ISyE 6501 HW2

September 5, 2019

Question 3.1

1. We first load the data, then use the function cv.kknn to find the best value of k based on cross validation.

```
# Load necessary packages
# install.packages("kknn")
library(kknn)

# load Data
data <- read.delim("credit_card_data-headers.txt", header=TRUE) # read data
cc_df = as.data.frame(data)
set.seed(123)  # set seed for reproducibility

# create an empty array to keep track of the accuracy of predictions
accuracy_array = vector()

# loop through each value of k, and test the accuracy of the cross validation
# assume 10-fold cross validation
# assume a max k of 100
for (k in 1:100){
  knn_fit = cv.kknn(R1 ~ .,
                    data = cc_df,
                    kcv = 10,
                    k = k,
                    scale = T)
  prediction = as.integer(round(knn_fit[[1]][,2]))
  accuracy_array[k] = sum(prediction == cc_df$R1) / length(cc_df$R1)
}

best_k = which.max(accuracy_array)
best_accuracy = round(accuracy_array[best_k] * 100, digits = 2)

print(best_k)
```

```
## [1] 5
```

```
print(best_accuracy)
```

```
## [1] 85.47
```

We find that the 'best' for k using cross validation is 5, and it produces an accuracy of 85.47%.

2. We create a function "cross_kknn" with arg for the number of training units. First, it splits the dataset into training and test sets. Second, it predicts the correct guesses using the test data. And last, it prints the model and percent of correct guesses. Note that validation is already embedded in train.kknn (or cv.kknn), so we do not need a separate step for splitting the data into a validation set. But we do need to test, however; and that is what we do here.

```r
cross_kknn <- function(split){
  n = nrow(cc_df)
  train = sample(1:n, split*n, replace = FALSE)
  training = cc_df[train,]
  test = cc_df[-train,]
  kernel = c("optimal","rectangular", "inv", "gaussian", "triangular")
  model = train.kknn(R1 ~ ., data = training, kmax = 100, kernel = kernel, scale = TRUE)
  pred <- round(predict(model, test))
  correct = sum(pred == test$R1)/length(test$R1)
  output <- list(model, correct)
  return(output)
}

# executing cross_kknn with training/test split 80/20
cross_kknn(0.8)
```

```
## [[1]]
##
## Call:
## train.kknn(formula = R1 ~ ., data = training, kmax = 100, kernel = kernel,     scale = TRUE)
##
## Type of response variable: continuous
## minimal mean absolute error: 0.1969407
## Minimal mean squared error: 0.1090043
## Best kernel: inv
## Best k: 21
##
## [[2]]
## [1] 0.8625954
```

As such, the 'best' model uses 40 neighbors, with an accuracy score of approximately 80.9%.

Question 4.1

One situation for a clustering model is the task of grouping stocks by their different characteristics. If I am an investor, I would cluster stocks based on specific factors that might ultimately affect their stock returns. Below are 5 predictors I would use:

1. **Size**: Large companies (let's say, with respect to market capitalization) tend to have different characteristics from smaller ones. They can be more stable and predictable, and they can even bully smaller companies out of business.
2. **Geography**: The location of a company can affect its tax rate, its target demographic, and its cost base, among many other factors. This can be further refined by indicating the geography of a company's largest revenue source, rather than the country in which it was incorporated.
3. **Industry**: Companies in the same industry tend to show similar characteristics in profitability, growth, and stock valuation.
4. **Growth**: The rate of a company's sales and/or profit growth has a meaningful impact on that company's stock valuation. Some investors only want exposure to high-growth companies (with the associated risk that comes with growth), while more conservative investors look for companies with stable performance.
5. **Debt levels**: High levels of debt (or 'leverage') may put a company at risk if business performance falters one year. As such, some investors choose to avoid stocks of companies with a lot of debt. On the other hand, taking on debt can provide the cash a company needs to invest in an important business initiative. Either way, a company's leverage can help put context into that company's stock performance.

