R Notebook - Bryan Honeck

Code

The Iris data set is known by just about anybody who has experimented with R or Python. It is a very popular data set because it can be used to demonstrate how the k-means clustering algorithm works. In this project, I implemented both the k-means and the k-nearest neighbors algorithms.

Hide

```
head(iris)
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
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

6 rows

Check to see if there are any null values in the set. If null values are present, we can write a function to impute values based on species if possible. If there are no null values, we are good to move on to the next step.

Hide

```
any(is.na(iris))
```

```
[1] FALSE
```

Now we need to check the variance of a couple variables to see if they are scaled. They are almost always never standardized. We need to save the species column to its own variable. Once that is done, we have to scale the data in order to avoid a faulty model.

Hide

```
species <- iris[, 5]

var(iris[, 1])
```

```
[1] 0.6856935
```

Hide

```
var(iris[, 2])
```

```
[1] 0.1899794
```

Hide

```
standardized.iris <- scale(iris[, 1:4])

var(standardized.iris[, 1])
```

```
[1] 1
```

Hide

```
var(standardized.iris[, 2])
```

```
[1] 1
```

Importing caTools allows us to split the data with the sample.split() function. We are basically assigning another variable here that will give a particular point in the data a boolean value.

Hide

```
library(caTools)

final.standardized.iris <- cbind(standardized.iris, species)

sample <- sample.split(final.standardized.iris[, 5], SplitRatio = 0.7)

train <- subset(final.standardized.iris, sample == TRUE)
test <- subset(final.standardized.iris, sample == FALSE)
```

Call in the class package to use the knn() function. After running the model, we can check the error rate for k.

Hide

```
library(class)

predictions.species <- knn(train[, 1:4], test[, 1:4], train[, 5], k =1)

#predictions.species

mean(test[, 5] != predictions.species)
```

```
[1] 0.06666667
```

Now, we run the model with multiple values for k.

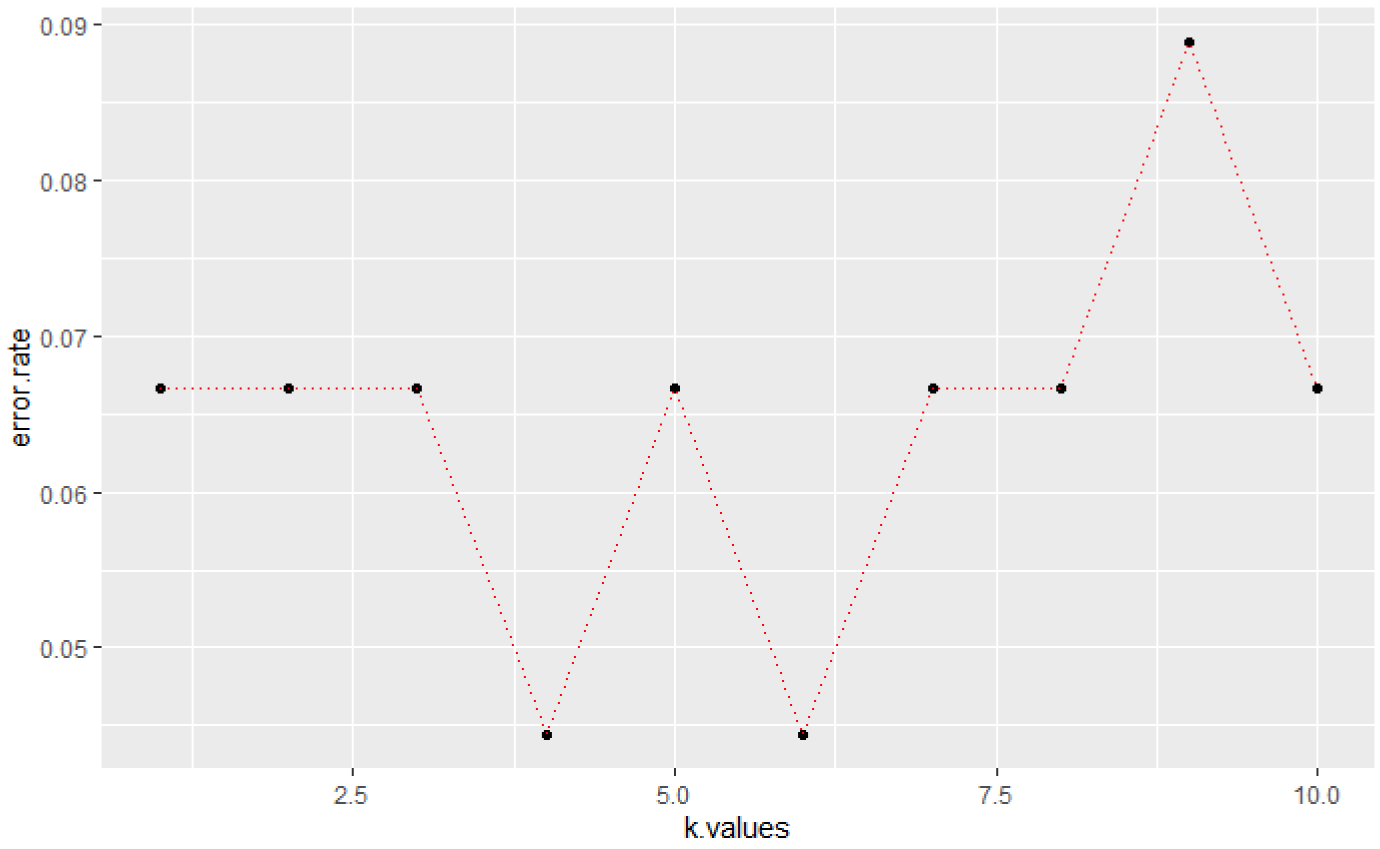
Hide

