KNN Classification Including K Fold Cross Validation for Model Selection

Kent Ng

May 21, 2018

Let's begin by loading the KKNN R package (already installed). We will also set seed value to 42 as best practice.

```
library(kknn)
set.seed(42)
```

Next let's load our data into a table.

```
data_2_3<-read.table("2.2credit_card_data-headersSummer2018.txt", header = T, sep='\t')
```

Train KNN model using the whole dataset (excluding the point being classified itself)

```
predicted<-rep(0,(nrow(data_2_3)))
for (i in 1:nrow(data_2_3)){
  model = kknn(R1~.,data_2_3[-i,],data_2_3[i,],k=10, scale = TRUE)
  predicted[i]<-as.integer(fitted(model)+0.5)
}
```

Finally, let's compare model predictions with actual classification. Note that we will not be using a test/validation dataset for the purpose of this activity; these topics will be touched upon in later projects. However, we will be using k-fold cross validation to select between two models below.

```
sum(predicted == data_2_3[,11]) / nrow(data_2_3)
```

```
## [1] 0.8501529
```

We can explore other values of K to try and see if we can get a better accuracy. First, let's try to vary the k value from 1-20.

```
for (x in 1:20){
  print (x)
  predicted_new<-rep(0,(nrow(data_2_3)))
  for (i in 1:nrow(data_2_3)){
    model = kknn(R1~.,data_2_3[-i,],data_2_3[i,],k=x, scale = TRUE)
    predicted_new[i]<-as.integer(fitted(model)+0.5)
  }
  accuracy = sum(predicted_new == data_2_3[,11]) / nrow(data_2_3)
  print(accuracy)
}
```

```
## [1] 1
## [1] 0.8149847
## [1] 2
## [1] 0.8149847
## [1] 3
## [1] 0.8149847
## [1] 4
## [1] 0.8149847
## [1] 5
## [1] 0.851682
## [1] 6
## [1] 0.8455657
## [1] 7
## [1] 0.8470948
## [1] 8
## [1] 0.8486239
## [1] 9
## [1] 0.8470948
## [1] 10
## [1] 0.8501529
## [1] 11
## [1] 0.851682
## [1] 12
## [1] 0.853211
## [1] 13
## [1] 0.851682
## [1] 14
## [1] 0.851682
## [1] 15
## [1] 0.853211
## [1] 16
## [1] 0.851682
## [1] 17
## [1] 0.851682
## [1] 18
## [1] 0.851682
## [1] 19
## [1] 0.8501529
## [1] 20
## [1] 0.8501529
```

It appears that k = 12 and 15 gives the best model accuracy. Let’s try varying the k value in bigger magnitudes.

```
for (x in c(15,30,50,100,200,400,600)){
  print (x)
  predicted_new<-rep(0,(nrow(data_2_3)))
  for (i in 1:nrow(data_2_3)){
    model = kknn(R1~.,data_2_3[-i,],data_2_3[i,],k=x, scale = TRUE)
    predicted_new[i]<-as.integer(fitted(model)+0.5)
  }
  accuracy = sum(predicted_new == data_2_3[,11]) / nrow(data_2_3)
  print(accuracy)
}
```

```
## [1] 15
## [1] 0.853211
## [1] 30
## [1] 0.8409786
## [1] 50
## [1] 0.8379205
## [1] 100
## [1] 0.8363914
## [1] 200
## [1] 0.8363914
## [1] 400
## [1] 0.8318043
## [1] 600
## [1] 0.8363914
```

It is obvious that as the value of k continues to increase, the model accuracy continues to decrease. Therefore, we can confidently say that 12 and 15 are good choices for K. Below we will use k fold cross validation to determine which model is better.

First we will need another library.

```
library(data.table)
```

To confirm that the our results are in line with what we discovered above, let's test k values from 1 to 20.

```
for(i in 1:20){
  cv = cv.kknn(R1~.,data_2_3,kcv=10,scale=TRUE, k = i)
  cv = data.table(cv[[1]])
  #print ('Cross Validation Accuracy')
  print(i)
  accuracy = sum(cv$y == round(cv$yhat))/nrow(data_2_3)
  print(accuracy)
}
```

```
## [1] 1
## [1] 0.8134557
## [1] 2
## [1] 0.8211009
## [1] 3
## [1] 0.8165138
## [1] 4
## [1] 0.8058104
## [1] 5
## [1] 0.8455657
## [1] 6
## [1] 0.853211
## [1] 7
## [1] 0.8348624
## [1] 8
## [1] 0.8501529
## [1] 9
## [1] 0.8501529
## [1] 10
## [1] 0.853211
## [1] 11
## [1] 0.8425076
## [1] 12
## [1] 0.8608563
## [1] 13
## [1] 0.8425076
## [1] 14
## [1] 0.8455657
## [1] 15
## [1] 0.8562691
## [1] 16
## [1] 0.8440367
## [1] 17
## [1] 0.8425076
## [1] 18
## [1] 0.851682
## [1] 19
## [1] 0.8501529
## [1] 20
## [1] 0.8470948
```

From this, we can see that the results align with what we concluded previously. Here, we can see that the knn model with k = 12 has the highest model accuracy; thus this is the classifier of choice.