K-Nearest Neighbors Example

Intro

So on the heels of this week's lecture perhaps you might be interested in a teaser for something involving a more hands on approach.

This notebook will help you to start becoming familiar with some of the activities common to Machine Learning. It's helpful for you to run the code on your own.

We'll dig deeper into similar examples later on and walk through then in greater detail later. However, getting some idea about how some of the ML concepts and methods work in actual life is useful.

Classifying Flowers

Consider the **iris** data frame whic is part of any default R installation. Note that this data set is also available as part of the scikit learn module in Python:

```
from sklearn import datasets
iris = datasets.load_iris()
```

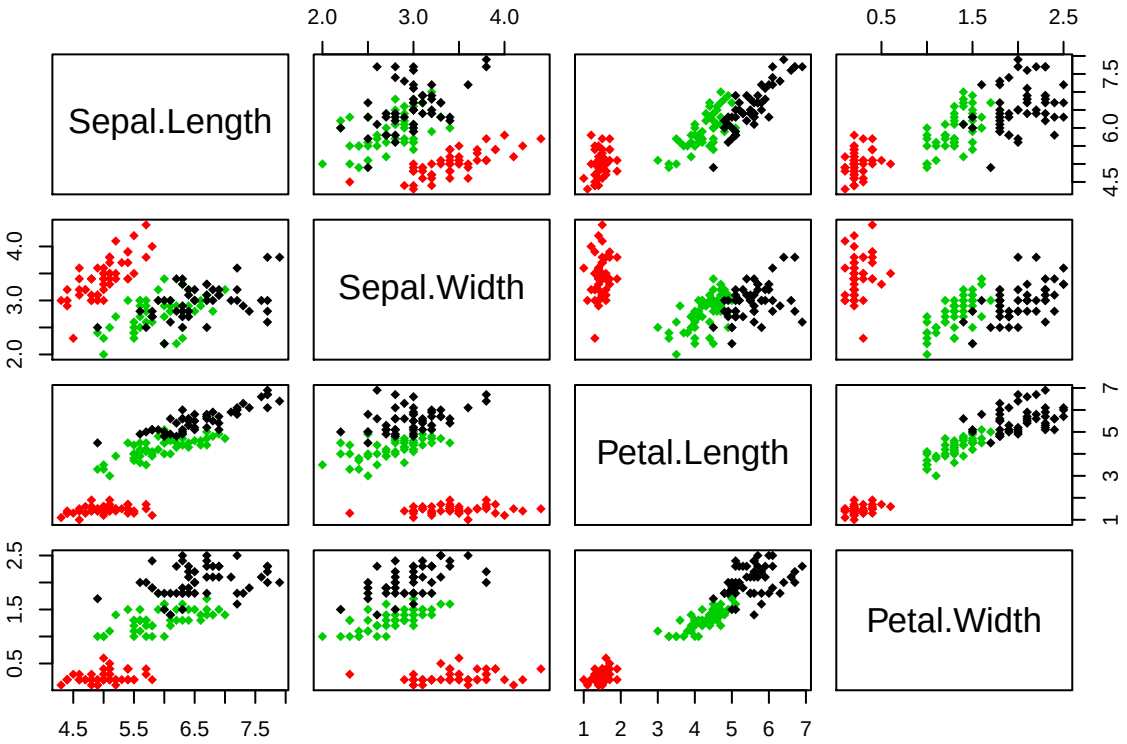
The help page for the dataset informs us that we have 150 flowers representing three species: setosa, virginica, and versicolor. There are 50 of each and there are 4 measurements for each flower.

```
data(iris)
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 ...
```

We might want to look at the relationships between the measured quantities to see if there is anything interesting going on.

```
some_colors <- c("red","green3","black")[unclass(iris$Species)]
pairs(iris[,-5],col=some_colors,pch=18)
```

