Question 3.1

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); and

For this question and with KNN models, I decided to use both the leave-one-out cross-validation (LOOCV) technique and the k-fold cross-validation (KCV) technique. When I was using the LOOCV technique, I was able to obtain the following accuracy values: For the training set, I had an accuracy of 0.8546845 and for the testing set, I had an accuracy of 0.8320611. The code for this can be found in appendix 3.1.A.1. As expected, the training accuracy was higher than the testing accuracy, although the two values were not drastically different which made me think that this was a good model. These were also the highest accuracy values possible for me to attain and they came when I set the k-value to 19. An interesting note for this situation is that I while the in-built function train.kknn gave me a "best" k-value of 48 (I had set kmax to 100), it was only when I iterated through the predicted fitted values for each k-value and manually calculated the accuracy associated with each that I was able to find the true best value in this scenario which was 19.

As mentioned before, I also decided to use the KCV technique for which I set the number of partitions to 10. The code for this can be found in appendix 3.1.A.2. I decided to iterate through values of k from 1 to 100 to see which would be best. In doing so, I obtained a maximum training accuracy of 0.8470363, which was obtained when k equaled 8. Then, I ran this model, with k=8, on the testing data I had set aside and obtained a testing accuracy value of 0.8473282, which is very slightly higher than the training accuracy. A possible explanation for this could be because of how the data was sampled (it was randomly sampled). Perhaps using a combination approach of random sampling and rotation when creating the training and testing datasets might have resolved it.

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

For this question, I created a training dataset containing 60% of the data and validation and testing datasets containing 20% each. Following this, I used the kknn function while iterating through values of k from 1 to 20 to obtain the accuracy for each k value. The code for this can be found in appendix 3.1.B.1. Of all the k values, the k value which gave the greatest accuracy on the training dataset (0.8545918) was 14. Although, k values of 5, 6, and 13 gave the next highest accuracy value of 0.8494898. Therefore, to investigate further, I created one model using k=14 and one model using k=6 and tested each on the validation dataset to determine which was the best. The model with k=14 gave a validation accuracy of 0.8167939 which was higher compared to the k=6 model validation accuracy of 0.8015267. Because the k=14 model was the best in the validation dataset, I then used it on the testing dataset where I received an accuracy value of 0.8244275. Surprisingly, the testing accuracy

value was once again slightly higher than the validation dataset. On the other hand, it was still lower than the accuracy obtained with the training dataset. I suspect that the reason why the testing accuracy was higher than the validation accuracy may be because of the random sampling used. As mentioned before, perhaps using a random sampling plus rotation combination when splitting the datasets might resolve this situation in the future.

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.

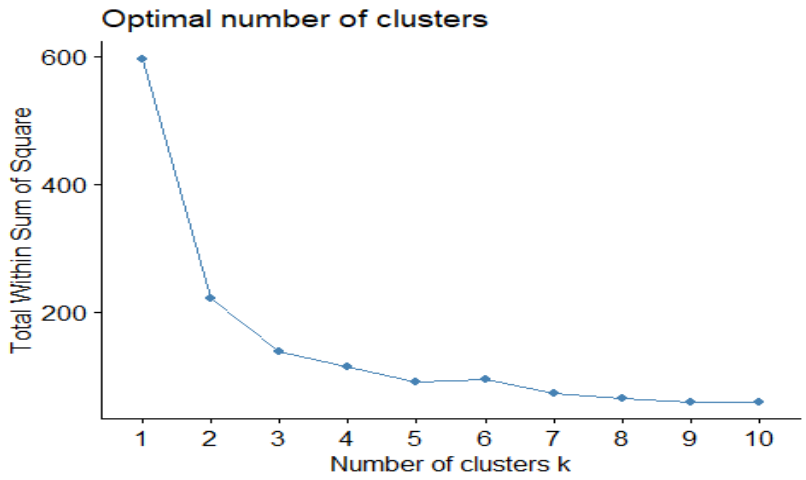
Clustering could be very valuable for me in business, especially when it comes to managing an online store. For instance, if I have a new assortment of potential products that I can add to my store and want to figure out how best to organize them and price them, then I can cluster them based on different predictors to see which ones group together and which ones do not. A few predictors that I might use are: the object's function, the object's size, the object's cost, the object's availability, and the object's seasonality (e.g., if there certain seasons where a product might sell more than in others). By clustering products with these predictors, it will be possible for me to figure out which groups of products share which attributes and then I can market and sell them accordingly.

