ISYE 6501 Homework 1

2023-08-30

```
knitr::opts_chunk$set(echo = TRUE)
```

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.

A company's marketing department may be interested in using classification models to predict customer behavior. For example, if Nike releases a new line of soccer cleats, they could use a classification model to better understand which of their customers are most likely to purchase the product. Predictors for this type of classification model may include age, gender, website engagement, subscription preferences, and customer satisfaction, among other predictors.

Question 2.2

Part 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.)

```
#Set up work space and load libraries
rm(list = ls()) #clear out environment
set.seed(42)
r = getOption("repos")
r["CRAN"] = "http://cran.us.r-project.org"
options(repos = r)
install.packages("kernlab")
```

```
## package 'kernlab' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\cgittelman001\AppData\Local\Temp\RtmpeQnKeO\downloaded_packages
```

```
library(kernlab)
library(kknn)

#load the data, making sure working directory is set up properly
data_df <- read.table("credit_card_data-headers.txt", stringsAsFactors=FALSE, header=TRUE)
```

```r
#convert the data to a matrix
data_mx <- data.matrix(data_df)
```

I fitted the SVM model using multiple values of C and observed the model accuracy for each. There is not significant variation in the model accuracy except when the value of C gets very small. Since a Type I error (a borrower is incorrectly deemed creditworthy) is more costly than a Type II error (a financial institution denies the application of a creditworthy borrower), I would choose a higher value of C. A higher value of C will decrease the size of the margin in the SVM model to decrease mis-classifications.

In this case, C=100 seems appropriate. C values higher than C = 100 do not impact model accuracy and greatly increase model run-time. Additionally, a really high value of C may lead to over-fitting the data.

```r
#fit KSVM model, testing with different values of C
for (c in c(.0001, .001, .01, 1, 10, 100, 1000, 10000, 100000)){ #repeat for each value of c
  ksvm_model <- ksvm(
              data_mx[,1:10],            #x, attributes
              data_mx[,11],              #y, response
              type="C-svc",              #type, classification
              C=c,                       #cost of constraints violation
              kernel="vanilladot",       #linear kernel
              scaled=TRUE
  )
  ksvm_prediction <- predict(ksvm_model) #see what the model predicts
  ksvm_accuracy <- sum(ksvm_prediction == data_mx[,11])/nrow(data_mx) #calculate accuracy of the model
  print(paste0("Accuracy (C=", c, "): ", round(ksvm_accuracy, 5)*100, "%")) #format and print result
}
```

```
##  Setting default kernel parameters
## [1] "Accuracy (C=1e-04): 54.74%"
##  Setting default kernel parameters
## [1] "Accuracy (C=0.001): 83.792%"
##  Setting default kernel parameters
## [1] "Accuracy (C=0.01): 86.391%"
##  Setting default kernel parameters
## [1] "Accuracy (C=1): 86.391%"
##  Setting default kernel parameters
## [1] "Accuracy (C=10): 86.391%"
##  Setting default kernel parameters
## [1] "Accuracy (C=100): 86.391%"
##  Setting default kernel parameters
## [1] "Accuracy (C=1000): 86.239%"
##  Setting default kernel parameters
## [1] "Accuracy (C=10000): 86.239%"
##  Setting default kernel parameters
## [1] "Accuracy (C=1e+05): 86.391%"
```

After determining the best value of C, I re-fitted the SVM model. Then, I used the following code to calculate the coefficients (a1…am) and intercept (a0) of a linear classifier. The following equation uses a C-value of C = 100 and classifies the data points with 86.391% accuracy:

ksvm_model <- (-0.0010065348 * A1) + (-0.0011729048 * A2) + (-0.0016261967 * A3) + (0.0030064203 * A8) + (1.0049405641 * A9) + (-0.0028259432 * A10) + (0.0002600295 * A11) + (-0.0005349551 * A12) + (-0.0012283758 * A14) + (0.1063633995 * A15) + 0.08158492

```
#fit KSVM model with C=100
ksvm_model <- ksvm(
  data_mx[,1:10],             #x, attributes
  data_mx[,11],               #y, response
  type="C-svc",               #type, classification
  C=100,                        #cost of constraints violation
  kernel="vanilladot",        #linear kernel
  scaled=TRUE
)
```

