Week1_Homework

YL

2023-08-29

#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 worked in an accounting company, and I review client's business balance sheet as daily task. We would use some vectors such as revenue, cost of goods, business liabilities, owner's equity and assets. We can simply using revenue and cost of goods to calculate the profit that the company earns. Furthermore, besides analyze our client's business. We can also use some vectors such as employee retension rate, new hire rate, average salary with the same title in the company to predict the company transition rate. Also for the managers prespective, they can analyze based on the transition rate and retension rate to create better management strategy.

#2.2.1

#Method 1 manually change C value.

```
library(kernlab)

#import data from txt.
data <- read.table("credit_card_data.txt", header = FALSE)

#define ksvm function. Call the linear equation to find the best C-value
model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type='C-svc',kernel='vanilladot',
C=10000,scaled=TRUE)
```

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

```
# calculate a1...am
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##            V1            V2            V3            V4            V5
##  0.0008936167  0.0016125725 -0.0003415921  0.0042114213  1.0014527518
##            V6            V7            V8            V9           V10
##  0.0023573637  0.0052757867  0.0002803662  0.0025097815  0.0035920442
```

```
# calculate a0
a0 <- -model@b
a0
```

```
## [1] 0.07046104
```

```
# see what the model predicts
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 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 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 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 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 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 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 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 0 0 1 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 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 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 1 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 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 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 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 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
## Levels: 0 1
```

```
# see what fraction of the model's predictions match the actual classification
sum(pred == data[,11]) / nrow(data)
```

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

This is my solution without for loop. Test result: 1. C=100, accuracy is 0.8639144. Then changed the C-value to C=50,000. Trying to see if I changed a larger c-value will the accuracy rate be different from the first test. When C=50,000, the accuracy is 0.8623853.In my third test. I tested C=100,000. The results came out the accuracy rate dropped from 0.8623853 to 0.6253823. In conclusion, I decided to keep the cost between 100 - 10,000 since the results shown the cost within this range has the highest accuracy.

#Method 2 For Loop

```
#for loop for different c values
cvalues <- c(0.00001,0.0001,0.01,1,10,100,10000,1000000)
#setup linear equation for kernel we want to use in this question
k_kernel = 'vanilladot'
#create a dataframe to store the data from the for loop
results <- data.frame("C-Values" = integer(),"accuracy" = numeric())

for(cvalue in cvalues)
{
  #create a ksvm model to run for the test.
  model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),
  type='C-svc',kernel=k_kernel,C=cvalue,scaled=TRUE)

  #use ksvm to run formula and loop for different c value.
  pred <- predict(model,data[,1:10])

  #result in fraction of the model's predictions match the actual classification
  accuracy <- sum(pred == data[,11]) / nrow(data)

  #create an empty list to store output values
  results[nrow(results) + 1,] <- c(cvalue,accuracy)
}
```

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

```
View(results)
```

##After used a for loop. I had better visulization on collecting accuracy results in a spreadsheet. I found out the highest accuracy rate was 0.8639 where the c value in between 0.01 to 100. Therefore, the best C value of the this model is in between 0.01 to 100.The higher the C value, it will lower the accuracy rate.

#2.2.2 (Optional)

```
#try different methods other than vanilladot
diffkernels <- c("rbfdot","splinedot","laplacedot","polydot","vanilladot")

#for loop for different c values
cvalues <- c(0.00001,0.0001,0.01,1,10,100,10000,1000000)

#create a dataframe to store the data from the for loop
results <- data.frame("C-Values" = integer(),"accuracy" = numeric(),
"kernel_name" = character(),stringsAsFactors = FALSE)

#n starts from 0.
n <- 0

for (ikernel in diffkernels) {
n <- n+1
for(cvalue in cvalues)
{
#setup model to loop for different kernels and c valuse.
model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type='C-svc',kernel=ikernel,
C=cvalue,scaled=TRUE)

#use ksvm to run formular in different c value.
pred <- predict(model,data[,1:10])

#result in fraction of the model's predictions match the actual classification
accuracy <- sum(pred == data[,11]) / nrow(data)

#store the output into data
results[nrow(results) + 1,] <- c(cvalue,accuracy,ikernel)
  }
}
```

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

```
View(results)
```

##I used another for loop over the accuracy rate from 2.2.1. Meaning I am not only looping different C value in the model, but I also looping different models other than ksvm to calculate the highest accuracy rate for the credit card test.In the results, I found out that

---

#2.2.3 #install.package("kknn")

