Iris Dataset Simple algorithms Examples

Bhushan Kamble

March 17, 2017

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

Iris data is loaded. We can see the summary that it has 150 observations of 5 variables of three species of the plant.

data(iris)  
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  
## 7 4.6 3.4 1.4 0.3 setosa  
## 8 5.0 3.4 1.5 0.2 setosa  
## 9 4.4 2.9 1.4 0.2 setosa  
## 10 4.9 3.1 1.5 0.1 setosa  
## 11 5.4 3.7 1.5 0.2 setosa  
## 12 4.8 3.4 1.6 0.2 setosa  
## 13 4.8 3.0 1.4 0.1 setosa  
## 14 4.3 3.0 1.1 0.1 setosa  
## 15 5.8 4.0 1.2 0.2 setosa  
## 16 5.7 4.4 1.5 0.4 setosa  
## 17 5.4 3.9 1.3 0.4 setosa  
## 18 5.1 3.5 1.4 0.3 setosa  
## 19 5.7 3.8 1.7 0.3 setosa  
## 20 5.1 3.8 1.5 0.3 setosa  
## 21 5.4 3.4 1.7 0.2 setosa  
## 22 5.1 3.7 1.5 0.4 setosa  
## 23 4.6 3.6 1.0 0.2 setosa  
## 24 5.1 3.3 1.7 0.5 setosa  
## 25 4.8 3.4 1.9 0.2 setosa  
## 26 5.0 3.0 1.6 0.2 setosa  
## 27 5.0 3.4 1.6 0.4 setosa  
## 28 5.2 3.5 1.5 0.2 setosa  
## 29 5.2 3.4 1.4 0.2 setosa  
## 30 4.7 3.2 1.6 0.2 setosa  
## 31 4.8 3.1 1.6 0.2 setosa  
## 32 5.4 3.4 1.5 0.4 setosa  
## 33 5.2 4.1 1.5 0.1 setosa  
## 34 5.5 4.2 1.4 0.2 setosa  
## 35 4.9 3.1 1.5 0.2 setosa  
## 36 5.0 3.2 1.2 0.2 setosa  
## 37 5.5 3.5 1.3 0.2 setosa  
## 38 4.9 3.6 1.4 0.1 setosa  
## 39 4.4 3.0 1.3 0.2 setosa  
## 40 5.1 3.4 1.5 0.2 setosa  
## 41 5.0 3.5 1.3 0.3 setosa  
## 42 4.5 2.3 1.3 0.3 setosa  
## 43 4.4 3.2 1.3 0.2 setosa  
## 44 5.0 3.5 1.6 0.6 setosa  
## 45 5.1 3.8 1.9 0.4 setosa  
## 46 4.8 3.0 1.4 0.3 setosa  
## 47 5.1 3.8 1.6 0.2 setosa  
## 48 4.6 3.2 1.4 0.2 setosa  
## 49 5.3 3.7 1.5 0.2 setosa  
## 50 5.0 3.3 1.4 0.2 setosa  
## 51 7.0 3.2 4.7 1.4 versicolor  
## 52 6.4 3.2 4.5 1.5 versicolor  
## 53 6.9 3.1 4.9 1.5 versicolor  
## 54 5.5 2.3 4.0 1.3 versicolor  
## 55 6.5 2.8 4.6 1.5 versicolor  
## 56 5.7 2.8 4.5 1.3 versicolor  
## 57 6.3 3.3 4.7 1.6 versicolor  
## 58 4.9 2.4 3.3 1.0 versicolor  
## 59 6.6 2.9 4.6 1.3 versicolor  
## 60 5.2 2.7 3.9 1.4 versicolor  
## 61 5.0 2.0 3.5 1.0 versicolor  
## 62 5.9 3.0 4.2 1.5 versicolor  
## 63 6.0 2.2 4.0 1.0 versicolor  
## 64 6.1 2.9 4.7 1.4 versicolor  
## 65 5.6 2.9 3.6 1.3 versicolor  
## 66 6.7 3.1 4.4 1.4 versicolor  
## 67 5.6 3.0 4.5 1.5 versicolor  
## 68 5.8 2.7 4.1 1.0 versicolor  
## 69 6.2 2.2 4.5 1.5 versicolor  
## 70 5.6 2.5 3.9 1.1 versicolor  
## 71 5.9 3.2 4.8 1.8 versicolor  
## 72 6.1 2.8 4.0 1.3 versicolor  
## 73 6.3 2.5 4.9 1.5 versicolor  
## 74 6.1 2.8 4.7 1.2 versicolor  
## 75 6.4 2.9 4.3 1.3 versicolor  
## 76 6.6 3.0 4.4 1.4 versicolor  
