**AIM: CLASSIFICATION ALGORITHMS - NAIVE BAYES, ID3, C 4.5, K NEAREST NEIGHBOUR**

**THEORY:**

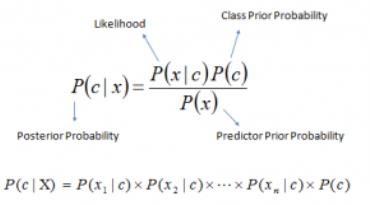
**Naive Bayesian:**

It is a [classification technique](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



Above,

* P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
* P(c) is the prior probability of class.
* P(x|c) is the likelihood which is the probability of predictor given class.
* P(x) is the prior probability of predictor. **Pros:**
* It is easy and fast to predict class of test data set. It also perform well in multi class prediction
* When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
* It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

**Cons:**

* If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
* On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.
* Another limitation of [Naive Bayes](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=naive-bayes-explained) is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

**K nearest Neighbors:**

KNN (K — Nearest Neighbors) is one of many (supervised learning) algorithms used in data mining and machine learning, it’s a classifier algorithm where the learning is based “how similar” is a data (a vector) from other .

**The KNN’s steps are:**

1. Receive an unclassified data;
2. Measure the distance (Euclidian, Manhattan, Minkowski or Weighted) from the new data to all others data that is already classified;
3. Gets the K(K is a parameter that you difine) smaller distances;
4. Check the list of classes had the shortest distance and count the amount of each class that appears;
5. Takes as correct class the class that appeared the most times; 6. Classifies the new data with the class that you took in step 5;

**Choosing the right value for K:**

To select the K that’s right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm’s ability to accurately make predictions when it’s given data it hasn’t seen before.

Here are some things to keep in mind:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I’m thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

# Advantages:

1. The algorithm is simple and easy to implement.
2. There’s no need to build a model, tune several parameters, or make additional assumptions.
3. The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

# Disadvantages:

1. The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

**R Packages:**

1. **tidyr:**

The word tidyr comes from the word tidy, which means clear. tidyr package is used to make the data' tidy'

1. **ggplot2:**

R provides the ggplot package for creating graphics declaratively. This package is famous for its elegant and quality graphs which sets it apart from other visualization packages.

1. **dplyr:**

R provides the dplyr library for performing data wrangling and data analysis.This library facilitates several functions for the data frame in R.

1. **caret:**

R allows us to perform classification and regression tasks by providing the caret package. CaretEnsemble is a feature of caret which is used for the combination of different models.

1. **e1071:**

The e1071 library provides useful functionsessential for data analysis like Naive Bayes, Fourier Transforms, SVMs, Clustering, and other miscellaneous functions

1. **IMPLEMENT NAIVE BAYES**

**INSTALLING LOADING LIBS:**

install.packages("e1071")

install.packages("klaR")

# Loading library e1071

library(e1071)

# Loading library k1aR

library(k1aR)

## Loading required package: caret

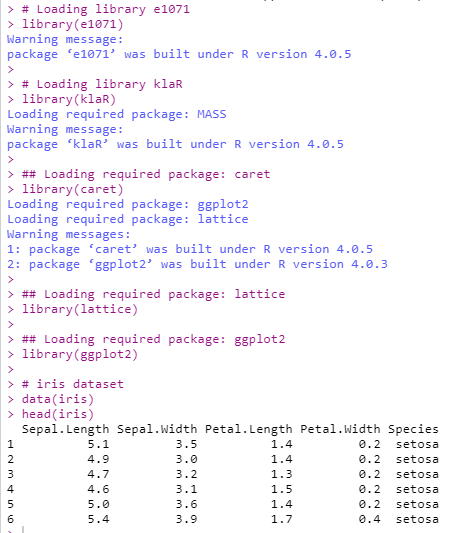
library(caret)

## Loading required package: lattice

library(lattice)

## Loading required package: ggplot2

library(ggplot2)

