Iris Data Set

Data Set Information:

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Predicted attribute: class of iris plant.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica

R Code:

Naïve Bayesian Classifier

- setwd('~/Desktop/dm') #set working directory to desktop
- library(e1071) #import library e1071
- library(klaR) #import library klaR
- iris<-read.table("iris.data",sep=",",header=TRUE) #read the dataset from the data file
- summary(iris)

```
 sepal_length    sepal_width     petal_length    petal_width               class
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
 Mean   :5.843   Mean   :3.054   Mean   :3.759   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
```

- classifier<-naiveBayes(iris[,1:4], iris[,5]) #Naive Bayesian function

In the above line we create a classifier using naive bayes using the first 4 columns as the data, and the last column as the class for each observation.

- predicted.classes <- predict(classifier, iris[,-5])

predict() method above receives the classifier object plus some observations (i.e. iris[,-5] is the original data without its class) and returns a suggested class for each observation.

- head(predicted.classes,n=12) #display 12 consecutive header attributes

```
 [1] Iris-setosa Iris-setosa Iris-setosa Iris-setosa Iris-setosa Iris-setosa Iris-setosa Iris-setosa Iris-setosa
[10] Iris-setosa Iris-setosa Iris-setosa
Levels: Iris-setosa Iris-versicolor Iris-virginica
```

- table(predicted.classes, iris[,5], dnn=list('predicted','actual'))

the method table() presents a confusion matrix between the suggested class vector with the real class vector. In this case the classification was very good.

```
                 actual
predicted         Iris-setosa Iris-versicolor Iris-virginica
  Iris-setosa              50               0              0
  Iris-versicolor           0              47              3
  Iris-virginica            0               3             47
```

- classifier$apriori / sum(classifier$apriori) #the prior distribution for the classes

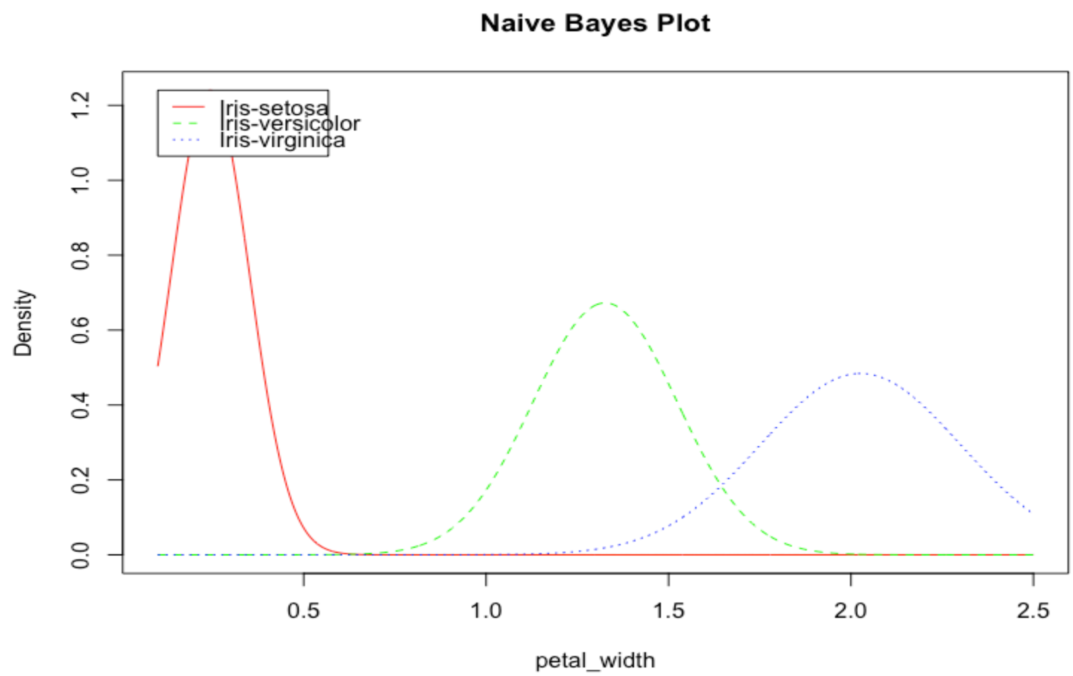
```
iris[, 5]
    Iris-setosa Iris-versicolor  Iris-virginica
      0.3333333       0.3333333       0.3333333
```