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.

For this problem, I worked on it in two ways. I created one approach with scaled data and one without. For the approach with scaling (which can be found in appendix 4.2.1), my suggested value was k=3. Using three clusters seemed to give the best output and made the most sense based on the categorical responses that were provided when checked against afterwards. To begin, in my approach I used the fviz_nbclust() function from the factoextra library to generate an elbow plot:

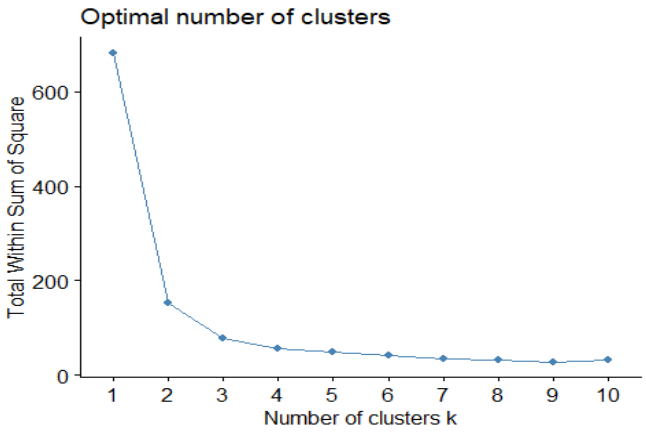


Using this plot, it was possible to see that after k=3, the decreases in total within sum of squares was minimal. That is one reason that I chose to use k=3. Another reason comes from the calculation of percent quality of partition (a value I obtained by dividing between sum of squares by the total sum of squares and multiplying by 100):

```
##    Percent quality of partition
## k2                      62.93972
## k3                      76.69658
## k4                      80.98463
```

As you can see, the value for k=3 clusters is still decently high (compared to k=2 clusters) and k=4 is not too different from it. This partitioning ratio essentially indicates the amount of information retained after clustering. However, it is worth noting that as the number of clusters increases, it will necessarily increase the value of this ratio since the partitioning will be better and the between sum of squares value will increase. Therefore, this ratio is just looked at alongside the elbow plot to verify that k=3 is a good choice for number of clusters. Following this, I created a function to calculate the accuracy of my clustering when given a specific selection of predictors. This was to remove noise and identify which combination of predictors gives the best accuracy. My results showed me that using only petal length and petal width gives the best accuracy at 96%. I also compared this 96% accuracy to the value obtained when calculating the accuracy of the function when all predictors are used (83%) and noted that it was much higher. All function outputs for the various predictor combinations are also available in the appendix.

Now, as mentioned before, I did also run this code without scaling the data. The elbow plot for it was as follows:



As can be seen, k=3 appears to be a reasonable number of clusters to choose in this case as well. The most surprising thing for me about this approach was that the accuracy values seemed somewhat inflated overall. For instance, compared to the 83% obtained when all four predictors were used with scaled data, the accuracy with four predictors without scaling was 89%. However, overall, the conclusion reached by both approaches was the same: The best combination of predictors obtained in the without scaling approach was also petal length and petal width. Furthermore, the accuracy value of this combination was also the same at 96%. The code for this approach can be found in appendix 4.2.2.