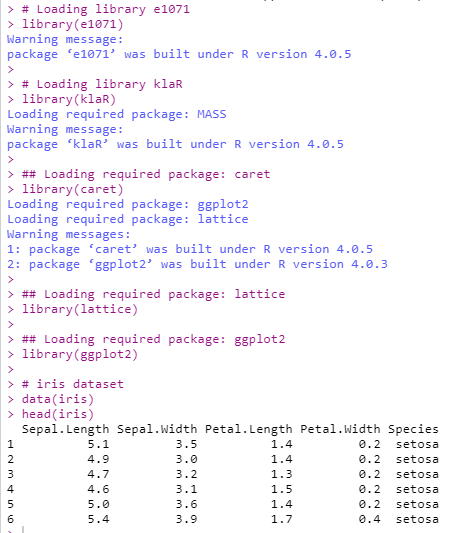


**PRINTING DATASET:**

# iris dataset

data(iris)

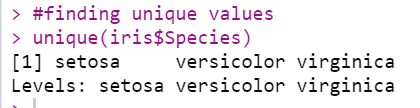
head(iris)



**PRINTING UNIQUE ELEMENTS:**

#finding unique values

unique(iris$Species)

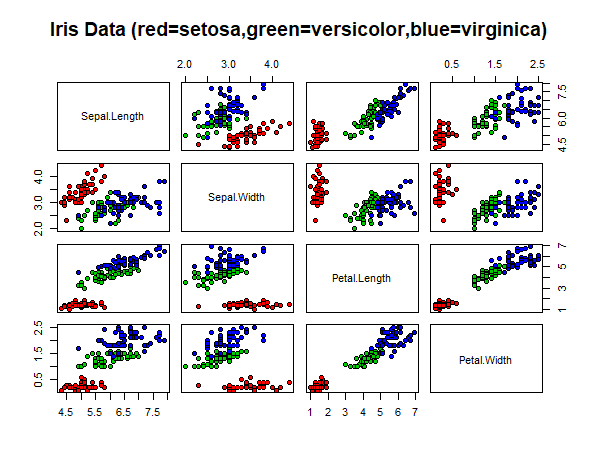


**PLOTTING GRAPH:**

#Plot graph

pairs(iris[1:4], main="Iris Data (red=setosa,green=versicolor,blue=virginica)",

pch=21, bg=c("red","green3","blue")[unclass(iris$Species)])



**TRAINING NAIVE BAYES:**

# training a naive Bayes model

index = sample(nrow(iris), floor(nrow(iris) \* 0.7)) #70/30 split.

train = iris[index,]

test = iris[-index,]

xTrain = train[,-5] # removing y-outcome variable.

yTrain = train$Species # only y.

xTest = test[,-5]

