ISyE_6501_Homework_2.R

Angela Ahn

Wed Jan 23, 2019

Question 3.1(a)

Revisiting the credit card data set, I first noted that the data includes 10 predictor variables (6 continuous and 4 binary predictors), and that the final column represents the outcome (either 0 or 1), with 0 equating to an individual being rejected or denied a loan, and 1 equating to an individual being accepted or granted a loan. I used the head and tail functions to view the first six and last six data points in the credit card data set.

I trained the k-nearest-neighbors model using "leave-one-out" cross validation (LOOCV) via the train.kknn function. I defined my kmax value as 30, and then created a loop function to test the accuracy of each model with different k values (1-30). I subsequently graphed a plot with k values in the x-axis and accuracy of the model in the y-axis to visualize which k value resulted in the highest accuracy. Based on the graph and in comparing the accuracies of each model (graph shown below, pg. 2), I found that the model's accuracy peaked at k = 12, k = 15, k = 16, and k =17. The accuracy of all four of these models was equivalent at 85.32%. Thus, I propose that any of these four k values would produce a k-nearest-neighbor model with the best accuracy. To be specific, **I would suggest k = 15 as the best model, which classifies the data points with 85.32% accuracy.** The code that supports these findings is included below.

```
###Homework Week 2###
#Clear the environment and load kknn package
rm(list = ls())
library(kknn)

#Set working directory
setwd("~/ISyE 6501")

#load data and view first six and last six data points
CCdata <- read.table("credit_card_data.txt", stringsAsFactors = FALSE, header =
FALSE)
head(CCdata)

##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1

tail(CCdata)

##     V1    V2     V3   V4 V5 V6 V7 V8  V9 V10 V11
## 649  1 40.58  3.290 3.50  0  1  0  0 400   0   0
## 650  1 21.08 10.085 1.25  0  1  0  1 260   0   0
## 651  0 22.67  0.750 2.00  0  0  2  0 200 394   0
## 652  0 25.25 13.500 2.00  0  0  1  0 200   1   0
```

```
## 653  1 17.92  0.205 0.04  0  1  0  1 280 750   0
## 654  1 35.00  3.375 8.29  0  1  0  0   0   0   0
```

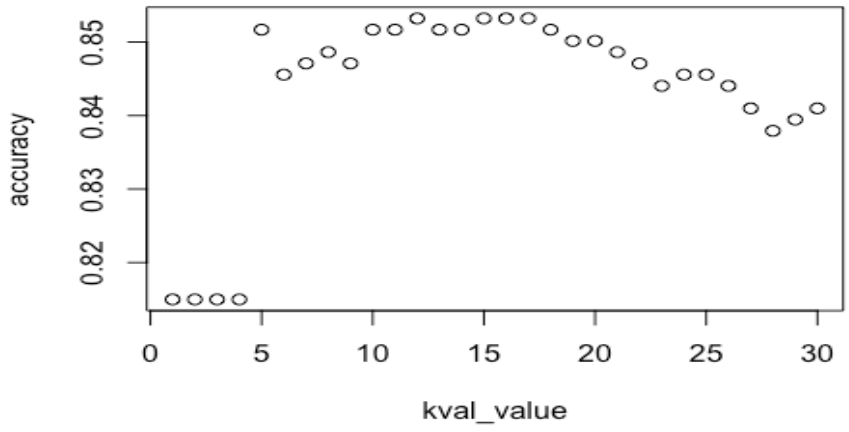
```
set.seed(1)
```

```
##Question 3.1a##
```

```
#Cross validation for k-nearest neighbor model using the "leave-out one" cross
validation via train.kknn().
#First set maximum value of k (number of neighbors) as 30
kmax <- 30
#Create an array for accuracies starting with values of 0
accuracy <- rep(0, kmax)

model_knn <- train.kknn(V11~., CCdata, kmax = kmax, scale = TRUE)

for (k in 1:kmax) {
  pred_knn <- as.integer(fitted(model_knn)[[k]][1:nrow(CCdata)] + 0.5) # adding
0.5 rounds the output to either 0 or 1
  accuracy[k] <- sum(pred_knn == CCdata$V11) / nrow(CCdata)
}

#Print accuracies
accuracy
```

```
##  [1] 0.8149847 0.8149847 0.8149847 0.8149847 0.8516820 0.8455657 0.8470948
##  [8] 0.8486239 0.8470948 0.8516820 0.8516820 0.8532110 0.8516820 0.8516820
## [15] 0.8532110 0.8532110 0.8532110 0.8516820 0.8501529 0.8501529 0.8486239
## [22] 0.8470948 0.8440367 0.8455657 0.8455657 0.8440367 0.8409786 0.8379205
## [29] 0.8394495 0.8409786
```

```
#Graph k-values (x axis) and accuracy (y axis) to see which k-value outputted
the best accuracy
kval_value <- c(1:30)
plot(kval_value, accuracy)
```



```
#k = 12, 15, 16, and 17 result in the highest accuracy rate of 85.32%
```

