	Marwadi University **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Introduction to R and R Studio (01CT0106)	**Aim: To obtain the classification of various classes using k-Nearest Neighbor approach**	
Experiment: 12	**Date: 20/04/2023**	**Enrollment No: 92200133030**

Aim: To obtain the classification of various classes using k-Nearest Neighbor approach

IDE: R Studio

Theory:

This algorithm is used to solve the classification model problems. K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line. Therefore, larger k value means smother curves of separation resulting in less complex models. Whereas, smaller k value tends to over fit the data and resulting in complex models. It's very important to have the right k-value when analyzing the dataset to avoid over fitting and under fitting of the dataset.

The model representation for KNN is the entire training dataset. It is as simple as that. KNN has no model other than storing the entire dataset, so there is no learning required. Efficient implementations can store the data using complex data structures like k-d trees to make look-up and matching of new patterns during prediction efficient. Because the entire training dataset is stored, you may want to think carefully about the consistency of your training data. It might be a good idea to curate it, update it often as new data becomes available and remove erroneous and outlier data.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

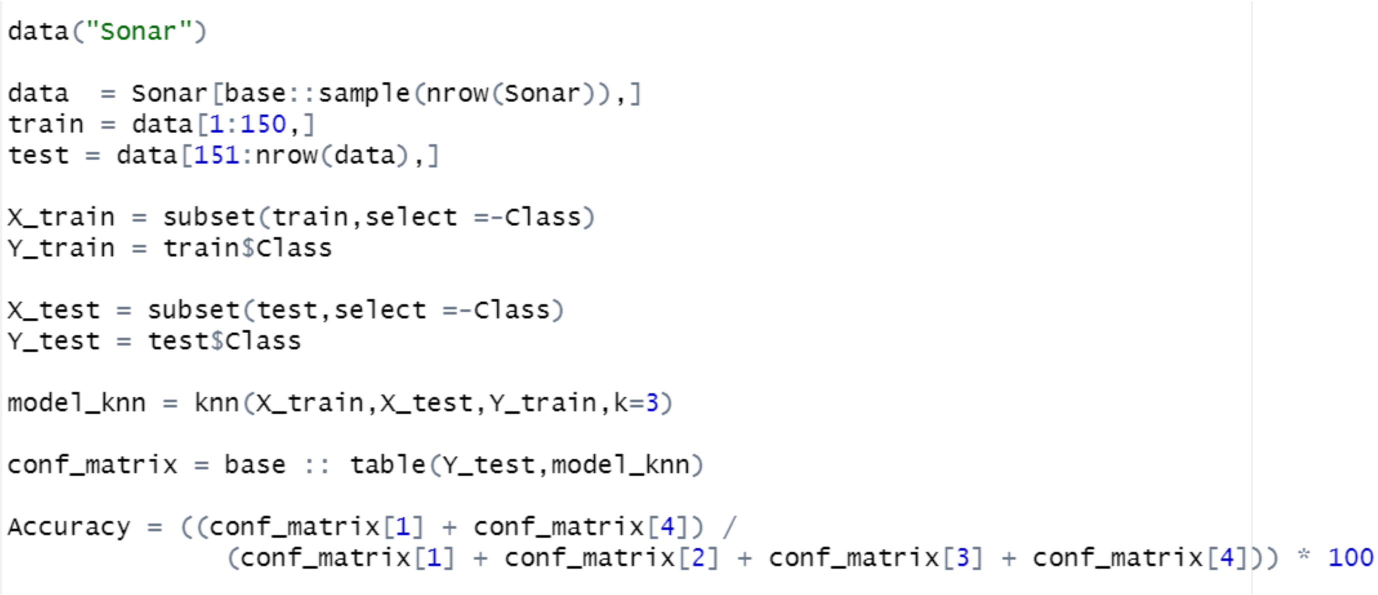
K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: Introduction to R and R Studio (01CT0106)	**Aim: To obtain the classification of various classes using k-Nearest Neighbor approach**	
Experiment: 12	**Date: 20/04/2023**	**Enrollment No: 92200133030**

