dictions <- predict(tree_model, newdata = test)
tree_error <- mean(tree_dictions != test$Y)
tree_confusion_matrix <- table(test$Y, as.factor(tree_dictions))

print(paste("Random Forest Classification Error:", tree_error))
print("Random Forest Confusion Matrix:")
print(tree_confusion_matrix)

set.seed(7979)

knn_predictions <- knn(train, test, train$Y, k = best_k)

predictions_total <- data.frame(LDA_predicts_list, knn_predictions, tree_dictions)
decision <- apply(predictions_total, 1, function(x) {
  class_counts <- table(x)
  class_counts[class_counts == max(class_counts)]
})

decision <- as.character(decision)
error <- mean(decision != test$Y)

tree_confusion_matrix <- table(test$Y, as.factor(tree_dictions))

print(paste("Random Forest Classification Error:", tree_error))
print("Random Forest Confusion Matrix:")
print(tree_confusion_matrix)

set.seed(7979)

knn_predictions <- knn(train, test, train$Y, k = best_k)

predictions_total <- data.frame(LDA_predicts_list, knn_predictions, tree_dictions)
decision <- apply(predictions_total, 1, function(x) {
  class_counts <- table(x)
  class_counts[class_counts == max(class_counts)]
})

decision <- as.character(decision)
error <- mean(decision != test$Y)
error <- tree_error

confusion_matrix <- table(test$Y, decision)
```

```
confusion_matrix <- tree_confusion_matrix

print(paste("Classification Error:", error))
print("Confusion Matrix:")
print(confusion_matrix)
```

Question 4

```
data <- data.frame(number = c(1, 2, 3, 4, 5, 6),
                   x_1 = c(3, 6, 0, 2, 4, 5),
                   x_2 = c(1, 2, 4, 1, 8, 0),
                   label = c(1, 2, 2, 1, 1, 2))
data

data$label <- as.factor(data$label)
plot(data$x_1, data$x_2, col = as.numeric(data$label), pch = 16,
     xlab = "x_1", ylab = "x_2", main = "Scatter Plot of Labeled Obs.")
legend("topright", legend = levels(data$label), col = 1:length(levels(data$label)), pch = 16)

centroids <- aggregate(cbind(x_1, x_2) ~ label, data = data, FUN = mean)

plot(data$x_1, data$x_2, col = as.numeric(data$label), pch = 16,
     xlab = "x_1", ylab = "x_2", main = "Scatter Plot of Labeled Obs. with Centroids")
points(centroids$x_1, centroids$x_2, col = as.numeric(centroids$label), pch = 4, cex = 2)

assigned_labels <- vector()
distances <- vector()

for (i in 1:nrow(data)) {
  point <- data[i, c("x_1", "x_2")]
  dist_to_centroids <- sqrt((centroids$x_1 - point$x_1)^2 + (centroids$x_2 - point$x_2)^2)
  closest_centroid_idx <- which.min(dist_to_centroids)
  assigned_labels <- c(assigned_labels, closest_centroid_idx)
  distances <- c(distances, dist_to_centroids[closest_centroid_idx])
}

for (i in 1:nrow(data)) {
  cat("Observation:", i, "Cluster:", assigned_labels[i], "Distance:", distances[i], "\n")
}

data_new <- data.frame(number = 1:6,
                   x_1 = c(3, 6, 0, 2, 4, 5),
                   x_2 = c(1, 2, 4, 1, 8, 0),
                   label = c(2,2,1,2,1,2))

centroids <- aggregate(cbind(x_1, x_2) ~ label, data = data_new, FUN = mean)

assigned_labels <- vector()
distances <- vector()

for (i in 1:nrow(data_new)) {
  point <- data_new[i, c("x_1", "x_2")]
  dist_to_centroids <- sqrt((centroids$x_1 - point$x_1)^2 + (centroids$x_2 - point$x_2)^2)
  closest_centroid_idx <- which.min(dist_to_centroids)
  assigned_labels <- c(assigned_labels, closest_centroid_idx)
  distances <- c(distances, dist_to_centroids[closest_centroid_idx])
}
```

```
for (i in 1:nrow(data_new)) {
  cat("Observation:", i, "Cluster:", assigned_labels[i], "Distance:", distances[i], "\n")
}

centroids <- aggregate(cbind(x_1, x_2) ~ label, data = data_new, FUN = mean)

plot(data$x_1, data$x_2, col = as.numeric(data$label), pch = 16,
     xlab = "x_1", ylab = "x_2", main = "Scatter Plot of Labeled Obs. with Centroids")
points(centroids$x_1, centroids$x_2, col = as.numeric(centroids$label), pch = 4, cex = 2)
```