Question 3.1(b)

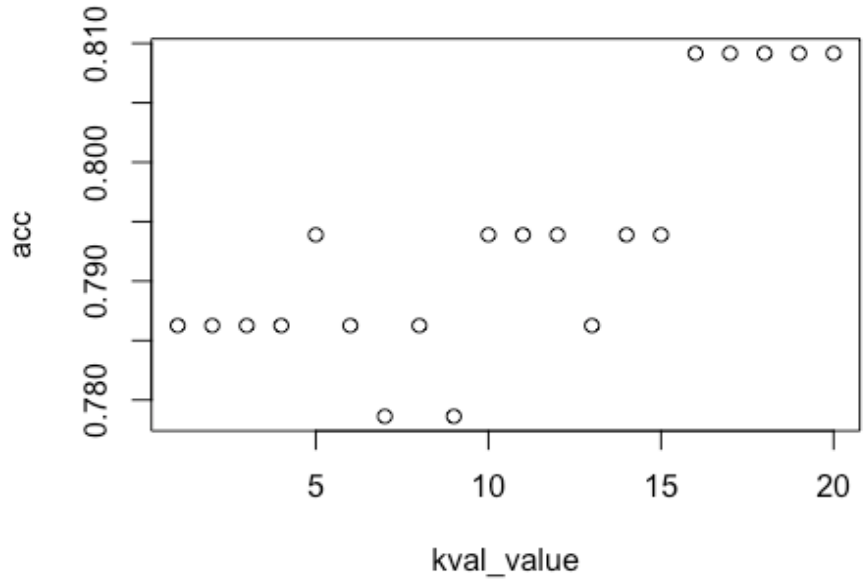
When splitting the credit card data set, I was mindful of the structure of the data set, and decided to randomly choose data points for each sub-data set so that 60% of the data would be included in the training set, 20% of the data in the validation set, and the remaining 20% in the test data set. In observing the data set more closely, I noted that the majority of the first couple hundred data points had a response variable of 1 (individuals who were approved for a loan). Without randomly splitting the data, the training set (comprised of 60% of the entire data set) would be biased, having an unproportional amount of data points that have a response variable of 1. Thus, randomizing the data set using the sample() function was necessary.

After successfully splitting the data into the training, validation, and testing sets (60/20/20), I trained and fit the knn model using the training and validation set. I utilized a loop to iterate through different values from k = 1 through k = 20. Again, I graphed a plot with k values in the x-axis and accuracy in the y-axis to visualize which k value resulted in the most accurate model. Based on the graph and in comparing the accuracies of each model (graph shown below, pg. 4), I found that the model's accuracy peaked at k = 16 at 80.92%. In other words, **the best knn model is when k = 16** with an accuracy on the validation set of 80.92%.