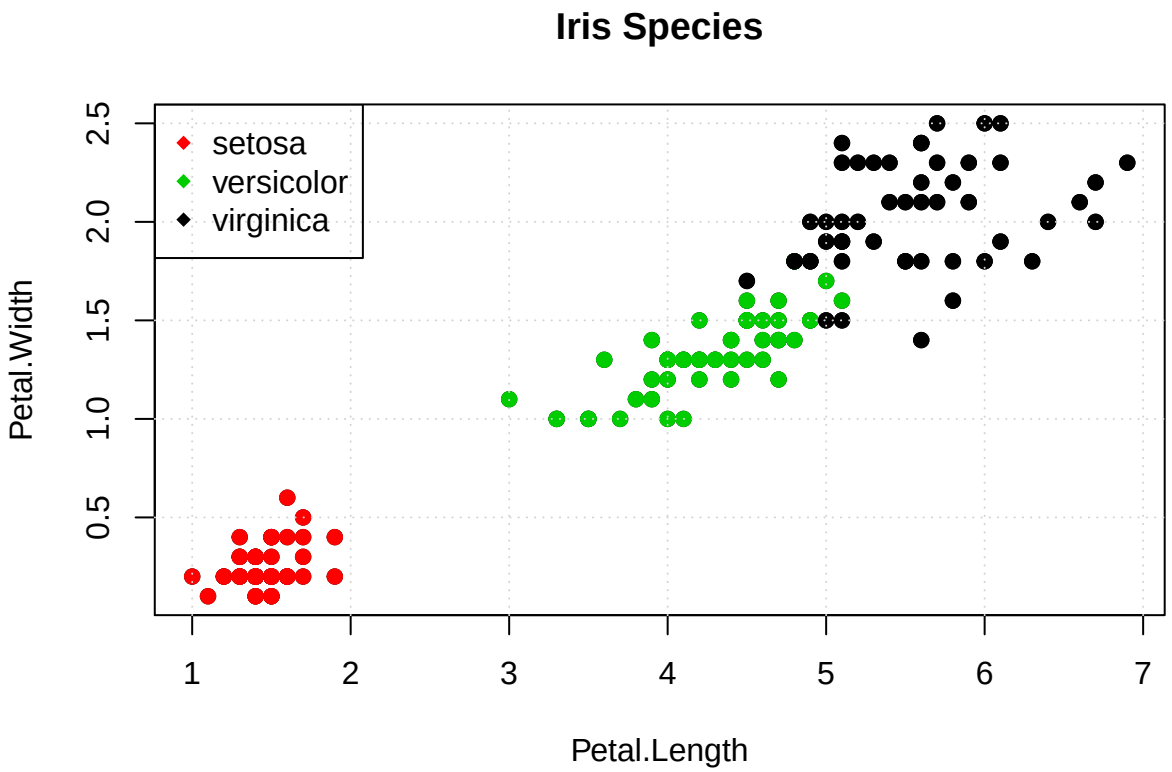


There are some linear relationships at work but we'll hold off that for now since we are here to learn about K Nearest Neighbors. Let's plot Sepal.Width as a function of Sepal.Length and shade the points according Species.

```
plot(Petal.Width ~ Petal.Length,
     data = iris,
     main = "Iris Species")

points(Petal.Width ~ Petal.Length,
       data=iris,
       pch = 19,
       col= some_colors)
grid()

legend("topleft",
       legend=levels(unclass(iris$Species)),
       col = c("red","green3","black"), pch=18)
```



This plot is encouraging in the sense that the Species all seem to cluster together. Another way to say this is that the distance between them (as computed by some formula) is close within a given Species - at least as it relates to Petal Width and Petal Length.

So What ?

Well, given the above information, you might want to know which variables (also known as "features") are the most influential in determing what Species a flower belongs to. It might not be just one variable. It could be several. Or it could some interaction between them. For now, let's just say that maybe our goal is the following:

- Given a new flower that is measured in the same way as the existing ones, we want to Classify the Species to which it belongs

K Nearest Neighbors

We could build a model using the K Nearest Neighbors function to apply to some "holdout" or "test" data. In reality we don't have any new flowers to measure or evaluate so we'll split the existing data into "training" and "test" datasets. We'll then use the "train" data to build a model and then apply it to the "test" data to see how well our model performs. We'll sample the iris data frame to pull out approximately 80% of the rows and save the remaining 20% into a test / holdout data set.

```
num_of_rows <- nrow(iris)

# Set the seed so this example can be reproduced
set.seed(123)

# Shuffle the data set
shuffled_iris <- iris[sample(num_of_rows),]
```

```
# Get some indices for making the training / test sets
idx <- sample(num_of_rows,round(.8*num_of_rows))

# This where we create test and train
# We save the actual Species names from the training

train <- shuffled_iris[idx,]
refs <-  train[,5]
test  <- shuffled_iris[-idx,]

# The number of rows in this data frame should be around
# 80 percent of the original 150 rows.

nrow(train)/nrow(iris)
```

```
## [1] 0.8
```

So now we'll use the K Nearest Neighbors function in R to build a classifier that, given a future observation that we have yet to see, will classify that observation as a one of the three Species types.

The way this works is that we give the function the following:

1) **Training Data** - This is all numeric in content. Note that we removed the species column since it is a factor. The number of rows in the data represent about 80% of the original data frame
2) **Test Data** - This is all numeric in content. Note that we removed the species column since it is a factor. The number of rows in the data represent about 20% of the original data frame
3) **refs** - This represents the actual / known Species labels corresponding to the training data labels. This will help us determine how close our model came to preficting the right Species.

Let's call the **knn** function to build a model for Species classification. What we expect back from the function are the names of the **predicted** Species corresponding to the rows of data in the **test / holdout** data set.

```
library(class)
(myKnn <- knn(train[,-5], test[,-5], refs))
```

```
##  [1] virginica  versicolor setosa     versicolor versicolor virginica
##  [7] setosa     versicolor setosa     setosa     setosa     setosa
## [13] virginica  virginica  setosa     versicolor virginica  virginica
## [19] setosa     setosa     versicolor virginica  versicolor setosa
## [25] virginica  virginica  versicolor setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

How close did we come ? Actually, pretty good. We can create a table that compares how frequently our classification model matched the actual label (stored in test$Species). If we have a "perfect" classififer then there will be no values in the off diagonal cells.

```
table(myKnn,test$Species)
```

```
##
## myKnn        setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0          9         0
##   virginica       0          1         9
```

Now, we got lucky in this case. Most predictive models aren't as good as this and there are a number of considerations to make here such as how many nearest neighbors did we compare observations to when

bulding the model. If you look at the help page for **knn* you will see that it defaults to 1 nearest neighbor. We could play with this number and see if this improves our model.

```
library(class)
(myKnn <- knn(train[,-5], test[,-5], refs, k = 3))
```

```
##  [1] virginica  versicolor setosa     versicolor versicolor virginica
##  [7] setosa     versicolor setosa     setosa     setosa     setosa
## [13] virginica  virginica  setosa     versicolor virginica  virginica
## [19] setosa     setosa     versicolor virginica  versicolor setosa
## [25] virginica  virginica  versicolor setosa     versicolor virginica
## Levels: setosa versicolor virginica
```

```
table(myKnn,test$Species)
```

```
##
## myKnn        setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0          9         0
##   virginica       0          1         9
```

There are better ways to evlauate this model such as using a "confusion matrix" from which we can compute a number of performance measures that allow us to judge the quality of the model.