Data_Modeling_hw1

Po-Han, Lai

2019/8/22

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.

Predict whether customers will open marketing email or not. We can predict this binary result (open or not open) based on the industry customers in, country customers from, last activity time, product type customers bought, and last time campaign outcome.

Question 2.2

The files 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

First of all, we should load the data before doing any analysis and predictions

R1 in column 11 is a variable that we want to predict(dependent variable). In this report, I will build **SVM Model** in R using ksvm(package kernlab) first and then use knn method(k-Nearest-Neighbor;package kknn) to predict the binary result.

```r
#install.packages("kernlab")
library(kernlab)
```

Because ksvm function requires a matrix data, I use as.matrix in the beginning.

```r
header<-as.matrix(header)
model <- ksvm(header[,1:10],header[,11],type="C-svc",kernel="vanilladot",C=100,scale=TRUE)
```

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

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

```r
# calculate a0
a0 <- -model@b
print(paste0("coefficient is ",a0))
```

```
## [1] "coefficient is 0.081584921659538"
```

```r
# see what the model predicts
```

```
pred <- predict(model,header[,1:10])
# see what fraction of the model???s predictions match the actual classification
accu<-sum(pred == header[,11]) / nrow(header)
print(paste0("Accuracy is ",accu))
```

```
## [1] "Accuracy is 0.863914373088685"
```

To trade off two components of correctness and margin is called c. Hence, I use for loop to test what value is the best. It looks that accuracy is the same while c goes from 0.01 to 100.

```
c<-c(0.0001,0.01,1,100,10000,100000)

different_c<-c()
for (i in c){
  model <- ksvm(header[,1:10],header[,11],type="C-svc",kernel="vanilladot",C=i,scaled=TRUE)
  pred=predict(model,header[,1:10])
  print(i)
  print(sum(pred==header[,11])/nrow(header))
  different_c[i]<-sum(pred==header[,11])/nrow(header)
  }
```

```
##  Setting default kernel parameters
## [1] 1e-04
## [1] 0.5474006
##  Setting default kernel parameters
## [1] 0.01
## [1] 0.8639144
##  Setting default kernel parameters
## [1] 1
## [1] 0.8639144
##  Setting default kernel parameters
## [1] 100
## [1] 0.8639144
##  Setting default kernel parameters
## [1] 10000
## [1] 0.8623853
##  Setting default kernel parameters
## [1] 1e+05
## [1] 0.8639144
```

After considing different **C**, I decide to build different model by using Different Kernel such as **Rbfdot, polydot, anovadot anovadot**

```
cost<-c()
accuracy<-c()
diff_model<-c()
kernel_choice<-c("rbfdot","polydot","vanilladot","anovadot")
for (i in c){
  for (w in kernel_choice){
  model <- ksvm(header[,1:10],header[,11],type="C-svc",kernel=w,C=i,scaled=TRUE)
  pred=predict(model,header[,1:10])
  cost<-append(cost,i)
  acc<-sum(pred==header[,11])/nrow(header)
  accuracy<-append(accuracy,acc)
  diff_model<-append(diff_model,w)}
  }
```

```
##  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
```

```
comparison<-data.frame(cost=cost,accuracy=accuracy,model=diff_model)

comparison<-comparison[order(-accuracy),]
head(comparison,5)
```

```
##     cost  accuracy    model
## 21 1e+05 0.9969419   rbfdot
## 17 1e+04 0.9954128   rbfdot
## 13 1e+02 0.9525994   rbfdot
## 20 1e+04 0.9082569 anovadot
## 16 1e+02 0.9067278 anovadot
```

Rbfdot have highest accuracy when cost is from 1000 to 10000 See the confusion matrix under.

```
best_model <- ksvm(header[,1:10],header[,11],type="C-svc",kernel="rbfdot",C=10000,scaled=TRUE)
pred=predict(best_model,header[,1:10])

table(pred,header[,11])
```

```
##
## pred   0   1
##    0 358   4
##    1   0 292
```