Lastly, I ran this model (k = 16) on the test data set. **The result shows that the k = 16 model results in an accuracy of 87.79%.** The code supporting these findings are included in the section below.

```
##Question 3.1b##
#Split the data for k-nearest neighbor model through random selection so that
the training data set includes 60%, validation data set includes 20%, and the
test data set includes 20%
train_ind <- sample(nrow(CCdata), size = floor(nrow(CCdata) * 0.6))
CC_train <- CCdata[train_ind, ] #training data set

remaining <- CCdata[-train_ind, ] #all rows except training

val_ind <- sample(nrow(remaining), size = floor(nrow(remaining)/2))

CC_val <- remaining[val_ind, ] #validation data set
CC_test <- remaining[-val_ind,  ] #test data set

#Now that we have successfully split the data set into training, validation,
and test data sets (60/20/20), train KNN models:

#Create an array for accuracies starting with values of 0
acc <- rep(0, 20)
for (k in 1:20) {
  knn_model <- kknn(V11~., CC_train, CC_val, k = k, scale = TRUE)
  pred_knn <- as.integer(fitted(knn_model) + 0.5)
  acc[k] = sum(pred_knn == CC_val$V11) / nrow(CC_val)
}
#Print accuracies
acc

##  [1] 0.7862595 0.7862595 0.7862595 0.7862595 0.7938931 0.7862595 0.7786260
##  [8] 0.7862595 0.7786260 0.7938931 0.7938931 0.7938931 0.7862595 0.7938931
## [15] 0.7938931 0.8091603 0.8091603 0.8091603 0.8091603 0.8091603
```

```
kval_value <- c(1:20)
plot(kval_value, acc)
```




```
#The best KNN model is k = 16 with accuracy on the validation set at 80.92%

#Now run this model (k = 16) on the test set data

knn_model <- kknn(V11~., CC_train, CC_test, k = 16, scale = TRUE)
pred <- as.integer(fitted(knn_model) + 0.5)

#Performance on test data:
sum(pred == CC_test$V11) / nrow(CC_test) #performance on test data

## [1] 0.8778626  #The k = 16 model produced an accuracy of 87.79%
```

Question 4.1

A clustering model would be an appropriate analytics tool to utilize at charter schools that are looking to improve the diversity of their student demographics. For example, one of the charter schools I work for recently established a new organizational goal— *to better reflect the demographic of the larger North Atlanta cluster by increasing the number of educationally disadvantaged students enrolled at the school by 30% over the next ten years*. The school recognizes that in order to actualize this goal, amongst other efforts, they would need to bolster their marketing strategy. Thus, the school could use clustering to help group people into clusters based on different predictor variables and inform a specific marketing strategy unique to each cluster. This marketing strategy would empower the school to reach diverse individuals and families in a targeted manner, knowing which qualities of the charter school to sell to which cluster, and how to best present the information. Below are 5 potential predictor variables that could be used in this clustering model and how the marketing strategy might differ for each group:

- **Geography & Socioeconomic Status/Family income**

    - Considering that the school is aiming to increase the number of educationally disadvantaged students, clustering models would allow the school could push more marketing efforts to clusters where more families are enrolled in the federal Free and Reduced Lunch program or fall below a specific family income line.
- **Families w/ children learning English as a second language**
    - For families who fall in this cluster, the school could specifically highlight the school's English Language Learners Program and the school's partnership with after-school tutoring and mentoring programs for Hispanic students.
- **Families w/ students with learning disabilities**
    - For families who fall in this cluster, the school could highlight the Special Education and Student Services team, the scope and breadth of services that are offered, and even success stories of the school's current special education students.
- **Parents/families who need after-school care**
    - For families who fall in this cluster, the school could share information about the after-school care program, including the variety of activities and services offered.
- **Families whose children previously attended private schools**
    - For families who fall in this cluster, the school could share information about the school's rigorous, content-rich curriculum and learning environment, along with statistics that highlight how this school's academic program mirrors that of a private school.

Question 4.2

In first orienting myself to the iris data, I noted that the dataset includes 4 predictor variables and one categorical response variable (i.e. type of flower). I used the head function to view the first six data points in the data set.

I first ran the table function and found that there are 50 of each type of flower. Because clustering is typically an unsupervised learning method, I made note of the fact that it would not be typical to know the response variable like we do for this particular problem.

Using the ggplot function, I graphed the relationships between different predictors to better understand the data and to get an initial idea of how the species would cluster best. Based on the results of these graphs, I made an initial prediction that the 2-variable combination of Petal Width and Petal Length would ouput a good clustering model.