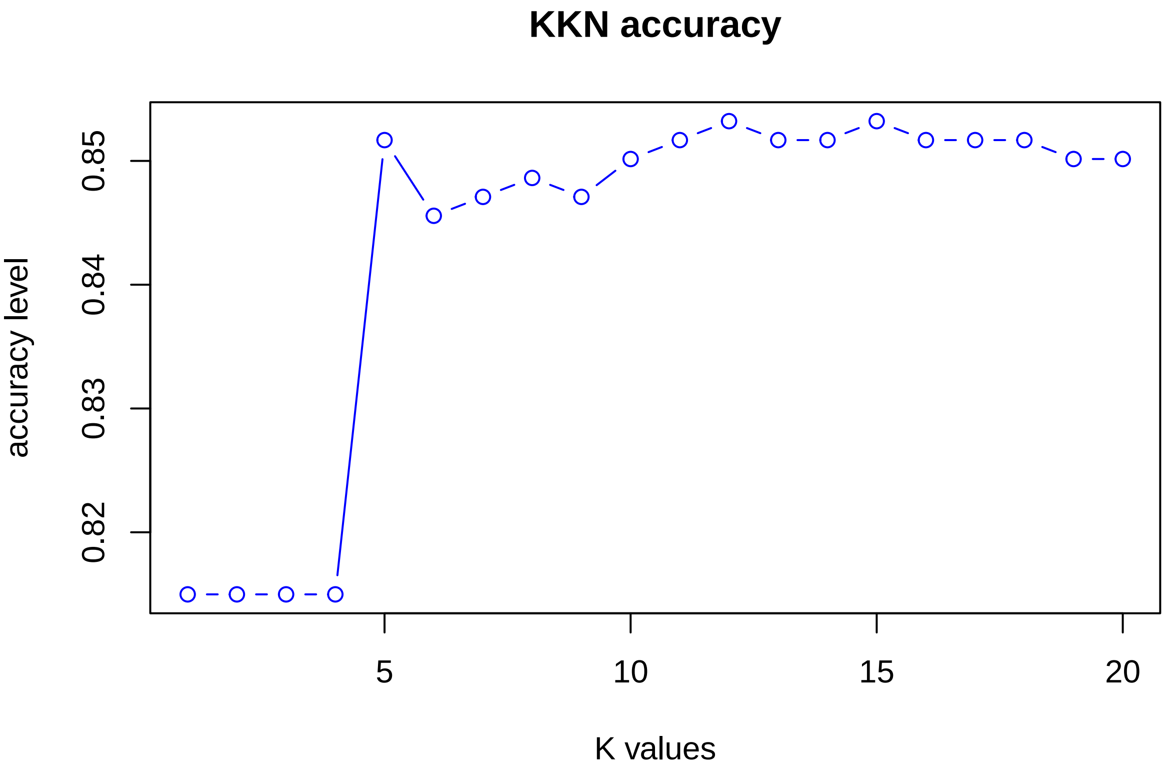
```
library(kknn)

knndata <- read.table("credit_card_data.txt", header = FALSE,stringsAsFactors = FALSE)

head(knndata)
```

```
##   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
```

```
#create function to run for the train and test value
accuracy_check = function(X){

  predicted <- rep(0,(nrow(knndata)))

  for (i in 1:nrow(data))
  {

    knn_credit=kknn(V11~.,knndata[-i,],knndata[i,],k=X,distance = 2,scale = TRUE)

    predicted[i] <- as.integer(fitted(knn_credit)+0.5) # round off to 0 or 1
  }

  # calculate % of correct predictions
  accuracy<- sum(predicted == knndata[,11]) / nrow(data)
  return(accuracy)
}

kknn_accuracy <- rep(0,20)

for (i in 1:20) {
kknn_accuracy[i] <- accuracy_check(i)
}

kknn_result <- data.frame(kknn_accuracy)
View(kknn_result)

best_kvalue <- which.max(kknn_accuracy)
View(best_kvalue)

print(paste0("The highest kknn accuracy rate is ", max(kknn_accuracy),
" and the best k value is ", best_kvalue))
```

```
## [1] "The highest kknn accuracy rate is 0.853211009174312 and the best k value is 12"
```

```
#Plot the graph for accuracy vs. K-values
par(bg="white")
plot(kknn_accuracy,ylab="accuracy level",xlab="K values",type='b',col='blue', main='KKN accuracy')
```


KKN accuracy
accuracy level
K values