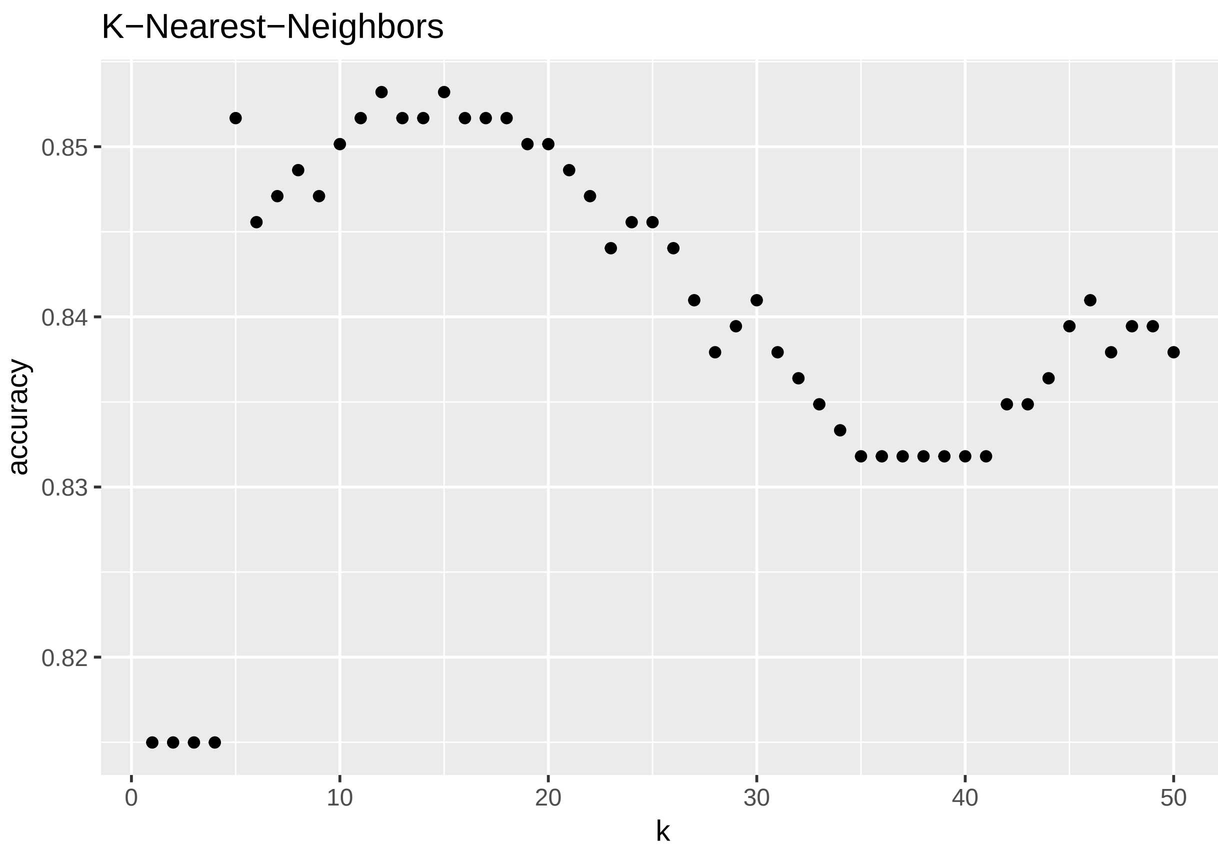
After having models by using SVM, Let us use knn method and compare the accuracy. One important thing is the number of k when you build the k-Nearest Neighbors. Hence, i build a function **knn_accuracy function** to drow a plot and figure out which k has the highest accuracy.

```
#install.packages("kknn")
library(kknn)
header.df<-as.data.frame(header)

knn_accuracy = function(k){
  prediction<-c()
  for (i in 1:nrow(header.df)){
    k_model<-kknn(R1~.,header.df[-i,],header.df[i,],k=k,scale=TRUE)
    prediction_ans<-ifelse(predict(k_model)>=0.5,1,0)
    prediction<-append(prediction,prediction_ans)}
```

```
    accuracy=sum(prediction==header.df[,11])/nrow(header.df)
    return(accuracy)
}

which_k=c()
for (i in 1:50){
  which_k[i]<-knn_accuracy(i)
}

k_model<-data.frame(accuracy<-which_k,k=1:50)
```

Here is the scatter plot 'x' is the k value; 'y' is the accuracy We can find that model has highest accuracy when k is **12** and the accuracy is around **85%**

```
library(ggplot2)
ggplot(aes(x=k,y=accuracy),data=k_model)+geom_point()+ggtitle("K-Nearest-Neighbors")
```



Last but not the least, I would like to use **train.kknn** method, which is a leave-one-out crossvalidation method and try different kerenel to evaluate our model by only train data. First, I decide that train data is 80% of data and test data is remaining 20%. Then, I include kernel method **optimal, rectangular, inv, gaussian, and triangular**.

```
ratio=round(nrow(header.df)*0.2)

sample.index<-sample(1:nrow(header.df),size=ratio,replace=FALSE)

train<-header.df[-sample.index,]
test<-header.df[sample.index,]
```

```r
model2<-train.kknn(R1~.,train, kmax = 100,kernel=c("optimal","rectangular","inv",
                                                   "gaussian","triangular"),scale=TRUE)

print(model2)
```

