Question 3.1

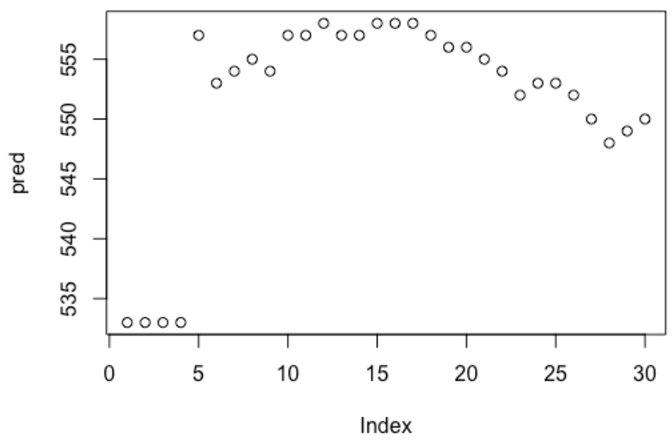
Using the same data set (`credit_card_data.txt` or `credit_card_data-headers.txt`) as in Question 2.2, use the `ksvm` or `kknn` function to find a good classifier:

1. (a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional); and
2. (b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

Answer 3.1a

```
# Clear global environment, load package, load and preview dataset
> rm(list = ls())
> library(kknn)
> df1 <- read.delim("/Users/…/credit_card_data-headers.txt", header = T)
> head(df1)
  A1    A2    A3   A8 A9 A10 A11 A12 A14 A15 R1
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

# Build model with train.kknn method to achieve cross-validation with maximum value of k set
to 30, data is scaled.
> model1 <- train.kknn(R1~., data = df1, kmax = 30, scale = T)

# Set variable x to be zero vector
> x <- rep(0, 30)

# Calculate fitted model predicted results for each k from 1 to 30 (maximum value of k). Store
results into variable pred.
> for (k in 1:30){
+    x <- round(fitted(model1)[[k]])
+    pred[i] <- sum(x == df1$R1)
+ }

# Seeing the predicted results, when k = 12, 15, 16, 17, the model achieves highest accuracy.
> pred
 [1] 533 533 533 533 557 553 554 555 554 557 557 558 557 557 558 558 558 557 556
[20] 556 555 554 552 553 553 552 550 548 549 550
> max(pred)
```

```
[1] 558
> which.max(pred)
[1] 12
> plot(pred)
```



Answer 3.1b

```
# Clear global environment, load package, load and preview dataset
> rm(list = ls())
> library(kknn)
> df1 <- read.delim("/Users/…/credit_card_data-headers.txt", header = T)
> head(df1)
  A1    A2    A3   A8 A9 A10 A11 A12 A14 A15 R1
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

# Create training, validation and test dataset by dividing the original dataset into 80%, 10%,
10% respectively.
> df2 <- sample(nrow(df1), size = floor(nrow(df1) * .8))
> df3 <- sample(nrow(df1[-df2,]), size = floor(nrow(df1[-df2,])/2))
> train <- df1[df2,]
> val <- df1[-df2,][df3,]
> test <- df1[-df2,][-df3,]
```

# Set variable x to be zero vector, then fit kknn model with validation dataset to find the predicted results for k value set from 1 to 30. According to the results, the optimal k value for the validation dataset is at 10, with accuracy at 87.69%.

```
> x <- rep(0,30)
> for (k in 1:30){
+     model2 <- kknn(R1~., train, val, k = k, scale = T)
+     pred <- round(fitted(model2))
+     x[k] <- sum(pred == val$R1) / nrow(val)
+ }
> x
 [1] 0.8307692 0.8307692 0.8307692 0.8307692 0.8307692 0.8307692 0.8307692
 [8] 0.8615385 0.8615385 0.8769231 0.8769231 0.8769231 0.8769231 0.8769231
[15] 0.8769231 0.8769231 0.8769231 0.8769231 0.8615385 0.8615385 0.8615385
[22] 0.8615385 0.8615385 0.8769231 0.8769231 0.8769231 0.8769231 0.8769231
[29] 0.8769231 0.8769231
> which.max(x)
[1] 10
```

# kknn model was fitted on test dataset to find the predicted result when k is at 10. This time around, the model accuracy on the test dataset is 83.33%

```
> model3 <- kknn(R1~., train, test, k = which.max(x), scale = T)
> pred2 <- round(fitted(model3))
> sum(pred2 == test$R1) / nrow(test)
[1] 0.8333333
```

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 4.1

My work in a manufacturing company involves analyzing various suppliers and the performance of the manufacturing yield, and make sure optimal transaction volume are assigned to suppliers. Clustering model can be applied to assess the similarities among suppliers given the same material. In order to achieve this, we create portfolio for each material using existing transaction data. Factors include brand, manufacturing quantities, production yields, molding quantities, etc. Factors that share similarities would be clustered together within the portfolio.

