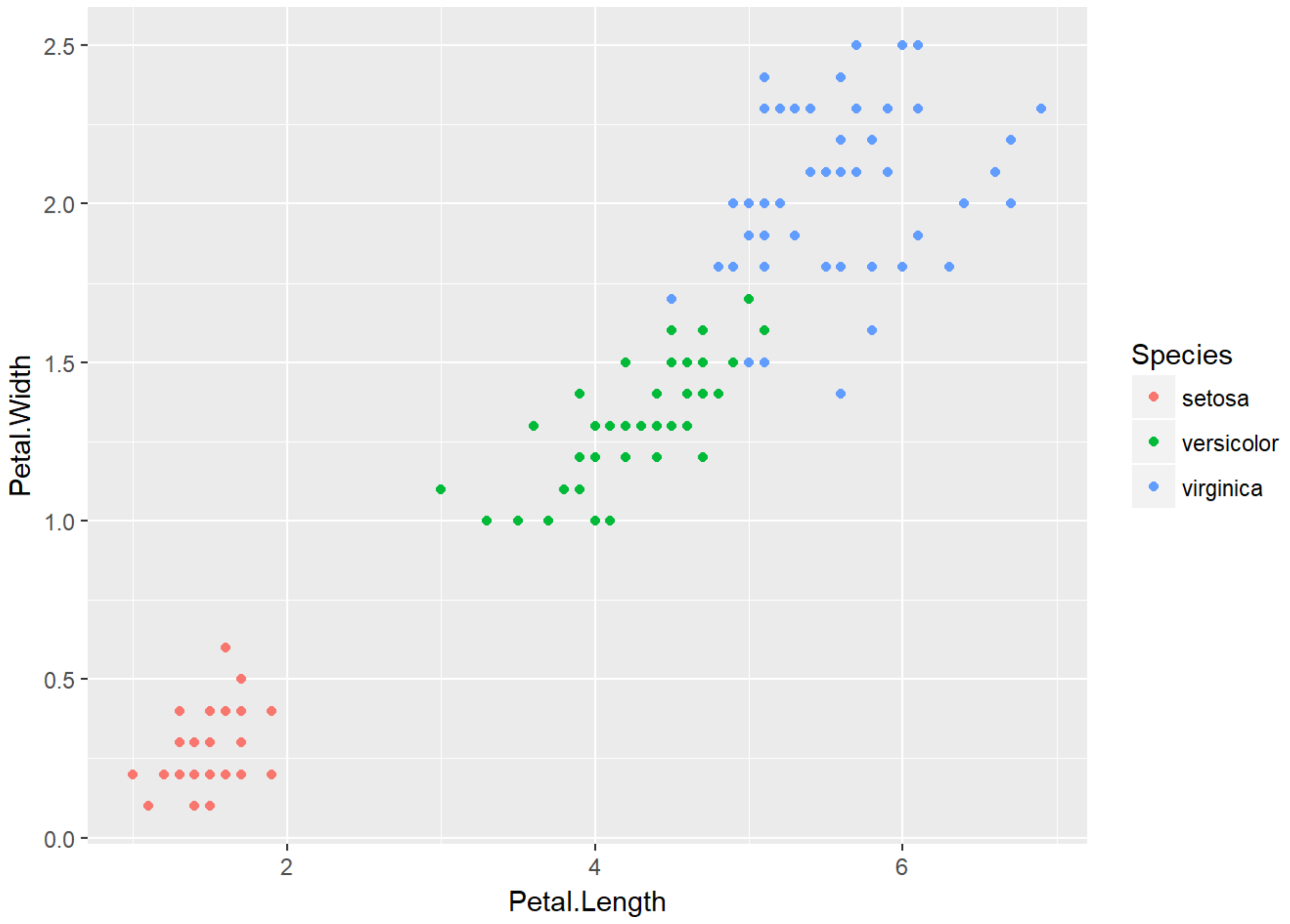
Classification and Clustering

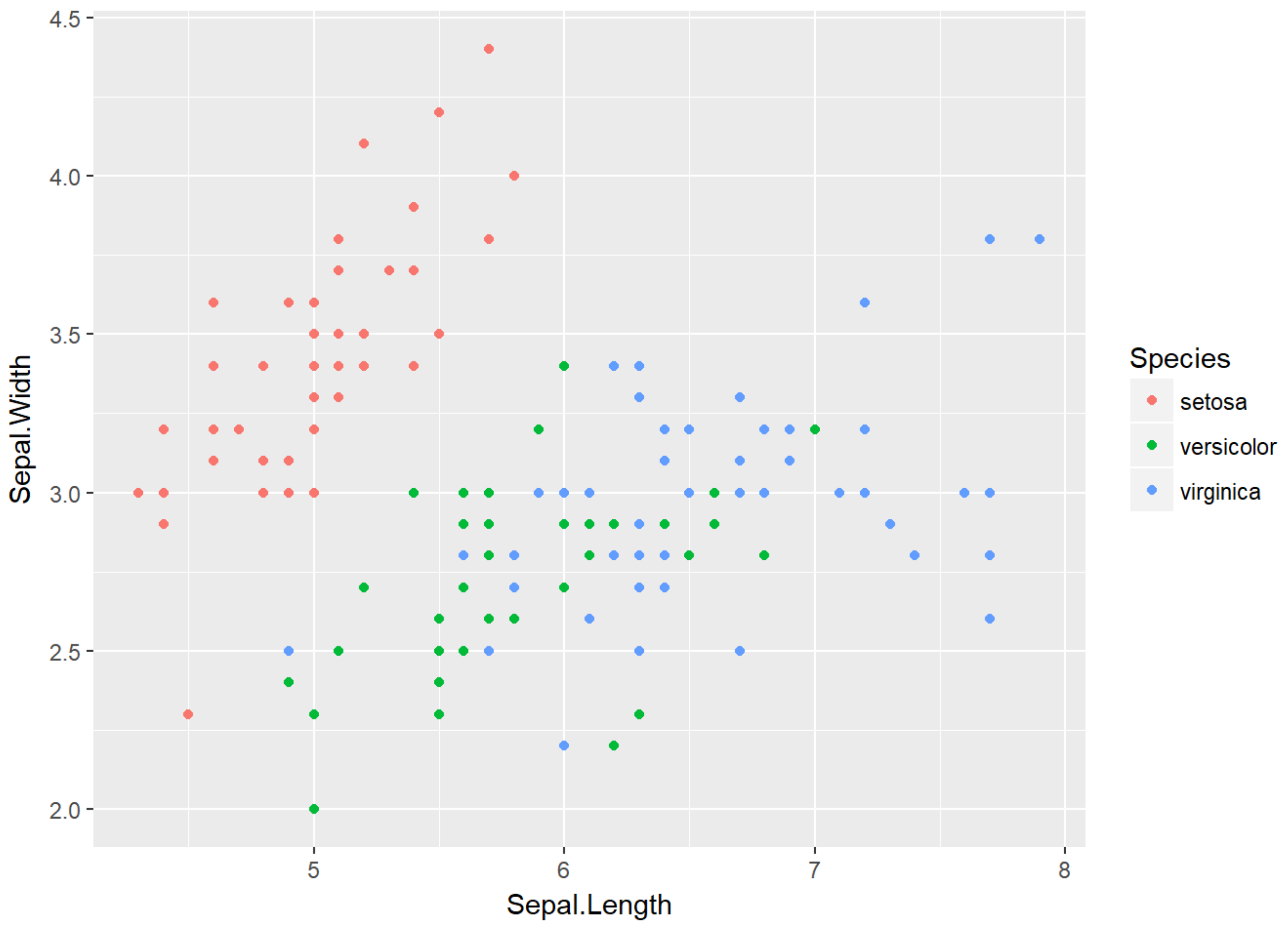
Analysis on Iris Dataset

Visualizing

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



```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point()
```



Splitting Data into training and test sets
We split the data into 70 percent for training data and 30 percent for testing.

```
set.seed(101)

sample <- sample.split(iris, SplitRatio = 0.7)

iris.train <- subset(iris, sample == T)
iris.test <- subset(iris, sample == F)
```

