Homework: Week 1

Question 2.1

Answer:

A classification model could probably be used to predict if a candidate will be admitted to Georgia Tech's Online Master of Science in Analytics program. (This idea occurs to me since I plan to apply for this program in the future.)
The predictors could include: The candidate's age, highest degree of education, GPA at undergraduate school, undergrad majors, number of years of full-time work experience, etc.

Question 2.2

Question 2.2.1

Answer:

First, we read the data for this question from text file and print the first 5 rows.

```
    #Read data for Question 2.2
    data = read.delim("C:\\Data\\week 1 data-summer\\data 2.2\\credit_card_data-headers.txt",
                      header = TRUE)
    print(data[1:5,])
```

```
##   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
```

Then we need to apply support vector machine model by running the code provided in homework PDF document. The key to success in this question is to determine the the right order of magnitude for C.

Here we will plug in several different values of C into the model, varying from 1*10^(-8) to 1*10^8. We will measure the prediction accuracy of this model under different C values by calculating what fraction of the model's predictions match the actual classification.

We will visualize the prediction accuracy under different C values in a line chart.

```
    #Load kernlab library
    library(kernlab)

    ksvm_model <- function(kernel_name){
      #Define C values we want to test
      cvalues = c()
       label <- c()
```

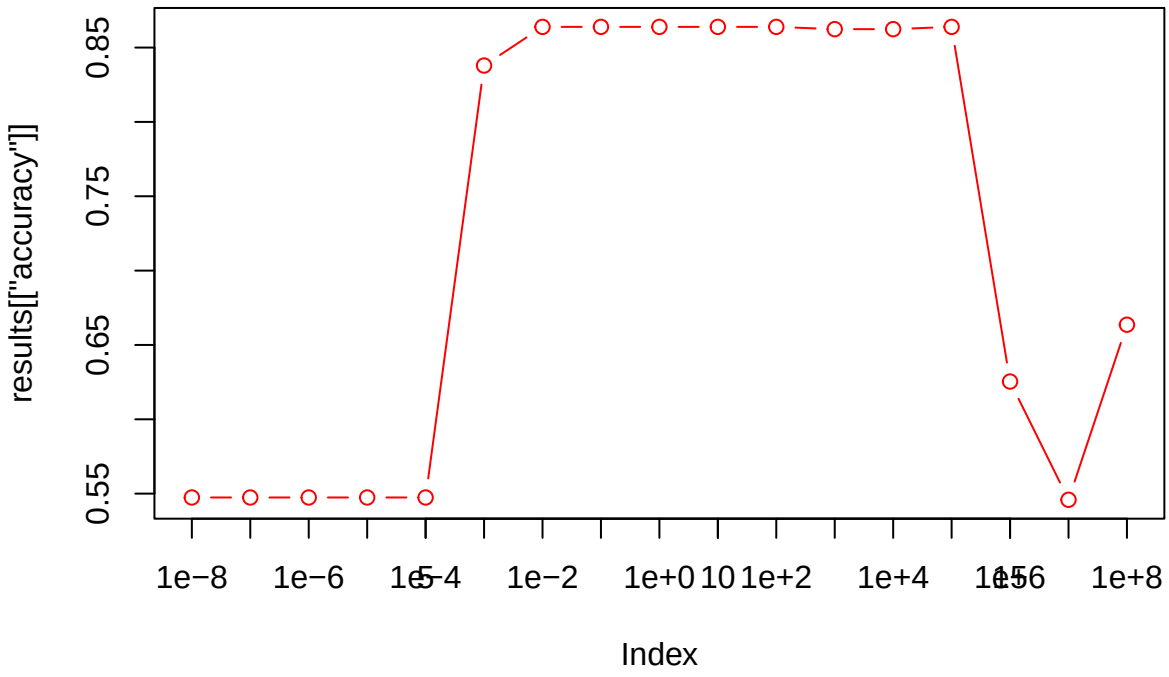
```
    for (i in -8:8){
      cvalues <-append(cvalues, 1 * 10^(i))

      if (i>=0){label <- append(label, paste("1e+",as.character(i),sep = ''))}
        else {label <- append(label, paste("1e",as.character(i),sep = ''))}
    }

    #call ksvm. with differert C values
    accuracy <- c()
    kernel <- c()

    for (c in cvalues){
      model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type="C-svc",
                  kernel=kernel_name, C=c,scaled=TRUE)

      # calculate a1...am
      a <- colSums(model@xmatrix[[1]] * model@coef[[1]])

      # calculate a0
      a0 <- -model@b

      # calculate what the model predicts
      pred <- predict(model,data[,1:10])

      # calculate what fraction of the model's predictions match the actual classification and store
      accuracy <- append(accuracy,sum(pred == data[,11]) / nrow(data))
      kernel <- append(kernel, kernel_name)
    }

    results = data.frame("kernel" = kernel, "c" = cvalues, "label" = label,
                         "accuracy" = accuracy)
    return (results)
  }

  results = ksvm_model("vanilladot")
```

```
  #Visualize results
  print(results)
```

```
##        kernel     c label  accuracy
## 1  vanilladot 1e-08  1e-8 0.5474006
## 2  vanilladot 1e-07  1e-7 0.5474006
## 3  vanilladot 1e-06  1e-6 0.5474006
## 4  vanilladot 1e-05  1e-5 0.5474006
## 5  vanilladot 1e-04  1e-4 0.5474006
## 6  vanilladot 1e-03  1e-3 0.8379205
## 7  vanilladot 1e-02  1e-2 0.8639144
## 8  vanilladot 1e-01  1e-1 0.8639144
## 9  vanilladot 1e+00  1e+0 0.8639144
## 10 vanilladot 1e+01  1e+1 0.8639144
## 11 vanilladot 1e+02  1e+2 0.8639144
## 12 vanilladot 1e+03  1e+3 0.8623853
## 13 vanilladot 1e+04  1e+4 0.8623853
```

```
## 14 vanilladot 1e+05  1e+5 0.8639144
## 15 vanilladot 1e+06  1e+6 0.6253823
## 16 vanilladot 1e+07  1e+7 0.5458716
## 17 vanilladot 1e+08  1e+8 0.6636086
```

```
    plot(results[["accuracy"]], cex = 1, col = "red", type = "b")
    axis(1, at = 1:nrow(results) ,labels=results[["label"]])
```