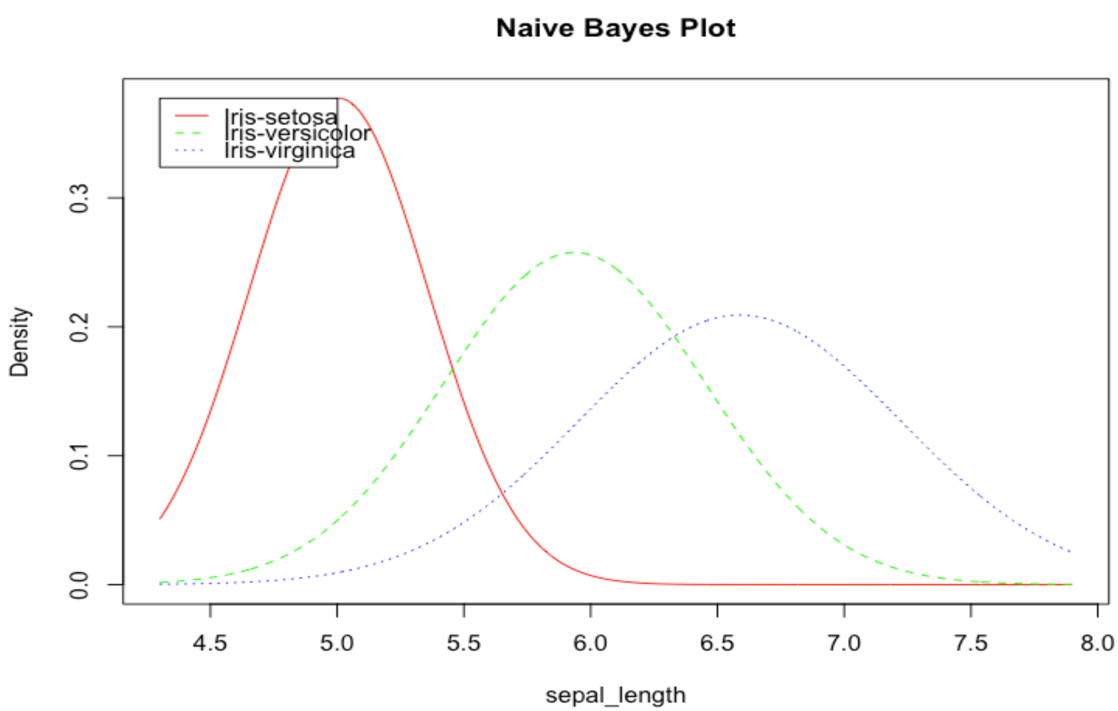
- classifier$tables$petal_length

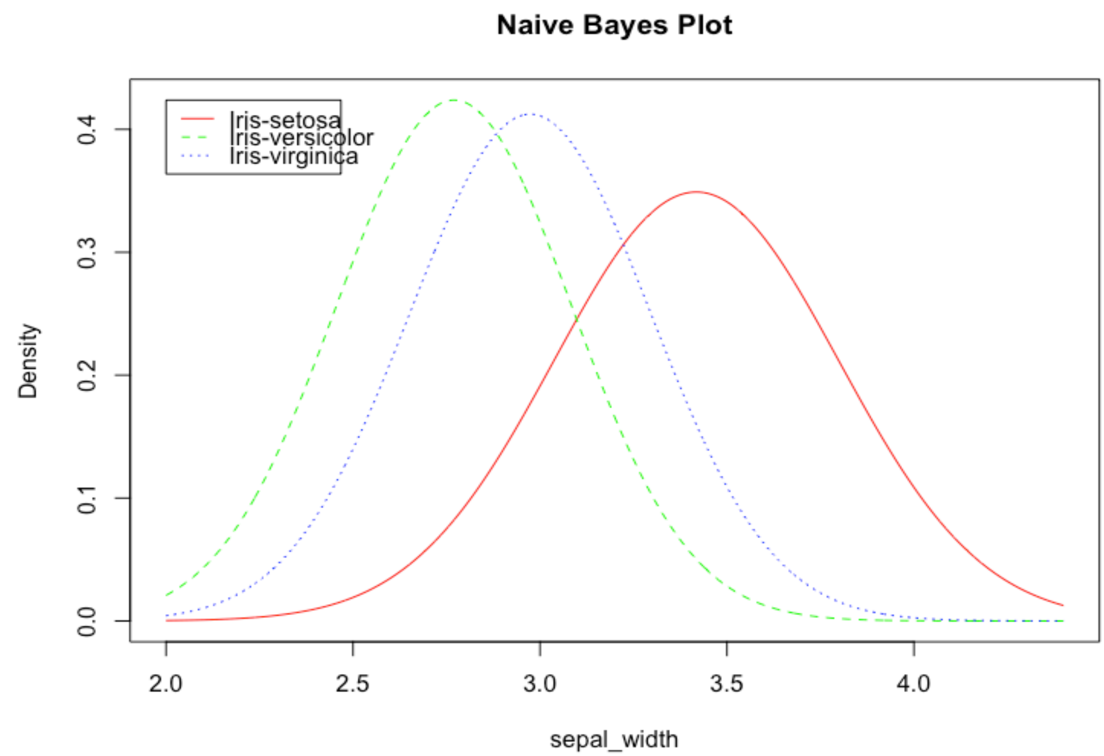
Since the predictor variables here are all continuous, the naive Bayes classifier generates three Gaussian (Normal) distributions for each predictor variable: one for each value of the class variable class. The first column is the mean, the 2nd column is the standard deviation.