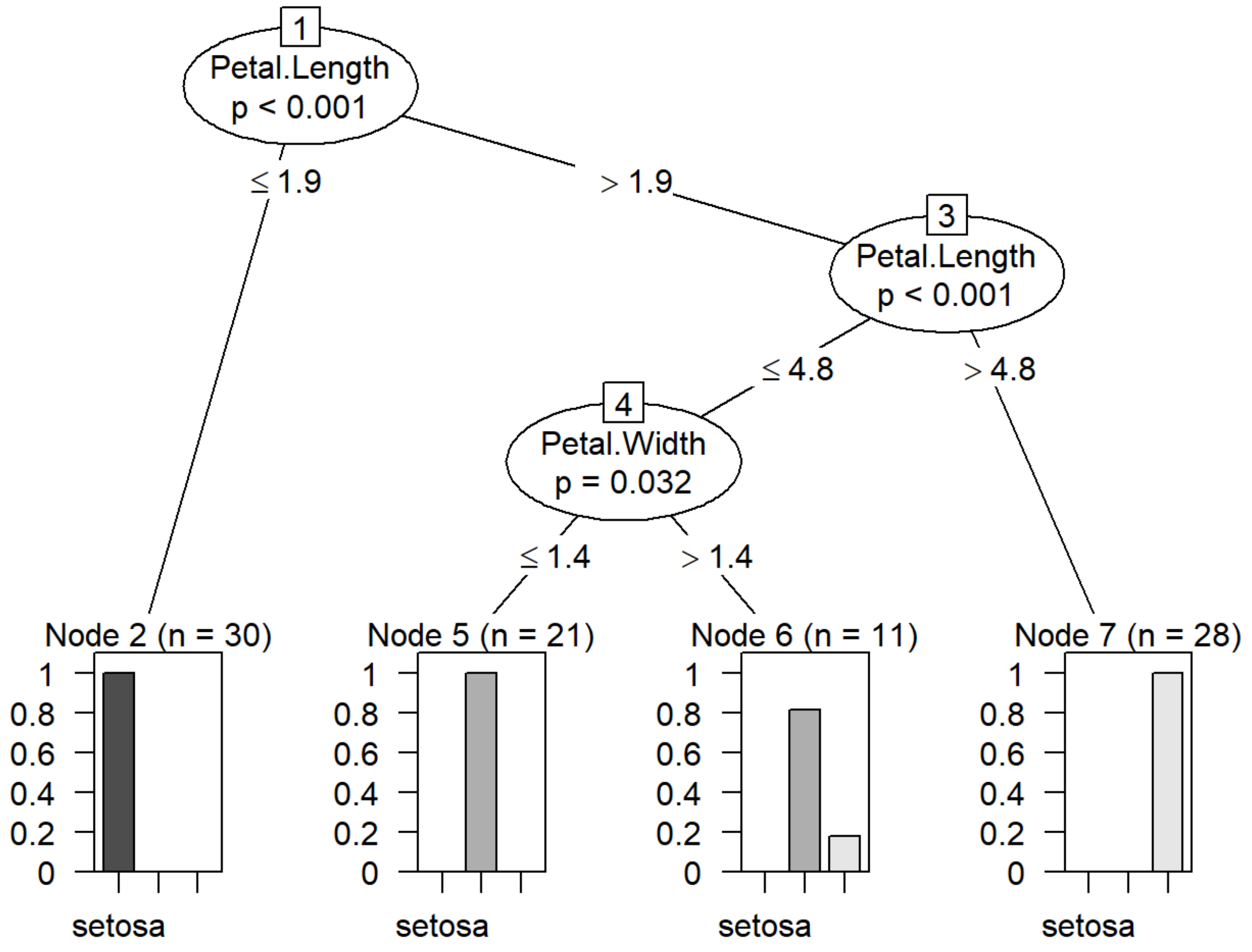
Classification

Using c-tree

Conditional inference trees are used in classification. We can see that the model has predicted setosa correctly. There is one misclassification in versicolor and 4 in virginica.

```
iris.ctree <- ctree(Species~., data = iris.train)
#plotting

plot(iris.ctree)
```



```
#prediction

p.ctree <- predict(iris.ctree, iris.test)
table(p.ctree,iris.test$Species)
```

```
##
## p.ctree      setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         16         1
##   virginica       0          4        19
```

