I. INTRODUCTION

Machine learning is defined as – “Field of study that gives computers the capability to learn without being explicitly programmed”. Machine learning is the study of algorithms, in easy terms, ML is the study of the data available from the and work onto that and predicts the future results automatically without any manual working. It is a part of AI. it builds a model based on the previous observations called the training data. ML has a very diverse use, it is almost used in every sector such as medicine, industry, speech recognition, government works, shopping (related to clothes or groceries, etc.) emails, etc.

ML provides various approaches to the computer even though there is no perfect solution to the data is available but still, it helps to provide data that has a higher probability of happening and gives the result in the format needed. working with ML, Past data is stored as the training data and the data new found is studied and used as testing data which is further used for the optimization of the algorithms.

Using this strategy discussed above we are going to use one or two algorithms to find the pattern (or relations) between the variables.

II. READING THE DATASET.

Heart attack is one of the deadliest diseases that a person suffers, if not treated properly could be led to the death of that person. the person suffering from this disease has to be prepared as it can happen at any time. So, to reduce the risk and keep your heart healthy one has to take preventive measures and should take necessary steps to reduce the risk.

In this study, we are available with the records of a heart clinic dataset, (the dataset was downloaded from the online source, *https://www.kaggle.com/datasets*) it contains records of the 298 patients, the dataset includes various variables/ factors that are responsible for the death of a person due to the heart attack. As only two states of the target data can be predicted so this is a classification problem. Since it's a Supervised Learning problem that the data itself shows which person died or not as it is shown in the 12th column.

The 12 variables are namely-

age, anemia, creatinine_phosphokinase, diabetes, ejection_fraction, high_blood_pressure, platelets, serum_creatinine, serum_sodium, sex, smoking, time.

III. THEORETICAL STUDY OF THE DATA GIVES US THE FOLLOWING INFORMATION.

- **Age-** Despite any gender, the age of the person plays a vital role in the resulting whether a person dies due to a heart attack or not.
- **Diabetes-** Also plays an important role in the health of a person, if the person is suffering through diabetes, then there is a fair chance that the person will die due to a heart attack.
- **High_blood_pressure-** This also counts to the contributions at majority, or say one of the prime reasons for the death of a person due to heart attack. If a person is suffering from high blood pressure, then there is a chance that the patient is going to die.
- **Smoking-** Non-smokers are more likely to live more than smoker's person. the probability of a person is going to live depends on whether he/ she smokes or not.
- More, the variables like anemia, sex do not play any vital role in the decision that the person is going to die.
- Factors like creatinine_phosphokinase, ejection_fraction, platelets, serum_creatinine, serum_sodium need deep learning as they alone do not contribute much but as a group, they do play a vital role in the person's health and contribute to the upbringing of the results.

IV. ALGORITHM CLASSIFICATION

After theoretical reading of the dataset, one can conclude that the problem is of Supervised Learning as the correct output of the past data is already available with us.

In Supervised learning, the past data available can be used to find accurate results for future predictions.

As we know, the Supervised Machine Learning algorithm can be broadly classified into Regression and Classification Algorithms. In Regression algorithms, we have predicted the output for continuous values, but to predict the categorical values, we need Classification algorithms.

Classification- The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations based on training data. In Classification, a program learns from the given dataset or observations and then classifies new observations into several classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, Classes can be called targets/labels or categories.

So, the dataset available to us is of the type of Classification algorithms, where we have 11 variable classes and death is the target class.

In this we are going to apply two types of algorithms (the K-nearest Neighbour & Naïve baies classifier) and will find out the results.

V. ALGORITHM APPLICATONS

KNN ALGORITHM

➢ INTRODUCTION

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- 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.
- Using the same technique, we are going to use the knn algorithm on the ‘heart clinical dataset’ to find the relation between the parameters.

Case A- Using 80% training and 20% testing with k=10.

➢ RULES & RESULTS:

Importing library file and reading the dataset

library(ggplot2)

library(e1071)

failure_data=read.csv("C:/Users/Hp-1/Desktop/MA722/PROJECT/heart_failure_clinical_records_dataset.csv")

