Homework 2

Anonymous

2024-01-22

Question 3.1a

Using the same data set (credit_card_data.txt or credit_card_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier:

(a) *using cross-validation (do this for the k-nearest-neighbors model; SVM is optional);*

```r
library(kknn)

data = read.table(file ="C:\\Users\\sheya\\OneDrive\\Desktop\\credit_card_data.txt"
                  ,header = FALSE)

#Set seed for random variable replication
set.seed(3)

#Random sample of 70% of the rows
random_set <- sample(1:nrow(data), as.integer(0.7*nrow(data)))


trainCCdata = data[random_set,]
#testCCdata is assigned to the residual 30% of the original set
testCCdata = data[-random_set,]

#Used LOOCV to determine kernel and best k-value
train.kknn(as.factor(V11)~.,
           data = trainCCdata,
           kmax = 25,
           scale = TRUE)
```

```
##
## Call:
## train.kknn(formula = as.factor(V11) ~ ., data = trainCCdata,     kmax = 25, scale = TRUE)
##
## Type of response variable: nominal
## Minimal misclassification: 0.1487965
## Best kernel: optimal
## Best k: 9
```

```r
pred_train <- rep(0, nrow(trainCCdata))
accuracy_CCtrain <- 0
for (i in 1:nrow(trainCCdata)){
  CCmodel = kknn(V11~.,
                 trainCCdata[-i,],
                 trainCCdata[i,],
                 k = 9,
                 kernel = "optimal",
                 scale = TRUE)

  pred_train[i] <- round(fitted(CCmodel))
}
#The best k-value is 9 and the best kernel is optimal
accuracy_CCtrain <- sum(pred_train == trainCCdata[,11])/ nrow(trainCCdata)

pred_test <- rep(0, (nrow(testCCdata)))
accuracy_CCtest <- 0

for(i in 1:nrow(testCCdata)){
  CCmodel = kknn(V11~.,
                 testCCdata[-i,],
                 testCCdata[i,],
                 k = 9,
                 kernel = "optimal",
                 scale = TRUE)
  pred_test[i] <- round(fitted(CCmodel))
}
accuracy_CCtest <- sum(pred_test == testCCdata[,11])/ nrow(testCCdata)

accuracy_CCtrain
```

```
## [1] 0.8490153
```

```r
accuracy_CCtest
```

```
## [1] 0.822335
```

I used k-nearest-neighbors model to find a good classifier. I also used Leave One Out Cross Validation to see the optimal hyper-parameters in the k-nearest-neighbors model. This calculated the train data accuracy and running the same model on the test data set for accuracy. The training accuracy will have random effects and the test data accuracy will measure a better accuracy for the model. When using LOOCV the determination of the best kernel and k-value is as follows: kernel = “optimal” and k=9 as the seed was set to 3 for randomization. The accuracy for the test data set is approximately 82% and the accuracy for the train data set is approximately 85%. This signifies random effects calculated on the train data set.

