CPTR330 – Homework 1

Pierre Visconti

4/3/2023

K-Nearest Neighbors Algorithm

K-Nearest Neighbor (KNN), is a supervised learning classification algorithm. It will classify a given input by looking at its k nearest neighbors and the class that those neighbors are categorized as. It will then classify the input based on the most frequent class observed from the nearest neighbors. The strengths of KNN is that it makes zero assumptions about the data and it is very simple to implement. Some weaknesses about KNN is that it needs to store the entire training data in memory, it does not produce a model which can later be used. It also has a time complexity of O(n) which means that large datasets will take a while to predict.

Step 1 - Collect Data

The Iris dataset gives the measurements for sepal length/width, and petal length/width in cm for flowers from three species of Iris'. The source of the data is Annals of Eugenics (1936) by R.A. Fisher and the Bulletin of the American Iris Society (1935) by Edgar Anderson.

```r
# import data
data("iris")
```

Step 2 - Exploring And Preparing The Data

```r
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 ...
```

```r
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

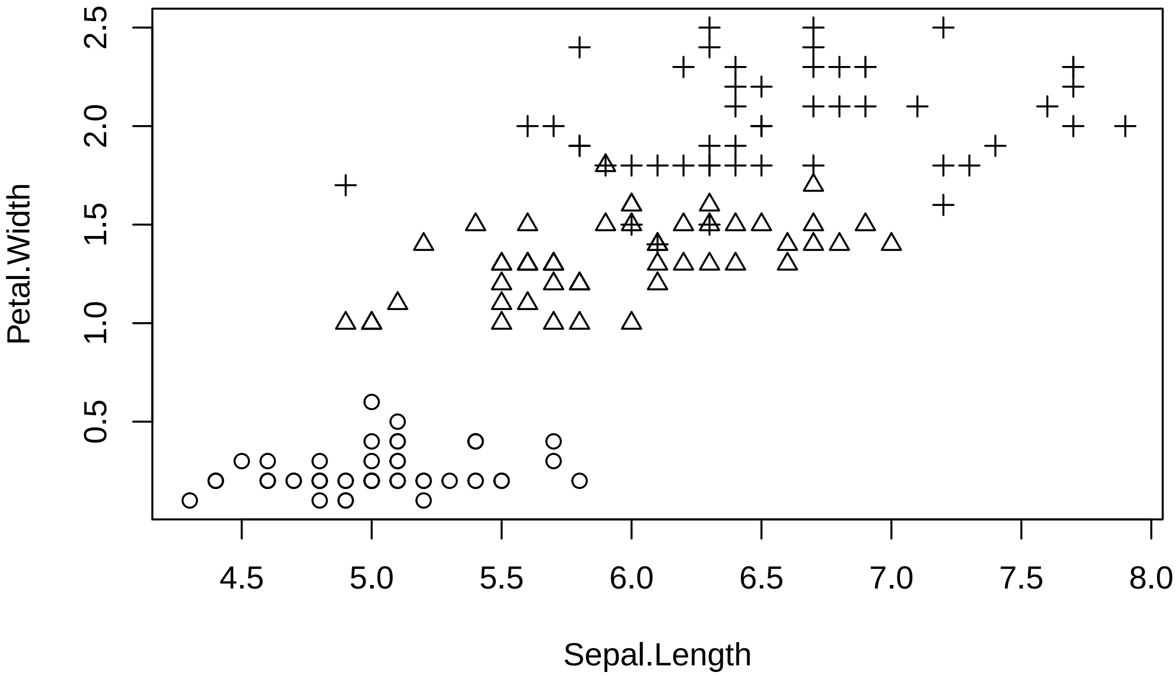
```r
round(prop.table(table(iris$Species)) * 100, digits = 1)
```

```
##
##     setosa versicolor  virginica
##       33.3       33.3       33.3
```

There are 50 flowers for each species of Iris, for a total of 150 observations. There are four numerical variables (length/width measurements) and one categorical variable (species of Iris). The data is not randomized, it is organized by species of Iris so it will need to be randomly sampled from when creating the training/testing datasets.

Lets focus on two features, "Sepal.Length" and "Petal width" to get a better picture of data.

```
plot(iris[c("Sepal.Length", "Petal.Width")], pch=c(iris$Species))
```



The three species are plotted on the graph with a different symbol for each species. There already appears to be a pretty good distinction (groups) between the species which means that a classification algorithm like KNN should perform well on this dataset.

```
summary(iris[c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")])
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

As can be seen from the code chunk above, the variables are not scaled the same, so the data needs to be normalized.

