ISYE 6501- HW 2 Clustering

Ryan Cherry

2024-01-18

R Markdown

#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:

Question 3.1a

#using cross-validation for the KNN model

I chose to do a 75-25 split into training and test on the kknn model from before. The train.kknn function was used to perform one-out cross validation on the training data. This function determined that the best value to use for k was 51.

Using a value of 51 for k, the classification rate on the training data ended up at 83.67%. This rate was high enough that I felt comfortable scoring the model on the test data.

The model, with k=51, performed better on the test data than the training data, obtaining a classification rate of 84.76%. This is slightly lower than the rate of 85.32% on the model from HW 1, but still a solid performance on new data. As a result, this model with k=51 may be more generalizable than the model from HW 1 that was not split.

```r
#the credit card data file was read in
credit_card_data <- read.delim("C:/Users/ryanc/Downloads/credit_card_data.txt", header=FALSE)

set.seed(123)

#split the credit card data set into training and test sets, where 75%
#of data points were used for training and the other 25% were used for cross-validation
first_split <- sample(1:nrow(credit_card_data),as.integer
(0.75*nrow(credit_card_data)))
train <- credit_card_data[first_split,]
test <- credit_card_data[-first_split, ]

#a kknn model was split up using leave-one-out cross validation
kknn_train <- train.kknn(V11 ~ ., data = train, kmax = 100)
summary(kknn_train)
```

```
##
## Call:
## train.kknn(formula = V11 ~ ., data = train, kmax = 100)
##
## Type of response variable: continuous
## minimal mean absolute error: 0.1897959
## Minimal mean squared error: 0.1063817
## Best kernel: optimal
## Best k: 51
```

```
#created an empty matrix of zeros for the model
#to fill out with probabilities
prediction <- rep(0, nrow(train))

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

#a weighted k-nearest neighbors model was created using the
  #training data
    kknn_model = kknn(V11 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
    train = train[-i,],test = train[i, ], k = 51,
    kernel = 'optimal', scale = TRUE)

#probabilities outputted to the model were rounded to 0 and 1 for
#classification purposes
 if (fitted(kknn_model) >= 0.5) {
     prediction[i] = 1
   } else if (fitted(kknn_model) < 0.5) {
     prediction[i] = 0
   }
}

#see the fraction of classifications that match the model
#predictions for each value of k
classification <- sum(prediction == train[,11]) / nrow(train)

print(paste0("Training set classification rate: ", format(classification * 100, digits = 4), "%"))
```

```
## [1] "Training set classification rate: 83.67%"
```

```
#
prediction_test <- rep(0, nrow(test))

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

#the performance of the weighted k-nearest neighbors model from
#above was scored on the test set using the kknn function
    kknn_model = kknn(V11 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
    train = test[-i,],test = test[i, ], k = 51, kernel = 'optimal', scale = TRUE)

#probabilities outputted to the model were rounded to 0 and 1 for
#classification purposes
 if (fitted(kknn_model) >= 0.5) {
     prediction_test[i] = 1
   } else if (fitted(kknn_model) < 0.5) {
     prediction_test[i] = 0
   }
}

#see the fraction of classifications that match the model
#predictions for each value of kon the training data
```

```r
classification_test <- sum(prediction_test == test[,11]) / nrow(test)

print(paste0("Test set classification rate: ",
format(classification_test * 100, digits = 4), "%"))
```

```
## [1] "Test set classification rate: 84.76%"
```

Question 3.2b

#splitting the data into training, validation, and test data sets for the KNN #model

I again chose to do a 75-25 split on the same kknn model from above. 75% of data was used for training, 12.5% (half of 25%) for validation, and the final 12.5% for test.

On the training data, different values for k up to 30 were tested and the highest classification rate of 84.08% came at k=9 and k=12, respectively.

The classification rate changed the most between k=7 and k=18 on the training data, so for the validation set k values in that range were tested. If the classification rate was close to the 84.08% obtained on the training data, the model would then be evaluated on test data.

Indeed, on the validation data the classification rate proved to be slightly higher, at 84.15% at k=8, k=9, and k=10. In order to reduce the chance of overfitting, k=8 was chosen for use of the test data.