```
predictions.species <- NULL
error.rate <- NULL


for (i in 1:10) {
  predictions.species <- knn(train[, 1:4], test[, 1:4], train[, 5], k =i)
  error.rate[i] <- mean(test[, 5] != predictions.species)
}
```

Plot the error rates.

Hide

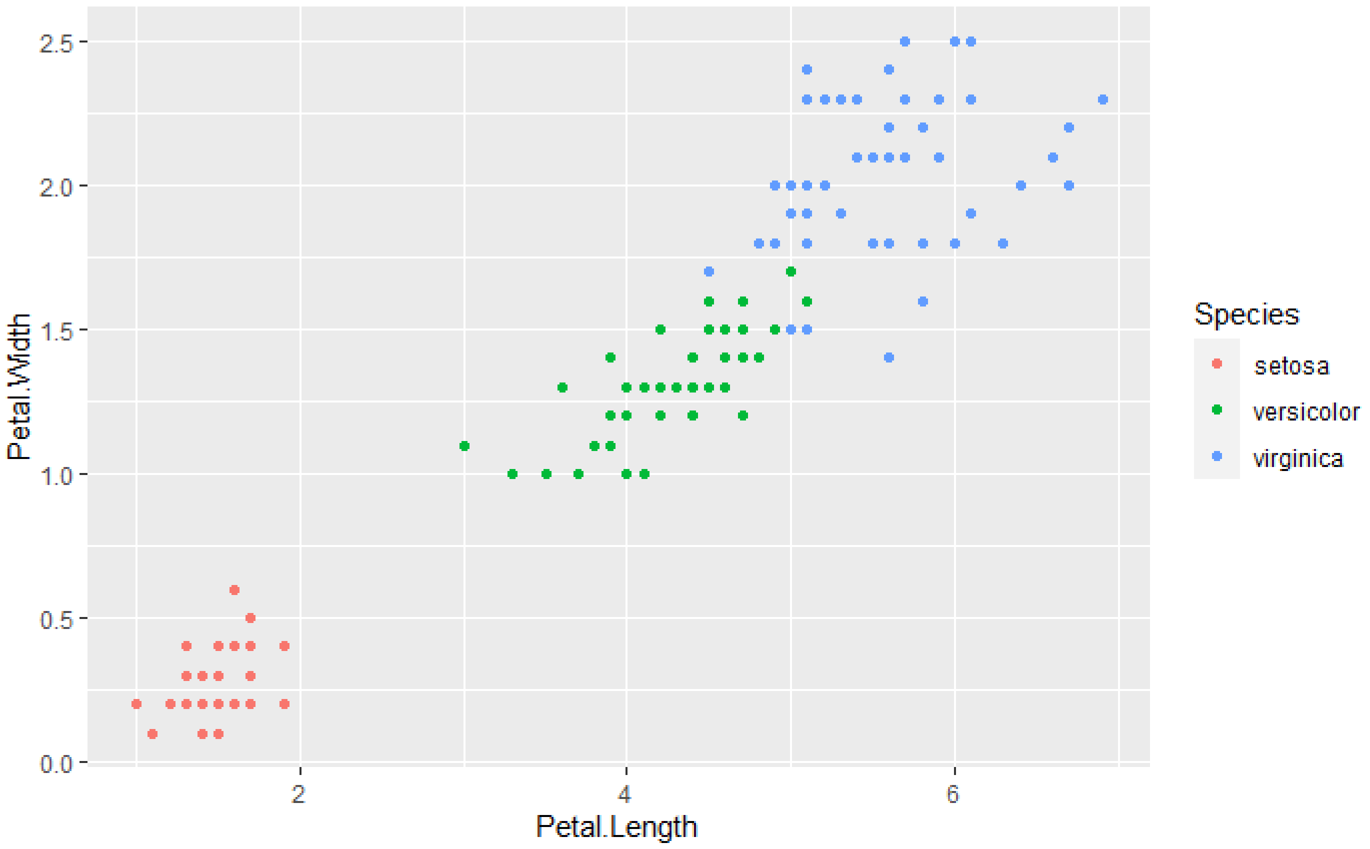
```
library(ggplot2)
k.values <- 1:10

error.df <- data.frame(error.rate, k.values)

pl <- ggplot(error.df, aes(k.values, error.rate)) + geom_point() + geom_line(lty="dotted", color
= "red")
pl
```



Now, we will use the k-means clustering algorithm to model the data. As we will see, this model performs significantly better than the knn model.

Hide

