R for Classification: *K* Nearest Neighbors
Dolores Romero Morales

The aim of this workshop is to work on *K* Nearest Neighbors using a dataset you are familiar with (newhousing.txt). This file can be found on the course directory. This file has thirteen explanatory variables. The last column is the class membership (‘pricelevel’). There are two classes: ‘below’ and ‘above’.

Recall that we will need to install the corresponding R packages.
Note: To build *K* Nearest Neighbors we will use the R package 'FNN'. Follow the two steps below to complete the installation and uploading of the package:

```
install.packages('FNN',dependencies=TRUE)
library('FNN')
```

Step 1. Read the newhousing.txt file into a data frame and get the dimension of the dataset.
Answer:

```
mynewhousing <- read.table(file.choose(),header=T, stringsAsFactors=TRUE)
dim(mynewhousing)
```

Note:
Recall that FNN uses the Euclidean distance to measure the similarity between individuals and therefore to be used you need ‘numeric’ or ‘integer’ explanatory variables. Recall you can see the type of data using the command str().

Step 2. Reshuffle the dataset and take a subsample of 400 observations. Call this the *minihousing*. Call the remaining dataset the *testhousing*. Split both datasets into explanatory variables and response variable.
Answer:

```
set.seed(1)
reshuffle <- mynewhousing[sample(nrow(mynewhousing)),]
minihousing <- reshuffle[1:400,]
testhousing <- reshuffle[401: nrow(reshuffle),]
myxtrain <- minihousing[,-14]
myytrain <- minihousing[,14]
myxtesting <- testhousing[,-14]
myytesting <- testhousing[,14]
```

Note1:
Recall that once you have reshuffled the data using ‘sample’, we are able to split the new dataset ‘reshuffle’ into two subsamples.
Note2:
Prior to the function that reshuffles the dataset, you may want to set the seed for the random generator using the function set.seed(). If you do so, then you are ensuring that you can reproduce in the future the same reshuffling.
Note3:
It is a good habit to use the ‘print’ function or the ‘summary’ function to get acquainted with the data, see whether we have created the two subsamples properly.
Note4:
It is not asked in the question, but if you want the class split in the minihousing you can use these couple of lines

```
priceleveltable <- table(minihousing$pricelevel)
aboveinminihousing <- priceleveltable[1]/nrow(minihousing)
belowinminihousing <- priceleveltable[2]/nrow(minihousing)
```

Note5:
Please note that we have been asked to split the minihousing and the testhousing into four pieces. First, we collect all the explanatory variables of the minihousing dataset, then we collect the response variable. We repeat this process with testhousing. We have to do this in order to be able to use the FNN package.

Step 3. Use the *minihousing* subsample to classify the *testhousing* sample using *K* Nearest Neighbors, with *K*=1.
Answer:
```
myk1nn <- knn(train= myxtrain, test= myxtesting, cl= myytrain, k=1)
myaccuracytablek1nn <- table (myk1nn,myytesting)
mytestingaccuracyk1nn <- sum(diag(myaccuracytablek1nn))/sum(myaccuracytablek1nn)
```
Note:
Please note that here we are just practicing the use of the knn() function, with a given value of k. In general, we do not know the value of k, and therefore we use crossvalidation to find a good value of this parameter for this dataset.

Step 4. Use the *minihousing* subsample to classify the *testhousing* sample using *K* Nearest Neighbors, with *K*=3.
Answer:
```
myk3nn <- knn(train= myxtrain, test= myxtesting, cl= myytrain, k=3)
myaccuracytablek3nn <- table (myk3nn,myytesting)
mytestingaccuracyk3nn <- sum(diag(myaccuracytablek3nn))/sum(myaccuracytablek3nn)
```
Note:
Please note that here we are just practicing the use of the knn() function, with a given value of k. In general, we do not know the value of k, and therefore we use crossvalidation to find a good value of this parameter for this dataset.