## 77 6.8 2.8 4.8 1.4 versicolor  
## 78 6.7 3.0 5.0 1.7 versicolor  
## 79 6.0 2.9 4.5 1.5 versicolor  
## 80 5.7 2.6 3.5 1.0 versicolor  
## 81 5.5 2.4 3.8 1.1 versicolor  
## 82 5.5 2.4 3.7 1.0 versicolor  
## 83 5.8 2.7 3.9 1.2 versicolor  
## 84 6.0 2.7 5.1 1.6 versicolor  
## 85 5.4 3.0 4.5 1.5 versicolor  
## 86 6.0 3.4 4.5 1.6 versicolor  
## 87 6.7 3.1 4.7 1.5 versicolor  
## 88 6.3 2.3 4.4 1.3 versicolor  
## 89 5.6 3.0 4.1 1.3 versicolor  
## 90 5.5 2.5 4.0 1.3 versicolor  
## 91 5.5 2.6 4.4 1.2 versicolor  
## 92 6.1 3.0 4.6 1.4 versicolor  
## 93 5.8 2.6 4.0 1.2 versicolor  
## 94 5.0 2.3 3.3 1.0 versicolor  
## 95 5.6 2.7 4.2 1.3 versicolor  
## 96 5.7 3.0 4.2 1.2 versicolor  
## 97 5.7 2.9 4.2 1.3 versicolor  
## 98 6.2 2.9 4.3 1.3 versicolor  
## 99 5.1 2.5 3.0 1.1 versicolor  
## 100 5.7 2.8 4.1 1.3 versicolor  
## 101 6.3 3.3 6.0 2.5 virginica  
## 102 5.8 2.7 5.1 1.9 virginica  
## 103 7.1 3.0 5.9 2.1 virginica  
## 104 6.3 2.9 5.6 1.8 virginica  
## 105 6.5 3.0 5.8 2.2 virginica  
## 106 7.6 3.0 6.6 2.1 virginica  
## 107 4.9 2.5 4.5 1.7 virginica  
## 108 7.3 2.9 6.3 1.8 virginica  
## 109 6.7 2.5 5.8 1.8 virginica  
## 110 7.2 3.6 6.1 2.5 virginica  
## 111 6.5 3.2 5.1 2.0 virginica  
## 112 6.4 2.7 5.3 1.9 virginica  
## 113 6.8 3.0 5.5 2.1 virginica  
## 114 5.7 2.5 5.0 2.0 virginica  
## 115 5.8 2.8 5.1 2.4 virginica  
## 116 6.4 3.2 5.3 2.3 virginica  
## 117 6.5 3.0 5.5 1.8 virginica  
## 118 7.7 3.8 6.7 2.2 virginica  
## 119 7.7 2.6 6.9 2.3 virginica  
## 120 6.0 2.2 5.0 1.5 virginica  
## 121 6.9 3.2 5.7 2.3 virginica  
## 122 5.6 2.8 4.9 2.0 virginica  
## 123 7.7 2.8 6.7 2.0 virginica  
## 124 6.3 2.7 4.9 1.8 virginica  
## 125 6.7 3.3 5.7 2.1 virginica  
## 126 7.2 3.2 6.0 1.8 virginica  
## 127 6.2 2.8 4.8 1.8 virginica  
## 128 6.1 3.0 4.9 1.8 virginica  
## 129 6.4 2.8 5.6 2.1 virginica  
## 130 7.2 3.0 5.8 1.6 virginica  
## 131 7.4 2.8 6.1 1.9 virginica  
## 132 7.9 3.8 6.4 2.0 virginica  
## 133 6.4 2.8 5.6 2.2 virginica  
## 134 6.3 2.8 5.1 1.5 virginica  
## 135 6.1 2.6 5.6 1.4 virginica  
## 136 7.7 3.0 6.1 2.3 virginica  
## 137 6.3 3.4 5.6 2.4 virginica  
## 138 6.4 3.1 5.5 1.8 virginica  
## 139 6.0 3.0 4.8 1.8 virginica  
## 140 6.9 3.1 5.4 2.1 virginica  
## 141 6.7 3.1 5.6 2.4 virginica  
## 142 6.9 3.1 5.1 2.3 virginica  
## 143 5.8 2.7 5.1 1.9 virginica  
## 144 6.8 3.2 5.9 2.3 virginica  
## 145 6.7 3.3 5.7 2.5 virginica  
## 146 6.7 3.0 5.2 2.3 virginica  
## 147 6.3 2.5 5.0 1.9 virginica  
## 148 6.5 3.0 5.2 2.0 virginica  
## 149 6.2 3.4 5.4 2.3 virginica  
## 150 5.9 3.0 5.1 1.8 virginica