Next, in order to identify the best k value, I used the kmeans function using all 4 predictorss and looped through different values of k from k = 2 to k = 10. I set nstart = 20, which means that R will generate 20 initial configurations -- i.e. try 20 different starting assignments and then choose the one with the lowest within cluster variation. Given that know there are 3 flower species, we can predict that k = 3 will be a good cluster value, but I test k values from 2 to 10 to confirm. The elbow graph confirmed that **k = 3 is the best k value**. As seen in the graph (below), **the elbow point falls at k = 3, meaning that this is the number where the benefit of adding another cluster starts to be marginal.**

Using k = 3, I tested the different combinations of variables. I tested all combinations of 2, 3, and 4 predictor variables, which was a total of 11 different combinations. For each predictor combination, I created a table that showed how many of each type of flower was correctly and incorrectly clustered. Finally, I created an accuracy function that used the data from each table to calculate the accuracy of the model.

In conclusion, the best k value is k = 3. The best combination of predictors is Petal Length and Petal Width. When using these 2 specific predictors and k = 3, the clustering model predicts flower type with 96% accuracy.

The code supporting all of these findings is included below.

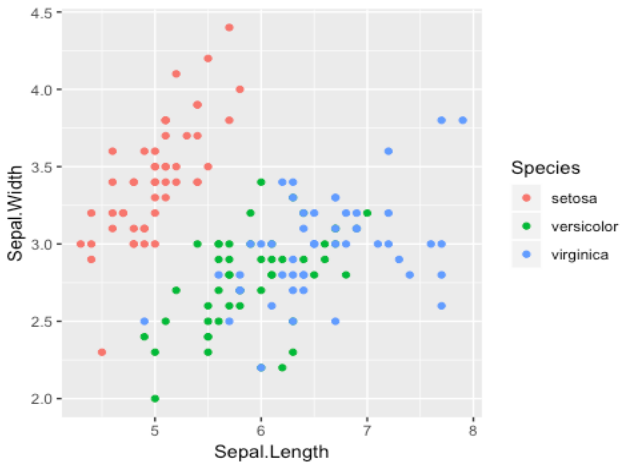
```r
##Question 4.2##
#clear environment
rm(list = ls())
#set working directory
setwd("~/ISyE 6501")

#load data and view first six and last six data points
irisdata <- read.table("iris.txt", header = TRUE)
head(irisdata)
```

```
##   Num Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1   1          5.1         3.5          1.4         0.2  setosa
## 2   2          4.9         3.0          1.4         0.2  setosa
## 3   3          4.7         3.2          1.3         0.2  setosa
## 4   4          4.6         3.1          1.5         0.2  setosa
## 5   5          5.0         3.6          1.4         0.2  setosa
## 6   6          5.4         3.9          1.7         0.4  setosa
```

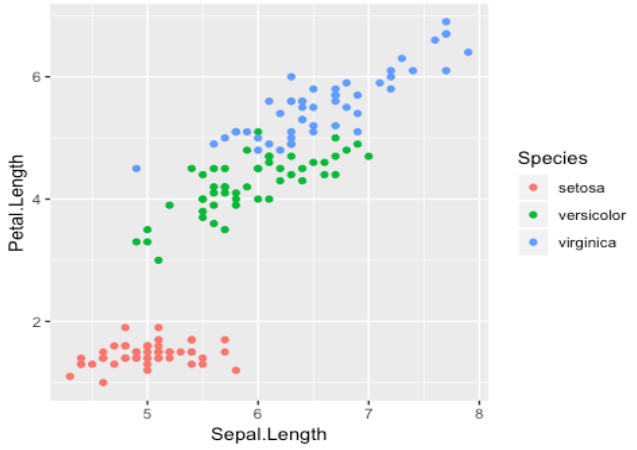
```r
set.seed(200)
irisdata <- irisdata[,2:6]
#By running the table function, we are able to see how many of each type of
flower species there are
table(irisdata$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

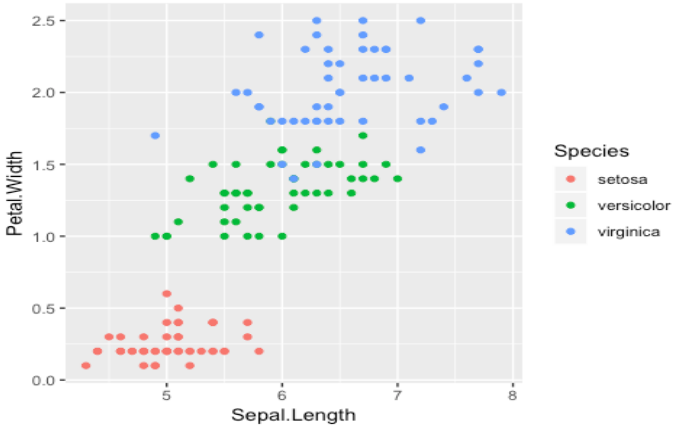
```r
#load ggplot2 package
#Graph relationships between the different variables to understand the data and
get an initial idea of how the species cluster best
library("ggplot2")

