Homework 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."

I am a Human Resources Information Systems Analyst at a hospital with over 4,000 employees that I think would greatly benefit from analyzing the risk of attrition, which is the likelihood of an employee either resigning or not, and to take countermeasures to reduce the risk of attrition. Therefore, I would use the following predictors in assessing an employee's risk of attrition:

1. Commute Distance (measured in miles): Calculating the distance to the hospital from the employee's home address since I expect the higher the commute distance, the higher the likelihood of the employee resigning.
2. Job Satisfaction (low, medium, high): I expect that the higher the employee's satisfaction is with their job the lower the likelihood of their resignation.
3. Salary Comparison level (Minimum, Midpoint, Maximum): I expect that if the employee's salary is relative to the market rate for a similar position, industry, and experience that they are less likely to resign.
4. Time since last promotion or raise (Calculated in years): I expect that if the employee receives a promotion or raise frequently that they are less likely to resign since they feel recognized for their work.
5. Hours Worked Per Week: I expect that if an employee has a greater work-life that they will be less likely to experience burnout and stress and will be less likely to resign.

Question 2.2.1

"1. Using the support vector machine function ksvm contained in the R package kernlab, find a good classifier for this data. Show the equation of your classifier, and how well it classifies the data points in the full data set. (Don't worry about test/validation data yet; we'll cover that topic soon.)"

Test Model 1

```
library("kernlab")
file_path <- "C:/Users/raque/OneDrive/Desktop/OMSA/ISYE 6501/HW/credit_card_data-headers.txt"
data<-read.table(file_path, header=TRUE)
model <- ksvm(as.matrix(data[,1:10]),data[,11],type="C-svc",kernel="vanilladot",C=100, scaled= TRUE)
```

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

```
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##            A1            A2            A3            A8            A9
## -0.0010065348 -0.0011729048 -0.0016261967  0.0030064203  1.0049405641
##           A10           A11           A12           A14           A15
## -0.0028259432  0.0002600295 -0.0005349551 -0.0012283758  0.1063633995
```

```
a0 <- -model@b
a0
```

```
## [1] 0.08158492
```

```
pred <- predict(model,data[,1:10])
pred
```

```
##   [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
##  [38] 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0
##  [75] 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
## [112] 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
## [149] 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 1 1 1 1 1 1 1 1
## [186] 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 0 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1
## [260] 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 0 0 0 1 0 0 0
## [297] 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
## [334] 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
## [371] 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
## [408] 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
## [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [482] 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
## [519] 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 0 1
## [556] 1 1 1 1 1 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 1 0 0 0 0
## [593] 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 1 0 0 0 0 0 0 0 0 0
## [630] 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
```

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

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

Summary

C Value: 100

The classifier's equation for Test Model 1 is then: (-0.0010065348A1)+(-0.0011729048A2)+(-0.0016261967A3) +(0.0030064203A8)+(1.0049405641A9)+(-0.0028259432A10)+(0.0002600295A11)+(-0.0005349551A12)+(-0.0012283758A14)+(0.1063633995A15)+(0.08158492)= 0

The rate of accuracy is equal to approximately 86.39%.

Test Model 2

```
model <- ksvm(as.matrix(data[,1:10]),data[,11],type="C-svc",kernel="vanilladot",C=1000, scaled= TRUE)
```

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

```
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##            A1            A2            A3            A8            A9
## -0.0002149185  0.0007097786  0.0011645166  0.0005673024  0.9987192088
##           A10           A11           A12           A14           A15
## -0.0005037912  0.0007155434 -0.0009130079  0.0007969700  0.0010062350
```

```
a0 <- -model@b
a0
```

```
## [1] 0.07017871
```

```
pred <- predict(model,data[,1:10])
pred
```

```
##   [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
##  [38] 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0
##  [75] 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
## [112] 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
## [149] 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 1 1 1 1 1 1 1 1
## [186] 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 0 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1
## [260] 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 0 0 0 1 0 0 0 0
## [297] 0 0 0 0 0 0 1 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 1 0
## [334] 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
## [371] 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
## [408] 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
## [445] 0 0 0 0 0 0 0 0 0 0 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
## [482] 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
## [519] 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 0 1
## [556] 1 1 1 1 1 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 1 0 0 0 0
## [593] 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 1 0 0 0 0 0 0 0 0 0 0
## [630] 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
```

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

```
## [1] 0.8623853
```

Summary

C Value: 1000

The classifier's equation for Test Model 2 is then: (-0.0002149185A1)+(0.0007097786A2)+(0.0011645166A3 +(0.0005673024A8)+(0.9987192088A9)+(-0.0005037912A10)+(0.0007155434A11)+(-0.0009130079A12)+(0.0007969700A14) 0

