ISYE 6501x - Week 1

Question 2.1

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

In a manufacturing environment from my original job, there's a classification to determine whether or not the manufactured product would work or not.

Examples are:

- Functional vs. Defective
- Pass vs. Fail
- Smooth vs. Unsmooth
- On-Time vs. Late for client orders
- Quick vs. Slow for product testing

Question 2.2.1

Loading the Libraries

```
library(kknn)
library(kernlab)
```

Reading the Dataset

```
credit_card <- read.table("credit_card_data.txt")
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
```

There are 654 rows and 11 columns in the dataset.

RNG

```
set.seed(1)
```

Creating the Scaled Models

```
ccd_scaled <- ksvm(as.matrix(credit_card[,1:10]),
as.factor(credit_card[,11]), # Using ksvm
              type="C-svc",
              kernel="vanilladot", # Simple Linear Model
              C=100,
              scaled=TRUE)

##  Setting default kernel parameters

ccd_scaled

## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 189
##
## Objective Function Value : -17887.92
## Training error : 0.136086

ccd_scaled@error

## [1] 0.1360856
```

Calculating Coefficients for Equation

```
ascl <- colSums(ccd_scaled@xmatrix[[1]] * ccd_scaled@coef[[1]])
ascl

##             V1            V2            V3            V4
V5
## -0.0010065348 -0.0011729048 -0.0016261967  0.0030064203
1.0049405641
##             V6            V7            V8            V9
V10
## -0.0028259432  0.0002600295 -0.0005349551 -0.0012283758
0.1063633995
```

Calculate a0

```
a0_scaled<- -ccd_scaled@b
a0_scaled

## [1] 0.08158492
```

Model Prediction

```
predictor <- predict(ccd_scaled,credit_card[,1:10], type = 'response')
predictor

##   [1] 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 1 1 1
##  [71] 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
1 1 1 1
## [176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
## [211] 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
## [246] 0 0 0 0 0 0 0 0 0 0 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
## [281] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1
0 0 0 0
## [316] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0
## [351] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0
## [386] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0
## [421] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0
## [456] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
## [491] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1
## [526] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
1 1 1 1
## [561] 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0
## [596] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0
## [631] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

See what fraction of the model’s predictions match the actual classification

```
sum(predictor == credit_card[,11]) / nrow(credit_card)

## [1] 0.8639144
```

By using the ksvm function, the model's prediction has came out to 86.4% accuracy. It has 13.6% misclassification, and the model has been overfitted, even by using all the available data that are accounted for.

Question 2.2.3

```
kknn_predict <- rep(0,(nrow(credit_card)))

for (n in 1:nrow(credit_card)){
  knn_credit = kknn(V11 ~ .,
                     credit_card[-n,],
                     credit_card[n,],
                     k=8,
                     scale = TRUE)
  kknn_predict[n] <- as.integer(fitted(knn_credit)+0.5)
}

sum((kknn_predict == credit_card$V11)/nrow(credit_card))

## [1] 0.8486239
```

After the calculations, the result comes to 84.9% by using the k-Nearest Neighbor at k = 8 clusters. When k is used at 1-4, the accuracy tends to be lower, and stays the same at 81.49%. When k starts to equal to 5, there has been a significant change to the accuracy. Although, by having more clusters, there has not been much difference with the accuracy at all when k is at 5 or higher amount of clusters as it hovers between 84-85% accuracy.