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 4.2

# Clear global environment, load and preview dataset. There are three types of species as the response in the given dataset.

```
> rm(list = ls())
> df1 <- read.table("/Users/…/iris.txt", header = T)
> head(df1)
    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
> df1$Species
Levels: setosa versicolor virginica
```

# Given three types of species, I performed kmeans with 2-4 clusters to see which number best fit the dataset.

```
> model1 <- kmeans(df1[,1:4], 2, nstart = 30)
> model2 <- kmeans(df1[,1:4], 3, nstart = 30)
> model3 <- kmeans(df1[,1:4], 4, nstart = 30)
```

# Set x to be zero vector, then calculate the sum from points to the cluster centers. The goal is to find a suitable number of clusters for which the sum is small enough so that cluster centers are close to the data points. Results show that when cluster is 2, 3, 4, the distance sum is 128, 97, 83 accordingly.

```
> x1 <- 0
> for (i in 1:nrow(df1)){
+    x1 = x1 + dist(rbind(df1[i, 1:4], model1$centers[model1$cluster[i],]))
+ }
> x2 <- 0
> for (i in 1:nrow(df1)){
+     x2 = x2 + dist(rbind(df1[i, 1:4], model2$centers[model2$cluster[i],]))
+ }
> x3 <- 0
> for (i in 1:nrow(df1)){
+     x3 = x3 + dist(rbind(df1[i, 1:4], model3$centers[model3$cluster[i],]))
+ }
> x1[1]
[1] 128.3367
> x2[1]
[1] 97.20457
> x3[1]
[1] 83.60772
```

# Now compare sum from points to cluster centers using Sepal Length and Sepal Width combination. With cluster set to 2 to 4, distance sum is 440, 430 and 429 accordingly.

```
> model1_1 <- kmeans(df1[,1:2], 2, nstart = 30)
> model2_1 <- kmeans(df1[,1:2], 3, nstart = 30)
> model3_1 <- kmeans(df1[,1:2], 4, nstart = 30)
> x1_1 <-0
> for (i in 1:nrow(df1)){
+     x1_1 = x1_1 + dist(rbind(df1[i, 1:4], model1_1$centers[model1_1$cluster[i],]))
+ }
> x2_1 <- 0
> for (i in 1:nrow(df1)){
+     x2_1 = x2_1 + dist(rbind(df1[i, 1:4], model2_1$centers[model2_1$cluster[i],]))
+ }
> x3_1 <- 0
> for (i in 1:nrow(df1)){
+     x3_1 = x3_1 + dist(rbind(df1[i, 1:4], model3_1$centers[model3_1$cluster[i],]))
+ }
> x1_1[1]
[1] 440.3481
> x2_1[1]
[1] 430.8609
> x3_1[1]
[1] 429.8963
```

# Now compare sum from points to cluster centers using Petal Length and Petal Width combination. With cluster set to 2 to 4, distance sum is 445, 431 and 427 accordingly.

```
> model1_2 <- kmeans(df1[,3:4], 2, nstart = 30)
> model2_2 <- kmeans(df1[,3:4], 3, nstart = 30)
> model3_2 <- kmeans(df1[,3:4], 4, nstart = 30)
> x1_2 <- 0
> for (i in 1:nrow(df1)){
+     x1_2 = x1_2 + dist(rbind(df1[i, 1:4], model1_2$centers[model1_2$cluster[i],]))
+ }
> x2_2 <- 0
> for (i in 1:nrow(df1)){
+     x2_2 = x2_2 + dist(rbind(df1[i, 1:4], model2_2$centers[model2_2$cluster[i],]))
+ }
> x3_2 <- 0
> for (i in 1:nrow(df1)){
+     x3_2 = x3_2 + dist(rbind(df1[i, 1:4], model3_2$centers[model3_2$cluster[i],]))
+ }
> x1_2[1]
[1] 445.2832
> x2_2[1]
[1] 431.7237
> x3_2[1]
[1] 427.8338
```

# In three combination cases, k = 3 has significantly better distribution over k = 2 and slightly better distribution over k = 4. When k = 3 with Petal Length and Petal Width combination, each of three species has nice cluster with cluster sum of squares by cluster being 94.3%

```
> table(model2_2$cluster, df1$Species)

    setosa versicolor virginica
  1      0          2        46
  2     50          0         0
  3      0         48         4
> model2_2
K-means clustering with 3 clusters of sizes 50, 52, 48

Cluster means:
  Petal.Length Petal.Width
1     1.462000    0.246000
2     4.269231    1.342308
3     5.595833    2.037500

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  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  39  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  59  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  79  80
```

```
  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   3   2   2 
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 
  2   2   2   3   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2 
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 
  3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   2 
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 
  3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   2   3 
141 142 143 144 145 146 147 148 149 150 
  3   3   3   3   3   3   3   3   3   3 

Within cluster sum of squares by cluster:
[1]  2.02200 13.05769 16.29167
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```