Appendix

Question 3.1.A.1

```
#This code uses train.kknn and leave-one-out cross validation

#Load library
library(kknn)

#Load data
data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 3.1\\credit_card
_data.txt",
                   stringsAsFactors = F, header = F)

#Set seed
set.seed(1)

#Sample 80% of the data and create training and testing data sets
random_rows <- sample(1:nrow(data), as.integer(0.80*nrow(data)))
train_data <- data[random_rows,]
test_data <- data[-random_rows,]

#Use leave-one-out cross validation on training data set
k_LOOCV <- train.kknn(V11~., train_data, kmax=100, scale = T)
k_LOOCV

##
## Call:
## train.kknn(formula = V11 ~ ., data = train_data, kmax = 100,     scale = T
)
##
## Type of response variable: continuous
## minimal mean absolute error: 0.2014814
## Minimal mean squared error: 0.1082319
## Best kernel: optimal
## Best k: 48

#Iterate through k-values to find best one in k_LOOCV
k_value_accs <- rep(0, 100)

for (x in 1:100){
  a <- fitted(k_LOOCV)[[x]][1:nrow(train_data)]
  b <- rep(0, nrow(train_data))
  for (i in 1:length(a)){
    b[i] <- as.integer(a[i] + 0.5)
  }
  k_value_accs[x] <- sum(b==train_data[,11])/nrow(train_data)
}


k_value_accs
```

```
##   [1] 0.7915870 0.7915870 0.7915870 0.7915870 0.8355641 0.8317400 0.837476
1
##   [8] 0.8413002 0.8413002 0.8470363 0.8451243 0.8413002 0.8451243 0.850860
4
##  [15] 0.8489484 0.8508604 0.8527725 0.8527725 0.8546845 0.8527725 0.848948
4
##  [22] 0.8508604 0.8489484 0.8489484 0.8451243 0.8451243 0.8451243 0.845124
3
##  [29] 0.8451243 0.8451243 0.8432122 0.8432122 0.8432122 0.8413002 0.841300
2
##  [36] 0.8432122 0.8413002 0.8393881 0.8393881 0.8413002 0.8413002 0.841300
2
##  [43] 0.8374761 0.8374761 0.8374761 0.8374761 0.8413002 0.8393881 0.837476
1
##  [50] 0.8374761 0.8393881 0.8374761 0.8355641 0.8374761 0.8374761 0.837476
1
##  [57] 0.8355641 0.8355641 0.8355641 0.8355641 0.8355641 0.8317400 0.831740
0
##  [64] 0.8336520 0.8336520 0.8317400 0.8317400 0.8317400 0.8317400 0.829827
9
##  [71] 0.8298279 0.8298279 0.8298279 0.8298279 0.8336520 0.8336520 0.833652
0
##  [78] 0.8355641 0.8355641 0.8374761 0.8374761 0.8374761 0.8374761 0.837476
1
##  [85] 0.8374761 0.8374761 0.8374761 0.8374761 0.8374761 0.8374761 0.837476
1
##  [92] 0.8374761 0.8374761 0.8374761 0.8355641 0.8374761 0.8374761 0.837476
1
##  [99] 0.8374761 0.8374761
```

```
max(k_value_accs)
```

```
## [1] 0.8546845
```

```
which.max(k_value_accs)
```

```
## [1] 19
```

```
#Double check using information generated from k_LOOCV (where best k is 19)
pred_train <- rep(0, nrow(train_data))

for (i in 1:nrow(train_data)){
  kknn_model_train <- kknn(V11~., train_data[-i,], train_data[i,], k=19, scal
e=T)
  pred_train[i] <- as.integer(fitted(kknn_model_train) + 0.5)
}

accuracy_train <- sum(pred_train==train_data[,11])/nrow(train_data)

#Now to see how a model functions with the test data
pred_test <- rep(0, nrow(test_data))
```

```
for (i in 1:nrow(test_data)){
  kknn_model_test <- kknn(V11~., test_data[-i,], test_data[i,], k=19, scale=T
)
  pred_test[i] <- as.integer(fitted(kknn_model_test) + 0.5)
}

accuracy_test <- sum(pred_test==test_data[,11])/nrow(test_data)
accuracy_test
```

```
## [1] 0.8320611
```