The table and graph above shows the prediction accuracy of SVM model under different C values. From the graph, we can see the accuracy of model reaches the highest level (around 86.39%) since C = 1*10^(-2) and remains at the same level until C>1*10^(5)

Considering the following formula that we want to minimize:

$$\underset{a_0,\ldots,a_m}{\text{Minimize}} \sum_{j=1}^{n} \max\left\{0,1 - \left(\sum_{i=1}^{m} a_i x_{ij} + a_0\right) y_j\right\} + \lambda \sum_{i=1}^{m} (a_i)^2$$

We know that when C gets larger, the important of margin becomes greater. When C is smaller, the importance of margin gets smaller.

Since we want to avoid the issue of overfitting, we want to maximize the margin of SVM model as much as possible. Therefore, we would choose C=1*10(5) as our C value, because this is the largest C value we can use while keeping the prediction accuracy at the highest level.

We can plug in C=100000 to the model and get the parameters a0, a1, a2...am as follows:

```
    model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type="C-svc",
                  kernel="vanilladot", C=100000,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```
    # calculate a1...am
    a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
    # calculate a0
    a0 <- -model@b
    # calculate what the model predicts
    pred <- predict(model,data[,1:10])

    a
```

```
##           A1           A2           A3           A8           A9          A10
## -0.004117738 -0.086896089  0.129715260 -0.083744032  0.988381368  0.031253888
##          A11          A12          A14          A15
## -0.055666972 -0.037281856  0.021940744  0.018521785
```

```
    a0
```

```
## [1] 0.08054451
```

```
    sum(pred == data[,11]) / nrow(data)
```

```
## [1] 0.8639144
```