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
# create a new variable for the normalized dataset
iris_n <- as.data.frame(lapply(iris[1:4], normalize))
summary(iris_n[c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")])
```

```
##   Sepal.Length     Sepal.Width      Petal.Length     Petal.Width     
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000  
##  1st Qu.:0.2222   1st Qu.:0.3333   1st Qu.:0.1017   1st Qu.:0.08333  
##  Median :0.4167   Median :0.4167   Median :0.5678   Median :0.50000  
##  Mean   :0.4287   Mean   :0.4406   Mean   :0.4675   Mean   :0.45806  
##  3rd Qu.:0.5833   3rd Qu.:0.5417   3rd Qu.:0.6949   3rd Qu.:0.70833  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
```

The data is now normalized.

```
# setting seed so that results do not change when knitting to pdf
set.seed(1)
# create training and test data
train.size <- round(nrow(iris_n)*0.8) # 80% of the dataset used for training
train.ind <- sample(1:nrow(iris_n), train.size)
iris_train <- iris_n[train.ind,]
iris_test <- iris_n[-train.ind,]
# create labels for training and test data
iris_train_labels <- iris[train.ind, 5]
iris_test_labels <- iris[-train.ind, 5]
```

Step 3 - Training A Model On The Data

The k-nearest neighbors takes a training and test dataset, the train labels and a value for k. A good starting value for k is typically the square root of the number of records. In this case, the training dataset has 150 records so our starting k will be 12 (after rounding down).

```
library(class)
iris_test_pred <- knn(train = iris_train,
                      test = iris_test,
                      cl = iris_train_labels,
                      k = 12)
```

Step 4 - Evaluating Model Performance

The model has three different species to predict. For each species there are two possible outcomes, a correct classification of the species and a classification as the wrong species. For the setosa species, there were 11 total observations in the testing set. Of these 11 the model predicted all 11 correctly. For the versicolor species, there were 12 total observations in the testing set. Of these 12 the model predicted 12 all correctly. For the virginica species there were 7 total observations in the testing set. Of these 7 the model predicted 6 correctly and 1 incorrectly. This means the model predicted 29/30 observations correctly resulting in an accuracy of 96.7%.

```
library(gmodels)
table(iris_test_labels, iris_test_pred)
```

```
##                   iris_test_pred
## iris_test_labels setosa versicolor virginica
##       setosa         11          0         0
##       versicolor      0         12         0
##       virginica       0          1         6
```

```
CrossTable(x = iris_test_labels,
           y = iris_test_pred,
           prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
```

```
## |-------------------------|
##
##
## Total Observations in Table:  30
##
##
##                  | iris_test_pred
## iris_test_labels |     setosa | versicolor |  virginica |  Row Total |
## -----------------|------------|------------|------------|------------|
##           setosa |         11 |          0 |          0 |         11 |
##                  |      1.000 |      0.000 |      0.000 |      0.367 |
##                  |      1.000 |      0.000 |      0.000 |            |
##                  |      0.367 |      0.000 |      0.000 |            |
## -----------------|------------|------------|------------|------------|
##       versicolor |          0 |         12 |          0 |         12 |
##                  |      0.000 |      1.000 |      0.000 |      0.400 |
##                  |      0.000 |      0.923 |      0.000 |            |
##                  |      0.000 |      0.400 |      0.000 |            |
## -----------------|------------|------------|------------|------------|
##        virginica |          0 |          1 |          6 |          7 |
##                  |      0.000 |      0.143 |      0.857 |      0.233 |
##                  |      0.000 |      0.077 |      1.000 |            |
##                  |      0.000 |      0.033 |      0.200 |            |
## -----------------|------------|------------|------------|------------|
##     Column Total |         11 |         13 |          6 |         30 |
##                  |      0.367 |      0.433 |      0.200 |            |
## -----------------|------------|------------|------------|------------|
##
##
```

Step 5 - Improving Model Performance

We have two options available to try and improve the model: The first is that we can normalize the data using a z-score standardization instead of min-max. The second is that we can tune the value of k to see if we can get better results.