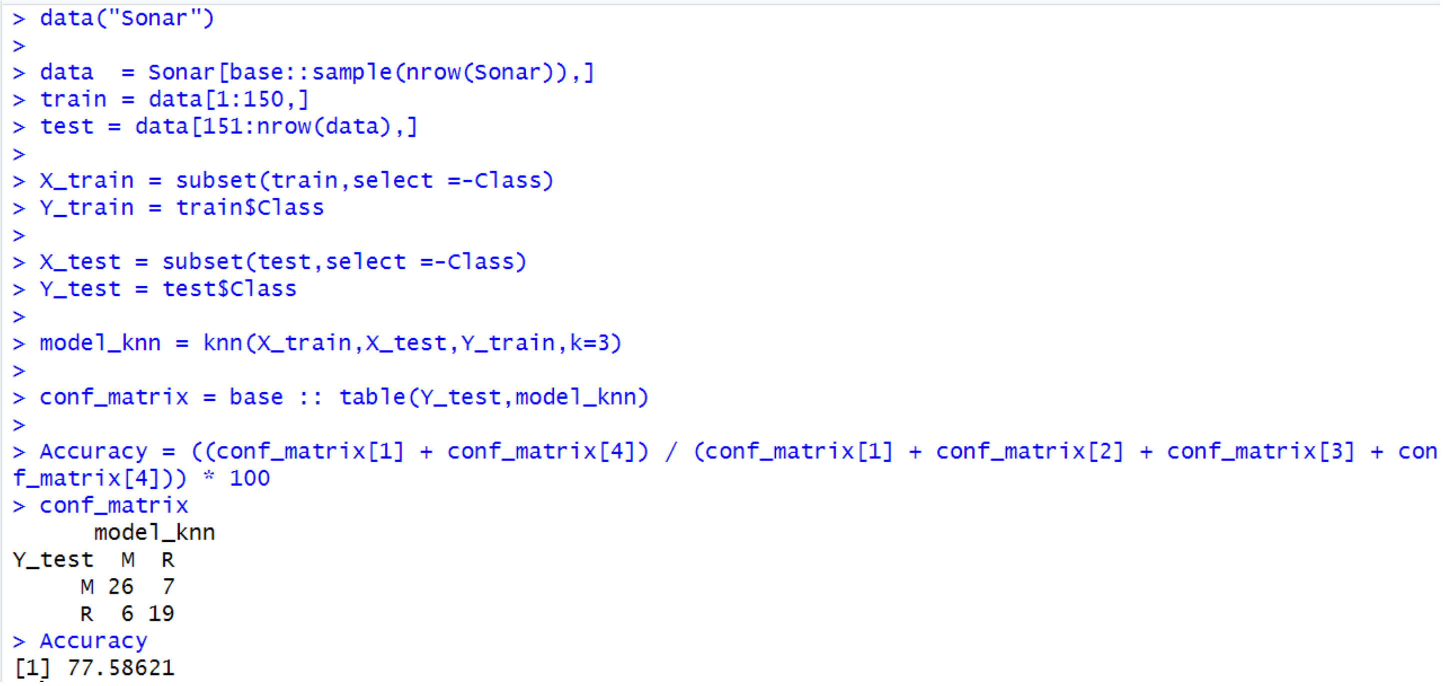
Program:

Write a R script to perform classification using K-Nearest Neighbour approach



```
data("Sonar")

data  = Sonar[base::sample(nrow(Sonar)),]
train = data[1:150,]
test = data[151:nrow(data),]

X_train = subset(train,select =-Class)
Y_train = train$Class

X_test = subset(test,select =-Class)
Y_test = test$Class

model_knn = knn(X_train,X_test,Y_train,k=3)

conf_matrix = base :: table(Y_test,model_knn)

Accuracy = ((conf_matrix[1] + conf_matrix[4]) /
              (conf_matrix[1] + conf_matrix[2] + conf_matrix[3] + conf_matrix[4])) * 100
```

Output:



```
> data("Sonar")
> 
> data  = Sonar[base::sample(nrow(Sonar)),]
> train = data[1:150,]
> test = data[151:nrow(data),]
> 
> X_train = subset(train,select =-Class)
> Y_train = train$Class
> 
> X_test = subset(test,select =-Class)
> Y_test = test$Class
> 
> model_knn = knn(X_train,X_test,Y_train,k=3)
> 
> conf_matrix = base :: table(Y_test,model_knn)
> 
> Accuracy = ((conf_matrix[1] + conf_matrix[4]) / (conf_matrix[1] + conf_matrix[2] + conf_matrix[3] + con
f_matrix[4])) * 100
> conf_matrix
      model_knn
Y_test  M  R
     M 26  7
     R  6 19
> Accuracy
[1] 77.58621
```

Observation and Learnings:

____

____

______.