

Question 3.1a

Using the same data set (*credit_card_data.txt* or *credit_card_data-headers.txt*) as in Question 2.2, use the *ksvm* or *kknn* function to find a good classifier:

a) using cross-validation (do this for the *k*-nearest-neighbors model; SVM is optional);

Answer:

In my code below I used `cv.kknn` function and a `kcv` (number of partitions) = 10. I then iterate over different values of `k` from 1:20 to find the optimal value of `K` (classifier). When I run the cross validation, it tells me that `k=15` is the slight winner in terms of accuracy (85.8%). Thus, cross-validation tells us that the model with 15 nearest neighbors is the best model to classify this data.

```
##
12 = for (k in 1:20){
13   x <- cv.kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,
14               ccddata,
15               kcv=10,
16               k=k,
17               scale = TRUE)
18   x1<-c()
19 = for (i in 1:654){
20     t=sum((x[[1]][i,1]==round(x[[1]][i,2])))
21     x1<-c(x1,t)
22 = }
23   accuracy <- sum(x1)/654
24   print(paste0("Accuracy for k = ", k, ": ", accuracy))
25 = }
26
27
```

17:7 (Top Level) ↕

Console	Terminal	Background Jobs
R 4.2.2 · ~/		
[1] "Accuracy for k = 2: 0.813455657492355"		
[1] "Accuracy for k = 3: 0.804281345565749"		
[1] "Accuracy for k = 4: 0.798165137614679"		
[1] "Accuracy for k = 5: 0.845565749235474"		
[1] "Accuracy for k = 6: 0.842507645259939"		
[1] "Accuracy for k = 7: 0.847094801223242"		
[1] "Accuracy for k = 8: 0.845565749235474"		
[1] "Accuracy for k = 9: 0.840978593272171"		
[1] "Accuracy for k = 10: 0.851681957186544"		
[1] "Accuracy for k = 11: 0.845565749235474"		
[1] "Accuracy for k = 12: 0.856269113149847"		
[1] "Accuracy for k = 13: 0.850152905198777"		
[1] "Accuracy for k = 14: 0.851681957186544"		
[1] "Accuracy for k = 15: 0.857798165137615"		
[1] "Accuracy for k = 16: 0.840978593272171"		
[1] "Accuracy for k = 17: 0.847094801223242"		
[1] "Accuracy for k = 18: 0.842507645259939"		
[1] "Accuracy for k = 19: 0.848623853211009"		
[1] "Accuracy for k = 20: 0.85474006116208"		
> x[[1]]		

Question 3.1b

Using the same data set (*credit_card_data.txt* or *credit_card_data-headers.txt*) as in Question 2.2, use the *ksvm* or *kknn* function to find a good classifier:

b.) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

Answer:

To run this analysis, here are my steps:

1. I first split my credit card data into three data sets. A training data set with 70%, a validation data set with 15%, and a test data set with 15%

```
#I am basically creating training, validation, and test data sets. Not the cleanest route, but it worked
split <- sample.split(ccdata$V11, SplitRatio = .7)
training <- subset(ccdata, split == TRUE)

rest_of_data <- subset(ccdata, split == FALSE)
split2 <- sample.split(rest_of_data, SplitRatio = .5)
validation <- subset(rest_of_data, split2 == TRUE)
test <- subset(rest_of_data, split2 == FALSE)
```

2. Then i trained a model using the training data set and a validation data set

```
#step 1 train the model.

predicted <- c()
for (k in 1:20){
  accuracy <- c()
  model <- kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,training,validation,k=k,scale = TRUE)
  predicted <- as.integer(fitted.values(model)+0.5)
```


3. I then checked which model had the best accuracy. The output of the code showed that k=15 through k=20 were all the highest accuracy models, so I chose 15 to test the true accuracy of

```

29 #step 2 validate which model is best
30
31 ~ for (i in 1:89){
32   accuracy1 <- (predicted[i]==validation[i,11])
33   accuracy <- c(accuracy,accuracy1)
34 ~ }
35 print(paste0("Accuracy for k = ", k, ": ", sum(100*accuracy/89)))
36 ~ }
37
38 #the validity at k = 5 and k=15-20 are optimal showing the highest accuracy at 83.14
39 #for the sake of true validation, I will choose k = 15 as my optimal k
40
41 #Step 3, I now test the accuracy of the optimal model using the test data set

```

45:19 (Top Level) ↕

Console	Terminal ×	Background Jobs ×
 R 4.2.2 · ~/		
<pre> [1] "Accuracy for k = 1: 76.4044943820224" [1] "Accuracy for k = 2: 76.4044943820224" [1] "Accuracy for k = 3: 76.4044943820224" [1] "Accuracy for k = 4: 76.4044943820224" [1] "Accuracy for k = 5: 83.1460674157302" [1] "Accuracy for k = 6: 82.0224719101122" [1] "Accuracy for k = 7: 82.0224719101122" [1] "Accuracy for k = 8: 80.8988764044943" [1] "Accuracy for k = 9: 80.8988764044943" [1] "Accuracy for k = 10: 82.0224719101122" [1] "Accuracy for k = 11: 82.0224719101122" [1] "Accuracy for k = 12: 82.0224719101122" [1] "Accuracy for k = 13: 82.0224719101122" [1] "Accuracy for k = 14: 82.0224719101122" [1] "Accuracy for k = 15: 83.1460674157302" [1] "Accuracy for k = 16: 83.1460674157302" [1] "Accuracy for k = 17: 83.1460674157302" </pre>		