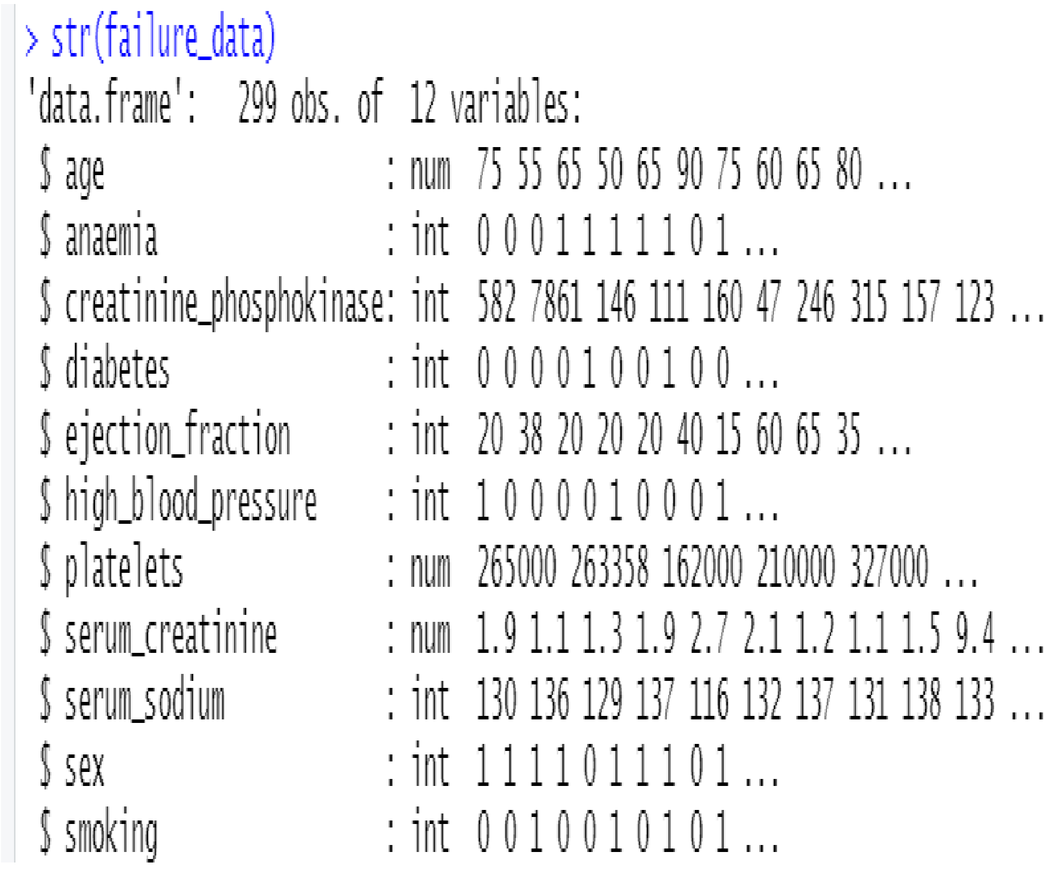
str(failure_data)

failure_data$DEATH_EVENT[failure_data$DEATH_EVENT ==0]<-'No'

failure_data$DEATH_EVENT[failure_data$DEATH_EVENT ==1]<-'Yes'

failure_data$DEATH_EVENT=factor(failure_data$DEATH_EVENT)

str(failure_data)



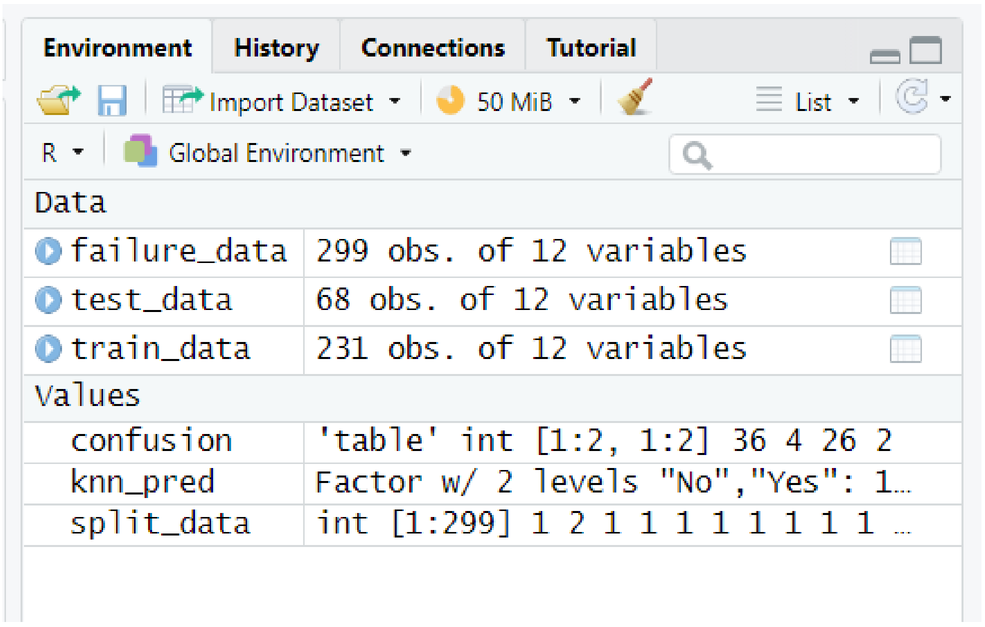
```
> str(failure_data)
'data.frame':	299 obs. of  12 variables:
 $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
 $ anaemia                 : int  0 0 0 1 1 1 1 1 0 1 ...
 $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
 $ diabetes                : int  0 0 0 0 1 0 0 1 0 0 ...
 $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
 $ high_blood_pressure     : int  1 0 0 0 0 1 0 0 0 1 ...
 $ platelets               : num  265000 263358 162000 210000 327000 ...
 $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
 $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
 $ sex                     : int  1 1 1 0 1 1 1 0 1 ...
 $ smoking                 : int  0 0 1 0 0 1 0 1 0 1 ...
```