With a value of k=8, the model correctly classified 82.93% of credit card applicants. This rate is lower than one would probably want in the fast-paced world of banking, and I would probably want to continue to try out different values for k and continue refining the model until a classification rate of at least 90% was reached.

However, it is not surprising that the model performed slightly worse on test data.

```r
set.seed(123)

#credit card data set was split with the first 75% going to
#the training set
split1 <- sample(1:nrow(credit_card_data),
as.integer(0.75*nrow(credit_card_data)))

training <- credit_card_data[split1,]

reduced_credit_card_data <- credit_card_data[-first_split,]

#remaining data was split again, with 12.5% of the data (50% of the
#remaining data) going for test and the other 12.5% for validation
split2 <- sample(1: nrow(reduced_credit_card_data), as.integer(0.5 * nrow(reduced_credit_card_data)))

validation <- reduced_credit_card_data[split2,]

test <- reduced_credit_card_data[-split2,]
```

```r
#an empty matrix of 0's was created for the kknn model to fill
#out with probabilities
prediction_train <- c(rep(0, nrow(training)))

#the k value that produces the best classification rate
#was determined from a range of values between 1 and 30
```

```
for (k in seq(1,30, 1)) {

#each of the data points were iterated through to prevent R
#from using i when determining the closest points to i itself
  for (i in 1:nrow(training)) {

#a weighted k-nearest neighbors model was created using the
#training data
kknn_model_training = kknn(V11 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +
V10, train = training[-i,],test = training[i, ], k = k, scale = TRUE)

#probabilities outputted to the model were rounded to 0 and 1 for
#classification purposes
   if (fitted(kknn_model_training) >= 0.5) {
     prediction_train[i] = 1
   } else if (fitted(kknn_model_training) < 0.5) {
     prediction_train[i] = 0
   }

#see the fraction of classifications that match the model
#predictions for each value of k for the training data
classification_train <- sum(prediction_train == training[,11]) /
nrow(training)

}

print(paste0(k, ": ", format(classification_train * 100,
digits = 4), "%"))

}
```

```
## [1] "1: 81.02%"
## [1] "2: 81.02%"
## [1] "3: 81.02%"
## [1] "4: 81.02%"
## [1] "5: 83.06%"
## [1] "6: 82.45%"
## [1] "7: 83.06%"
## [1] "8: 83.67%"
## [1] "9: 84.08%"
## [1] "10: 83.88%"
## [1] "11: 83.67%"
## [1] "12: 84.08%"
## [1] "13: 83.88%"
## [1] "14: 83.88%"
## [1] "15: 83.27%"
## [1] "16: 83.27%"
## [1] "17: 83.47%"
## [1] "18: 83.27%"
## [1] "19: 83.06%"
## [1] "20: 83.06%"
## [1] "21: 83.06%"
## [1] "22: 83.06%"
## [1] "23: 83.06%"
## [1] "24: 82.86%"
## [1] "25: 82.65%"
## [1] "26: 82.65%"
## [1] "27: 82.86%"
## [1] "28: 82.86%"
## [1] "29: 83.27%"
## [1] "30: 83.47%"
```

```
#an empty matrix of 0's was created for the kknn model to fill
#out with probabilities
prediction_validation <- c(rep(0, nrow(validation)))

#the k value that produces the best classification rate
#was determined from a range of values between 7 and 18
for (k in seq(7, 18, 1)) {

#each of the data points were iterated through to prevent R
#from using i when determining the closest points to i itself
  for (i in 1:nrow(validation)) {

#a weighted k-nearest neighbors model was tested on validation data to see if
#further model adjustments were warranted
kknn_model_validation = kknn(V11 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10, train = validation

#probabilities outputted to the model were rounded to 0 and 1 for
#classification purposes on the validation data
   if (fitted(kknn_model_validation) >= 0.5) {
     prediction_validation[i] = 1
   } else if (fitted(kknn_model_validation) < 0.5) {
     prediction_validation[i] = 0
   }

#see the fraction of classifications that match the model
#predictions for each value of k on the validation data
classification_validation <- sum(prediction_validation == validation[,11]) / nrow(validation)

}

print(paste0(k, ": ", format(classification_validation * 100, digits = 4), "%"))

}
```

```
## [1] "7: 82.93%"
## [1] "8: 84.15%"
## [1] "9: 84.15%"
## [1] "10: 84.15%"
## [1] "11: 82.93%"
## [1] "12: 82.93%"
## [1] "13: 82.93%"
## [1] "14: 82.93%"
## [1] "15: 82.93%"
## [1] "16: 82.93%"
## [1] "17: 82.93%"
## [1] "18: 82.93%"
```

```
#an empty matrix of 0's was created for the kknn model to fill
#out with probabilities
prediction_test <- c(rep(0, nrow(test)))


#each of the data points were iterated through to prevent R
#from using i when determining the closest points to i itself
for (i in 1:nrow(test)) {

#the model refined on the validation data was scored using new data
#from the test set
kknn_model_test = kknn(V11 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10, train = test[-i,],test =

#an is-else loop was used to convert the outputted probabilities
#to classifications of one's and zero's
   if (fitted(kknn_model_test) >= 0.5) {
     prediction_test[i] = 1
   } else if (fitted(kknn_model_test) < 0.5) {
     prediction_test[i] = 0
   }

#see the fraction of classifications that match the model
#predictions for each value of k  on the test data
classification_test2 <- sum(prediction_test == test[,11]) / nrow(test)

}

print(paste0("8: ", format(classification_test2 * 100, digits = 4), "%"))
```

```
## [1] "8: 82.93%"
```