yTest = test$Species

# nb - tells to use naive bayes

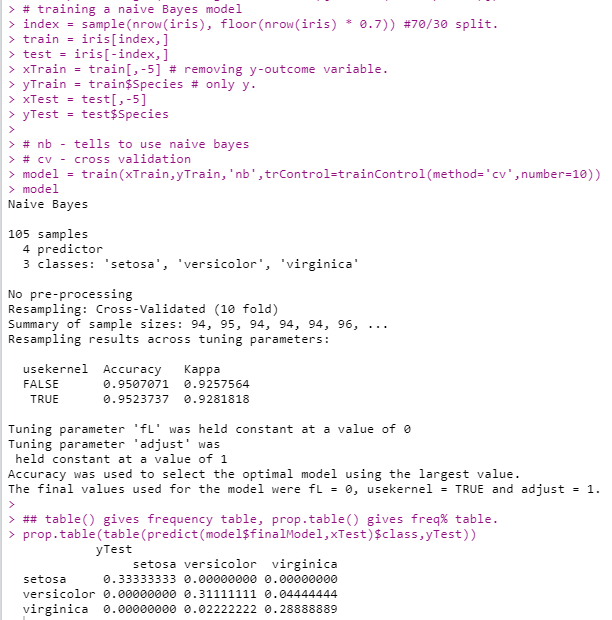
# cv - cross validation

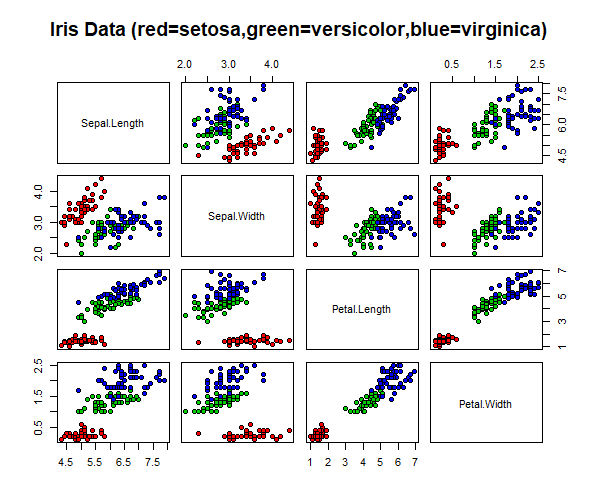
model = train(xTrain,yTrain,'nb',trControl=trainControl(method='cv',number=10))

model

## table() gives frequency table, prop.table() gives freq% table.

prop.table(table(predict(model$finalModel,xTest)$class,yTest))





1. **IMPLEMENT K NEAREST NEIGHBOUR**

**LOADING DATASET:**

#K nearest Neighbour

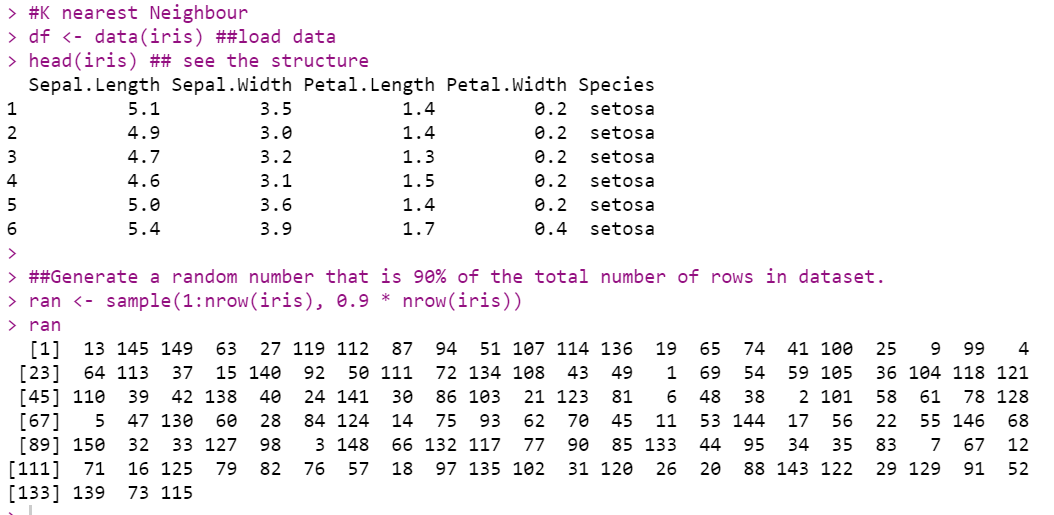
df <- data(iris) ##load data

head(iris) ## see the structure

##Generate a random number that is 90% of the total number of rows in dataset.

ran <- sample(1:nrow(iris), 0.9 \* nrow(iris))