Question 3.1.A.2

```
#This code uses k-fold cross validation

#Load library
library(kknn)

#Load data
data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 3.1\\credit_card
_data.txt",
                   stringsAsFactors = F, header = F)

#Set seed
set.seed(2)

#Sample 80% of the data and create training and testing data sets
random_rows <- sample(1:nrow(data), as.integer(0.80*nrow(data)))
train_data <- data[random_rows,]
test_data <- data[-random_rows,]

#Use k-fold cross-validation with folds set to 10
k_max_accuracy <- rep(0, 100)

for (k in 1:length(k_max_accuracy)) {
  k_KFCV <- cv.kknn(V11~., train_data, kcv = 10, k = k, scale = T)
  k_fit <- rep(0, nrow(train_data))

  for (i in 1:nrow(k_KFCV[[1]])){
    k_fit[i] <- as.integer(k_KFCV[[1]][i, 2] + 0.5)
  }

  accuracy <- sum(k_fit==train_data[,11])/nrow(train_data)
  k_max_accuracy[k] <- accuracy
}

max(k_max_accuracy)
```

```
## [1] 0.8470363
```

```
which.max(k_max_accuracy)
```

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

```
#Using the k-values giving the highest accuracy, run a model on the test data
pred_test <- rep(0, nrow(test_data))

for (i in 1:nrow(test_data)){
  model <- kknn(V11~., test_data[-i,], test_data[i,],
                k=which.max(k_max_accuracy),
                scale=T)
  pred_test[i] <- as.integer(fitted(model) + 0.5)
}

pred_test
```

```
##   [1] 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0
##  [38] 1 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1
1 1 0
## [112] 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
#View the accuracy of the test model
accuracy_test <- sum(pred_test==test_data[,11])/nrow(test_data)
accuracy_test
```

```
## [1] 0.8473282
```

Question 3.1.B.1

```
#Load library
library(kknn)

#Load data
data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 3.1\\credit_card
_data.txt", stringsAsFactors = F, header = F)

#Set seed
set.seed(10)

#Sample 60% of the data and create training set
rr_training <- sample(1:nrow(data), as.integer(0.60*nrow(data)))
train_data <- data[rr_training,]

#Validation and testing Data (20% each)
tv_data <- data[-rr_training,]

rr_remaining <- sample(1:nrow(tv_data), as.integer(0.50*nrow(tv_data)))
```

```r
validation_data <- tv_data[rr_remaining,]

test_data <- tv_data[-rr_remaining,]

#Function to generate k values with highest accuracies for training data

checking_accuracy_train <- function(A) {
  pred <- rep(0, nrow(train_data))

  for (i in 1:nrow(train_data)){
    model <- kknn(V11~., train_data[-i,], train_data[i,], k = A, scale = T)
    pred[i] <- as.integer(fitted(model) + 0.5)
  }

  accuracy <- sum(pred==train_data[,11])/nrow(train_data)
  return(accuracy)

}

acc <- rep(0, 20)
for (A in 1:length(acc)){
  acc[A] <- checking_accuracy_train(A)
}

acc
```

```
##  [1] 0.8214286 0.8214286 0.8214286 0.8214286 0.8494898 0.8494898 0.8469388
##  [8] 0.8469388 0.8494898 0.8469388 0.8469388 0.8469388 0.8494898 0.8545918
## [15] 0.8469388 0.8469388 0.8418367 0.8443878 0.8443878 0.8469388
```

```r
max(acc)
```

```
## [1] 0.8545918
```

```r
which.max(acc)
```

```
## [1] 14
```

```r
#k=14 is the highest
#k values of 5, 6, and 13 are tied for next highest

#Test obtained k values on validation data set
pred1 <- rep(0, nrow(validation_data))

for (i in 1:nrow(validation_data)){
  maxk_model1 <- kknn(V11~., validation_data[-i,], validation_data[i,],
                k=14,
                scale=T)
  pred1[i] <- as.integer(fitted(maxk_model1) + 0.5)
}
```

```
accuracy1 <- sum(pred1==validation_data[,11])/nrow(validation_data)
accuracy1
```

```
## [1] 0.8167939
```

```
pred2 <- rep(0, nrow(validation_data))

for (i in 1:nrow(validation_data)){
  maxk_model2 <- kknn(V11~., validation_data[-i,], validation_data[i,],
                k=6,
                scale=T)
  pred2[i] <- as.integer(fitted(maxk_model2) + 0.5)
}

accuracy2 <- sum(pred2==validation_data[,11])/nrow(validation_data)
accuracy2
```

```
## [1] 0.8015267
```

```
#Since both accuracy1 is higher, I'll use k = 14
pred_test <- rep(0, nrow(test_data))

for (i in 1:nrow(test_data)){
  test_model <- kknn(V11~., test_data[-i,], test_data[i,],
                k=14,
                scale=T)
  pred_test[i] <- as.integer(fitted(test_model) + 0.5)
}

accuracy_test <- sum(pred_test==test_data[,11])/nrow(test_data)
accuracy_test
```

```
## [1] 0.8244275
```