Question 3.1b

(b) *splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).*

```r
library(kknn)

data = read.table(file ="C:\\Users\\sheya\\OneDrive\\Desktop\\credit_card_data.txt",
                  stringsAsFactors = FALSE,
                  header = FALSE)

#Set seed for random varible replication
set.seed(3)

random_set <- sample(1:nrow(data), as.integer(0.7*nrow(data)))

trainCCdata = data[random_set,]

residualCCdata = data[-random_set,]

#Generate a random sample of 30% of the residual rows
random_set2 <- sample(1:nrow(residualCCdata), as.integer(0.3*nrow(residualCCdata)))

validateCCdata = residualCCdata[random_set2,]
testCCdata = residualCCdata[-random_set2,]

pred_train <- rep(0,(nrow(trainCCdata)))

accuracy_CCtrain <- 0

Z <- 0
#Created a table the k-values and their accuracies
df_acc <- data.frame(matrix(nrow = 25, ncol = 2))
colnames(df_acc) <- c("K", "Accuracy")

for(Z in 1:25){

  for(i in 1:nrow(trainCCdata)){
    CCmodel = kknn(V11~.,
                 trainCCdata[-i,],
                 trainCCdata[i,],
                 k = Z,
                 kernel = "optimal",
                 scale = TRUE)
    pred_train[i] <- round(fitted(CCmodel))
  }

  accuracy_CCtrain <- sum(pred_train == trainCCdata[,11]) / nrow(trainCCdata)

  df_acc[Z, 1] <- Z
  df_acc[Z, 2] <- accuracy_CCtrain

}
df_acc
```

```
##     K  Accuracy
## 1   1 0.8052516
## 2   2 0.8052516
## 3   3 0.8052516
## 4   4 0.8052516
## 5   5 0.8424508
## 6   6 0.8446389
## 7   7 0.8424508
## 8   8 0.8468271
## 9   9 0.8490153
## 10 10 0.8490153
## 11 11 0.8468271
## 12 12 0.8468271
## 13 13 0.8446389
## 14 14 0.8424508
## 15 15 0.8402626
## 16 16 0.8336980
## 17 17 0.8336980
## 18 18 0.8336980
## 19 19 0.8380744
## 20 20 0.8402626
## 21 21 0.8358862
## 22 22 0.8380744
## 23 23 0.8358862
## 24 24 0.8358862
## 25 25 0.8358862
```

```
#Seems as though k-value of 12 with holds the
#best accuracy for training data
#Although, 13, 14, & 15 show close accuracies as well
#I will check those values as well against validation to verify k-value of 12
#holding the best accuracy

#Now to Validate those values
pred_validate <- rep(0,(nrow(validateCCdata)))
acc_validate <- 0
Z <- 0
df_validate <- data.frame(matrix(nrow =4, ncol =2))
colnames(df_validate) <- c("K", "Validate Accuracy")
counter <- 0
#Range k-values from 12 to 15
for(Z in 12:15){
  counter <- counter + 1
  for( i in 1:nrow(validateCCdata)){
    CCmodel = kknn(V11~.,
                   validateCCdata[-i,],
                   validateCCdata[i,],
                   k = Z,
                   kernel = "optimal",
                   scale = TRUE)
    pred_validate[i] <- round(fitted(CCmodel))
 }

acc_validate <- sum(pred_validate == validateCCdata[,11])/ nrow(validateCCdata)
df_validate[counter, 1] <- Z
df_validate[counter, 2] <- accuracy_CCtrain
}
```

```r
df_validate
```

```
##    K Validate Accuracy
## 1 12          0.8358862
## 2 13          0.8358862
## 3 14          0.8358862
## 4 15          0.8358862
```

```r
#From the findings above k-values of 13, 14, & 15 perform similarly
#Therefore, will use the k-value 12 for the model.

#Now to run test accuracy for the final result.
pred_test <- rep(0, (nrow(testCCdata)))
accuracy_CCtest <- 0

for(i in 1:nrow(testCCdata)){
  CCmodel = kknn(V11~., testCCdata[-i,], testCCdata[i,],
                 k=12,
                 kernel = "optimal",
                 scale = TRUE)
  pred_test[i] <- round(fitted(CCmodel))

}

accuracy_CCtest <- sum(pred_test == testCCdata[,11]) / nrow(testCCdata)

accuracy_CCtrain
```

```
## [1] 0.8358862
```

```r
accuracy_CCtest
```

```
## [1] 0.8188406
```

Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

A situation for which a clustering model would be efficient is a customer segmentation. The clusters can be assigned to each customer based on *recent visits, frequency, and monetary value*. However, if we examine the size of the clusters, we can range them into smaller clusters as:

- **very recent**
- **excessively frequent**
- **highly valued customers**
- **not very recent**
- **infrequent**

- **low spending customers**

This can mean a lot of clusters or predictors if say the amount of customers evaluated was 600, which then can be labeled into the amount of customers that fall into each cluster as: **regulars, VIP's and travelers**. Those who do not frequent often can be labeled as *travelers* considering they may only frequent when they are in town or *in the neighborhood.* Where as those who frequent often (*regulars), * live near by, caters to convenience, or comfortability. Moreover, the *VIP's* always spend in high amounts that are considered high value every time they indulge. This way the marketing team can use these clusters in an easier and more practical manner.

Question 4.2

Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

```
library(stats)
library(cluster)
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
iris_data = read.table(file ="C:\\Users\\sheya\\OneDrive\\Desktop\\iris.txt",
                  header = TRUE)
data("iris")
head(iris, 4)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
```

```
str(iris)
```

