**K-Nearest Neighbour(KNN):**

Definition:

**k-NN** is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

**R CODE:**

data(iris): The data function will attach the iris dataset to the present environment.

str(iris): This function will give the overall structure of the iris data set .In other words it will tell the schema of the iris data set.

'data.frame': 150 obs. of 5 variables:

$ Sepal.Length: num 6.7 7.7 5.4 4.7 5.7 6.4 5.9 5.1 5.1 5.8 ...

$ Sepal.Width : num 3.1 3.8 3.9 3.2 2.5 2.9 3 3.7 3.8 2.8 ...

$ Petal.Length: num 4.4 6.7 1.3 1.6 5 4.3 5.1 1.5 1.9 5.1 ...

$ Petal.Width : num 1.4 2.2 0.4 0.2 2 1.3 1.8 0.4 0.4 2.4 ...

$ Species : Factor w/ 3 levels "setosa","versicolor",..: 2 3 1 1 3 2 3 1 1 3 ...

summary(iris): This will summarize all the required calculations as shows in the console.

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

table(iris$Species): **table** uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.

set.seed(9850): .Random.seed is an integer vector, containing the random number generator (RNG) state for random number generation in R. It can be saved and restored, but should not be altered by the user.

random\_variables=runif(nrow(iris)): These functions provide information about the uniform distribution on the interval from min to max. dunif gives the density, punif gives the distribution function qunif gives the quantile function and **runif** generates random deviates.

In our case it will genearate **iris number of** random numbers and assign it to random\_variables.

**nrow()** ---> number of rows in a given data set.

iris = iris[order(random\_variables),]: **order** returns a permutation which rearranges its first argument into ascending or descending order, breaking ties by further arguments. sort.list is the same, using only one argument.

In our case we will order the random variables generated as per the rows of iris data set so that we can alter all the data.

summary(iris[,c(1,2,3,4)]):

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

min\_max\_norm =

function(x)

{

return((x-min(x)) / (max(x)-min(x)))

} :

The above function is created to normalize the data.As they are varying in many ways,we will normalize them such that the max will be one and the min will be 0.

iris\_norm = as.data.frame(lapply(iris[,c(1,2,3,4)],min\_max\_norm)):

Functions to check if an object is a data frame, or coerce it if possible **(as.data.frame)**.

This function will change the data set into a dataframe normalizing all the columns except the last one.

summary(iris\_norm):

Sepal.Length Sepal.Width Petal.Length Petal.Width

Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.00000

1st Qu.:0.2222 1st Qu.:0.3333 1st Qu.:0.1017 1st Qu.:0.08333

Median :0.4167 Median :0.4167 Median :0.5678 Median :0.50000

Mean :0.4287 Mean :0.4406 Mean :0.4675 Mean :0.45806

3rd Qu.:0.5833 3rd Qu.:0.5417 3rd Qu.:0.6949 3rd Qu.:0.70833

Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.00000

From the above results we can see that our nim\_max\_norm function has sucessfuly normalized the data.

iris\_train = iris\_norm[1:130,]:

This function will split the data .First 130 rows will be training data.

iris\_test = iris\_norm[131:150,]:

This function will split the data .Last 20 rows will be testing data.

iris\_train\_target = iris[1:130,5]:

This function will split the first 130 values of the 5th column.

iris\_train\_test = iris[131:150,5]:

This function will split the last 20 values of the 5th column.

install.packages("class"): The function for **knn** model is present in the class package .

model\_1 = knn(train = iris\_train,test= iris\_test,cl = iris\_train\_target,k=1):

k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

Train = training data

Test = testing data

Cl = factor of true classifications of training set (or) training target variable

table(iris\_train\_test,model\_1):

**table** uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.

model\_1

iris\_train\_test setosa versicolor virginica

setosa 5 0 0

versicolor 0 10 1

virginica 0 0 4

mean(iris\_train\_test==model\_1):

This function gives us the accuracy of the model that we have predicted.

[1] 0.95