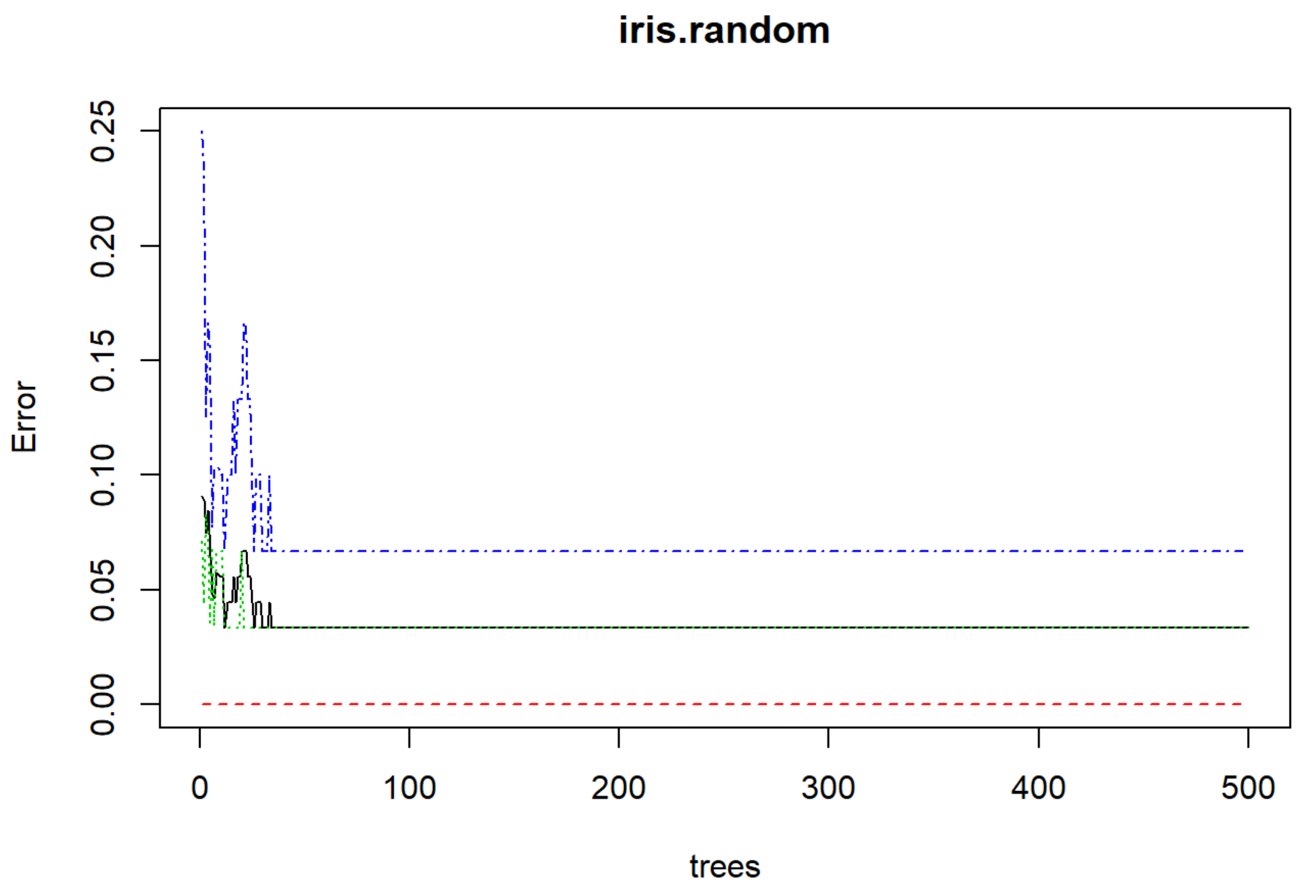
Using random Forest

We can see that random forest gave a better performance than c tree. It predicted setosa and versicolor correctly but gave an error of 3 in virginica.

```
iris.random <- randomForest(Species ~ ., data = iris.train)
iris.random$importance
```

```
##              MeanDecreaseGini
## Sepal.Length         6.191868
## Sepal.Width          1.392539
## Petal.Length        28.526133
## Petal.Width         23.122882
```

```
plot(iris.random)
```



```
p <- predict(iris.random, iris.test)
table(p, iris.test$Species)
```

```
##
## p            setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         17         0
##   virginica       0          3        20
```

```
##Using Support vector machine

model <- svm(Species~. , data = iris)
summary(model)
```

```
##
## Call:
## svm(formula = Species ~ ., data = iris)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.25
##
## Number of Support Vectors:  51
##
##  ( 8 22 21 )
##
##
## Number of Classes:  3
##
## Levels:
##  setosa versicolor virginica
```

```
predicted.values <- predict(model, iris[,1:4])

table(predicted.values, iris$Species)
```

```
##
## predicted.values setosa versicolor virginica
##       setosa         50          0         0
##       versicolor      0         48         2
##       virginica       0          2        48
```

Using neural network

Such systems “learn” (i.e. progressively improve performance on) tasks by considering examples, generally without task-specific programming.
I have normalized the data first then applied neuralnet function which can be plotted to give weights to the values of the variables.

```
labels <- class.ind(as.factor(iris$Species))

standardiser <- function(x){
  (x-min(x))/(max(x)-min(x))
}

iris[, 1:4] <- lapply(iris[, 1:4], standardiser)

pre_process_iris <- cbind(iris[,1:4], labels)

f <- as.formula("setosa + versicolor + virginica ~ Sepal.Length + Sepal.Width + Pet
al.Length + Petal.Width")

iris_net <- neuralnet(f, data = pre_process_iris, hidden = c(16, 12), act.fct = "ta
nh", linear.output = FALSE)

plot(iris_net)
```

Decision tree using r part

We can also use rpart to perform classification.

