HOMEWORK 1 ISYE

Sheyanne Rodriguez

2024-01-15

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.

1. I bartend at a casual fine-dining restaurant and I thought it would be fun to implement predictors for a classification model that could see when different restaurants would be busy or not. I find that an issue in the restaurant industry faces is the target market. The type of guests they are trying to attract into their restaurant plays a significant role in tracking their busy days and times. Most upscale restaurants have elevated prices on their menu that invites those who can afford to pay those prices which means they probably won't have specials or a *happy hour*. There are places that make specific foods (e.g. tacos, vegan, etc.) or a broad spectrum of foods to invite a non-specific or a specific type of crowd. This effort can differ from busy date-nights, four hour lunch rushes, or slow weekdays. I believe the predictors that would be useful in this situation are:

    a) Having a unique value proposition(UVP) to attract desired guests: affordability, special menu offerings(happy hour), atmosphere, service and possibly ingredients.
    b) Menu plans: display which days have specials, features, and limited time offers during specific days and times.
    c) Seeking competition: are there other established businesses near by that already secured foot traffic? Do have different offers at different times and days that bring curious guests to see what is new (if it is a new restaurant)? Are there stores nearby that give people the privilege of walking to the restaurant instead of moving their car (e.g. a mall).
    d) Marketing: is there a social media page that shows the time and day happy hour occurs or for example is there a post advertising half off bottles of wine on Wednesdays(if offering).

Question 2.2.1

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.

```r
library(kernlab)

data = read.table(file ="C:\\Users\\sheya\\OneDrive\\Desktop\\credit_card_data.txt",
                  header = FALSE)
model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type=
                "C-svc",kernel="vanilladot",C=1,scaled=TRUE)
```

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

```
a <- colSums(model@xmatrix[[1]]* model@coef[[1]])
a0 <- -model@b
a0
```

```
## [1] 0.08148382
```

```
# Get the coefficients df
df <- data.frame(Coefficients = a)
print(df)
```

```
##      Coefficients
## V1  -0.0011026642
## V2  -0.0008980539
## V3  -0.0016074557
## V4   0.0029041700
## V5   1.0047363456
## V6  -0.0029852110
## V7  -0.0002035179
## V8  -0.0005504803
## V9  -0.0012519187
## V10  0.1064404601
```

```
# Multiply by 100 to convert the variable into a percentage

pred <- predict(model,data[,1:10])
sprintf("The accuracy is %.2f%%", sum(pred == data[,11])/nrow(data)*100)
```

```
## [1] "The accuracy is 86.39%"
```

```
#paste("The accuracy is", sum(pred == data[,11])/ nrow(data)* 100,"%")
#Initially I found paste() to help me print the accuracy formally.
#However, I found the sprintf() function more helpful to put the accuracy in a string
#and attach a percent sign.
```

When C is 1 in the ksvm function, the equation of the classifier is as follows:

$$\begin{gathered}-0.0011026642v_1 - 0.0008980539v_2 - 0.0016074557v_3 + 0.0029041700v_4 \\ +1.0047363456v_5 - 0.0029852110v_6 - 0.0002035179v_7 - 0.0005504803v_8 \\ -0.0012519187v_9 + 0.1064404601v_{10} + 0.08148382 = 0\end{gathered}$$

C-Val	0.0001	0.01	1	10	100	10000
Accuracy	54.74%	83.8%	86.39%	86.39%	86.39%	86.24%

When I changed C to find a fluctuation in accuracy, it did not seem to improve the accuracy. Even with an extremely small number nor did a high number like 10,000 impact the accuracy significantly.

Question 2.2.2

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.