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.

One thing that clustering is used for, is when streaming services such as Max and Netflix recommend shows to you. They may use predictors such as: 1. Genre, plot, or cast of previously watched TV shows, so the algorithm can recommend something in the same genre or with the same actors 2. Time spent watching, so the algorithm can recommend something shorter or longer depending on how often or how much time a user watches per day 3. Location, so the algorithm can recommend a show that is popular with users nearby 4. Scrolling activity, so the algorithm can recommend something in a category or genre the user spends the most time looking through 5. Time of day the viewer watches, because viewing patterns are different in morning or afternoon as opposed to later at night

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.

The irises data was scaled before kmeans was run on the set. An elbow diagram, which plots the within sums of squares (wss) against k, can show the optimal value for k to use. The elbow point, where the within-sums-of-squares starts to decrease sharply and the shape of the graph shifts. In this case, the optimal value of k is 3. In this case, the optimal value for k is at 3.

Several kmeans models were built, using every possible combination of 4, 3, and 2 predictors. From each model, an accuracy score was calculated, which simply took the sum of the number of flowers correctly classified in each category and dividing by 150, the total number of observations in this data set.

The best combination of predictors turned out to be only petal length and petal width, where a classification rate of 96% was obtained. As a result, if further data analysis is needed, I would go with the combination of petal length and petal width as predictors at k=3.

Predictor Combination	Classification rate
Petal length, Petal width	96%
Sepal length, Petal length, Petal width	86.67%
Sepal width, Petal length, Petal width	86%

Note: Since over 10 combinations of predictors were tested, only the three with the highest classification rate are shown in the output below.

