ISYE HW1

2024-01-16

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 could use the classification model to optimize my rental business by predicting the occupancy of the properties during a specific time period. The informed decisions could help ensure that the properties are marketed effectively, priced competitively, and occupied at optimal levels throughout the year. Predictors: Pricing trends Seasonal trends Local events Historical booking history

Question 2.2

The files credit_card_data.txt (without headers) and credit_card_data-headers.txt (with headers) contain a dataset with 654 data points, 6 continuous and 4 binary predictor variables. It has anonymized credit card applications with a binary response variable (last column) indicating if the application was positive or negative. The dataset is the "Credit Approval Data Set" from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Credit+Approval) without the categorical variables and without data points that have missing values.

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.)

```r
#Load the data
data <- read.table("credit_card_data-headers.txt", header = TRUE)

#Look at the data
head(data)
```

```
##   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
## 6  1 32.08 4.000 2.50  1   1   0   0 360   0  1
```

```r
tail(data)
```

```
##     A1    A2     A3   A8 A9 A10 A11 A12 A14 A15 R1
## 649  1 40.58  3.290 3.50  0   1   0   0 400   0  0
```

```
## 650  1 21.08 10.085 1.25  0   1   0   1 260   0  0
## 651  0 22.67  0.750 2.00  0   0   2   0 200 394  0
## 652  0 25.25 13.500 2.00  0   0   1   0 200   1  0
## 653  1 17.92  0.205 0.04  0   1   0   1 280 750  0
## 654  1 35.00  3.375 8.29  0   1   0   0   0   0  0

#Load the package kernlab which contains ksvm
library(kernlab)

#Run the model; use the ksvm function with simple linear kernel Vanilladot
#convert to matrix format
model1 <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]), C = 100, scaled =
TRUE, kernel = "vanilladot", type = "C-svc")

##  Setting default kernel parameters

#Calculate coefficients
a <- colSums(model1@xmatrix[[1]]* model1@coef[[1]])
print(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
a0 <- -model1@b
print(a0)

## [1] 0.08158492

#Show the model predictions
pred <- predict(model1,data[,1:10])

#Test the accuracy of the model’s predictions
accuracy <- sum(pred == data[,11])/nrow(data)* 100
print(accuracy)

## [1] 86.39144

# 0.86391.. -> 86.391%, This means the models’ accuracy is 86.391%

#I calculated the accuracy at different C values and found that adjusting the
value of C did not change the outcome of the model.

#the classifier’s equation: -0.001A1 - 0.00117A2 - 0.0016A3 + 0.003A8 +
1.0049A9 - 0.0028A10 + 0.00026A11 - 0.0005A12 - 0.0012A14 + 0.10636A15 +
0.08158
```

2. You are welcome, but not required, to try other (nonlinear) kernels as well; we’re not covering them in this course, but they can sometimes be useful and might provide better predictions than vanilladot.

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).

```
#load package
library(kknn)

kknn_accuracy_test = function(Z){
  Pred_kknn <- rep(0,nrow(data))
  for (i in 1:nrow(data)){

    #model creation using scaled data; ensuring it doesnt use i itself
    kknn_model <- kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15, data[-i,],
data[i,], k = Z, scale = TRUE)

    #to round values
    Pred_kknn[i] <- as.integer(fitted(kknn_model) + 0.5)
  }

  #accuracy calculation
  accuracy_out <- sum(Pred_kknn == data[,11]) / nrow(data)

  return(accuracy_out)
}
acc <- rep(0,20)
for (Z in 1:20){
  acc[Z] = kknn_accuracy_test(Z)
}

#accuracy percentage
kknn_acc = as.matrix(acc * 100)

kknn_acc

##           [,1]
##  [1,] 81.49847
##  [2,] 81.49847
##  [3,] 81.49847
##  [4,] 81.49847
##  [5,] 85.16820
##  [6,] 84.55657
##  [7,] 84.70948
##  [8,] 84.86239
##  [9,] 84.70948
## [10,] 85.01529
## [11,] 85.16820
## [12,] 85.32110
## [13,] 85.16820
## [14,] 85.16820
```

```
## [15,] 85.32110
## [16,] 85.16820
## [17,] 85.16820
## [18,] 85.16820
## [19,] 85.01529
## [20,] 85.01529

#maximum accuracy is 85.321%
#12 and 15 have the highest accuracy.
```