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

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

```
a <- colSums(model@xmatrix[[1]]* model@coef[[1]])
a0 <- -model@b

# Multiply by 100 to create a percentage

pred <- predict(model,data[,1:10])
sprintf("The accuracy for Spline is %.2f%%",sum(pred == data[,11])/ nrow(data)* 100)
```

```
## [1] "The accuracy for Spline is 96.64%"
```

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

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

```
a <- colSums(model@xmatrix[[1]]* model@coef[[1]])
a0 <- -model@b

# Multiply by 100 to create a percentage

pred <- predict(model,data[,1:10])
sprintf("The accuracy for Hyperbolic Tangent is %.2f%%",sum(pred == data[,11])/ nrow(data)* 100)
```

```
## [1] "The accuracy for Hyperbolic Tangent is 72.17%"
```

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

a <- colSums(model@xmatrix[[1]]* model@coef[[1]])
a0 <- -model@b

# Multiply by 100 to create a percentage

pred <- predict(model,data[,1:10])
sprintf("The accuracy for Laplacian is %.2f%%",sum(pred == data[,11])/ nrow(data)* 100)
```

```
## [1] "The accuracy for Laplacian is 86.39%"
```

Trying other (nonlinear) kernels in R when C = 1 showed a significant difference in accuracy. I tried Spline, Hyperbolic tangent, and Laplacian in this case. Spline seemed to have the best accuracy as it showed at 96.63%, Laplacian revealed an accuracy close to *Vanilladot* at 86.39% and Hyperbolic Tangent had the worst outcome of 72.17%

Question 2.2.3

Using the k-nearest-neighbors classification 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).

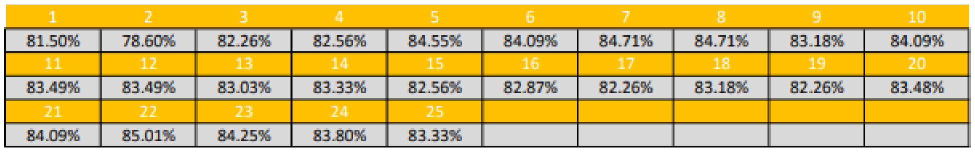
```r
library(kknn)
knn_max = 25
accuracy_k <- rep(0, knn_max)
pred_CCmodel_knn <- rep(0, nrow(data))
 for(k in 1:knn_max){
   for(i in 1:nrow(data)){
    CCmodel_knn = kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,
                     data[-i,1:11],
                     data[i,1:10],
                     k = k,
                     distance = 2,
                     kernel = "rectangular",
                     scale = TRUE)
   pred_CCmodel_knn[i] <- round(fitted(CCmodel_knn))
   CCmodel_acck <- sum(pred_CCmodel_knn == data[,11])/ nrow(data)

   }
   accuracy_k[k] <- CCmodel_acck

 }
print(accuracy_k)
```

```
##  [1] 0.8149847 0.7859327 0.8226300 0.8256881 0.8455657 0.8409786 0.8470948
##  [8] 0.8470948 0.8318043 0.8409786 0.8348624 0.8348624 0.8302752 0.8333333
## [15] 0.8256881 0.8287462 0.8226300 0.8318043 0.8226300 0.8348624 0.8409786
## [22] 0.8501529 0.8425076 0.8379205 0.8333333
```

```r
paste("The highest accuracy of the first maximum k value",which.max(accuracy_k),
       "is", round(max(accuracy_k),4)*100,"%")
```

```
## [1] "The highest accuracy of the first maximum k value 22 is 85.02 %"
```

As I evaluated the accuracy from 1 neighbor to 25 neighbors I found the following results:



1	2	3	4	5	6	7	8	9	10
81.50%	78.60%	82.26%	82.56%	84.55%	84.09%	84.71%	84.71%	83.18%	84.09%
11	12	13	14	15	16	17	18	19	20
83.49%	83.49%	83.03%	83.33%	82.56%	82.87%	82.26%	83.18%	82.26%	83.48%
21	22	23	24	25					
84.09%	85.01%	84.25%	83.80%	83.33%					

Evidently from the table above, neighbor 22 reveals the highest accuracy in the prediction with 85.02%.

.

.

Works Cited

Kernlab - An S4 Package for Kernel Methods in R

https://cran.r-project.org/web/packages/kernlab/vignettes/kernlab.pdf

R Markdown from RStudio

https://rmarkdown.rstudio.com/lesson-2.html

Print a formatted string in R Programming

https://www.geeksforgeeks.org/print-a-formatted-string-in-r-programming-sprintf-function/

Mathematics in R Markdown

https://rpruim.github.io/s341/S19/from-class/MathinRmd.html