ran



**NORMILIZATION:**

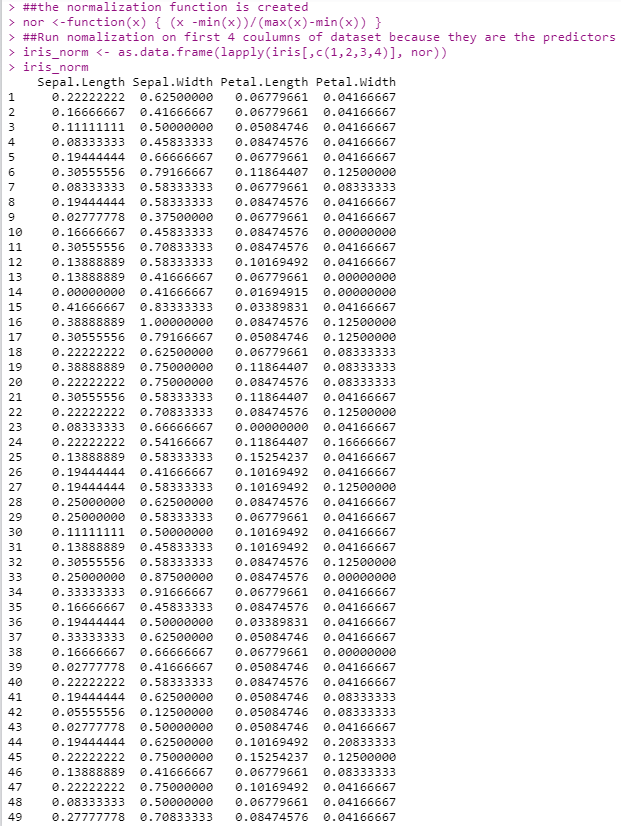
##the normalization function is created

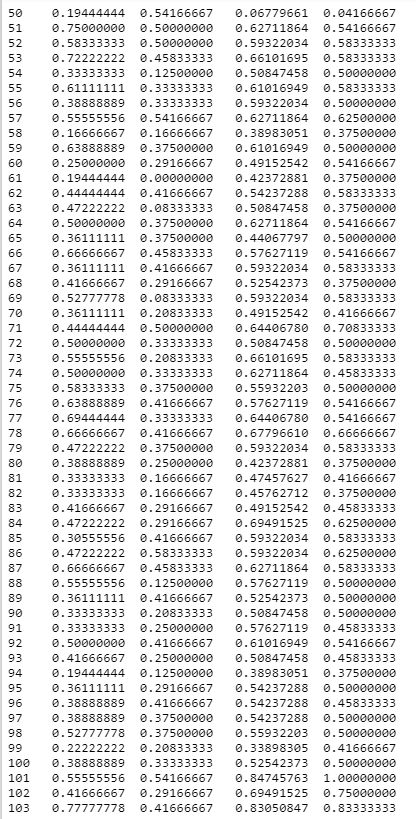
nor <-function(x) { (x -min(x))/(max(x)-min(x)) }

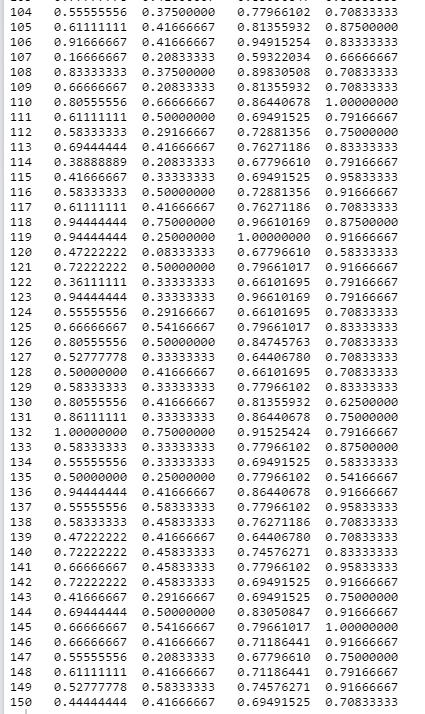
##Run nomalization on first 4 coulumns of dataset because they are the predictors

iris\_norm <- as.data.frame(lapply(iris[,c(1,2,3,4)], nor))

iris\_norm





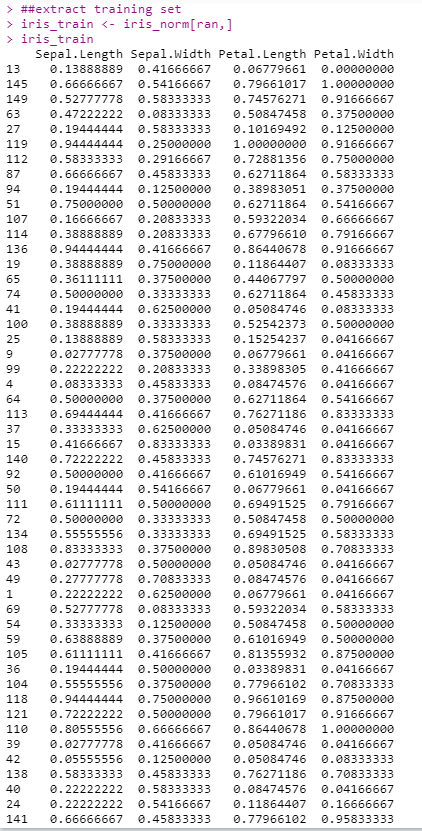


**EXTRACTING TRAINING SET:**

##extract training set

iris\_train <- iris\_norm[ran,]