The rate of accuracy is equal to approximately 86.24%

Test Model 3

```
model <- ksvm(as.matrix(data[,1:10]),data[,11],type="C-svc",kernel="vanilladot",C=1000000, scaled= TRUE)
```

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

```
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##         A1         A2         A3         A8         A9        A10        A11
## -0.8283471 -0.2217216 -0.3301782  0.2825488  0.5750731  0.6143978  0.2607774
##        A12        A14        A15
## -0.5943042 -1.1175369  0.9336833
```

```
a0 <- -model@b
a0
```

```
## [1] -0.1281168
```

```
pred <- predict(model,data[,1:10])
pred
```

```
##   [1] 0 1 1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0
##  [38] 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
##  [75] 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 1 0 1 1 1
## [112] 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 1 1
## [149] 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1
## [186] 1 0 0 0 1 1 0 1 0 0 1 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1
## [223] 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0
## [260] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0
## [297] 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
## [334] 1 0 1 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1
## [371] 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0
## [408] 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0
## [445] 1 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 1
## [482] 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1 1 1
## [519] 1 0 0 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 0
## [556] 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
## [593] 0 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 0
## [630] 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 1 0 0 1
```

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

```
## [1] 0.6253823
```

Summary

C Value= 1000000

The classifier's equation for Test Model 3 is then: (-0.8283471A1)+(-0.2217216A2)+(-0.3301782A3 +(0.2825488A8)+(0.5750731A9)+(0.6143978A10)+(0.2607774A11)+(-0.5943042A12)+(-1.1175369A14)+(0.9336833A15)+(-0.1281168)= 0

The rate of accuracy is equal to approximately 62.54%

Test Model 4

```
model <- ksvm(as.matrix(data[,1:10]),data[,11],type="C-svc",kernel="vanilladot",C=.01, scaled= TRUE)
```

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

```
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##            A1            A2            A3            A8            A9
## -0.0001500738 -0.0014818294  0.0014083130  0.0072863886  0.9916470037
##           A10           A11           A12           A14           A15
## -0.0044661236  0.0071482899 -0.0005468386 -0.0016930578  0.1054824270
```

```
a0 <- -model@b
a0
```

```
## [1] 0.08198854
```

```
pred <- predict(model,data[,1:10])
pred
```

```
##   [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
##  [38] 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0
##  [75] 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
## [112] 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
## [149] 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 1 1 1 1 1 1 1 1
## [186] 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 0 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1
## [260] 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 0 0 0 0 1 0 0 0
## [297] 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
## [334] 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
## [371] 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
## [408] 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
## [445] 0 0 0 0 0 0 0 0 0 0 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
## [482] 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
## [519] 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 0 1
## [556] 1 1 1 1 1 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 1 0 0 0 0
## [593] 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 1 0 0 0 0 0 0 0 0 0
## [630] 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
```

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

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

Summary

C Value=.01

The classifier's equation for Test Model 4 is then: (-0.0001500738A1)+(-0.0014818294A2)+(0.0014083130A3 +(0.0072863886A8)+(0.9916470037A9)+(-0.0044661236A10)+(0.0071482899A11)+(-0.0005468386A12)+(-0.0016930578A14)+(0.1054824270A15)+(0.08198854)= 0

The rate of accuracy is equal to approximately 86.39%.

Test Model 5

```
model <- ksvm(as.matrix(data[,1:10]),data[,11],type="C-svc",kernel="vanilladot",C=.00000000000001, scal
```

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

```
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##            A1            A2            A3            A8            A9
## -6.479335e-14  6.730631e-13  9.364570e-13  1.803160e-12  4.127070e-12
##           A10           A11           A12           A14           A15
## -2.416310e-12  2.334032e-12 -1.403036e-13 -3.383522e-13  1.011458e-12
```

```
a0 <- -model@b
a0
```

```
## [1] -1
```

```
pred <- predict(model,data[,1:10])
pred
```

```
##   [1] 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 0 0
##  [38] 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 0 0
##  [75] 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 0 0
## [112] 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 0 0
## [149] 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 0 0
## [186] 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 0 0
## [223] 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 0 0
## [260] 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 0 0
## [297] 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 0 0
## [334] 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 0 0
## [371] 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 0 0
## [408] 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 0 0
## [445] 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 0 0
## [482] 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 0 0
## [519] 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 0 0
## [556] 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 0 0
## [593] 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 0 0
## [630] 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
```

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

```
## [1] 0.5474006
```

Summary

C Value=.00000000000001

The classifier's equation for Test Model 5 is then: (-6.479335e-14A1)+(6.730631e-13A2)+(9.364570e-13A3 +(1.803160e-12A8)+(4.127070e-12A9)+(-2.416310e-12 A10)+(2.334032e-12A11)+(-1.403036e-13A12)+(-3.383522e-13A14)+(1.011458e-12A15)+(-1)= 0

The rate of accuracy is equal to approximately 54.74%.