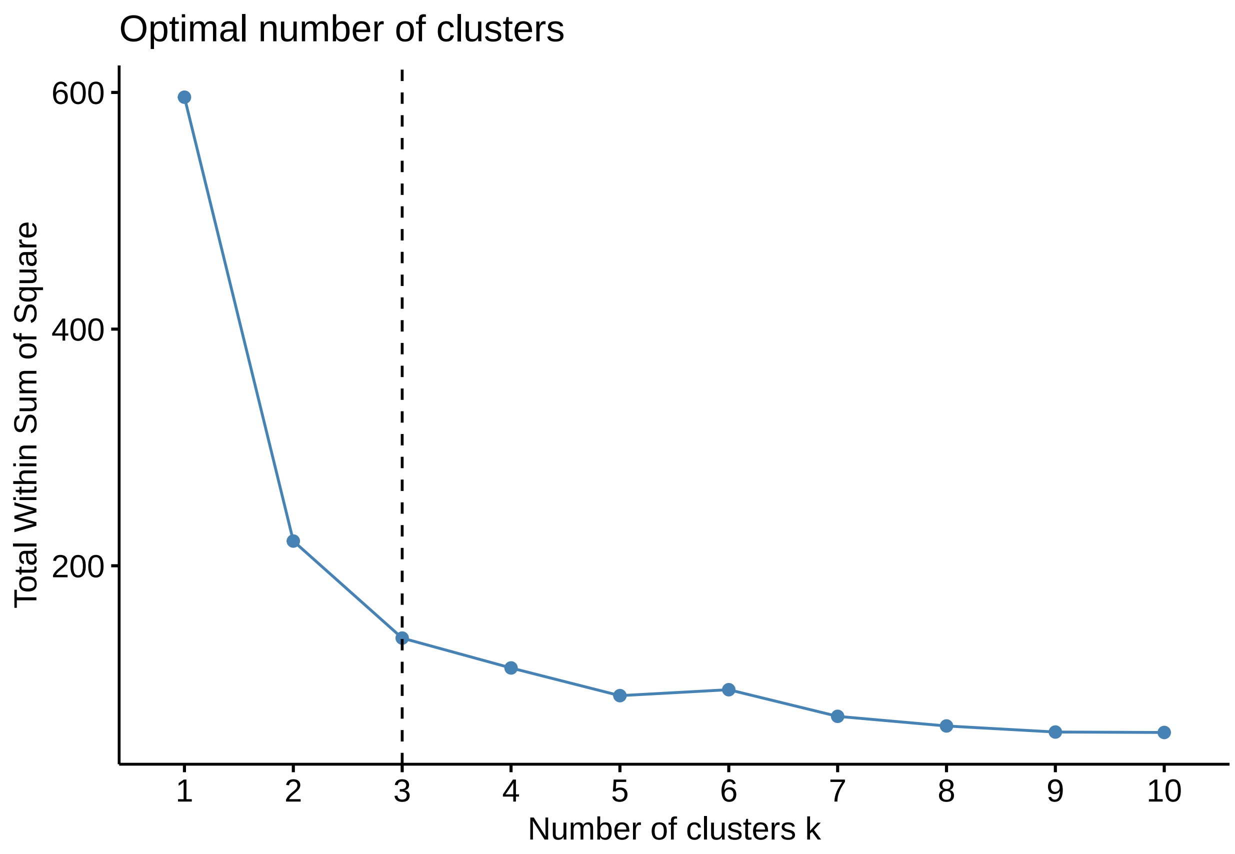
```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
#Excluding the column "Species"
df <- iris[,-5]
df <- scale(df) #standarize the variables


set.seed(1)
```

```r
#Elbow method

fviz_nbclust(df, kmeans, method = "wss") +
  geom_vline(xintercept = 3, linetype = 2)+
  labs(substitute = "Elbow method")
```



```r
iris_cluster <- kmeans(df, 3, nstart = 25)

str(iris_cluster)
```

```
## List of 9
##  $ cluster     : int [1:150] 3 3 3 3 3 3 3 3 3 3 ...
##  $ centers     : num [1:3, 1:4] -0.0501 1.1322 -1.0112 -0.8804 0.0881 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "1" "2" "3"
##   .. ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##  $ totss       : num 596
##  $ withinss    : num [1:3] 44.1 47.5 47.4
##  $ tot.withinss: num 139
##  $ betweenss   : num 457
##  $ size        : int [1:3] 53 47 50
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```r
print(iris_cluster)
```

```
## K-means clustering with 3 clusters of sizes 53, 47, 50
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -0.05005221 -0.88042696    0.3465767   0.2805873
## 2   1.13217737  0.08812645    0.9928284   1.0141287
## 3  -1.01119138  0.85041372   -1.3006301  -1.2507035
##
## Clustering vector:
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1
##  [75] 1 2 2 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
## [112] 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 2 2 2 2 2 2 1 1 2 2 2 1 2 2 2 1 2 2 2 1 2
## [149] 2 1
##
## Within cluster sum of squares by cluster:
## [1] 44.08754 47.45019 47.35062
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
#Best clusting model predicts flower type
best_model <- kmeans(df[,1:4], 3, nstart = 25, iter.max = 30)
table(iris[,5], best_model$cluster)
```

```
##
##               1  2  3
##   setosa     50  0  0
##   versicolor  0 11 39
##   virginica   0 36 14
```

According to my observations, it's possible to define k = 3 as the optimal number of clusters in the data using the "elbow method". In my output, I have three clusters where the size of each cluster is 53, 47 and 50.

From my understanding, using all four predictors Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width give the smallest total within-cluster sum of Squares. The values are quite low which means they are more compact and dense in the cloud of points around its own centroid. If there are compact clusters then this signifies they are well separated from each other and that reveals a ratio close to 100%. Therefore, the ratio I have above is (between_SS / total_SS = 76.7%) which translates to the ratio tends closely to 1 (or 100%) of the flowers.

Using the best model, *setosa* has all of its species in cluster 1 while *versicolor* and *virginica* have some overlapping in each of their cluster.

Works Cited

LOOCV (Leave One Out Cross-Validation) in R Programming https://www.geeksforgeeks.org/loocvleave-one-out-cross-validation-in-r-programming/

RDocumentaion

https://www.rdocumentation.org/packages/kknn/versions/1.3.1/topics/train.kknn

Customer Segementaion Using K-Means Clustering

https://medium.com/data-and-beyond/customer-segmentation-using-k-means-clustering-with-pyspark-unveiling-insights-for-business-8c729f110fab

K Means parameters and results (in R Studio) explained

https://andrea-grianti.medium.com/kmeans-parameters-in-rstudio-explained-c493ec5a05df