iris\_train

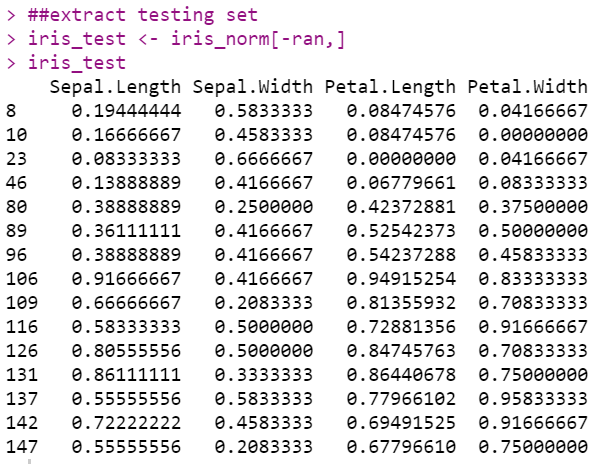


**EXTRACT TESTING DATASET:**

##extract testing set

iris\_test <- iris\_norm[-ran,]

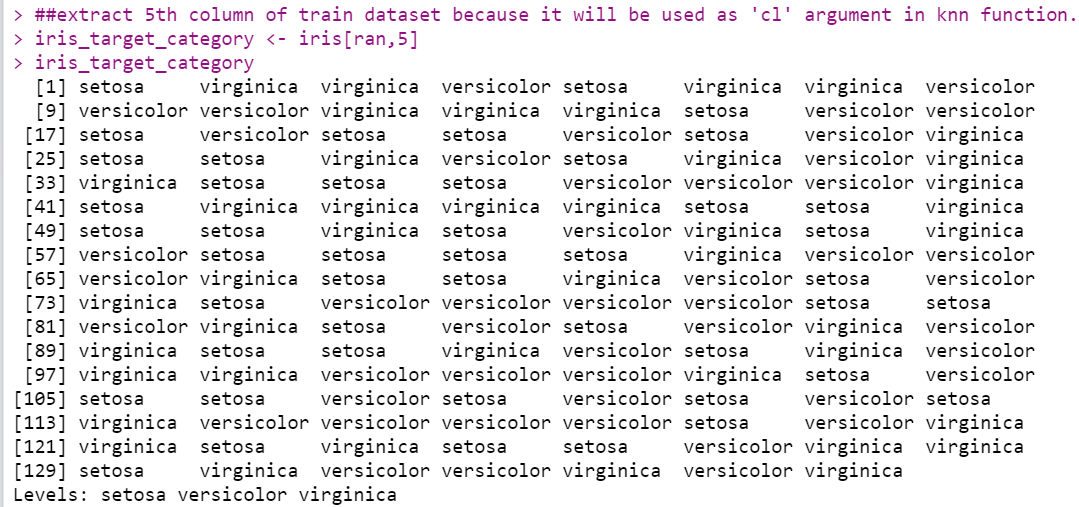
iris\_test



##extract 5th column of train dataset because it will be used as 'cl' argument in knn function.

iris\_target\_category <- iris[ran,5]