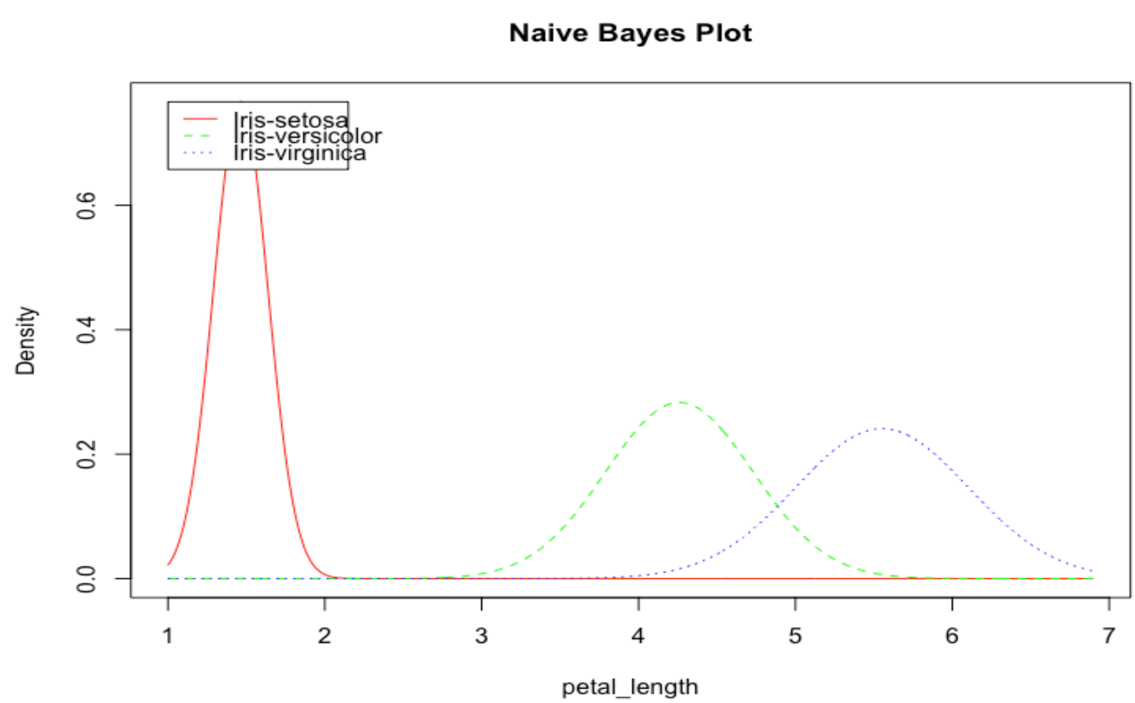
```
                 petal_length
iris[, 5]          [,1]      [,2]
  Iris-setosa     1.464 0.1735112
  Iris-versicolor 4.260 0.4699110
  Iris-virginica  5.552 0.5518947
```

- naive_iris <- NaiveBayes(iris$class ~ ., data=iris)
- plot(naive_iris) #plot


Naive Bayes Plot
Iris-setosa
Iris-versicolor
Iris-virginica
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.5
1.0
1.5
2.0
2.5
petal_width


Naive Bayes Plot
Iris-setosa
Iris-versicolor
Iris-virginica
Density
0.0
0.1
0.2
0.3
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal_length