Overall Analysis of Results:

A smaller value of C allows for a larger margin, pontentially leading to more misclassifications. On the other hand, a larger value of C leads to a smaller margin that reduces the amount of misclassifications. By testing the 5 models above using ksvm with vanilladot, we are able to see how changing the hyperparameter of C (ranging from 00000000000001 to 1000000) depicts how C helps control the overall accuracy rate in our test models above. Even with these randomly chose 5 C values, we can see how some C parameters achieve greater accuracy than others.Finding the best C parameter involves testing various C parameters to achieve a certain level of accuracy.

Overall, the C parameters that best classified the data points in the dataset above ended up being equal for both Test Model 1 and 4, in where both had an accuracy rate of 86.39% (despite Test Model 1's C parameter being 100 and Test Model 4's parameter being .01). Test Model 2's C parameter of 1000 ended up being close to both Test Model 1 and Test Model 4's results with an accuracy rate of 86.24%. Also, in the other testing models above we can see that for very tiny values or very large values of C (Test Models 3 and 5), both lead to lower accuracy rates due to underfitting or overfitting.

Question 2.2.3

3. "Using the k-nearest-neighbors classification function kknn contained in the R kknn package, suggest a good value of k, and show how well it classifies that data points in the full data set. Don't forget to scale the data (scale=TRUE in kknn)."

```
library(kknn)
file_path <- "C:/Users/raque/OneDrive/Desktop/OMSA/ISYE 6501/HW/credit_card_data-headers.txt"
data<-read.table(file_path, header=TRUE)
set.seed(454)
kknnmodel<-train.kknn(R1~.,
                      data,
                      kmax=95,
                      kernel="optimal",
                      scale=TRUE)
pred<-predict(kknnmodel, data)
rounded<-round(pred)
accuracy <-sum(rounded==data[,11])/(nrow(data))
accuracy
```

```
## [1] 0.8776758
```

```
kknnmodel
```

```
##
## Call:
## train.kknn(formula = R1 ~ ., data = data, kmax = 95, kernel = "optimal",     scale = TRUE)
##
## Type of response variable: continuous
## minimal mean absolute error: 0.1850153
## Minimal mean squared error: 0.1073792
## Best kernel: optimal
## Best k: 58
```

K-Nearest Neighbor Analysis

The weighted kknn uses the kernel functions to weigh the neighbors according to their distance. By using the leave out cross validation method otherwise known as (train.kkkn) we are able to determine the best k value between the k values of 1 to 95. In the results, shown above we can see that a k value of 95 with a kernel set to optimal that k=58 is the best k for this model and achieves an accuracy of approximately 87.77%.

References

All analyses were performed using R Statistical Software (R version 4.3.2 (2023-10-31 ucrt)).

Allaire, J., Xie, Y., Dervieux, C., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., & Iannone, R. (2023). rmarkdown: Dynamic Documents for R. https://github.com/rstudio/rmarkdown

Baeldung, W. by: (2023, June 22). The C parameter in support Vector Machines. Baeldung on Computer Science. https://www.baeldung.com/cs/ml-svm-c-parameter

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2013). An introduction to statistical learning : with applications in R. New York :Springer,

Hechenbichler, K. and Schliep, K. (2004): Weighted k-Nearest-Neighbor Techniques and Ordinal Classification. Collaborative Research Center 386, Discussion Paper 399 [PDF, 229kB]

Karatzoglou A, Smola A, Hornik K (2023). kernlab: Kernel-Based Machine Learning Lab. R package version 0.9-32, https://CRAN.R-project.org/package=kernlab.

Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "kernlab – An S4 Package for Kernel Methods in R." Journal of Statistical Software, 11(9), 1–20. doi:10.18637/jss.v011.i09.

Quinlan,J. R.. Credit Approval. UCI Machine Learning Repository. https://doi.org/10.24432/C5FS30.

Vemulapati, P. (2020, May 20). The conundrum of "C": SVM Hyperparameters. Medium. https://medium.com/swlh/the-conundrum-of-c-svm-hyperparameters-3327dfc7354a

YouTube. (2019). YouTube. Retrieved January 17, 2024, from https://www.youtube.com/watch?v=efR1C6CvhmE.

YouTube. (2023). YouTube. Retrieved January 17, 2024, from https://www.youtube.com/watch?v=yZ0bV2Afkjc.