-

Fig- Structure of the data

Splitting the data into training and testing (splitting of 80% training and 20% testing)

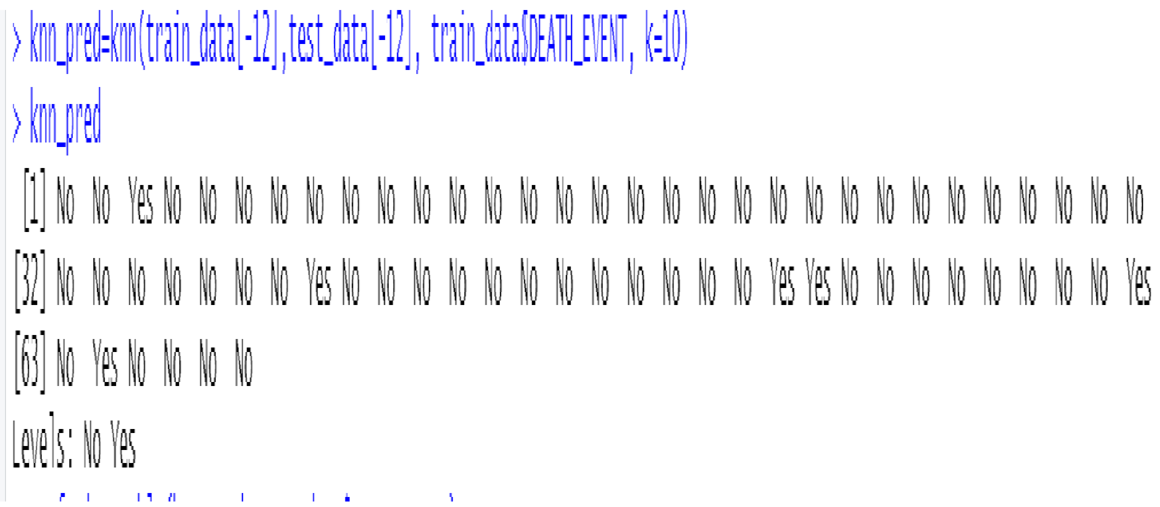
split_data=sample(2,nrow(failure_data), replace = T, prob=c(0.8,0.2))
train_data=failure_data[split_data==1,]
test_data=failure_data[split_data==2,]



```
Data
failure_data   299 obs. of 12 variables
test_data      68 obs. of 12 variables
train_data     231 obs. of 12 variables
Values
confusion      'table' int [1:2, 1:2] 36 4 26 2
knn_pred       Factor w/ 2 levels "No","Yes": 1…
split_data     int [1:299] 1 2 1 1 1 1 1 1 1 1 …
```

Applying the K-Nearest Neighbour's Algorithm (assuming k=10)

knn_pred=knn(train_data[-12],test_data[-12], train_data$DEATH_EVENT, k=10)
knn_pred



```
> knn_pred=knn(train_data[-12],test_data[-12], train_data$DEATH_EVENT, k=10)
> knn_pred
 [1] No  No  Yes No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
[32] No  No  No  No  No  No  No  Yes No  No  No  No  No  No  No  No  No  No  No  No  Yes Yes No  No  No  No  No  No  No  No  Yes
[63] No  Yes No  No  No  No
Levels: No Yes
```

Creating the Confusion Matrix

confusion=table (knn_pred, test_data$DEATH_EVENT)

confusion

confusionMatrix(knn_pred, test_data$DEATH_EVENT)

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  39  15
       Yes  9   1

               Accuracy : 0.625
                 95% CI : (0.4951, 0.743)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : 0.9909

                  Kappa : -0.1429

 Mcnemar's Test P-Value : 0.3074

            Sensitivity : 0.8125
            Specificity : 0.0625
         Pos Pred Value : 0.7222
         Neg Pred Value : 0.1000
             Prevalence : 0.7500
         Detection Rate : 0.6094
   Detection Prevalence : 0.8438
      Balanced Accuracy : 0.4375

       'Positive' Class : No
```

Misclassification error

> 1-sum(diag(confusion))/nrow(test_data)