Question 4.2.1

```
#This code will be with scaling

#Load library
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://g
oo.gl/ve3WBa
```

```
#Load data
original_data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 4.2\\ir
is.txt",
                    stringsAsFactors = F, header = T)

#Scaling the data and getting rid of the categorical variables
```

```r
data <- scale(original_data[,1:4])

set.seed(1)

#Generate an elbow plot to see which number of clusters might be the best
fviz_nbclust(data, kmeans, method="wss")

#Based on the results of the elbow plot, k=3 clusters seems best
#But generate a few more to see how they would do
cluster2 <- kmeans(as.matrix(data), 2, nstart=10)
cluster2

## K-means clustering with 2 clusters of sizes 50, 100
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1   -1.0111914   0.8504137    -1.300630  -1.2507035
## 2    0.5055957  -0.4252069     0.650315   0.6253518
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  1
9  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  3
9  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  5
9  60
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2
2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  7
9  80
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  9
9 100
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 11
9 120
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
9 140
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 141 142 143 144 145 146 147 148 149 150
##   2   2   2   2   2   2   2   2   2   2
```

```
##
## Within cluster sum of squares by cluster:
## [1]  47.35062 173.52867
##  (between_SS / total_SS =  62.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(cluster2$cluster, original_data$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         50        50
```

```
cluster3 <- kmeans(as.matrix(data), 3, nstart=10)
cluster3
```

```
## K-means clustering with 3 clusters of sizes 53, 50, 47
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -0.05005221 -0.88042696    0.3465767   0.2805873
## 2  -1.01119138  0.85041372   -1.3006301  -1.2507035
## 3   1.13217737  0.08812645    0.9928284   1.0141287
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  1
9  20
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  3
9  40
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  5
9  60
##   2   2   2   2   2   2   2   2   2   2   3   3   3   1   1   1   3   1
1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  7
9  80
##   1   1   1   1   1   3   1   1   1   1   3   1   1   1   1   3   3   3
1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  9
9 100
##   1   1   1   1   1   3   3   1   1   1   1   1   1   1   1   1   1   1
1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 11
```

```
9 120
##   3   1   3   3   3   3   1   3   3   3   3   3   3   1   1   3   3   3
3   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
9 140
##   3   1   3   1   3   3   1   3   3   3   3   3   3   1   1   3   3   3
1   3
## 141 142 143 144 145 146 147 148 149 150
##   3   3   1   3   3   3   1   3   3   1
##
## Within cluster sum of squares by cluster:
## [1] 44.08754 47.35062 47.45019
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(cluster3$cluster, original_data$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         39        14
##   2     50          0         0
##   3      0         11        36
```

```
cluster4 <- kmeans(as.matrix(data), 4, nstart=10)
cluster4
```

```
## K-means clustering with 4 clusters of sizes 25, 53, 47, 25
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -0.71894419  1.50198969   -1.2972312  -1.2165934
## 2  -0.05005221 -0.88042696    0.3465767   0.2805873
## 3   1.13217737  0.08812645    0.9928284   1.0141287
## 4  -1.30343857  0.19883774   -1.3040289  -1.2848136
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  1
9  20
##   1   4   4   4   1   1   4   4   4   4   1   4   4   4   1   1   1   1
1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  3
9  40
##   1   1   1   4   4   4   4   1   1   4   4   1   1   1   4   4   1   1
4   4
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  5
9  60
```

```
##   1   4   4   1   1   4   1   4   1   4   3   3   3   2   2   2   3   2
2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  7
9  80
##   2   2   2   2   2   3   2   2   2   2   3   2   2   2   2   3   3   3
2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  9
9 100
##   2   2   2   2   2   3   3   2   2   2   2   2   2   2   2   2   2   2
2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 11
9 120
##   3   2   3   3   3   3   2   3   3   3   3   3   3   2   2   3   3   3
3   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