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

```
#Calculate a1...am for KSVM model where C=100
a <- colSums(ksvm_model@xmatrix[[1]] * ksvm_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
```

```
#Calculate a0 for KSVM model where C=100
a0 <- -ksvm_model@b
a0
```

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

Part 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)

I fitted the K-NN model using multiple values of K and observed the model accuracy for each. For this model, I was careful to loop through each row in the data frame to make sure the model does not use i itself (which is in the data set) as one of the nearest neighbors. In this case, K = 22 seems appropriate since it generates the highest accuracy.

```
#fit KNN Model, testing with different values of k
knn_prediction <- rep(0, nrow(data_df))  #create a list of zeros equal to the length of the data frame

for (k in 1:50){ #repeat for each value of k between 1-50
  for (i in 1:nrow(data_df)){ #repeat for every row in the data frame
    knn_model <- kknn(
                  R1~.,                         #formula object
                  data_df[-i,],                 #train with every row except i
                  data_df[i,],                  #test with row i
                  k=k,                          #k value
                  kernel="rectangular",
                  scale=TRUE)
    knn_prediction[i] <- round(predict(knn_model)) #update list of zeros at index i with actual predict
  }
knn_accuracy <- sum(knn_prediction==data_df[,11])/nrow(data_df) #calculate accuracy of the model
print(paste0("Accuracy (K=", k, "): ", round(knn_accuracy, 5)*100, "%")) #format and print result
}
```

```
## [1] "Accuracy (K=1): 81.498%"
## [1] "Accuracy (K=2): 78.593%"
## [1] "Accuracy (K=3): 82.263%"
## [1] "Accuracy (K=4): 82.569%"
## [1] "Accuracy (K=5): 84.557%"
## [1] "Accuracy (K=6): 84.098%"
## [1] "Accuracy (K=7): 84.709%"
## [1] "Accuracy (K=8): 84.709%"
## [1] "Accuracy (K=9): 83.18%"
## [1] "Accuracy (K=10): 84.098%"
## [1] "Accuracy (K=11): 83.486%"
## [1] "Accuracy (K=12): 83.486%"
## [1] "Accuracy (K=13): 83.028%"
## [1] "Accuracy (K=14): 83.333%"
## [1] "Accuracy (K=15): 82.569%"
## [1] "Accuracy (K=16): 82.875%"
## [1] "Accuracy (K=17): 82.263%"
## [1] "Accuracy (K=18): 83.18%"
## [1] "Accuracy (K=19): 82.263%"
## [1] "Accuracy (K=20): 83.486%"
## [1] "Accuracy (K=21): 84.098%"
## [1] "Accuracy (K=22): 85.015%"
## [1] "Accuracy (K=23): 84.251%"
## [1] "Accuracy (K=24): 83.792%"
## [1] "Accuracy (K=25): 83.333%"
## [1] "Accuracy (K=26): 83.792%"
## [1] "Accuracy (K=27): 83.792%"
## [1] "Accuracy (K=28): 83.792%"
## [1] "Accuracy (K=29): 83.945%"
## [1] "Accuracy (K=30): 83.639%"
## [1] "Accuracy (K=31): 83.333%"
## [1] "Accuracy (K=32): 83.945%"
## [1] "Accuracy (K=33): 83.486%"
## [1] "Accuracy (K=34): 84.251%"
## [1] "Accuracy (K=35): 83.945%"
## [1] "Accuracy (K=36): 84.098%"
## [1] "Accuracy (K=37): 84.098%"
## [1] "Accuracy (K=38): 84.098%"
## [1] "Accuracy (K=39): 84.098%"
## [1] "Accuracy (K=40): 84.404%"
## [1] "Accuracy (K=41): 84.404%"
## [1] "Accuracy (K=42): 84.709%"
## [1] "Accuracy (K=43): 84.251%"
## [1] "Accuracy (K=44): 85.015%"
## [1] "Accuracy (K=45): 84.251%"
## [1] "Accuracy (K=46): 84.557%"
## [1] "Accuracy (K=47): 83.945%"
## [1] "Accuracy (K=48): 84.862%"
## [1] "Accuracy (K=49): 84.098%"
## [1] "Accuracy (K=50): 84.404%"
```