```
pl2 <- ggplot(iris, aes(Petal.Length, Petal.Width)) + geom_point(aes(color = Species))
pl2
```



Simple to use the kmeans() functions to create the model.

Hide

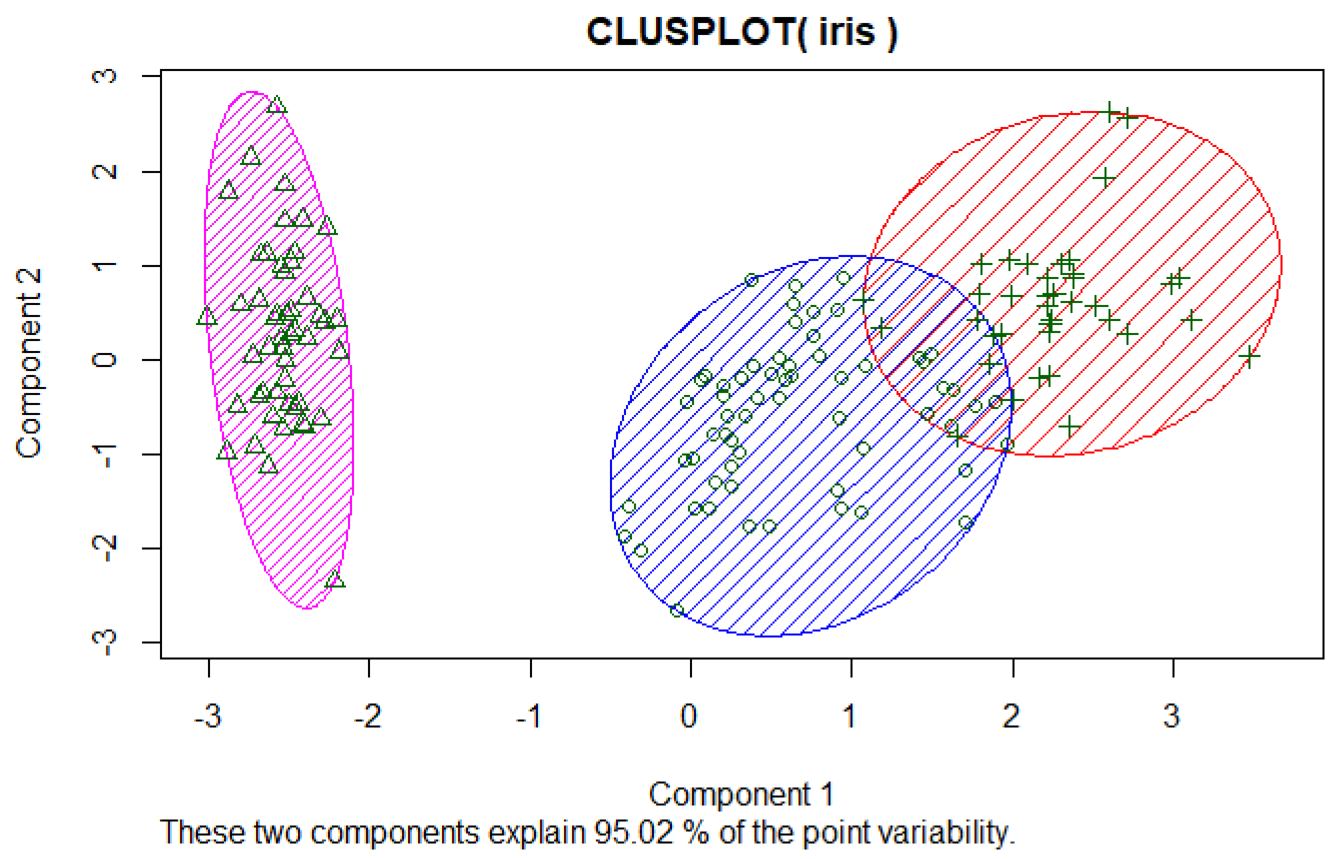
```
cluster.iris <- kmeans(iris[, 1:4], 3, nstart = 20)
cluster.iris
```

```
K-means clustering with 3 clusters of sizes 62, 50, 38

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.901613    2.748387     4.393548    1.433871
2     5.006000    3.428000     1.462000    0.246000
3     6.850000    3.073684     5.742105    2.071053

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 1 1 3 1
 [55] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 3 1 3 3 3 3 1 3
[109] 3 3 3 3 3 1 1 3 3 3 3 1 3 1 3 1 3 3 1 1 3 3 3 3 3 1 3 3 3 3 1 3 3 3 1 3 3 3 1 3 3 1

Within cluster sum of squares by cluster:
[1] 39.82097 15.15100 23.87947
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "s
ize"
[8] "iter"         "ifault"
```

Plot out the clusters.

Hide

```
library(cluster)

clusplot(iris, cluster.iris$cluster, color = TRUE, shade = TRUE, labels = 0, lines = 0)
```



Using factoextra, we can see how accurate our model was. The average silhouette width is better when it's higher. In this case, clusters 1 and 2 performed well, but cluster 3 performed exceptionally well.

Hide