The formula of the classifier will look like the following (variable A1, A2... A15 have been scaled):

$$W^T \times A + b = 0$$

Where W and A are vectors and b is a number:

$$W = [-0.004117738, \\ -0.086896089, \\ 0.129715260, \\ -0.083744032, \\ 0.988381368, \\ 0.031253888, \\ -0.055666972, \\ -0.037281856, \\ 0.021940744, \\ 0.018521785]$$

$$A = [A1, A2, A3, A8, A9, A10, A11, A12, A14, A15]$$

$$b = 0.08054451$$

Question 2.2.3

Answer:

I referred to the following post to determine the method of select the right k value: https://stats.stackexchange.com/a/126138

The strategy is: we will iterate all the values between k=1 and k=40 (hope 40 is a number big enough) and measure the model's prediction accuracy under different k by calculating what fraction of the model's predictions match the actual classification. Then we will visualize the results and choose the largest k before the model's prediction accuracy noticeably drops.

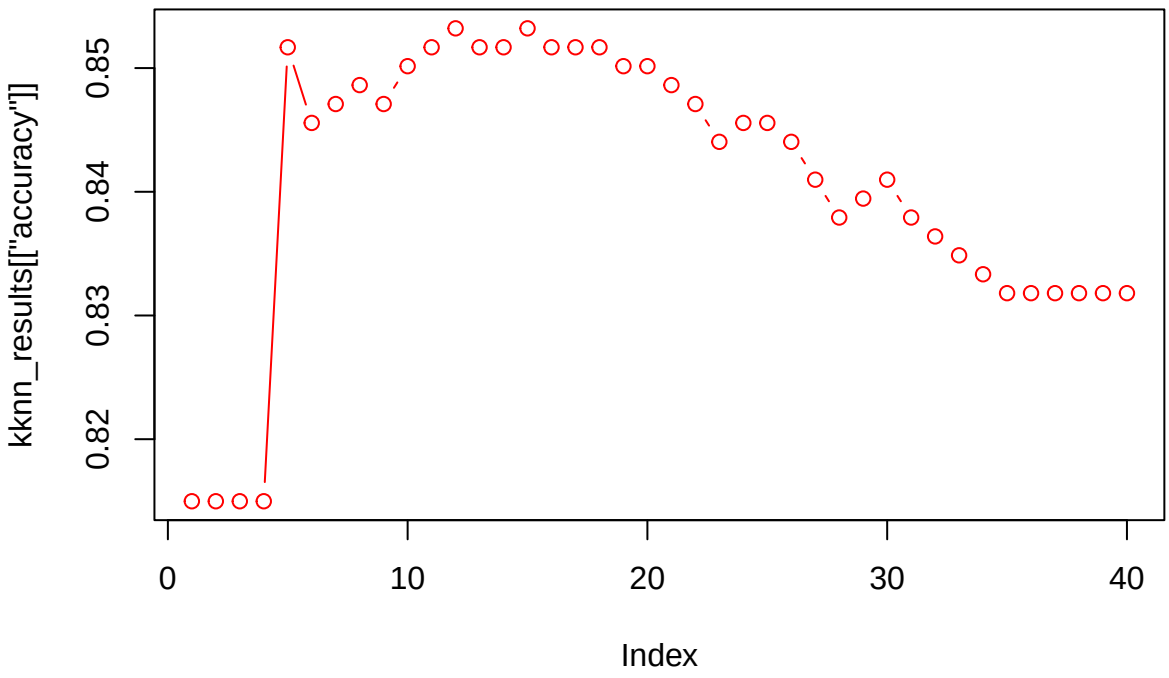
Note: the prediction result returned by kknn will be continuous. We use round() function to convert the results into integers

```
library(kknn)

data = read.delim("C:\\Data\\week 1 data-summer\\data 2.2\\credit_card_data-headers.txt",
                  header = TRUE)

kknn_accuracy <- function(kk){
  # Make predictions of each data point contained in the dataset
  predictions <- c()
  for (i in 1:nrow(data)){
    #set up train / test dataset
    #excluding data point i from training datasets and use it as testing dataset
    training <- data[-i,]
    testing <- data[i,]
    #apply KNN model
    kknn_model <- kknn(R1~., training, testing, k = kk, scale = TRUE)
    predictions <- append(predictions,round(fitted(kknn_model)))
  }
  accuracy = sum(predictions == data[,11])/nrow(data)
  return (accuracy)
}

#Summarize and visualize the prediction accuracy for each k value between 1-40
kknn_accu = c()
k = c()
for (kk in 1:40){
  accuracy = kknn_accuracy(kk)
  k = append(k,kk)
  kknn_accu = append(kknn_accu, accuracy)
}
kknn_results = data.frame("k"=k, "accuracy"=kknn_accu)
print(kknn_results)
```

```
##     k  accuracy
## 1   1 0.8149847
## 2   2 0.8149847
## 3   3 0.8149847
## 4   4 0.8149847
## 5   5 0.8516820
## 6   6 0.8455657
## 7   7 0.8470948
## 8   8 0.8486239
```

```
## 9   9 0.8470948
## 10 10 0.8501529
## 11 11 0.8516820
## 12 12 0.8532110
## 13 13 0.8516820
## 14 14 0.8516820
## 15 15 0.8532110
## 16 16 0.8516820
## 17 17 0.8516820
## 18 18 0.8516820
## 19 19 0.8501529
## 20 20 0.8501529
## 21 21 0.8486239
## 22 22 0.8470948
## 23 23 0.8440367
## 24 24 0.8455657
## 25 25 0.8455657
## 26 26 0.8440367
## 27 27 0.8409786
## 28 28 0.8379205
## 29 29 0.8394495
## 30 30 0.8409786
## 31 31 0.8379205
## 32 32 0.8363914
## 33 33 0.8348624
## 34 34 0.8333333
## 35 35 0.8318043
## 36 36 0.8318043
## 37 37 0.8318043
## 38 38 0.8318043
## 39 39 0.8318043
## 40 40 0.8318043
```

```
    plot(kknn_results[['accuracy']], cex = 1, col = "red", type = "b")
```