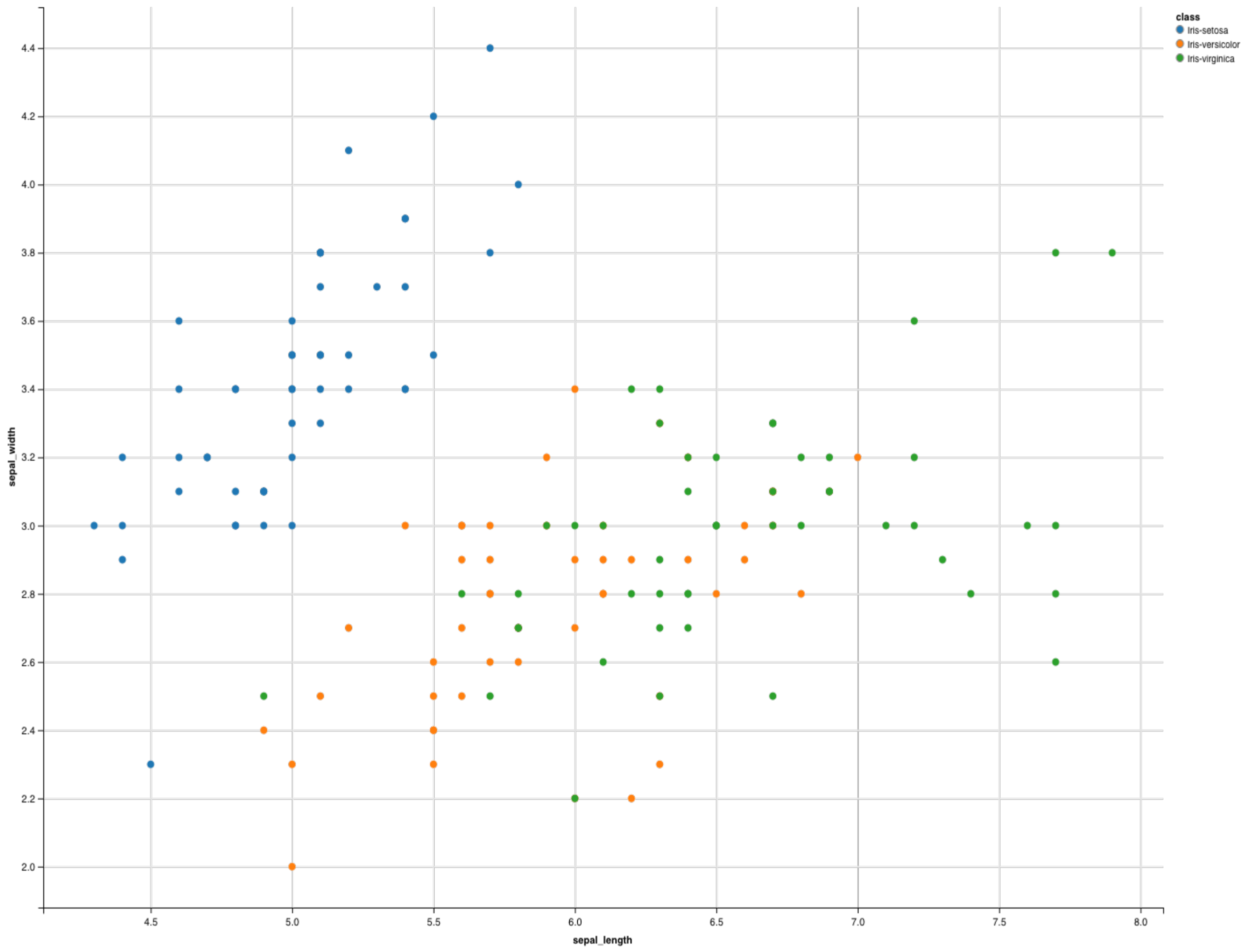


K Nearest Neighbour Classifier

- library(ggvis) #load ggvis package for scatter plots

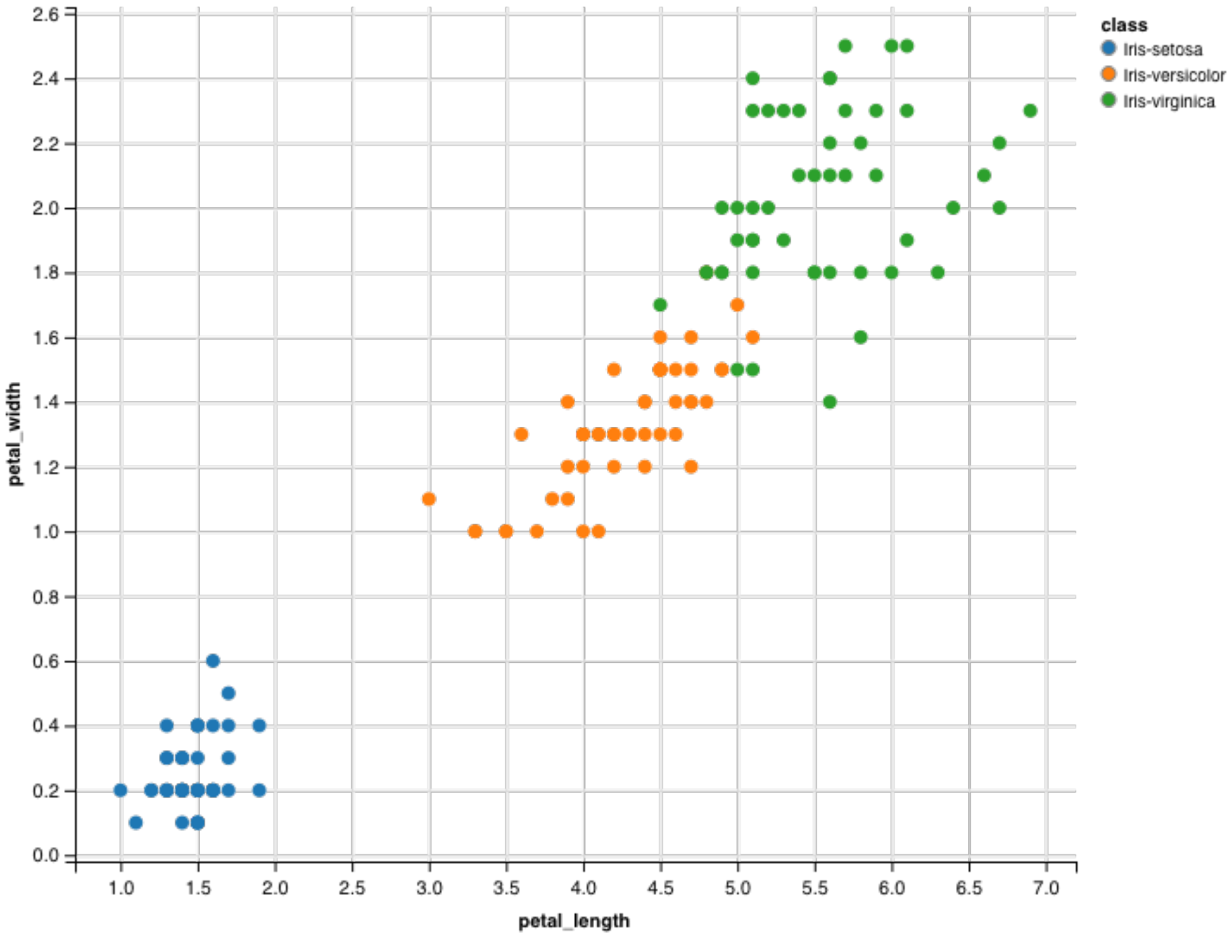
- iris %>% ggvis(~sepal_length, ~sepal_width, fill = ~class) %>% layer_points()

We see that there is a high correlation between the sepal length and the sepal width of the Setosa iris flowers, while the correlation is somewhat less high for the Virginica and Versicolor flowers.



- iris %>% ggvis(~petal_length, ~petal_width, fill = ~class) %>% layer_points()

The scatter plot shown below maps the petal length and the petal which is similar to the above graph and indicates a positive correlation between the petal length and the petal width for all different classes that are included into the Iris data set.



- str(iris) #generalized view of the data showing attributes as num or factors

```
'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 ...
 $ class       : Factor w/ 3 levels "Iris-setosa",..: 1 1 1 1 1 1 1 1 1 1 ...
```

- table(iris$class)

A quick look at the class attribute through tells us that the division of the class of flowers is 50-50-50:

```
Iris-setosa Iris-versicolor  Iris-virginica
         50              50              50
```

- summary(iris) #summary of iris data set

```
  sepal_length    sepal_width     petal_length    petal_width               class
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Iris-setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Iris-versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Iris-virginica :50
 Mean   :5.843   Mean   :3.054   Mean   :3.759   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
```

- library(class)
- set.seed(1234)

To make your training and test sets, we first set a seed. This is a number of R's random number generator. The major advantage of setting a seed is that we can get the same sequence of random numbers whenever we supply the same seed in the random number generator.

- ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))

Then, we make sure that your Iris data set is shuffled and that you have the same ratio between class in your training and test sets. We use the sample() function to take a sample with a size that is set as the number of rows of the Iris data set, or 150. You sample with replacement: you choose from a vector of 2 elements and assign either 1 or 2 to the 150 rows of the Iris data set. The assignment of the elements is subject to probability weights of 0.67 and 0.33.
We then use the sample that is stored in the variable ind to define our training and test sets:

- iris.training <- iris[ind==1, 1:4]
- iris.test <- iris[ind==2, 1:4]

- iris.trainLabels <- iris[ind==1, 5]
- iris.testLabels <- iris[ind==2, 5]

The above two lines are performed to divide the target attribute which are class labels with the training and test sets.

- iris_pred <- knn(train = iris.training, test = iris.test, cl = iris.trainLabels, k=3)

the above code actually implements the knn function which takes the arguments like training set, test set and train labels as well as amount of neighbours to be found out using this algorithm. The result of this function is a factor vector with the predicted classes for each row of the test data.

- iris_pred #Retrieve knn function result

```
 [1] Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa
 [8] Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-setosa     Iris-versicolor Iris-versicolor
[15] Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor Iris-versicolor
[22] Iris-versicolor Iris-versicolor Iris-versicolor Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica
[29] Iris-versicolor Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica
[36] Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica
Levels: Iris-setosa Iris-versicolor Iris-virginica
```

- library(gmodels)
- CrossTable(x = iris.testLabels, y = iris_pred, prop.chisq=FALSE)

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


Total Observations in Table:  40


                | iris_pred
iris.testLabels |     Iris-setosa | Iris-versicolor |  Iris-virginica |       Row Total |
----------------|-----------------|-----------------|-----------------|-----------------|
    Iris-setosa |              12 |               0 |               0 |              12 |
                |           1.000 |           0.000 |           0.000 |           0.300 |
                |           1.000 |           0.000 |           0.000 |                 |
                |           0.300 |           0.000 |           0.000 |                 |
----------------|-----------------|-----------------|-----------------|-----------------|
Iris-versicolor |               0 |              12 |               0 |              12 |
                |           0.000 |           1.000 |           0.000 |           0.300 |
                |           0.000 |           0.923 |           0.000 |                 |
                |           0.000 |           0.300 |           0.000 |                 |
----------------|-----------------|-----------------|-----------------|-----------------|
 Iris-virginica |               0 |               1 |              15 |              16 |
                |           0.000 |           0.062 |           0.938 |           0.400 |
                |           0.000 |           0.077 |           1.000 |                 |
                |           0.000 |           0.025 |           0.375 |                 |
----------------|-----------------|-----------------|-----------------|-----------------|
   Column Total |              12 |              13 |              15 |              40 |
                |           0.300 |           0.325 |           0.375 |                 |
----------------|-----------------|-----------------|-----------------|-----------------|
```

From the above output we can determine that there is an anomaly in the 1st row of the Iris-virginica column under iris.testLabels while others are accurately predicted. But as the amount of anomaly is less this model is acceptable.

Neural Network Classifier

- library('neuralnet') #import neuralnet package

First we need to create the training dataset with 50 rows selected randomly. To do this we can use the sample() function:

- itrain <- iris[sample(1:150, 50),]

Now when we call itrain, there will be 50 random rows from the original iris dataset.

Next we need to change the categorical output of itrain$class (names of the flower species) to columns with Boolean (true/false) values. To do this, we can add three columns to the data frame:

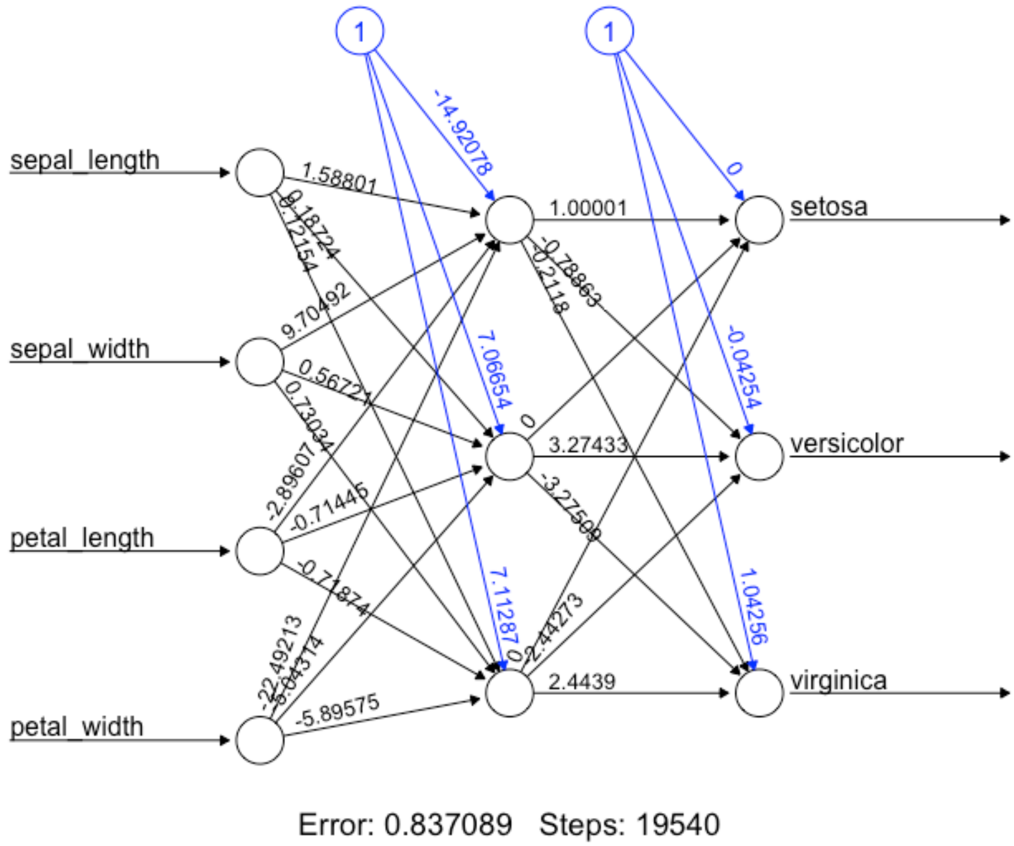
- itrain$setosa <- c(itrain$class == 'Iris-setosa')
- itrain$versicolor <- c(itrain$class== 'Iris-versicolor')
- itrain$virginica <- c(itrain$class== 'Iris-virginica')
- itrain$class <- NULL #drop the class column
- inet <- neuralnet(setosa + versicolor + virginica ~ sepal_length + sepal_width + petal_length + petal_width, itrain, hidden=3, lifesign='full') #training the neural network

This will use setosa, versicolor and virginica as output nodes and Sepal Length, Sepal Width, Petal Length and Petal Width as input nodes. The algorithm will use 3 hidden nodes, and print all information during the iterations and save the results to inet.