9 140
##   3   2   3   2   3   3   2   3   3   3   3   3   3   2   2   3   3   3
2   3
## 141 142 143 144 145 146 147 148 149 150
##   3   3   2   3   3   3   2   3   3   2
##
## Within cluster sum of squares by cluster:
## [1] 12.147537 44.087545 47.450194  9.646348
##  (between_SS / total_SS =  81.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(cluster4$cluster, original_data$Species)
```

```
##
##     setosa versicolor virginica
##   1     25          0         0
##   2      0         39        14
##   3      0         11        36
##   4     25          0         0
```

```
#For each cluster, look at the quality of the partition
#This ratio indicates the amount of information retained after clustering
quality <- rep(0, 3)

for (i in 1:4) {
  k_model <- kmeans(as.matrix(data), i, nstart=10)
  quality[i] <- (k_model$betweenss / k_model$totss) * 100

}
```

```r
quality_matrix <- as.matrix(quality[2:4])
rownames(quality_matrix) <- c("k2", "k3", "k4")
colnames(quality_matrix) <- "Percent quality of partition"
quality_matrix
```

```
##    Percent quality of partition
## k2                     62.93972
## k3                     76.69658
## k4                     80.98463
```

```r
#Generate a function to find the best combination of predictors
#The accuracy with all four predictors

accuracy_four <- (50 + 39 + 36)/150
accuracy_four
```

```
## [1] 0.8333333
```

```r
#Using only three inputs because we already got the prediction with four abov
e
accuracy <- function(p1, p2, p3){
  if (nargs() == 3) {
    input <- c(p1, p2, p3)
  }
  if (nargs() == 2) {
    input <- c(p1, p2)
  }
  if (nargs() == 1) {
    input <- p1
  }
  model <- kmeans(data[,input], 3, nstart=10)
  table(model$cluster, original_data$Species)
}

#Calculate accuracy
accuracy(1,2,3)
```

```
##
##     setosa versicolor virginica
##   1      1         35        13
##   2      0         15        37
##   3     49          0         0
```

```r
acc1 <- (35 + 37 + 49)/150

accuracy(1,2,4)
```

```
##
##     setosa versicolor virginica
##   1      0         12        34
```

```
##   2     49          0         0
##   3      1         38        16
```

```
acc2 <- (34 + 49 + 38)/150
```

```
accuracy(1,3,4)
```

```
##
##     setosa versicolor virginica
##   1      0         44        14
##   2     50          1         0
##   3      0          5        36
```

```
acc3 <- (44 + 50 + 36)/150
```

```
accuracy(2,3,4)
```

```
##
##     setosa versicolor virginica
##   1     49          0         0
##   2      0          8        38
##   3      1         42        12
```

```
acc4 <- (49 + 38 + 42)/150
```

```
accuracy(1,2)
```

```
##
##     setosa versicolor virginica
##   1      1         36        19
##   2      0         14        31
##   3     49          0         0
```

```
acc5<- (49 + 36 + 31)/150
```

```
accuracy(1,3)
```

```
##
##     setosa versicolor virginica
##   1      0          9        34
##   2     50          4         0
##   3      0         37        16
```

```
acc6<- (50 + 37 + 34)/150
```

```
accuracy(1,4)
```

```
##
##     setosa versicolor virginica
##   1      0         40        15
##   2      0          6        35
##   3     50          4         0
```

```
acc7<- (50 + 40 + 35)/150

accuracy(2,3)

##
##     setosa versicolor virginica
##   1     49          0         0
##   2      0         22        38
##   3      1         28        12

acc8<- (49 + 27 + 40)/150

accuracy(2,4)

##
##     setosa versicolor virginica
##   1      0         15        39
##   2     49          0         0
##   3      1         35        11

acc9<- (49 + 39 + 35)/150

accuracy(3,4)

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

acc10<- (50 + 48 + 46)/150

accuracy(1)

##
##     setosa versicolor virginica
##   1      5         35        23
##   2     45          6         1
##   3      0          9        26

acc11<- (47 + 22 + 31)/150

accuracy(2)

##
##     setosa versicolor virginica
##   1     31          1         5
##   2      1         27        19
##   3     18         22        26
```

```
acc12<- (31 + 21 + 34)/150

accuracy(3)

##
##     setosa versicolor virginica
##   1      0         48         6
##   2     50          0         0
##   3      0          2        44

acc13<- (48 + 44 + 50)/150

accuracy(4)

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

acc14<-  (48 + 46 + 50)/150

acc_vec <- c(acc1, acc2, acc3, acc4, acc5, acc6, acc7, acc8, acc9, acc10, acc
11, acc12, acc13, acc14)
max(acc_vec)

## [1] 0.96

which.max(acc_vec)

## [1] 10

#Using predictors 3 and 4 (petal length and width) gives the best results
```

