ISYE 6501x - Week 1

Question 3.1

Part A

Loading the Libraries

```
library(kknn)
library(kernlab)
library(caret)
library(quantreg)
```

Reading the Dataset

```
credit_card <- read.table("credit_card_data.txt", stringsAsFactors =
FALSE, header = FALSE)
head(credit_card)
```

```
##   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
```

Looking at Number of Rows and Columns in the Dataset

```
nrow(credit_card)
```

```
## [1] 654
```

```
ncol(credit_card)
```

```
## [1] 11
```

RNG

```
set.seed(157)
```

Training the Model for K Clusters

```
kc_mod <- train(as.factor(V11) ~ .,
                credit_card, method = "knn",
                trControl = trainControl(method ="repeatedcv",
                                         number = 10, # number of
validations
                                         repeats = 7),
                preProcess=c("center", "scale"),
```

```
                tuneLength = 20) # setting the number of k
kc_mod
```

```
## k-Nearest Neighbors
##
## 654 samples
##  10 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 7 times)
## Summary of sample sizes: 588, 590, 588, 588, 589, 589, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8420021  0.6820022
##    7  0.8417691  0.6824991
##    9  0.8365609  0.6721049
##   11  0.8304638  0.6593827
##   13  0.8317792  0.6615785
##   15  0.8260681  0.6499148
##   17  0.8278399  0.6535361
##   19  0.8330612  0.6636188
##   21  0.8358985  0.6687780
##   23  0.8315028  0.6596752
##   25  0.8334840  0.6630119
##   27  0.8338770  0.6630226
##   29  0.8358183  0.6665555
##   31  0.8389121  0.6725575
##   33  0.8389020  0.6725326
##   35  0.8395614  0.6735695
##   37  0.8399976  0.6744856
##   39  0.8417526  0.6779768
##   41  0.8413131  0.6769220
##   43  0.8402041  0.6744508
##
## Accuracy was used to select the optimal model using the largest
value.
## The final value used for the model was k = 5.
```

Training the Model for SVM (Linear Version)

```
svm_mod <- train(as.factor(V11)~., credit_card, method = "svmLinear", #
changing the method
                trControl = trainControl(method ="repeatedcv",
                                         number = 10,
                                         repeats = 7),
                preProcess=c("center", "scale"),
                tuneLength = 20)
svm_mod
```

```
## Support Vector Machines with Linear Kernel
##
## 654 samples
##  10 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 7 times)
## Summary of sample sizes: 588, 589, 588, 589, 589, 588, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.862415  0.7268798
##
## Tuning parameter 'C' was held constant at a value of 1
```

Nonlinear SVM

```
svm_nonlinear <- train(as.factor(V11)~., credit_card, method =
"svmRadial", # Nonlinear SVM
                trControl = trainControl(method ="repeatedcv",
                                         number = 10,
                                         repeats = 7),
                preProcess=c("center", "scale"),
                tuneLength = 20)
svm_nonlinear
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 654 samples
##  10 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 7 times)
## Summary of sample sizes: 588, 589, 588, 588, 589, 589, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##        0.25  0.8593332  0.7211984
##        0.50  0.8606486  0.7237327
##        1.00  0.8591336  0.7204290
##        2.00  0.8497091  0.7007634
##        4.00  0.8401085  0.6799480
##        8.00  0.8311674  0.6606515
##       16.00  0.8344544  0.6661705
##       32.00  0.8278870  0.6526213
##       64.00  0.8169948  0.6296253
##      128.00  0.8130717  0.6211017
##      256.00  0.8095713  0.6137261
##      512.00  0.8054152  0.6054356
```

```
##     1024.00  0.7927506  0.5796723
##     2048.00  0.7872991  0.5686964
##     4096.00  0.7838567  0.5620674
##     8192.00  0.7768771  0.5480482
##    16384.00  0.7757679  0.5460328
##    32768.00  0.7739790  0.5428481
##    65536.00  0.7724335  0.5393300
##   131072.00  0.7724537  0.5395788
##
## Tuning parameter 'sigma' was held constant at a value of 0.09009635
## Accuracy was used to select the optimal model using the largest
value.
## The final values used for the model were sigma = 0.09009635 and C =
0.5.
```

By setting k = 20, the accuracy comes out as 84.7% and the best k value is at 7. SVM also have a similar performance for both the Linear and Non-Linear versions at 86.2% and 85.9% respectively. This method was done by having 7 repeats, 10 cross-validations. At C = 0.25 to 1, the Non-Linear SVM has almost close to identical accuracy as with K-Nearest Neighbors.