```
hidden: 3    thresh: 0.01    rep: 1/1    steps:    1000 min thresh: 0.07572347698
                                                   2000 min thresh: 0.03353433366
                                                   3000 min thresh: 0.01996086366
                                                   4000 min thresh: 0.01214582765
                                                   5000 min thresh: 0.01072514252
                                                   6000 min thresh: 0.01072514252
                                                   7000 min thresh: 0.01072514252
                                                   8000 min thresh: 0.01072514252
                                                   9000 min thresh: 0.01072514252
                                                  10000 min thresh: 0.01072514252
                                                  11000 min thresh: 0.01072514252
                                                  12000 min thresh: 0.01072514252
                                                  13000 min thresh: 0.01072514252
                                                  14000 min thresh: 0.01072514252
                                                  15000 min thresh: 0.01072514252
                                                  16000 min thresh: 0.01072514252
                                                  17000 min thresh: 0.01072514252
                                                  18000 min thresh: 0.01072514252
                                                  19000 min thresh: 0.0104074797
                                                  19540 error: 0.83709   time: 3.88 secs
```

Once the algorithm reaches the threshold of 0.01 (finishes) we can plot the result using:

- plot(inet, rep='best')



- predict outputs for the whole iris dataset (150 rows) using the compute() function:
- predict <- compute(inet, iris[1:4])
- predict$net.result

Calling back predict$net.result shows the predictions for the neural network. We will notice that for most rows one column contains a value close to 1 and the other two have values close to 0. The column with the value close to 1 (i.e. the largest value) indicates TRUE while a value near 0 indicates FALSE.