Step 5. Use the *minihousing* subsample to tune the value of *K* using leave-one-out crossvalidation.
Answer:
```
bestk=0
bestaccuracy=0
accuracy <- NULL
for(auxk in  1:15){
mycv <- knn.cv(train= myxtrain, cl= myytrain, k=auxk)
mytable <- table (mycv, myytrain)
accuracy[auxk] <- sum(diag(mytable))/sum(mytable)
if(bestaccuracy< accuracy[auxk]) bestk=auxk
if(bestaccuracy< accuracy[auxk]) bestaccuracy = accuracy[auxk]}
plot(accuracy, xlab="K", ylab="Crossvalidated Accuracy")
```
Note1:
Note that it would have been nicer to have a tuning function to choose the best value of K. We have seen these functions in previous weeks. That depends on the R package we are using. FNN has not got this function.
Note2:
We have decided to plot in Figure 1 the performance of the crossvalidation step K-NN for different values of K.
Note3:
Please note that in the lines of coding above, you will find the best value of K (bestk), i.e., the value in which the crossvalidated accuracy is the highest.

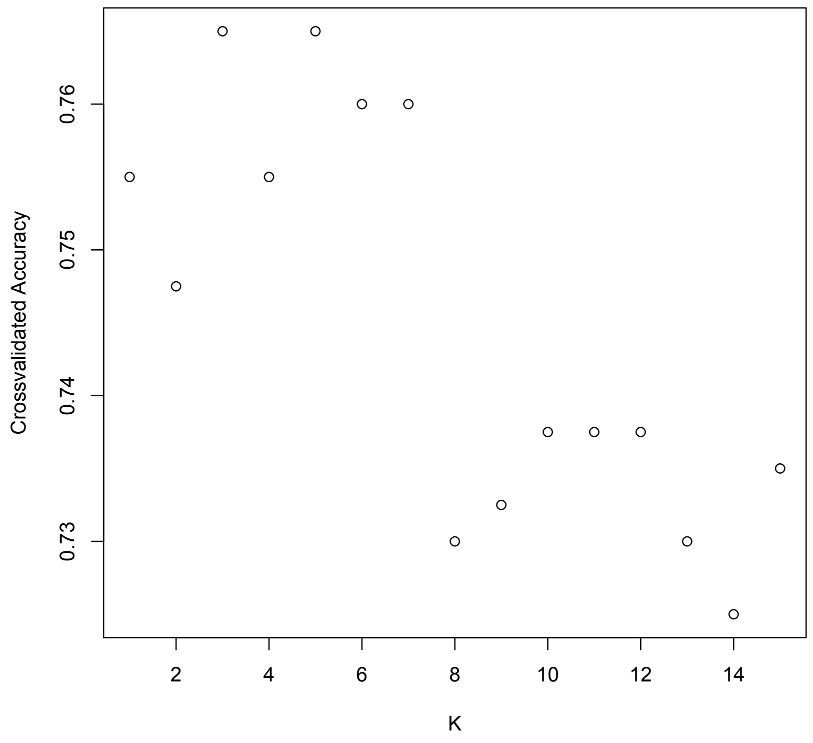

Figure 1 Visualizing the accuracy results from K-NN

Step 6. Use the *minihousing* subsample to classify the *testhousing* sample using *K* Nearest Neighbors, with the best *K* found in Step 5.

Answer:

```
mybestknn <- knn(train= myxtrain, test= myxtesting, cl= myytrain, k=bestk)
mytestingaccuracytablebestknn <- table (mybestknn,myytesting)
mytestingaccuracybestknn<-
sum(diag(mytestingaccuracytablebestknn))/sum(mytestingaccuracytablebestknn)
```

Step 7. Repeat Step 2 through Step 6, but with data being normalized in Step 2.