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 ...

summary(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 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   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

From the data, it can be seen that the observations are given in order of the sepcies.

To randomize the iris data set lets use the runif fucntion. It creates a uniform distribution of 150 nos. And we can use there order as a rank for our data set to mix it up.

set.seed(1234)  
  
random <- runif(150)  
iris\_random <- iris[order(random),]  
head(iris\_random)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 7 4.6 3.4 1.4 0.3 setosa  
## 64 6.1 2.9 4.7 1.4 versicolor  
## 73 6.3 2.5 4.9 1.5 versicolor  
## 98 6.2 2.9 4.3 1.3 versicolor  
## 101 6.3 3.3 6.0 2.5 virginica  
## 110 7.2 3.6 6.1 2.5 virginica

The data set is randomized. Lets normalize the numerical variables of the data set. Normalizing the numerical values is really effective for algorithms, as it provide a measure from 0 to 1 which corresponds to min value to the max value of the data column.

We define a normal function which will normalize the set of values according to its minimum value and maximum value. Lets create a new data set iris\_new applying the function.

normal <- function(x) (  
 return( (x-min(x) / (max(x)-min(x))) )  
)  
normal(1:12)

## [1] 0.9090909 1.9090909 2.9090909 3.9090909 4.9090909 5.9090909  
## [7] 6.9090909 7.9090909 8.9090909 9.9090909 10.9090909 11.9090909

iris\_new <- as.data.frame(lapply(iris\_random[,-5], normal))  
summary(iris\_new)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :3.106 Min. :1.167 Min. :0.8305 Min. :0.05833   
## 1st Qu.:3.906 1st Qu.:1.967 1st Qu.:1.4305 1st Qu.:0.25833   
## Median :4.606 Median :2.167 Median :4.1805 Median :1.25833   
## Mean :4.649 Mean :2.224 Mean :3.5885 Mean :1.15767   
## 3rd Qu.:5.206 3rd Qu.:2.467 3rd Qu.:4.9305 3rd Qu.:1.75833   
## Max. :6.706 Max. :3.567 Max. :6.7305 Max. :2.45833

Lets create test and train data sets. Train data set is what we will build our model on. We will test our model on test data set Lets have 20 observation out of 150 for test and the rest as training data set. Lets create respective column of the observation's species to use in the model and check the accuracy of the test data.