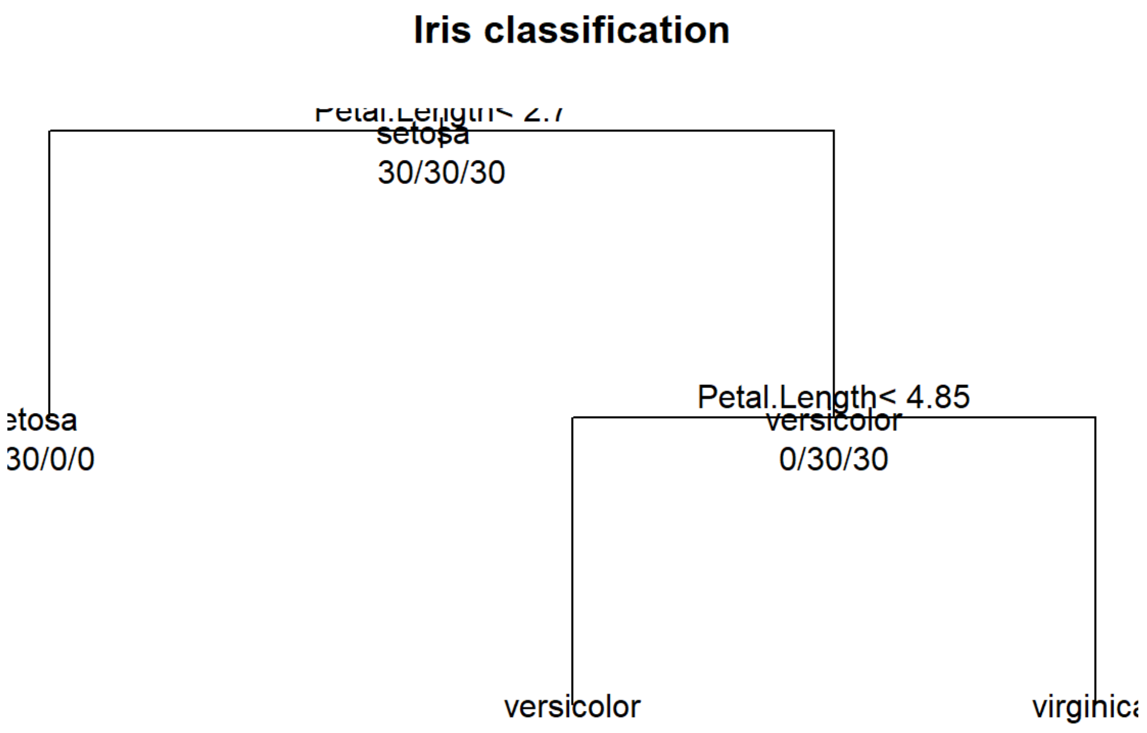
```
iris.tree <- rpart(Species~. , method = 'class',data = iris.train)

printcp(iris.tree)
```

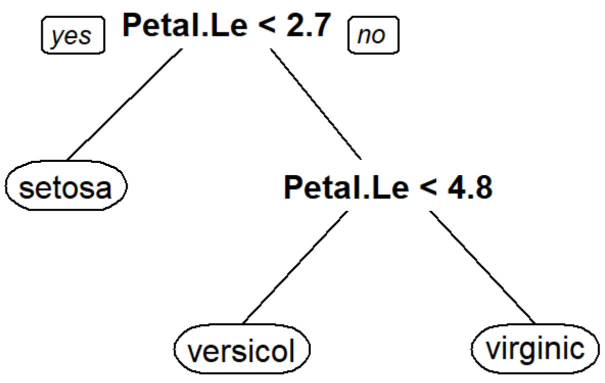
```
##
## Classification tree:
## rpart(formula = Species ~ ., data = iris.train, method = "class")
##
## Variables actually used in tree construction:
## [1] Petal.Length
##
## Root node error: 60/90 = 0.66666667
##
## n= 90
##
##           CP nsplit   rel error      xerror        xstd
## 1 0.50000000      0 1.000000000 1.216666667 0.061888989
## 2 0.46666667      1 0.500000000 0.933333333 0.076658615
## 3 0.01000000      2 0.033333333 0.033333333 0.023306863
```

```
#plotting

plot(iris.tree, uniform = T, main = 'Iris classification')
text(iris.tree, use.n = T, all = T)
```



```
prp(iris.tree)
```



```
#prediction
p.rpart <- predict(iris.tree, newdata = iris.test[,1:4])
```