We can see from the table and graph above that the model reaches the highest level of accuracy (85.32%) at k=12. (Then the accuracy starts to decrease as k increases. Therefore k=12 may be a good value in this model.

Question 3.1

Answer:

In the last exercise, i think we are essentially already doing the cross validation for the KNN model. We basically created as many validation data sets as the total number of rows in the data frame and did the cross validation.

In this excercise, i am going to adopt the k-fold cross validation approach by randomly dividing the original data set into 5 different sub data sets. We will use the first four data sets for training and cross validation. We are going to use the 5th data set to calculate the prediction accuracy of our model (testing). A KNN model will be adopted in this exercise.

First, we need to divide the data into 5 data set. Here's the approach:

For each of the 654 rows in the original data sets, we need to assign it into one of the 5 sub data sets. We will assign row 1 into data set 1, row 2 into data set 2, row 3 into data set 3, row 4 into data set 4 and row 5 into data set 5. Then we will start over again from assigning row 6 to data set 1... Until we finish assigning all the rows.

```
#Load dataset
data <- read.delim("C:\\Data\\week 1 data-summer\\data 3.1\\credit_card_data-headers.txt",
                   header = TRUE)
print(data[1:5,])
```

```
##   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
```

```
    #divide data into 5 groups
    nr <- nrow(data)
    datasets <- split(data, rep(1:5, each=1, length.out=nr))

    train_and_val <- datasets[1:4]
```

Then we will take similar approach as **Question 2.2.3**

We will iterate k=1 to 100 and find out the best value of k using cross validation.