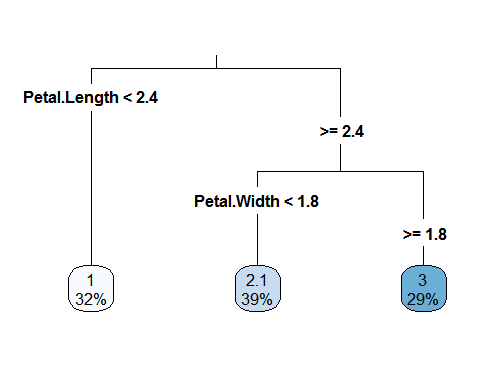
train <- iris\_new[1:100,]  
test <- iris\_new[101:150,]  
train\_sp <- iris\_random[1:100,5]  
test\_sp <- iris\_random[101:150,5]

Now we can use K-NN algorithm. Lets call the "class" package which contains the K-NN algorithm. In k-NN algorithm, we have to provide 'k' value which is no of nearest neighbours(NN) to look for in order to classify it. In common we take an odd value, let's take it as square root of the observation. Lets build a model on it. cl is the class of the training data set and k is the no of neighbours to look for in order to classify it accordingly. I have included CART(from rpart library), C5.0 algorithms

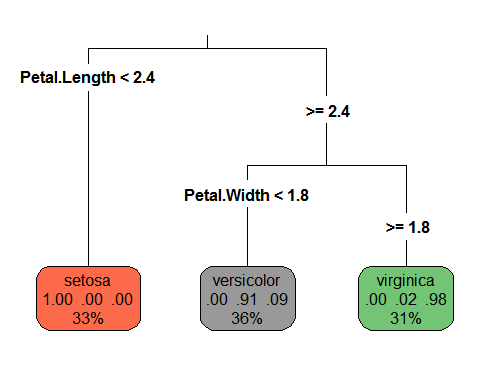
require(class)

## Loading required package: class

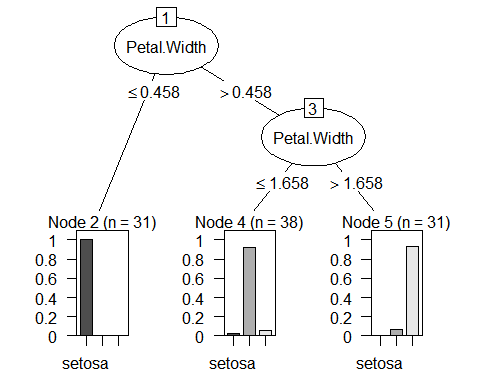
model\_knn <- knn(train= train,test=test,cl= train\_sp,k=13)  
  
library(rpart)  
library(rpart.plot)  
model\_dectree\_anova <- rpart(iris\_random[1:100,]$Species~ .,data = iris\_random[1:100,] ,method = "anova")  
predict\_anova <- predict(model\_dectree\_anova,test)  
  
predict\_anova[predict\_anova < 1.5] <- "setosa"  
predict\_anova[predict\_anova < 2.5 & predict\_anova >=1.5] <- "versicolor"  
predict\_anova[predict\_anova < 3.0 & predict\_anova >=2.5] <- "virginica"  
  
  
model\_dectree\_class <- rpart(iris\_random$Species~ .,data = iris\_random)  
predict\_class <- predict(model\_dectree\_class,test)  
  
  
rpart.plot(model\_dectree\_anova,type = 3)



rpart.plot(model\_dectree\_class, type = 3)  
  
  
library(C50)



model\_c50 <- C5.0(train,train\_sp)  
predict\_c50 <- predict(model\_c50,test)  
plot(model\_c50)