Clustering

Using K means

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

As we know in advance that there are 3 species we have kept the value of clusters as 3.
We can see it has predicted setosa correctly but has few misclassifications in versicolor and virginica.
But still it is a good prediction as it is an unsupervised learning algorithm.

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

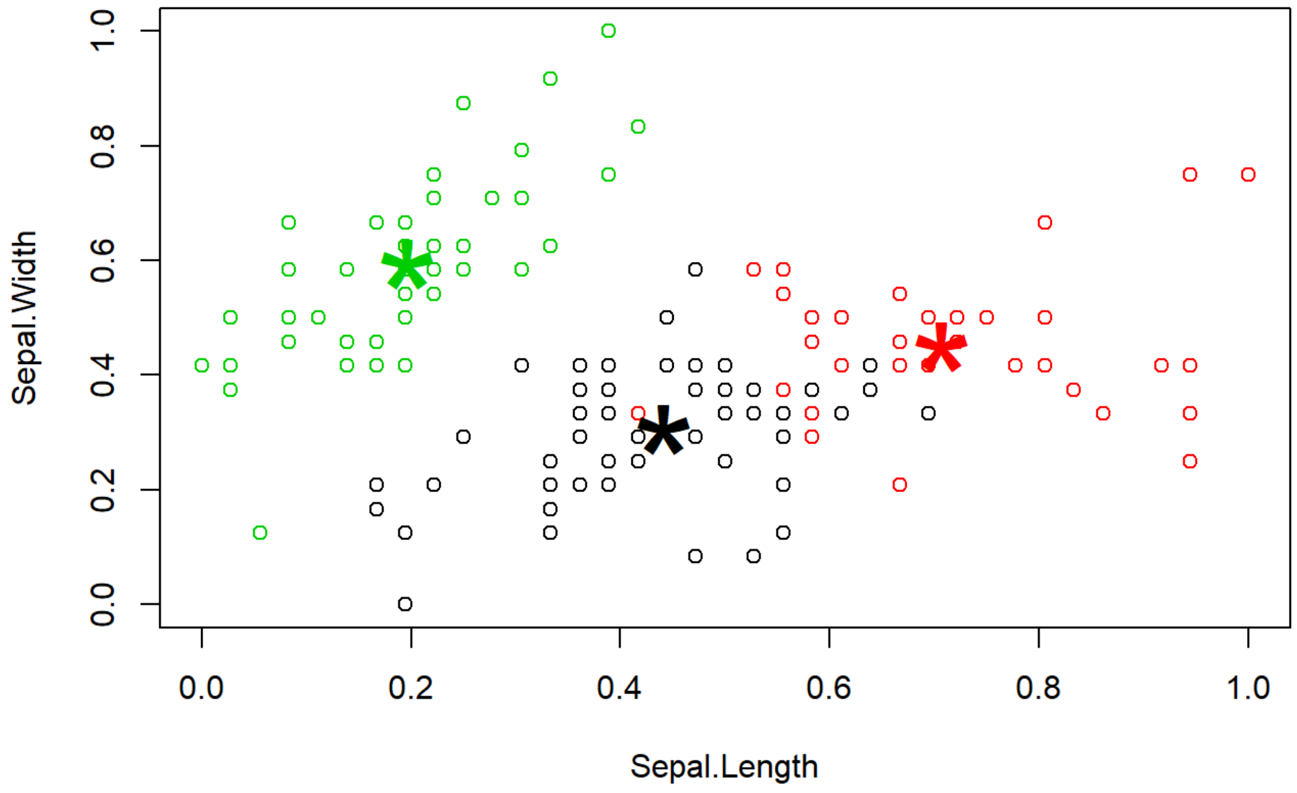
```
## K-means clustering with 3 clusters of sizes 61, 39, 50
##
## Cluster means:
##   Sepal.Length  Sepal.Width  Petal.Length   Petal.Width
## 1 0.4412568306 0.3073770492 0.57571547652 0.54918032787
## 2 0.7072649573 0.4508547009 0.79704476315 0.82478632479
## 3 0.1961111111 0.5950000000 0.07830508475 0.06083333333
##
## 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
##  [36] 3 3 3 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
##  [71] 1 1 1 1 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 1 1 2 1 2 2 2
## [106] 2 1 2 2 2 2 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
## [141] 2 2 1 2 2 2 1 2 2 1
##
## Within cluster sum of squares by cluster:
## [1] 3.079830278 2.073324081 1.829062114
##  (between_SS / total_SS =  83.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
table(irisCluster$cluster, iris$Species)
```