#Plot Sepal Length vs. Sepal Width
ggplot(irisdata, aes(Sepal.Length, Sepal.Width, color = Species)) +
geom_point() #infer that these two attributes do not make a good combination
because there is a signficant amount of overlap between the versicolor and
virginica species
```



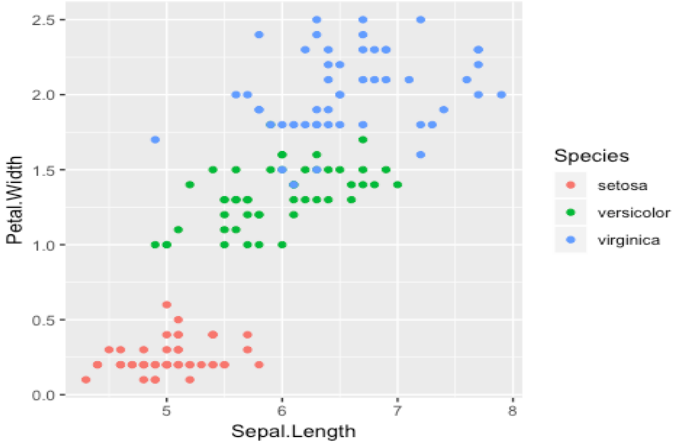
```r
#Plot Sepal.Length vs. Petal Length
ggplot(irisdata, aes(Sepal.Length, Petal.Length, color = Species)) +
geom_point() #clusters are better defined
```



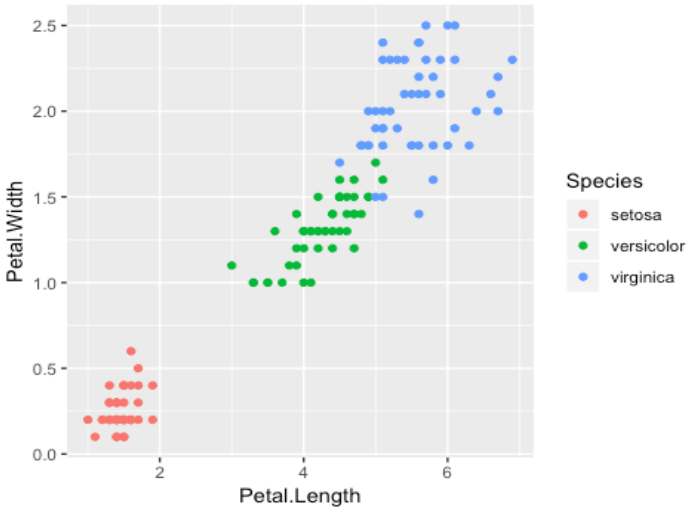

```r
#Plot Sepal.Length vs. Petal.Width
ggplot(irisdata, aes(Sepal.Length, Petal.Width, color = Species)) +
geom_point() #clusters are better defined
```