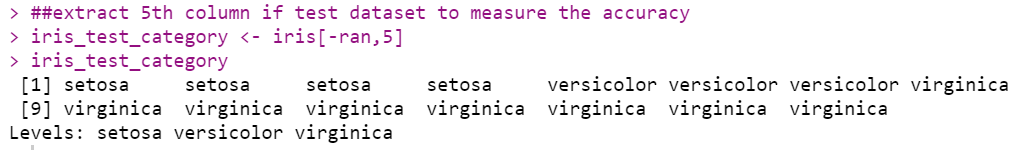
iris\_target\_category



##extract 5th column if test dataset to measure the accuracy

iris\_test\_category <- iris[-ran,5]

iris\_test\_category



**EXECUTE KNN FUNCTION:**

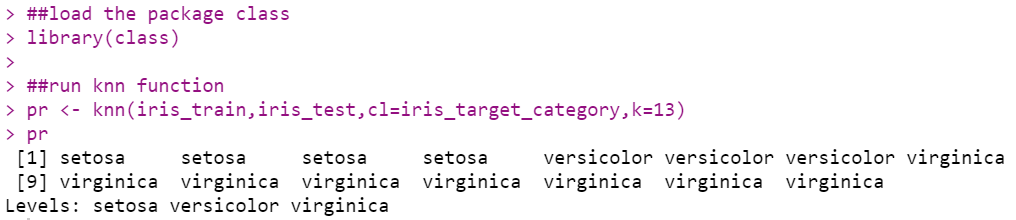
##load the package class

library(class)

##run knn function

pr <- knn(iris\_train,iris\_test,cl=iris\_target\_category,k=13)

pr

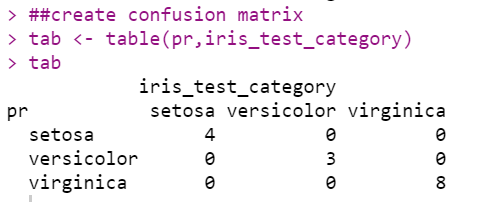


**CREATE CONFUSION MATRIX:**

##create confusion matrix

tab <- table(pr,iris\_test\_category)

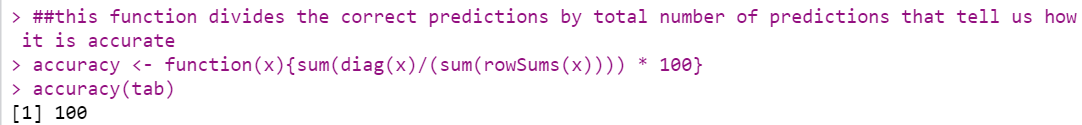
tab



**CHECKING ACCURACY:**

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}

accuracy(tab)

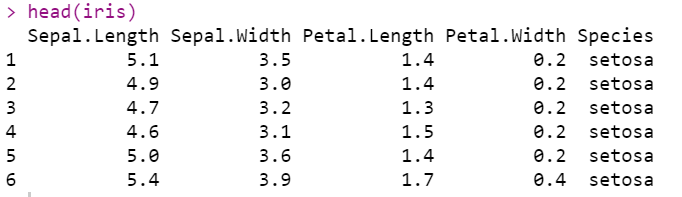


1. **IMPLEMENT K-MEANS CLUSTERING**

**LOADING DATASET:**

#K-Means clustering

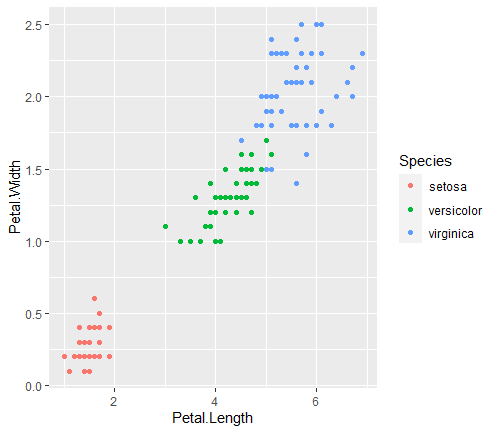
head(iris)



**LOADING LIBRARY FOR PLOTING GRAPH:**

library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom\_point()



**EXECUTING K-MENAS AND PLOTTING GRAPH:**

#Setting a seed in R means to initialize a pseudorandom number generator.

set.seed(20)