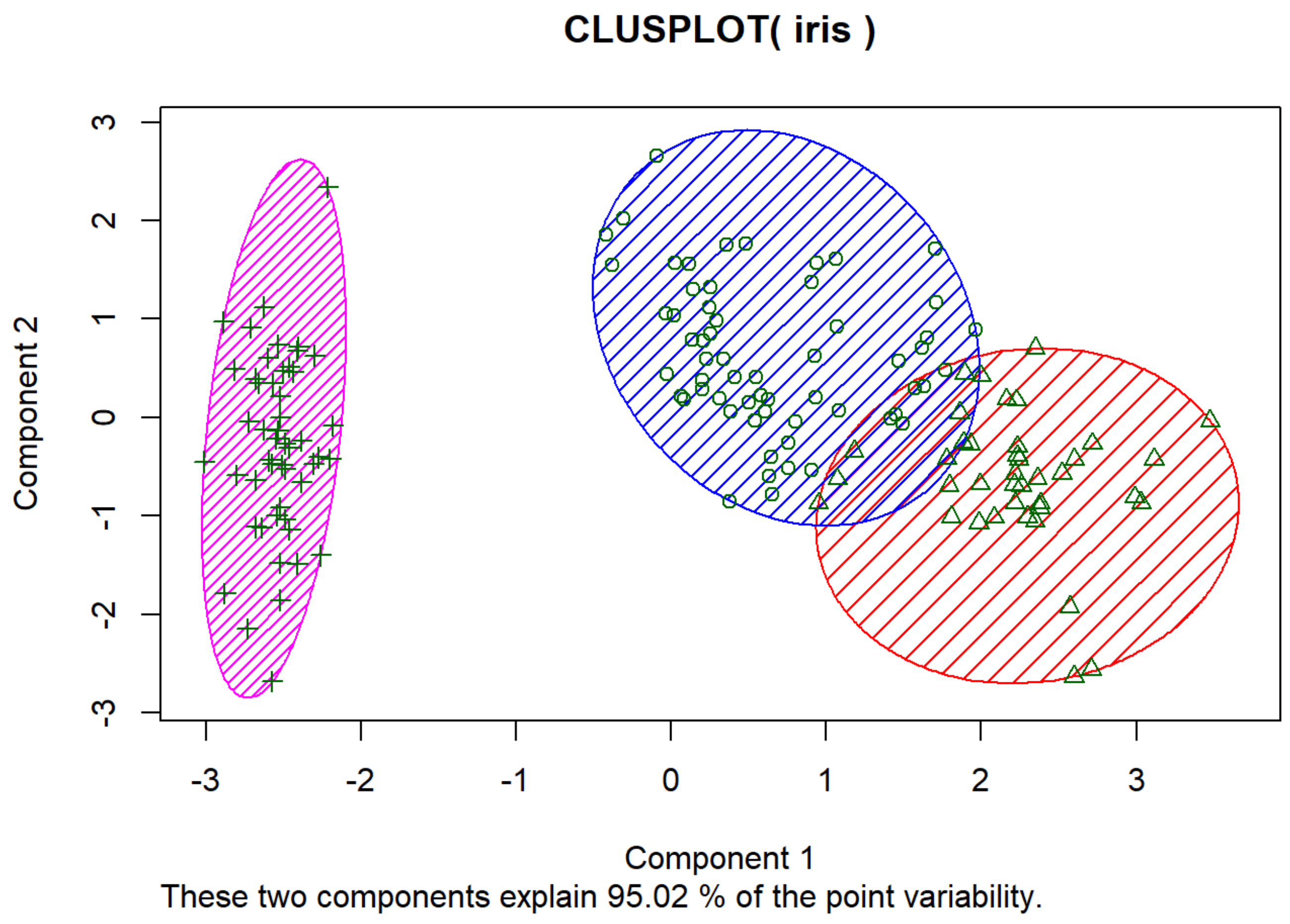
```
##
##     setosa versicolor virginica
##   1      0         47        14
##   2      0          3        36
##   3     50          0         0
```

```
#Plot

plot(iris[c("Sepal.Length", "Sepal.Width")], col = irisCluster$cluster)
points(irisCluster$centers[, c("Sepal.Length", "Sepal.Width")], col = 1:3, pch = "*
", cex = 5)
```



```
#Using clusplot
clusplot(iris, irisCluster$cluster, color = T, shade = T, labels = 0, lines = 0)
```



```
table(irisCluster$cluster,iris$Species)
```