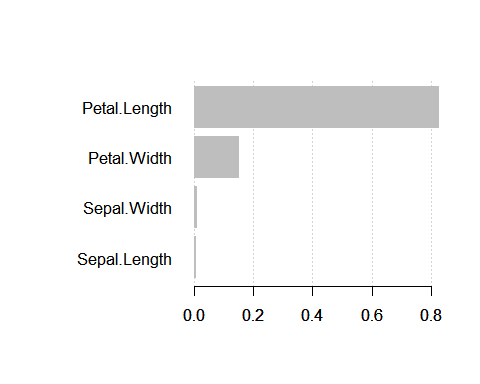


I know I should not use it at this scale. But then again!

library(xgboost)  
model\_xgboost <- xgboost(  
 data= data.matrix(train),  
 label = as.factor(train\_sp),  
 nrounds = 15,  
 objective= "reg:linear"  
)

## [1] train-rmse:1.204551   
## [2] train-rmse:0.862699   
## [3] train-rmse:0.622392   
## [4] train-rmse:0.454554   
## [5] train-rmse:0.340005   
## [6] train-rmse:0.256243   
## [7] train-rmse:0.194087   
## [8] train-rmse:0.148642   
## [9] train-rmse:0.122280   
## [10] train-rmse:0.097089   
## [11] train-rmse:0.078371   
## [12] train-rmse:0.064456   
## [13] train-rmse:0.053747   
## [14] train-rmse:0.043997   
## [15] train-rmse:0.036353

names <- dimnames(data.matrix(train))[[2]]  
importance\_matrix <- xgb.importance(names, model = model\_xgboost)  
xgb.plot.importance(importance\_matrix[1:4,])



pred\_test <- predict(model\_xgboost,data.matrix(test))  
  
predict\_xgb <- NULL  
predict\_xgb[pred\_test <=1.5] <- "setosa"  
predict\_xgb[pred\_test > 1.5 & pred\_test < 2.5 ] <- "versicolor"  
predict\_xgb[pred\_test > 2.5] <- "virginica"

Lets have a look at the confusion matrix from each model. Although we have very small no of observations. Powerful algorithms like xgboost work really well for very large dataset and so are the other algorithms.

table(model\_knn,test\_sp)

## test\_sp  
## model\_knn setosa versicolor virginica  
## setosa 18 0 0  
## versicolor 0 13 2  
## virginica 0 0 17

table(predict\_anova,test\_sp)

## test\_sp  
## predict\_anova setosa versicolor virginica  
## setosa 18 0 0  
## versicolor 0 13 2  
## virginica 0 0 17

table(predict\_c50,test\_sp)

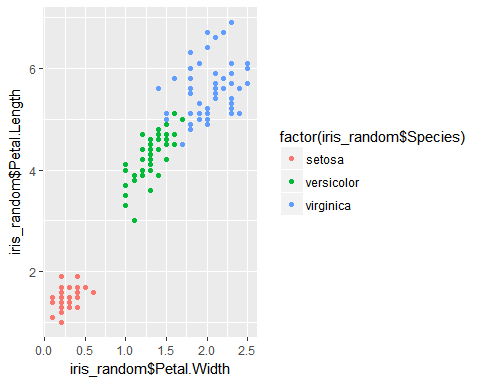
## test\_sp  
## predict\_c50 setosa versicolor virginica  
## setosa 17 0 0  
## versicolor 1 13 2  
## virginica 0 0 17

table(predict\_xgb,test\_sp)

## test\_sp  
## predict\_xgb setosa versicolor virginica  
## setosa 18 0 0  
## versicolor 0 13 2  
## virginica 0 0 17

The table(test\_sp, model) matrix is also called confusion matrix. It has test\_sp on one axis and model prediction on the other. The diagonal elements are the no of correctly predicted observations for that species. We can see how the model performed. It predicted all the species correctly.

library(ggplot2)  
  
ggplot(aes(iris\_random$Petal.Width,iris\_random$Petal.Length), data = iris\_random)+ geom\_point(aes(color= factor(iris\_random$Species)))



From the above graph and the decision trees, we can see that how model considered petal width and petal length as most important factor to classify the species. 3 virginica samples are lying in versicolor samples. And that is where the algorithms are most probably making wrong choices.