```
                        [,1]                [,2]              [,3]
 [1,] 1.000008068948536222  0.000067998243220 -0.00007614473477
 [2,] 1.000006595722761515  0.000004347035647 -0.00001109027993
 [3,] 1.000007858646450698  0.000019347611670 -0.00002731128649
 [4,] 1.000006869733803816 -0.000068096047266  0.00006108591814
 [5,] 1.000008073615460624  0.000065395299819 -0.00007353523095
 [6,] 1.000008054642093747 -0.000241559121919  0.00023315650012
 [7,] 1.000007611354717829 -0.000164800843399  0.00015697543222
 [8,] 1.000008042591075341  0.000019546285839 -0.00002768573978
 [9,] 0.999999426192546892 -0.000079933973499  0.00008034615658
[10,] 1.000007998224434758  0.000104060060505 -0.00011211136744
[11,] 1.000008076350125563  0.000093501496811 -0.00010164247451
[12,] 1.000008013527804884 -0.000039768298734  0.00003165045841
[13,] 1.000007894144116172  0.000108509755586 -0.00011645992505
[14,] 1.000007907326826206  0.000126831679529 -0.00013477790798
[15,] 1.000008077291758779  0.000207447302251 -0.00021555395793
[16,] 1.000008077434643150 -0.000065081875802  0.00005683490391
[17,] 1.000008070361335433 -0.000077557128781  0.00006923109475
[18,] 1.000007997679639660 -0.000060283116077  0.00005209328045
[19,] 1.000008073508812378 -0.000038251998100  0.00003000501992
[20,] 1.000008071662341180 -0.000071395548223  0.00006317865850
```

There are more such results upto 150 (whole data set)

We can use the which.max() function on each row to workout which column has the largest value. For example which.max(predict$net.result[1,]) returns the column number with the highest value for the first row.
To do this we can create a variable called result and then insert which column has the maximum value for each row using a for loop:

- result<-0
- for (i in 1:150) { result[i] <- which.max(predict$net.result[i,]) }

Now we can change the column numbers to reflect the name of the class using:

for (i in 1:150) { if (result[i]==1) {result[i] ='Iris-setosa'} }
for (i in 1:150) { if (result[i]==2) {result[i] ='Iris-versicolor'} }
for (i in 1:150) { if (result[i]==3) {result[i] ='Iris-virginica'} }

Finally, we can combine the actual data with the predicted data using:

- comparison <- iris
- comparison$Predicted <- result

You can call back comparison to show the results of the actual species and the predicted species using the neural network.

- comparison

51	7.0	3.2	4.7	1.4	Iris-versicolor	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor	Iris-versicolor
54	5.5	2.3	4.0	1.3	Iris-versicolor	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor	Iris-versicolor
56	5.7	2.8	4.5	1.3	Iris-versicolor	Iris-versicolor
57	6.3	3.3	4.7	1.6	Iris-versicolor	Iris-virginica
58	4.9	2.4	3.3	1.0	Iris-versicolor	Iris-versicolor
59	6.6	2.9	4.6	1.3	Iris-versicolor	Iris-versicolor
60	5.2	2.7	3.9	1.4	Iris-versicolor	Iris-versicolor
61	5.0	2.0	3.5	1.0	Iris-versicolor	Iris-versicolor
62	5.9	3.0	4.2	1.5	Iris-versicolor	Iris-versicolor
63	6.0	2.2	4.0	1.0	Iris-versicolor	Iris-versicolor
64	6.1	2.9	4.7	1.4	Iris-versicolor	Iris-versicolor
65	5.6	2.9	3.6	1.3	Iris-versicolor	Iris-versicolor
66	6.7	3.1	4.4	1.4	Iris-versicolor	Iris-versicolor
67	5.6	3.0	4.5	1.5	Iris-versicolor	Iris-versicolor
68	5.8	2.7	4.1	1.0	Iris-versicolor	Iris-versicolor
69	6.2	2.2	4.5	1.5	Iris-versicolor	Iris-virginica
70	5.6	2.5	3.9	1.1	Iris-versicolor	Iris-versicolor
71	5.9	3.2	4.8	1.8	Iris-versicolor	Iris-virginica
72	6.1	2.8	4.0	1.3	Iris-versicolor	Iris-versicolor
73	6.3	2.5	4.9	1.5	Iris-versicolor	Iris-virginica
74	6.1	2.8	4.7	1.2	Iris-versicolor	Iris-versicolor
75	6.4	2.9	4.3	1.3	Iris-versicolor	Iris-versicolor

As we can see, the neural network was able to correctly predict the majority of species using only the initial training set of 50 rows.

References:

http://rischanlab.github.io/NaiveBayes.html
https://www.datacamp.com/community/tutorials/machine-learning-in-r
http://hodgett.co.uk/get-started-with-neural-networks-in-r/