```r
#Plot Sepal.Width vs. Petal.Width
ggplot(irisdata, aes(Sepal.Length, Petal.Width, color = Species)) +
geom_point() #clusters are better defined
```



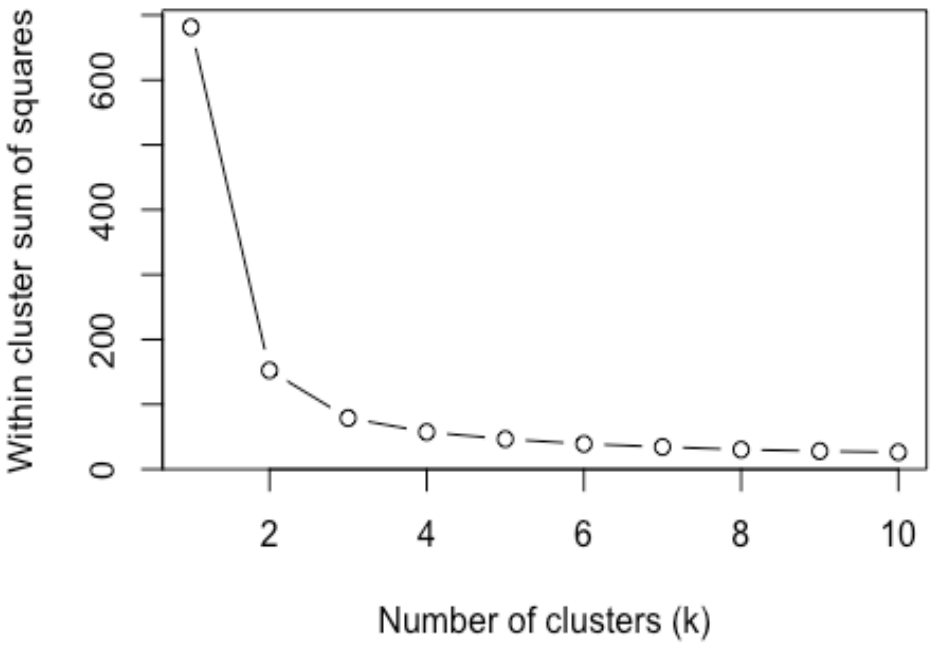
```r
#Plot Petal Length vs. Petal Width
ggplot(irisdata, aes(Petal.Length, Petal.Width, color = Species)) +
geom_point() #clusters are better defined
```




```r
#Use the kmeans function to cluster the points using all 4 attributes, looping through different values of k starting with k = 1 to k = 10:
k.max <- 10
wss <- sapply(1:k.max, function(k){kmeans(iris[,1:4], k, nstart = 20, iter.max = 20)$tot.withinss})
wss  #within cluster sum of square values for each k value
```

```
##  [1] 681.37060 152.34795  78.85144  57.22847  46.44618  39.03999  34.29823
##  [8]  30.18455  27.86026  26.16505
```

```r
#graph elbow plot
plot(1:k.max, wss, type = "b", xlab = "Number of clusters (k)", ylab = "Within cluster sum of squares")
```




```
#Graph results show that k = 3 leads to the best model
#Use k = 3 for models when testing different combinations of variables

#test combination of Sepal.Length and Sepal.Width (cluster_1)
cluster_1 <- kmeans(irisdata[, 1:2], 3, nstart = 20)
cluster_1$tot.withinss

## [1] 37.0507

table_1 <- table(cluster_1$cluster, irisdata$Species)
table_1

##
##     setosa versicolor virginica
##   1      0         38        15
##   2      0         12        35
##   3     50          0         0

#test combination of Sepal.Length and Petal.Length (cluster_2)
cluster_2 <- kmeans(irisdata[c(1,3)], 3, nstart = 20)
cluster_2$tot.withinss

## [1] 53.80998

table_2 <- table(cluster_2$cluster, irisdata$Species)
table_2
```

```
##
##     setosa versicolor virginica
##   1     50          1         0
##   2      0          4        37
##   3      0         45        13
```

```
#test combination of Sepal.Length and Petal.Width (cluster_3)
cluster_3 <- kmeans(irisdata[c(1,4)], 3, nstart = 20)
cluster_3$tot.withinss
```

```
## [1] 32.73348
```

```
table_3 <- table(cluster_3$cluster, irisdata$Species)
table_3
```

```
##
##     setosa versicolor virginica
##   1      0          9        37
##   2     50          4         0
##   3      0         37        13
```

```
#test combination of Sepal.Width and Petal.Length (cluster_4)
cluster_4 <- kmeans(irisdata[c(2,3)], 3, nstart = 20)
cluster_4$tot.withinss
```

```
## [1] 40.73707
```

```
table_4 <- table(cluster_4$cluster, irisdata$Species)
table_4
```

```
##
##     setosa versicolor virginica
##   1      0         48         9
##   2     50          0         0
##   3      0          2        41
```

```
#test combination of Sepal.Width and Petal.Width (cluster_5)
cluster_5 <- kmeans(irisdata[c(2,4)], 3, nstart = 20)
cluster_5$tot.withinss
```