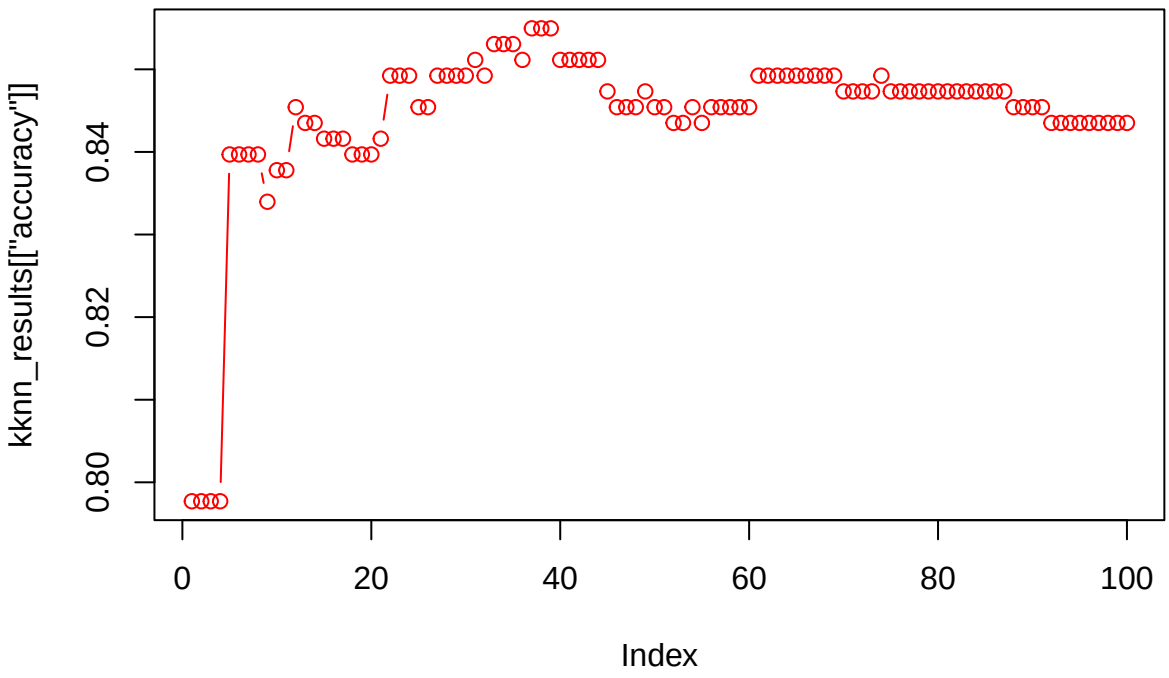
```
    kknn_cx_val_accuracy <- function(kk){
      accuracy <- c()
      for (i in 1:4){
        #set up train / validation dataset
        #excluding data frame i from training datasets and use it as validation dataset
        validation <- train_and_val[[i]]

        training_datasets <- train_and_val[-i]
        training <- data.frame()

        for (ds in training_datasets){
          training <- rbind(training,ds)
        }

        #apply KNN model
        kknn_model <- kknn(R1~., training, validation, k = kk, scale = TRUE)
        pred <- round(fitted(kknn_model))
        accuracy = append(accuracy,sum(pred == validation[,11])/nrow(validation))
      }
      #return the average accuracy of cross validations
      return (mean(accuracy))
    }

    #Summarize and visualize the prediction accuracy for each k value between 1-40
    kknn_accu = c()
    k = c()
    for (kk in 1:100){
      accuracy = kknn_cx_val_accuracy(kk)
      k = append(k,kk)
      kknn_accu = append(kknn_accu, accuracy)
    }
    kknn_results = data.frame("k"=k, "accuracy"=kknn_accu)
    print(kknn_results)
```

```
##       k  accuracy
## 1     1 0.7977099
## 2     2 0.7977099
```

```
## 3     3 0.7977099
## 4     4 0.7977099
## 5     5 0.8396947
## 6     6 0.8396947
## 7     7 0.8396947
## 8     8 0.8396947
## 9     9 0.8339695
## 10   10 0.8377863
## 11   11 0.8377863
## 12   12 0.8454198
## 13   13 0.8435115
## 14   14 0.8435115
## 15   15 0.8416031
## 16   16 0.8416031
## 17   17 0.8416031
## 18   18 0.8396947
## 19   19 0.8396947
## 20   20 0.8396947
## 21   21 0.8416031
## 22   22 0.8492366
## 23   23 0.8492366
## 24   24 0.8492366
## 25   25 0.8454198
## 26   26 0.8454198
## 27   27 0.8492366
## 28   28 0.8492366
## 29   29 0.8492366
## 30   30 0.8492366
## 31   31 0.8511450
## 32   32 0.8492366
## 33   33 0.8530534
## 34   34 0.8530534
## 35   35 0.8530534
## 36   36 0.8511450
## 37   37 0.8549618
## 38   38 0.8549618
## 39   39 0.8549618
## 40   40 0.8511450
## 41   41 0.8511450
## 42   42 0.8511450
## 43   43 0.8511450
## 44   44 0.8511450
## 45   45 0.8473282
## 46   46 0.8454198
## 47   47 0.8454198
## 48   48 0.8454198
## 49   49 0.8473282
## 50   50 0.8454198
## 51   51 0.8454198
## 52   52 0.8435115
## 53   53 0.8435115
## 54   54 0.8454198
## 55   55 0.8435115
## 56   56 0.8454198
```