#Executing Kmeans

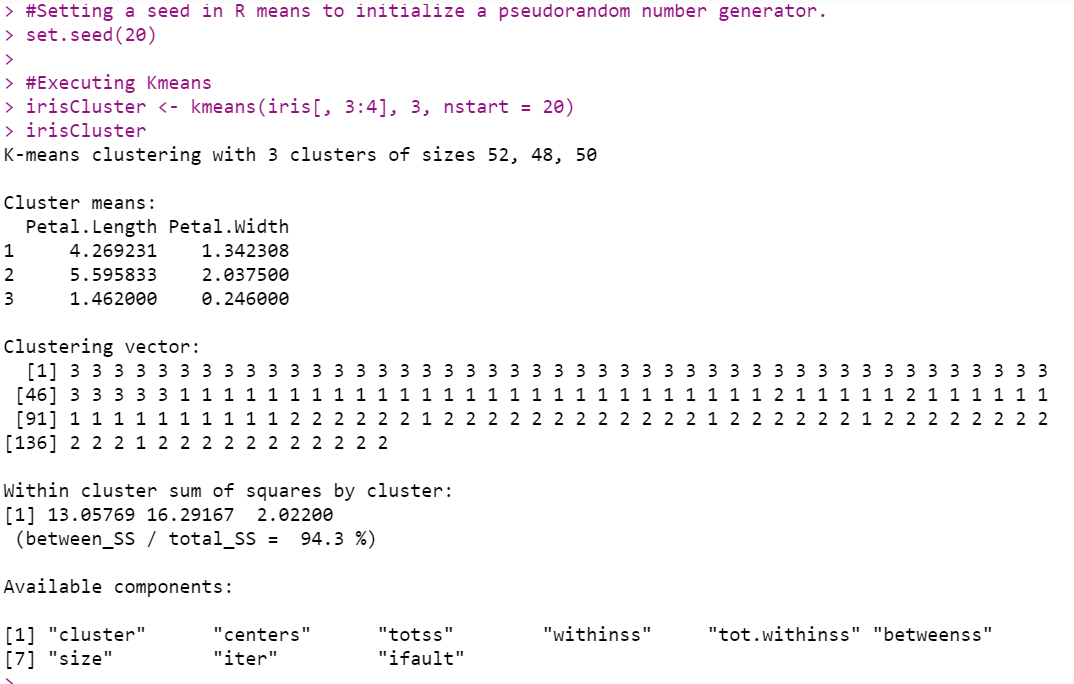
irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)

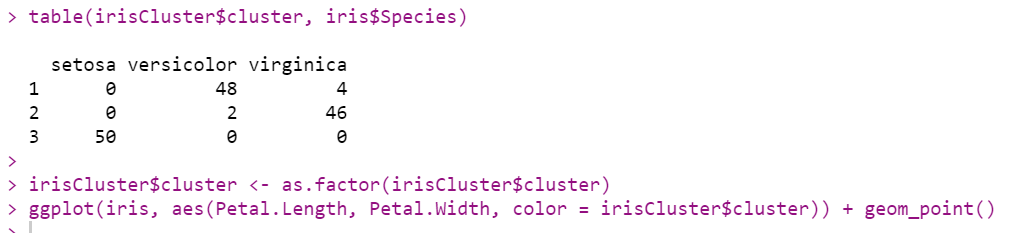
irisCluster

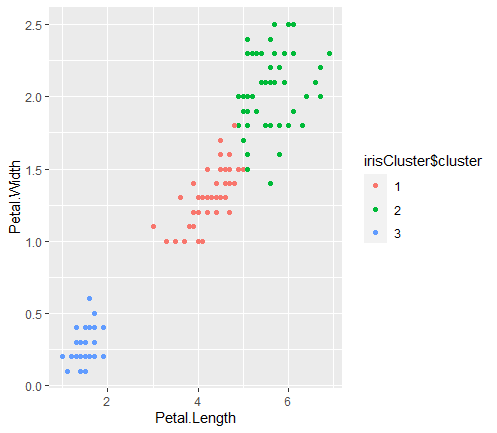
table(irisCluster$cluster, iris$Species)

irisCluster$cluster <- as.factor(irisCluster$cluster)

ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom\_point()







**CONCLUSION:**

From this practical, I have learned how to implement naive bayes, k – nn, k – means clustering in R.