4. Finally, i tested the model against the test data set and got that my true accuracy was 41%, which was pretty low!

```

#Step 3, I now test the accuracy of the optimal model using the test data set

model1 <- kkn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,training,test,k=15,scale = TRUE)
predicted <- as.integer(fitted.values(model)+0.5)
sum(predicted)/107

```

Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

Answer:

In my role as an Analytics professional for an advertising agency, it is often necessary to figure out distinct targeting segments to serve separate advertising campaigns to. You can use predictors like geo, income, race, gender, and age to form various cluster groups. There are many data sets that contain this information at an individual level. For

example, census like data. Once you have created your clusters, you need a matching key to be able to target the individuals in the cluster groups with advertising. There are many companies who specialize in providing this key. Essentially, they can map from census like data (demographic information on an individual) to the same individuals device or cookie id. This allows you to take your clusters and serve them with online advertising.

Question 4.2

The *iris* data set `iris.txt` contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with `iris` once the library is loaded. It is also available at the UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/Iris>). *The response values are only given to see how well a specific method performed and should not be used to build the model.*

Use the R function `kmeans` to cluster the points as well as possible. Report the best combination of predictors, your suggested value of `k`, and how well your best clustering predicts flower type.

Answer:

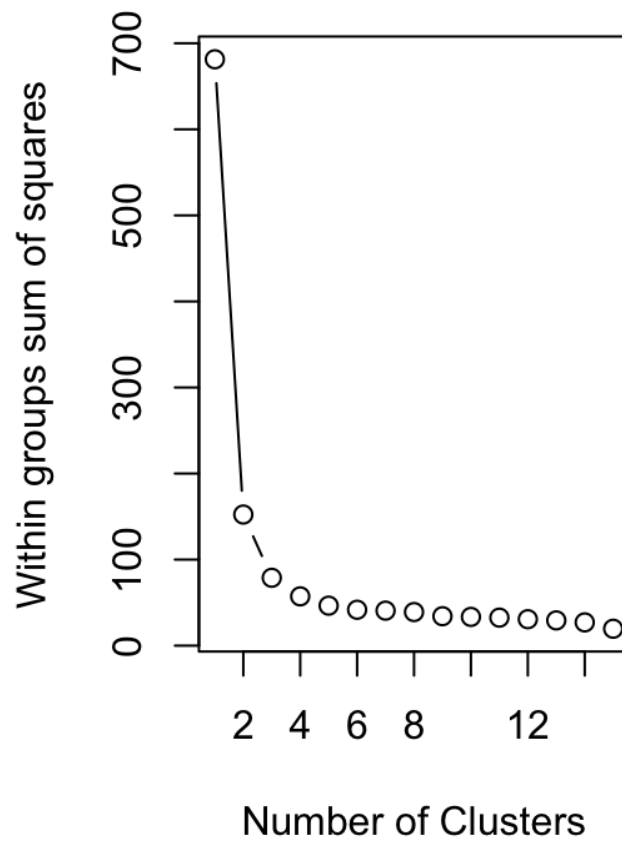
(Full code at end of answer)

(Note: in researching this, i stumbled on a youtube video which was very helpful, but i cross-checked and validated everything from the video – did not blindly copy)

For this question, i had to spend some time researching how to do the elbow plot. I found this website (<https://rpubs.com/violetgirl/201598>) which gives the code for a `wssplot` function that worked to create an elbow plot.

Steps:

1. loaded libraries `stats`, `dplyr`, `ggplot2`, and `ggfortify`
2. Loaded *iris* data set (excluded actuals, last column)
3. Ran code for `wssplot` to create the elbow graph (below)



From the elbow graph, i was able to determine that the optimal number of clusters was 3. Three is where the marginal return starts to diminish.

Next, i ran the `kkmeans` function on the iris data set using 3 clusters. I then used `autoplot` to plot the clustering



Finally, i printed a table of the cluster centers to make sure that all of the dimensions were distinct for each cluster, which they reasonably were.

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	6.850000	3.073684	5.742105	2.071053
2	5.901613	2.748387	4.393548	1.433871
3	5.006000	3.428000	1.462000	0.246000

```
irisdata <-select(iris,c(1,2,3,4))

wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")
  wss
}

wssplot(irisdata)

kmeans = kmeans(irisdata,3)

autoplot(kmeans,irisdata,frame=TRUE)

kmeans$centers
```