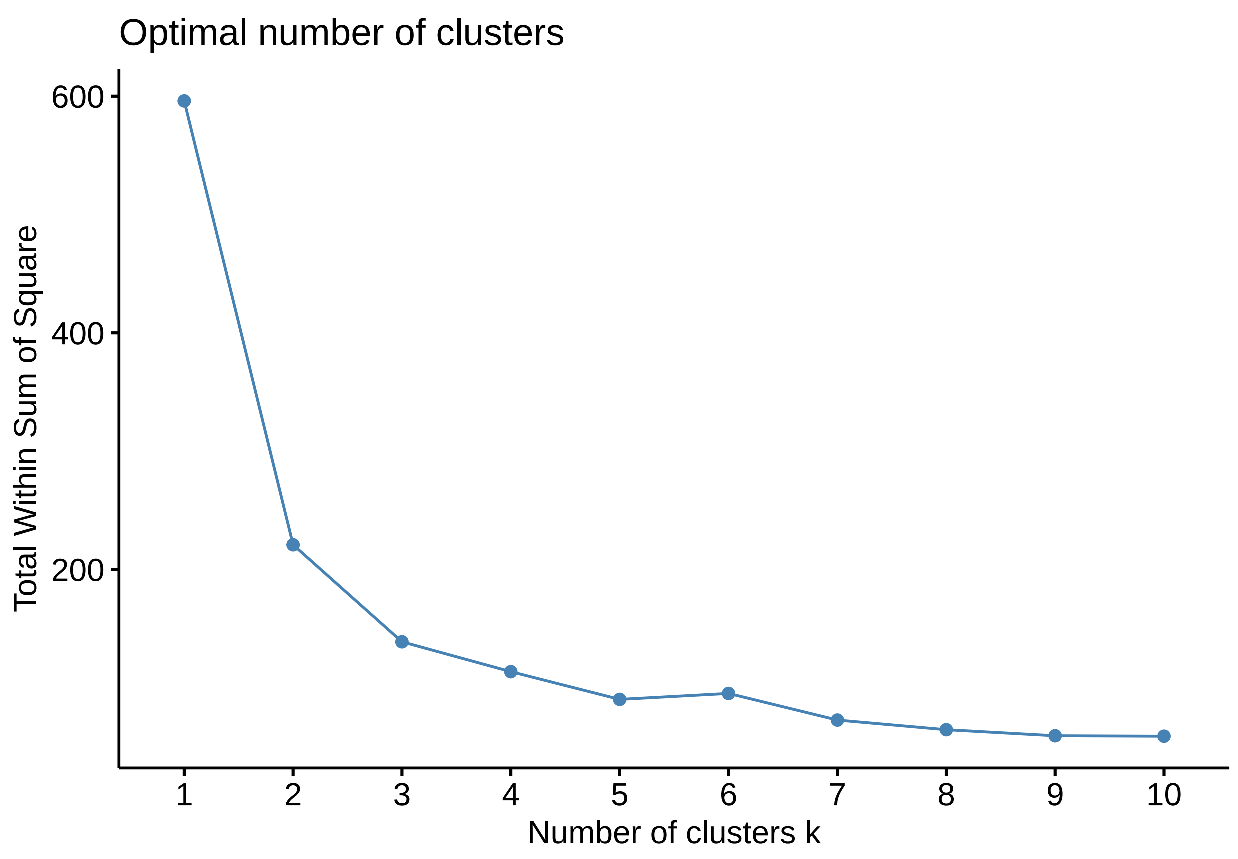
```r
install.packages("datasets")
```

```
## Warning: package 'datasets' is in use and will not be installed
```

```r
library(datasets)
library(kknn)
data(iris)

irises <- iris
head(irises)
```

```
##   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
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
#the dataset was scaled in preparation for the k-means algorithm
set.seed(123)

irises_scale <- scale(irises[c(1:4)])

#an elbow chart was created to determine the optimal value for k
fviz_nbclust(irises_scale, kmeans, method = "wss")
```



```r
set.seed(24)
irises_scale <- scale(irises[c(3,4)])

#tested a kmeans model with only petal length and petal width as predictors
K3_model_34 <- kmeans(irises_scale, centers = 3, nstart = 25)

table(K3_model_34$cluster, irises[, 5])
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         48         4
##   3      0          2        46
```

```r
#classification rate for this model was calculated
classificationK3_34 = (50+48+46)/150
classificationK3_34
```

```
## [1] 0.96
```

```r
set.seed(1234)
irises_scale <- scale(irises[c(1, 3:4)])

#built a kmeans model with only sepal length, petal width, and petal length as
```

```r
#predictors
K3_model_134 <- kmeans(irises_scale, centers = 3, nstart = 25)

table(K3_model_134$cluster, irises[, 5])
```

```
##
##     setosa versicolor virginica
##   1     50          1         0
##   2      0         44        14
##   3      0          5        36
```

```r
#determined the classification rate for the model with sepal length, pedal
#width, and pedal length as predictors
classificationK3_134 = (50+44+36)/150
classificationK3_134
```

```
## [1] 0.8666667
```

```r
set.seed(2)
irises_scale <- scale(irises[c(2:4)])

#tested a model with sepal length, petal width, and petal length as predictors
K3_model_234 <- kmeans(irises_scale, centers = 3,nstart = 25)

table(K3_model_234$cluster, irises[, 5])
```

```
##
##     setosa versicolor virginica
##   1     49          0         0
##   2      1         42        12
##   3      0          8        38
```

```r
classificationK3_234 = (49+42+38)/150
classificationK3_234
```

```
## [1] 0.86
```