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

Density based clustering

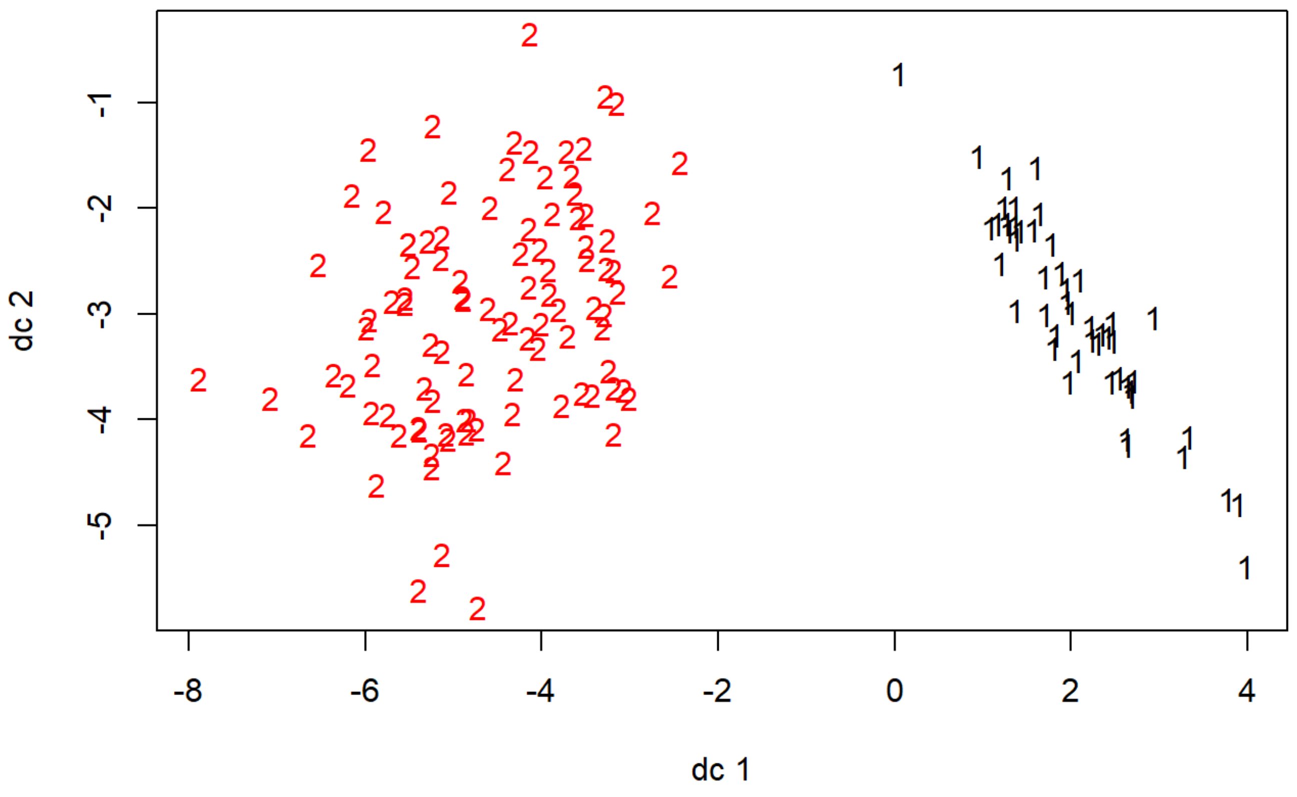
Unlike K-Means, DBSCAN does not require the number of clusters as a parameter. Rather it infers the number of clusters based on the data, and it can discover clusters of arbitrary shape (for comparison, K-Means usually discovers spherical clusters).
It has discovered two clusters on its own and mixes versicolor and virginica as a single cluster.

```
ds <- dbscan(iris[,1:4], eps = 0.42, MinPts = 5)

table(ds$cluster,iris$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         50        50
```

```
plotcluster(iris[,1:4], ds$cluster)
```


dc 1
dc 2