Answer:

```
%This reads the data
mynewhousing <- read.table(file.choose(),header=T, stringsAsFactors=TRUE)
%This leaves out a sample testhousing to report the final performance
set.seed(1)
reshuffle <- mynewhousing[sample(nrow(mynewhousing)),]
minihousing <- reshuffle[1:400,]
testhousing <- reshuffle[401: nrow(reshuffle),]
%This normalizes the training data, and uses the same normalization for the testing data
%Please note that in the training data all variables are now between 0 and 1
%However, this does not need to be the case in the testing data
minihousingNORM<- minihousing
testhousingNORM<- testhousing
for(i in 1:length(colnames(minihousing))-1) {
    if(class(minihousing[,i]) == "numeric" || class(minihousing[,i]) == "integer") {
        minimum<-min(minihousing[,i])
```

```
          maximum<-max(minihousing[,i])
          minihousingNORM[,i] <-
as.vector(scale(minihousing[,i],center=minimum,scale=maximum-minimum))
          testhousingNORM[,i] <-
as.vector(scale(testhousing[,i],center=minimum,scale=maximum-minimum))
}
      }
summary(minihousingNORM)
summary(testhousingNORM)
%This constructs the blocks that K-NN needs
myxtrainNORM <- minihousingNORM[,-14]
myytrainNORM <- minihousingNORM[,14]
myxtestingNORM <- testhousingNORM[,-14]
myytestingNORM <- testhousingNORM[,14]
%This runs cross validation to find the best value of K
bestk=0
bestaccuracy=0
accuracy <- NULL
for(auxk in  1:15){
mycv <- knn.cv(train= myxtrainNORM, cl= myytrainNORM, k=auxk)
mytable <- table (mycv, myytrainNORM)
accuracy[auxk] <- sum(diag(mytable))/sum(mytable)
if(bestaccuracy< accuracy[auxk]) bestk=auxk
if(bestaccuracy< accuracy[auxk]) bestaccuracy = accuracy[auxk]}
plot(accuracy, xlab="K", ylab="Crossvalidated Accuracy")
%This now runs K-NN with the best value of K found in the crossvalidation step, and
%reports the accuracy on testhousing
mybestknn <- knn(train= myxtrainNORM, test= myxtestingNORM, cl= myytrainNORM,
k=bestk)
mytestingaccuracytablebestknn <- table (mybestknn,myytestingNORM)
mytestingaccuracybestknn<-
sum(diag(mytestingaccuracytablebestknn))/sum(mytestingaccuracytablebestknn)
```

Note1:

Please note we have not performed Steps 3 and 4, as these are practicing steps.

Note2:

We have decided to plot in Figure 2 the performance of the crossvalidation step K-NN for different values of K.

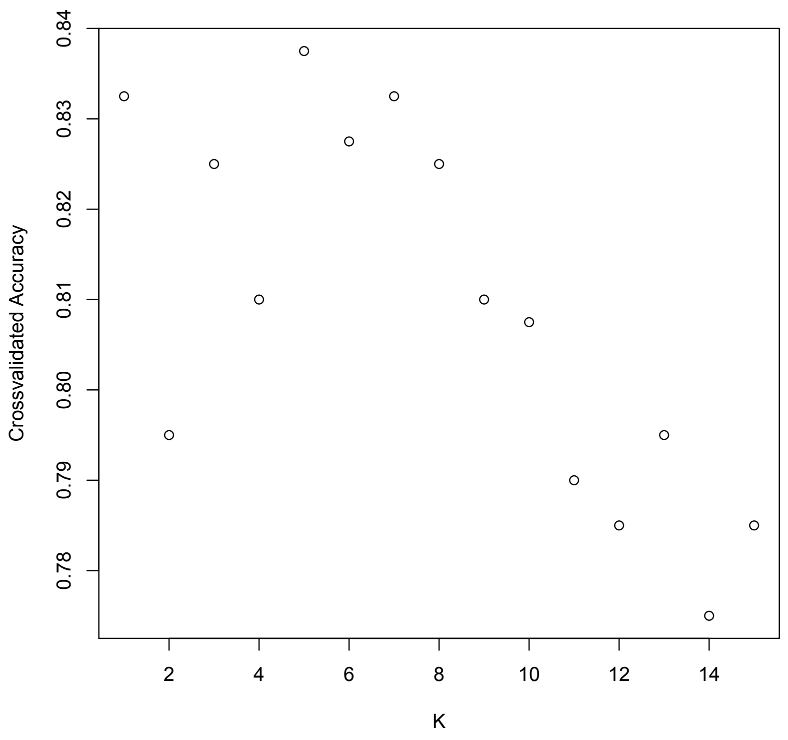

Figure 2 Visualizing the accuracy results from K-NN (with normalized data)