```
## [1] 20.6024
```

```
table_5 <- table(cluster_5$cluster, irisdata$Species)
table_5
```

```
##
##     setosa versicolor virginica
##   1      0          4        44
##   2      1         46         6
##   3     49          0         0
```

```
#test combination of Petal.Length and Petal.Width (cluster_6)
cluster_6 <- kmeans(irisdata[c(3,4)], 3, nstart = 20)
cluster_6$tot.withinss
```

```
## [1] 31.37136
```

```
table_6 <- table(cluster_6$cluster, irisdata$Species)
table_6
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         48         4
##   3      0          2        46
```

```
##now test combinations of 3 variables
#test combination of Sepal.Length, Sepal.Width, and Petal.Length (cluster 7)
cluster_7 <- kmeans(irisdata[, 1:3], 3, nstart = 20)
cluster_7$tot.withinss
```

```
## [1] 69.42974
```

```
table_7 <- table(cluster_7$cluster, irisdata$Species)
table_7
```

```
##
##     setosa versicolor virginica
##   1      0          5        37
##   2      0         45        13
##   3     50          0         0
```

```
#test combination of Sepal.Width, Petal.Length, and Petal.Width (cluster 8)
cluster_8 <- kmeans(irisdata[, 2:4], 3, nstart = 20)
cluster_8$tot.withinss
```

```
## [1] 47.86643
```

```
table_8 <- table(cluster_8$cluster, irisdata$Species)
table_8
```

```
##
##     setosa versicolor virginica
##   1      0          2        45
##   2      0         48         5
##   3     50          0         0
```

```
#test combination of Sepal.Length, Sepal.Width, and Petal.Width (cluster 9)
cluster_9 <- kmeans(irisdata[c(1, 2, 4)], 3, nstart = 20)
cluster_9$tot.withinss
```

```
## [1] 48.66078
```

```
table_9 <- table(cluster_9$cluster, irisdata$Species)
table_9
```

```
##
##     setosa versicolor virginica
##   1      0         39        15
```

```
##   2      0         11        35
##   3     50          0         0
```

```
#test combination of Sepal.Length,Petal.Length, and Petal.Width (cluster 10)
cluster_10 <- kmeans(irisdata[c(1, 3, 4)], 3, nstart = 20)
cluster_10$tot.withinss
```

```
## [1] 63.34212
```

```
table_10 <- table(cluster_10$cluster, irisdata$Species)
table_10
```

```
##
##     setosa versicolor virginica
##   1      0         48        14
##   2      0          2        36
##   3     50          0         0
```

```
##test combination of all 4 predictor variables (cluster 11)
cluster_11 <- kmeans(irisdata[, 1:4], 3, nstart = 20)
cluster_11$tot.withinss
```

```
## [1] 78.85144
```

```
table_11 <- table(cluster_11$cluster, irisdata$Species)
table_11
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0          2        36
##   3      0         48        14
```

```
#create a function that calculates accuracy
accuracy <- function(x){
  return((max(x[,1])+max(x[,2])+max(x[,3]))/nrow(irisdata))
}

#Run each table through the accuracy function and determine which table
corresponds to the highest accuracy.
accuracy(table_1)
```

```
## [1] 0.82
```

```
accuracy(table_2)
```

```
## [1] 0.88
```

```
accuracy(table_3)
```

```
## [1] 0.8266667
```

```
accuracy(table_4)
```

```
## [1] 0.9266667
```

```r
accuracy(table_5)
```

```
## [1] 0.9266667
```

```r
accuracy(table_6)
```

```
## [1] 0.96
```

```r
accuracy(table_7)
```

```
## [1] 0.88
```

```r
accuracy(table_8)
```

```
## [1] 0.9533333
```

```r
accuracy(table_9)
```

```
## [1] 0.8266667
```

```r
accuracy(table_10)
```

```
## [1] 0.8933333
```

```r
accuracy(table_11)
```

```
## [1] 0.8933333
```

```r
#Because each table represents a different combination of predictor variables, we can conclude that table_6 outputs the highest accuracy at 96%. This corresponds to the 2 variable combination of Petal.Length and Petal.Width.
```