Question 4.2.2

```
#This code will be without scaling

#Load library
library(factoextra)

#Load data
original_data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 4.2\\ir
is.txt",
                   stringsAsFactors = F, header = T)

#Scaling the data and getting rid of the categorical variables
data <- original_data[,1:4]

set.seed(2)
```

```r
#Generate an elbow plot to see which number of clusters might be the best
fviz_nbclust(data, kmeans, method="wss")

#Based on the results of the elbow plot, k=3 clusters seems best
#But generate a few more to see how they would do
cluster2 <- kmeans(as.matrix(data), 2, nstart=10)
cluster2
```

```
## K-means clustering with 2 clusters of sizes 53, 97
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.005660    3.369811     1.560377    0.290566
## 2     6.301031    2.886598     4.958763    1.695876
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  1
9  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  3
9  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  5
9  60
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   1
2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  7
9  80
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  9
9 100
##   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2
1   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 11
9 120
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
9 140
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 141 142 143 144 145 146 147 148 149 150
##   2   2   2   2   2   2   2   2   2   2
##
## Within cluster sum of squares by cluster:
## [1]  28.55208 123.79588
##  (between_SS / total_SS =  77.6 %)
```

```
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"

table(cluster2$cluster, original_data$Species)

##
##     setosa versicolor virginica
##   1     50          3         0
##   2      0         47        50

cluster3 <- kmeans(as.matrix(data), 3, nstart=10)
cluster3

## K-means clustering with 3 clusters of sizes 62, 50, 38
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.901613    2.748387     4.393548    1.433871
## 2     5.006000    3.428000     1.462000    0.246000
## 3     6.850000    3.073684     5.742105    2.071053
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  1
9  20
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  3
9  40
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  5
9  60
##   2   2   2   2   2   2   2   2   2   2   1   1   3   1   1   1   1   1
1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  7
9  80
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   3
1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  9
9 100
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 11
9 120
##   3   1   3   3   3   3   1   3   3   3   3   3   3   1   1   3   3   3
3   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
```

```
9 140
##   3   1   3   1   3   3   1   1   3   3   3   3   3   1   3   3   3   3
1   3
## 141 142 143 144 145 146 147 148 149 150
##   3   3   1   3   3   3   1   3   3   1
##
## Within cluster sum of squares by cluster:
## [1] 39.82097 15.15100 23.87947
##  (between_SS / total_SS =  88.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"

table(cluster3$cluster, original_data$Species)

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

cluster4 <- kmeans(as.matrix(data), 4, nstart=10)
cluster4

## K-means clustering with 4 clusters of sizes 32, 40, 50, 28
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     6.912500    3.100000     5.846875    2.131250
## 2     6.252500    2.855000     4.815000    1.625000
## 3     5.006000    3.428000     1.462000    0.246000
## 4     5.532143    2.635714     3.960714    1.228571
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  1
9  20
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
3   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  3
9  40
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
3   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  5
9  60
##   3   3   3   3   3   3   3   3   3   3   2   2   2   4   2   4   2   4
2   4
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  7
9  80
```

```
##   4   4   4   2   4   2   4   4   2   4   2   4   2   2   2   2   2   2
2   4
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  9
9 100
##   4   4   4   2   4   2   2   2   4   4   4   2   4   4   4   4   4   2
4   4
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 11
9 120
##   1   2   1   1   1   1   4   1   1   1   2   2   1   2   2   1   1   1
1   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
9 140
##   1   2   1   2   1   1   2   2   1   1   1   1   1   2   2   1   1   1
2   1
## 141 142 143 144 145 146 147 148 149 150
##   1   1   2   1   1   1   2   2   1   2
##
## Within cluster sum of squares by cluster:
## [1] 18.703437 13.624750 15.151000  9.749286
##  (between_SS / total_SS =  91.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withi
nss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(cluster4$cluster, original_data$Species)
```

```
##
##     setosa versicolor virginica
##   1      0          0        32
##   2      0         23        17
##   3     50          0         0
##   4      0         27         1
```

```
#For each cluster, look at the quality of the partition
#This ratio indicates the amount of information retained after clustering
quality <- rep(0, 3)

for (i in 1:4) {
  k_model <- kmeans(as.matrix(data), i, nstart=10)
  quality[i] <- (k_model$betweenss / k_model$totss) * 100

}


quality_matrix <- as.matrix(quality[2:4])
rownames(quality_matrix) <- c("k2", "k3", "k4")
```

```r
colnames(quality_matrix) <- "Percent quality of partition"
quality_matrix
```

```
##    Percent quality of partition
## k2                     77.64096
## k3                     88.42753
## k4                     91.60098
```

```r
#Generate a function to find the best combination of predictors
#The accuracy with all four predictors

accuracy_four <- (50 + 48 + 36)/150
accuracy_four
```