```
#install.packages("factoextra")
library(factoextra)

sil <- silhouette(cluster.iris$cluster, dist(iris))
```

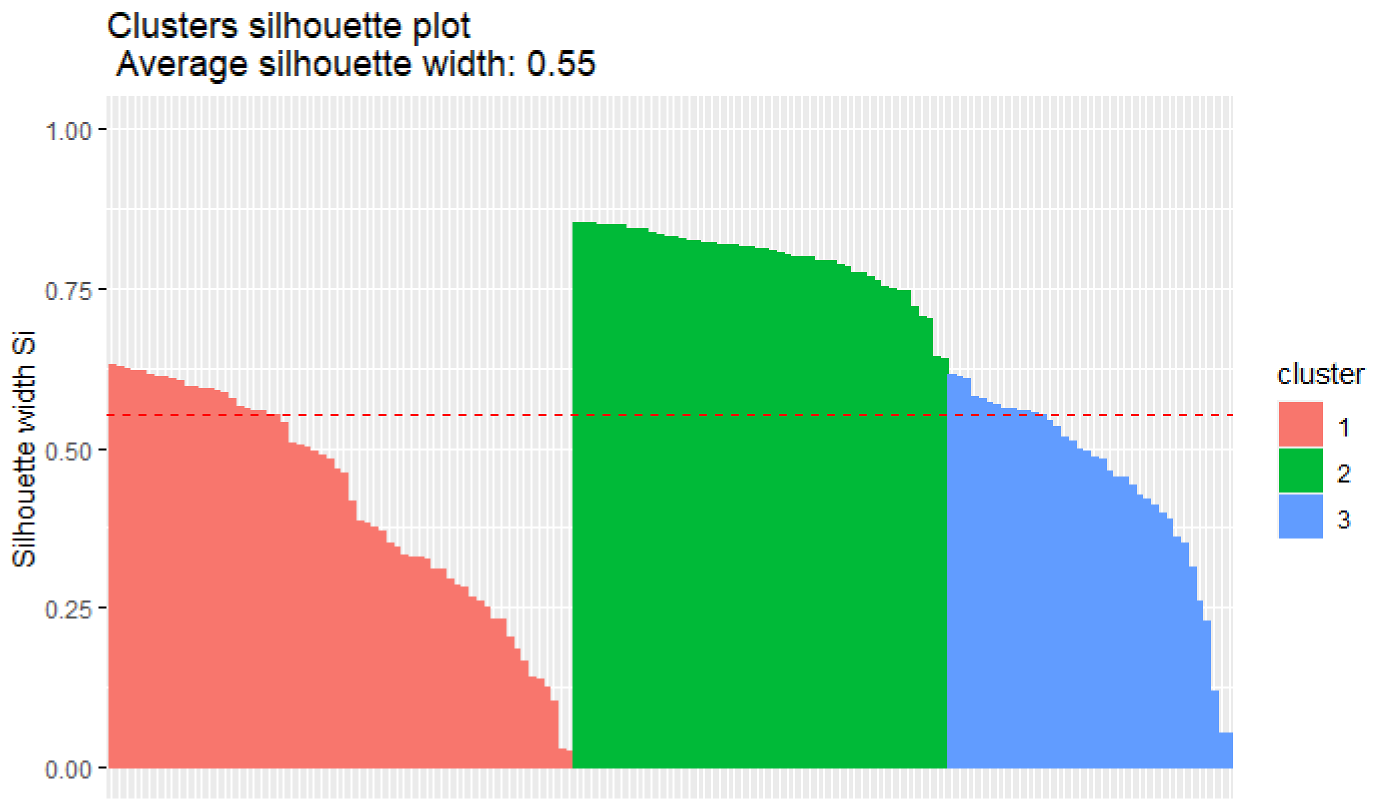
```
NAs introduced by coercion
```

Hide

```
fviz_silhouette(sil)
```

	cluster <fctr>	size <int>	ave.sil.width <dbl>
1	1	62	0.42
2	2	50	0.80
3	3	38	0.45

3 rows



The iris data set is very interesting. It's clusters are defined well. They are noticeable when they are plotted out. In this case, the k-means clustering algorithm was a great fit for the data because we knew there clusters present after viewing the data with ggplot. We knew exactly how many clusters we were looking for, too. KNN did not do a poor job of capturing these groups, but it still lacked the same accuracy that the k-means clustering model produced.