However, because we should not test the accuracy of a model on the data we use to train it, I fitted the model again using 70% of the data to train and the other 30% to test. First, I split the data up into training and test sets. Then, I re-ran the K-NN model.

```
#split data into training and test sets
sample <- sample(1:nrow(data_df), round(nrow(data_df)*.7)) #randomly choose 30% of data
train_sample <- data_df[sample,] #train sample = 70% of data
test_sample <- data_df[-sample,] #test sample = 30% of data not in train sample

#fit KNN Model, testing with different values of k
for (k in 1:50){ #repeat for each value of k between 1-50
    knn_model_2 <- kknn(
      R1~.,                     #formula object
      train_sample,             #train
      test_sample,               #test
      k=k,                      #k value
      kernel="rectangular",
      scale=TRUE)
    knn_prediction <- round(predict(knn_model_2))
    knn_accuracy <- sum(knn_prediction==test_sample[,11])/nrow(test_sample) #calculate accuracy of the
    print(paste0("Accuracy (K=", k, "): ", round(knn_accuracy, 5)*100, "%")) #format and print result
  }
```

```
## [1] "Accuracy (K=1): 81.633%"
## [1] "Accuracy (K=2): 80.102%"
## [1] "Accuracy (K=3): 84.694%"
## [1] "Accuracy (K=4): 83.163%"
## [1] "Accuracy (K=5): 85.204%"
## [1] "Accuracy (K=6): 85.204%"
## [1] "Accuracy (K=7): 86.224%"
## [1] "Accuracy (K=8): 85.204%"
## [1] "Accuracy (K=9): 84.694%"
## [1] "Accuracy (K=10): 83.673%"
## [1] "Accuracy (K=11): 85.204%"
## [1] "Accuracy (K=12): 84.694%"
## [1] "Accuracy (K=13): 85.714%"
## [1] "Accuracy (K=14): 85.204%"
## [1] "Accuracy (K=15): 84.694%"
## [1] "Accuracy (K=16): 85.204%"
## [1] "Accuracy (K=17): 86.224%"
## [1] "Accuracy (K=18): 83.673%"
## [1] "Accuracy (K=19): 84.694%"
## [1] "Accuracy (K=20): 82.653%"
## [1] "Accuracy (K=21): 82.653%"
## [1] "Accuracy (K=22): 82.653%"
## [1] "Accuracy (K=23): 83.673%"
## [1] "Accuracy (K=24): 82.653%"
## [1] "Accuracy (K=25): 83.673%"
## [1] "Accuracy (K=26): 83.673%"
## [1] "Accuracy (K=27): 83.673%"
## [1] "Accuracy (K=28): 82.653%"
## [1] "Accuracy (K=29): 83.673%"
## [1] "Accuracy (K=30): 83.163%"
## [1] "Accuracy (K=31): 83.673%"
## [1] "Accuracy (K=32): 83.673%"
## [1] "Accuracy (K=33): 84.184%"
## [1] "Accuracy (K=34): 83.163%"
## [1] "Accuracy (K=35): 83.673%"
## [1] "Accuracy (K=36): 83.163%"
## [1] "Accuracy (K=37): 84.184%"
## [1] "Accuracy (K=38): 83.163%"
## [1] "Accuracy (K=39): 84.184%"
## [1] "Accuracy (K=40): 84.184%"
## [1] "Accuracy (K=41): 85.204%"
## [1] "Accuracy (K=42): 84.184%"
## [1] "Accuracy (K=43): 84.184%"
## [1] "Accuracy (K=44): 84.184%"
## [1] "Accuracy (K=45): 84.694%"
## [1] "Accuracy (K=46): 84.184%"
## [1] "Accuracy (K=47): 84.184%"
## [1] "Accuracy (K=48): 84.184%"
## [1] "Accuracy (K=49): 84.694%"
## [1] "Accuracy (K=50): 84.694%"
```

Since this model splits the data into train and test sets, it gives us a more accurate evaluation of model performance. Because the data is randomly sorted into training and test sets each time we run the model, the model accuracy will change accordingly. Thus, I ran the model multiple times to make sure my observations were not greatly impacted by the randomness of one specific set of training/test data.

After running the model multiple times, I noticed a pattern where model accuracy would decrease as the value of K got larger. I suspect this is because a high value of K introduces more noise into each prediction and results in underfitting the data. All things considered, I would suggest choosing a K value between 15 and 20. This uses a high enough value of K so that we are not overfitting the data, while maintaining a high model accuracy.