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

```r
#Using only three inputs because we already got the prediction with four abov
e
accuracy <- function(p1, p2, p3){
  if (nargs() == 3) {
    input <- c(p1, p2, p3)
  }
  if (nargs() == 2) {
    input <- c(p1, p2)
  }
  if (nargs() == 1) {
    input <- p1
  }
  model <- kmeans(data[,input], 3, nstart=10)
  table(model$cluster, original_data$Species)
}

#Calculate accuracy
accuracy(1,2,3)
```

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

```r
acc1 <- (50 + 45 + 37)/150

accuracy(1,2,4)
```

```
##
##     setosa versicolor virginica
##   1      0         11        35
##   2      0         39        15
##   3     50          0         0
```

```r
acc2 <- (50 + 39 + 35)/150

accuracy(1,3,4)
```

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

```r
acc3 <- (50 + 48 + 36)/150

accuracy(2,3,4)
```

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

```r
acc4 <- (50 + 48 + 45)/150

accuracy(1,2)
```

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

```r
acc5<- (50 + 38 + 35)/150

accuracy(1,3)
```

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

```r
acc6<- (50 + 45 + 37)/150

accuracy(1,4)
```

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

```
acc7<- (50 + 40 + 35)/150

accuracy(2,3)

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

acc8<- (50 + 37 + 37)/150

accuracy(2,4)

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

acc9<- (49 + 46 + 44)/150

accuracy(3,4)

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

acc10<- (50 + 48 + 46)/150

accuracy(1)

##
##     setosa versicolor virginica
##   1      5         35        23
##   2     45          6         1
##   3      0          9        26

acc11<- (47 + 22 + 31)/150

accuracy(2)

##
##     setosa versicolor virginica
##   1     31          1         5
##   2      1         27        19
##   3     18         22        26
```

```r
acc12<- (31 + 21 + 34)/150

accuracy(3)

##
##     setosa versicolor virginica
##   1      0         48         6
##   2     50          0         0
##   3      0          2        44

acc13<- (48 + 44 + 50)/150

accuracy(4)

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

acc14<-  (48 + 46 + 50)/150

acc_vec <- c(acc1, acc2, acc3, acc4, acc5, acc6, acc7, acc8, acc9, acc10, acc
11,
          acc12, acc13, acc14)
max(acc_vec)

## [1] 0.96

which.max(acc_vec)

## [1] 10

#Using predictors 3 and 4 (petal length and width) gives the best results
```