```
## 57   57 0.8454198
## 58   58 0.8454198
## 59   59 0.8454198
## 60   60 0.8454198
## 61   61 0.8492366
## 62   62 0.8492366
## 63   63 0.8492366
## 64   64 0.8492366
## 65   65 0.8492366
## 66   66 0.8492366
## 67   67 0.8492366
## 68   68 0.8492366
## 69   69 0.8492366
## 70   70 0.8473282
## 71   71 0.8473282
## 72   72 0.8473282
## 73   73 0.8473282
## 74   74 0.8492366
## 75   75 0.8473282
## 76   76 0.8473282
## 77   77 0.8473282
## 78   78 0.8473282
## 79   79 0.8473282
## 80   80 0.8473282
## 81   81 0.8473282
## 82   82 0.8473282
## 83   83 0.8473282
## 84   84 0.8473282
## 85   85 0.8473282
## 86   86 0.8473282
## 87   87 0.8473282
## 88   88 0.8454198
## 89   89 0.8454198
## 90   90 0.8454198
## 91   91 0.8454198
## 92   92 0.8435115
## 93   93 0.8435115
## 94   94 0.8435115
## 95   95 0.8435115
## 96   96 0.8435115
## 97   97 0.8435115
## 98   98 0.8435115
## 99   99 0.8435115
## 100 100 0.8435115
```

```
    plot(kknn_results[['accuracy']], cex = 1, col = "red", type = "b")
```



From the graph above, we can see that the model reaches its highest level of prediction accuracy (85.50%) at k=37. Therefore we are going to choose k=37 in our KNN model, and test its prediction accuracy using the 5th sub data set. The training data set will be the combined data set generated from the first 4 sub data sets.

```
kknn_cx_val_testing <- function(kk){

    #set up training / testing dataset
    testing <- datasets[[5]]
    training <- data.frame()

    for (ds in train_and_val){
      training <- rbind(training,ds)
    }

    #apply KNN model
    kknn_model <- kknn(R1~., training, testing, k = kk, scale = TRUE)
    pred <- round(fitted(kknn_model))
    accuracy = sum(pred == testing[,11])/nrow(testing)

  return (accuracy)
}

#calculate the prediction accuracy for each k=37
accuracy = kknn_cx_val_testing(37)
```

```
    print(accuracy)
```

```
## [1] 0.8
```

Form the result above, we can see that our KNN model has a prediction accuracy of 80% with k=37.