```r
# z-scale standardization
iris_z <- as.data.frame(scale(iris[-5]))

# confirm that the transformation was applied correctly
summary(iris_z[c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")])
```

```
##   Sepal.Length       Sepal.Width       Petal.Length      Petal.Width
##  Min.   :-1.86378   Min.   :-2.4258   Min.   :-1.5623   Min.   :-1.4422
##  1st Qu.:-0.89767   1st Qu.:-0.5904   1st Qu.:-1.2225   1st Qu.:-1.1799
##  Median :-0.05233   Median :-0.1315   Median : 0.3354   Median : 0.1321
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.67225   3rd Qu.: 0.5567   3rd Qu.: 0.7602   3rd Qu.: 0.7880
##  Max.   : 2.48370   Max.   : 3.0805   Max.   : 1.7799   Max.   : 1.7064
```

```r
# create training and test data using the same indexes as before so a direct comparison can be made
iris_train <- iris_z[train.ind,]
iris_test <- iris_z[-train.ind,]

# re-classify test cases
```

```r
iris_test_pred <- knn(train = iris_train,
                      test = iris_test,
                      cl = iris_train_labels,
                      k = 12)

# Create the cross tabulation of predicted vs. actual
CrossTable(x = iris_test_labels,
           y = iris_test_pred,
           prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  30
##
##
##                  | iris_test_pred
## iris_test_labels |     setosa | versicolor |  virginica |  Row Total |
## -----------------|------------|------------|------------|------------|
##           setosa |         11 |          0 |          0 |         11 |
##                  |      1.000 |      0.000 |      0.000 |      0.367 |
##                  |      1.000 |      0.000 |      0.000 |            |
##                  |      0.367 |      0.000 |      0.000 |            |
## -----------------|------------|------------|------------|------------|
##       versicolor |          0 |         12 |          0 |         12 |
##                  |      0.000 |      1.000 |      0.000 |      0.400 |
##                  |      0.000 |      0.923 |      0.000 |            |
##                  |      0.000 |      0.400 |      0.000 |            |
## -----------------|------------|------------|------------|------------|
##        virginica |          0 |          1 |          6 |          7 |
##                  |      0.000 |      0.143 |      0.857 |      0.233 |
##                  |      0.000 |      0.077 |      1.000 |            |
##                  |      0.000 |      0.033 |      0.200 |            |
## -----------------|------------|------------|------------|------------|
##     Column Total |         11 |         13 |          6 |         30 |
##                  |      0.367 |      0.433 |      0.200 |            |
## -----------------|------------|------------|------------|------------|
##
##
```

The results for z-score standardization are exactly the same as our original previous model.

Now lets try different values for *k*. To keep the output down, all results will be shown with a simple table. For these checks, the percentages are not required to understand the performance.

```r
# try several different values of k
iris_train <- iris_n[train.ind,]
```

```r
iris_test <- iris_n[-train.ind, ]


iris_test_pred <- knn(train = iris_train, test = iris_test, cl = iris_train_labels, k = 1)
table(iris_test_labels, iris_test_pred)
```

```
##                 iris_test_pred
## iris_test_labels setosa versicolor virginica
##       setosa         11          0         0
##       versicolor      0         12         0
##       virginica       0          1         6
```

```r
iris_test_pred <- knn(train = iris_train, test = iris_test, cl = iris_train_labels, k = 5)
table(iris_test_labels, iris_test_pred)
```

```
##                 iris_test_pred
## iris_test_labels setosa versicolor virginica
##       setosa         11          0         0
##       versicolor      0         12         0
##       virginica       0          1         6
```

```r
iris_test_pred <- knn(train = iris_train, test = iris_test, cl = iris_train_labels, k = 11)
table(iris_test_labels, iris_test_pred)
```

```
##                 iris_test_pred
## iris_test_labels setosa versicolor virginica
##       setosa         11          0         0
##       versicolor      0         12         0
##       virginica       0          1         6
```

```r
iris_test_pred <- knn(train = iris_train, test = iris_test, cl = iris_train_labels, k = 15)
table(iris_test_labels, iris_test_pred)
```

```
##                 iris_test_pred
## iris_test_labels setosa versicolor virginica
##       setosa         11          0         0
##       versicolor      0         12         0
##       virginica       0          1         6
```

```r
iris_test_pred <- knn(train = iris_train, test = iris_test, cl = iris_train_labels, k = 21)
table(iris_test_labels, iris_test_pred)
```

```
##                 iris_test_pred
## iris_test_labels setosa versicolor virginica
##       setosa         11          0         0
##       versicolor      0         12         0
##       virginica       0          1         6
```

```r
iris_test_pred <- knn(train = iris_train, test = iris_test, cl = iris_train_labels, k = 27)
table(iris_test_labels, iris_test_pred)
```

```
##                 iris_test_pred
## iris_test_labels setosa versicolor virginica
##       setosa         11          0         0
##       versicolor      0         12         0
##       virginica       0          1         6
```

“‘ For all values of k the results are exactly the same as our original model. Therefore an improvement was not made over the original model from either possible options.

Autograding

```
.AutograderMyTotalScore()
```

```
##
## Step 1:      0/1
## Step 2:      0/1
## Step 3:      0/1
## Step 4:      0/1
## Step 5:      0/1
## Total Score: 0/5
```