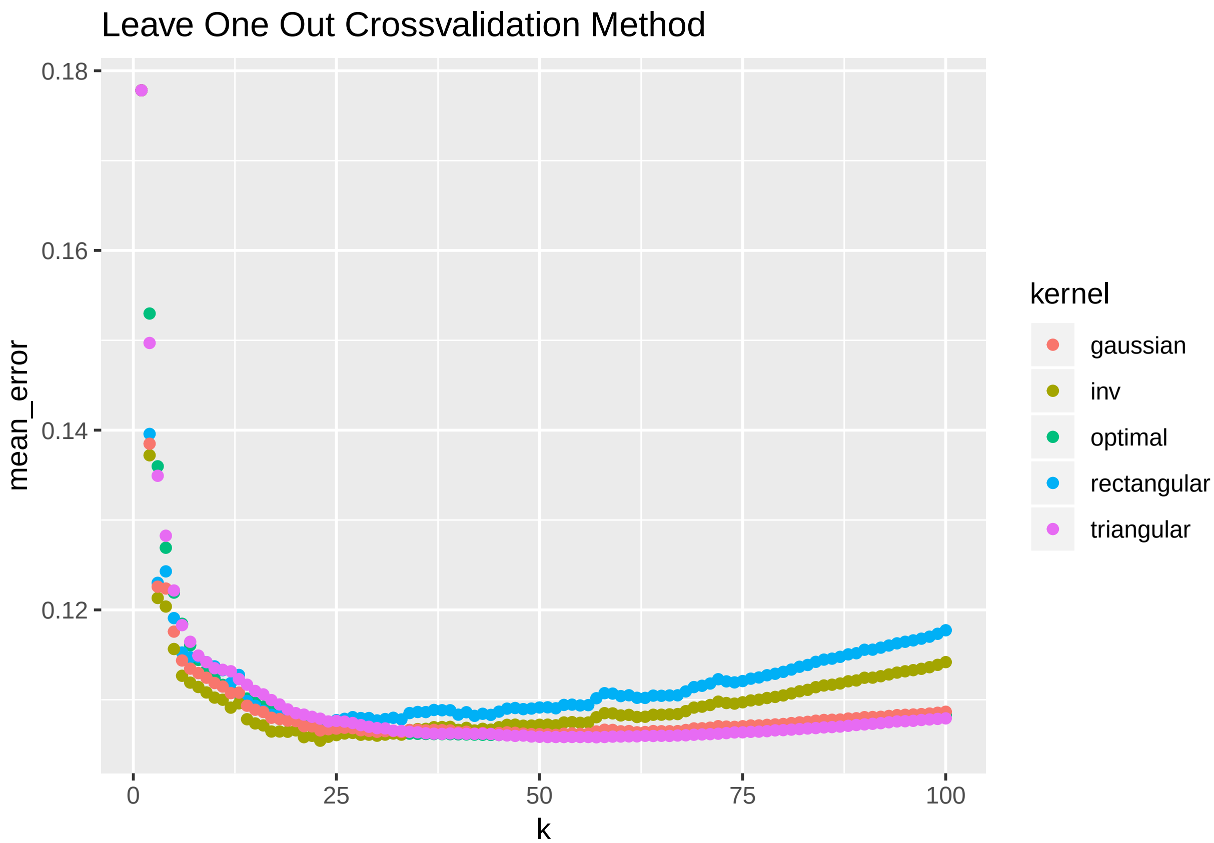
```
##
## Call:
## train.kknn(formula = R1 ~ ., data = train, kmax = 100, kernel = c("optimal",     "rectangular", "inv
##
## Type of response variable: continuous
## minimal mean absolute error: 0.1778203
## Minimal mean squared error: 0.1054376
## Best kernel: inv
## Best k: 23
```

```r
compare_5_kernel<-as.data.frame(model2$MEAN.SQU)

apply(compare_5_kernel,2,which.min)
```

```
##     optimal rectangular         inv    gaussian  triangular
##          46          23          23          52          57
```

Plot the comparison.

```r
library(tidyverse)
compare_5_kernel$k<-seq(1,100,1)
compare_5_kernel_viz<-compare_5_kernel %>% gather(key=kernel,value = mean_error,1:5)

ggplot(aes(x=k,y=mean_error,color=kernel),data=compare_5_kernel_viz) +
  geom_point()+ggtitle("Leave One Out Crossvalidation Method")
```


Leave One Out Crossvalidation Method
0.18
0.16
0.14
0.12
mean_error
0
25
50
75
100
k
kernel
gaussian
inv
optimal
rectangular
triangular