Question 4.2

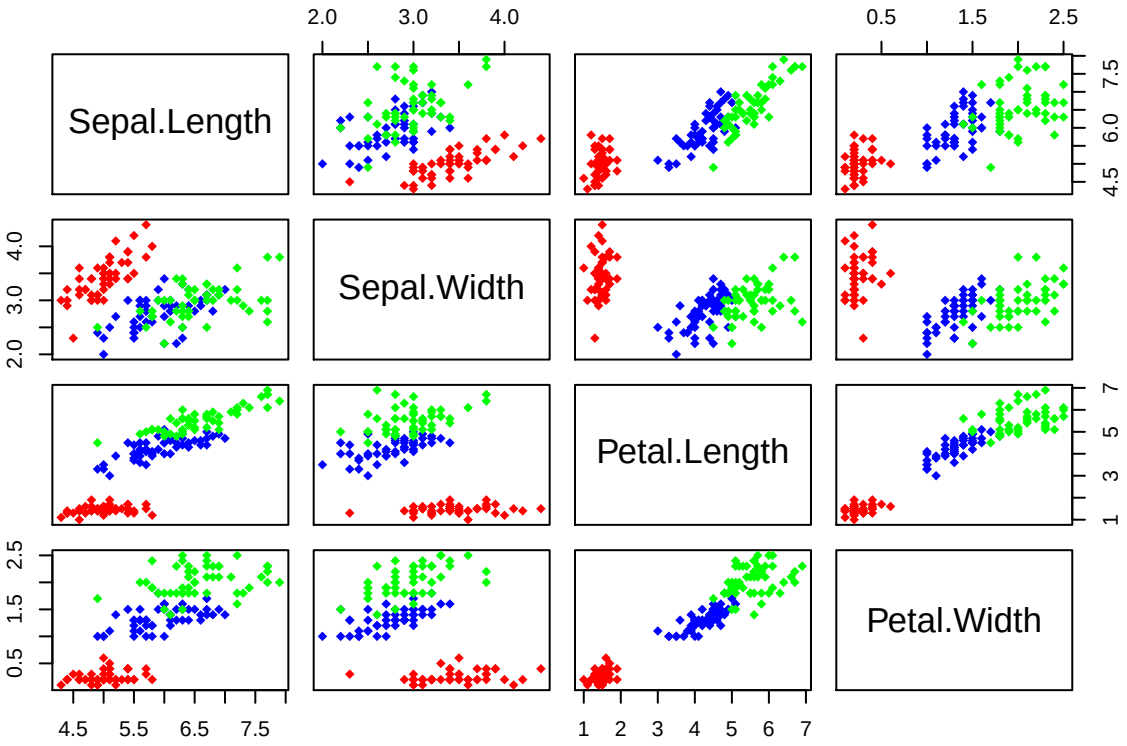
Upon opening the data, we see that there are 3 groups of species: Setosa, Versicolor, and Virginica. In addition, for a clustering problem, we would intuitively start with a simple plot to see if there are any obvious patterns. Since there are 4 dimensions in the data, we need to be a little creative and try a scatterplot matrix.

```
set.seed(1)
require("datasets")
head(iris)
```

```
##   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
```

```
predictors = iris[,c(1,2,3,4)]
response = iris[,5]

leg_colors = c("red", "blue", "green")
pairs(iris[,1:4], pch=18,
      col = leg_colors[iris$Species])
```



We start by standardizing values with the min-max method, which is typically preferred for clustering over mean-SD scaling.

```
stand <- function(x){return((x-min(x))/(max(x)-min(x)))}
predictors$Sepal.Length <- stand(predictors$Sepal.Length)
predictors$Sepal.Width <- stand(predictors$Sepal.Width)
predictors$Petal.Length <- stand(predictors$Petal.Length)
```

```
predictors$Petal.Width <- stand(predictors$Petal.Width)
```

Now we run the k-means model. The documentation suggests the Hartigan-Wong (default) typically performs best, so we do not change it. The algorithm generates 3 clusters of size 61, 39, and 50.

```
model <- kmeans(predictors, centers = 3, trace = FALSE)
model
```

```
## K-means clustering with 3 clusters of sizes 61, 39, 50
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1    0.4412568   0.3073770   0.57571548  0.54918033
## 2    0.7072650   0.4508547   0.79704476  0.82478632
## 3    0.1961111   0.5950000   0.07830508  0.06083333
##
## 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 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 2 1 1 1 1 1 1 1 1 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 2 2 2 2 2 1 2 1 2 1 2 2 1 1 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] 3.079830 2.073324 1.829062
##  (between_SS / total_SS =  83.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(Predicted = model$cluster, Actual = response)
```

```
##          Actual
## Predicted setosa versicolor virginica
##         1      0         47        14
##         2      0          3        36
##         3     50          0         0
```

We now calculate Count R2 as the number of correct classifications/total cases.
Cluster 1 is Versicolor and misclassifies 14 cases.
Cluster 2 is Virginica and misclassifies 3 cases.
Cluster 3 is Setosa and does not misclassify any case.
Count R2 = 150 - (3+14) / 150 is approximately 0.89.
Thus, about 89% of the cases were correctly classified.