```
[1] 0.375
```

Plotting the Graph
plot(knn_pred~test_data$DEATH_EVENT)

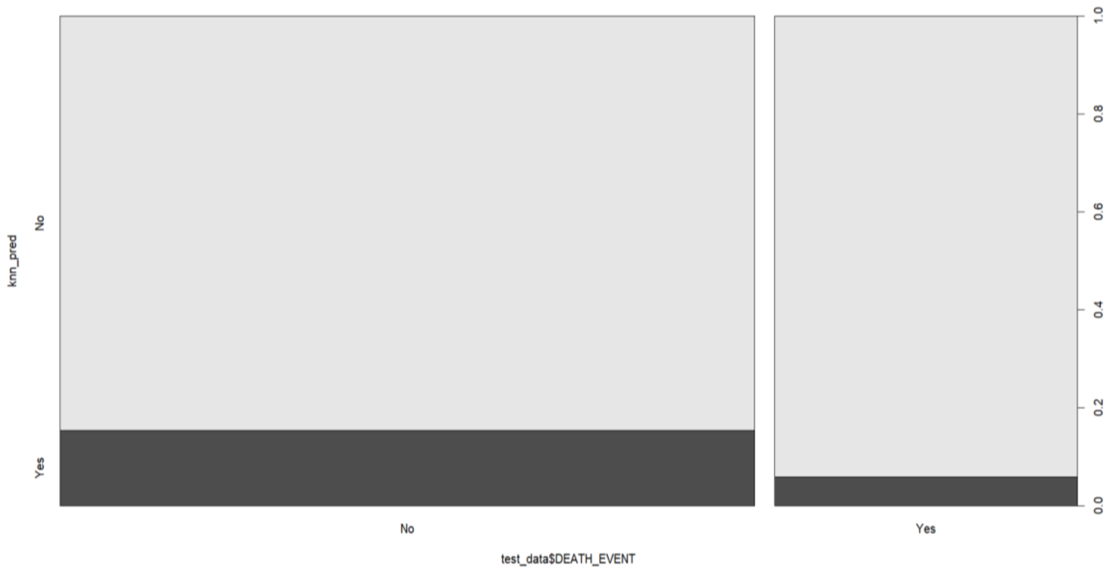


Fig- knn vs test_data

The graph is plotted between the knn predicted algorithms and the available target data (i.e., DEATH_EVENT). As we can see 67.5% of the data was accurately predicted.

➢ INFERENCE:

So far, we have applied the knn algorithm on the given clinical heart dataset in which we have taken the random samples as the training and the testing datasets. of the 80 percent of the samples were considered as the training data and 20 percent of the data were considered for testing purposes. as the dataset was mostly symmetric so for evenness in testing, we deduce the random splitting of the data.

Then after splitting the data, we applied the algorithm keeping the value of k=10 where our target data is DEATH EVENT. A total of 64 observations were available. They were random of 'yes' and 'no' category. after applying the algorithm, we found that only 40 observations were accurately predicted and 24 data were wrongly predicted

If we calculate the misclassification error it is found out to be 37.5% which is too high for any of the algorithms.

So now, we are going to change the parameter to improve the accuracy of the algorithm.

Case B- Using 70% training and 30% testing with k=20

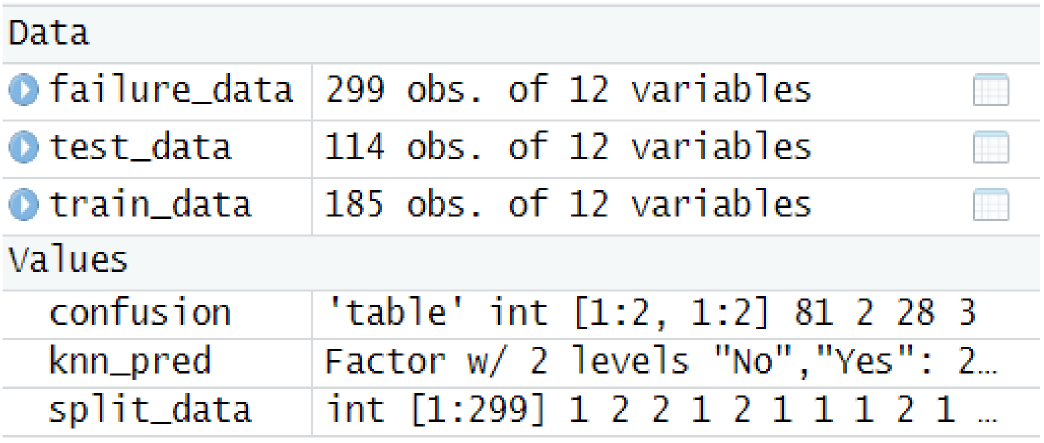
➢ **RULES & RESULTS:**

Splitting the data into training and testing (splitting of 70% training and 30% testing)

split_data=sample (2, nrow(failure_data), replace = T, prob=c (0.6,0.4))

train_data=failure_data[split_data==1,]

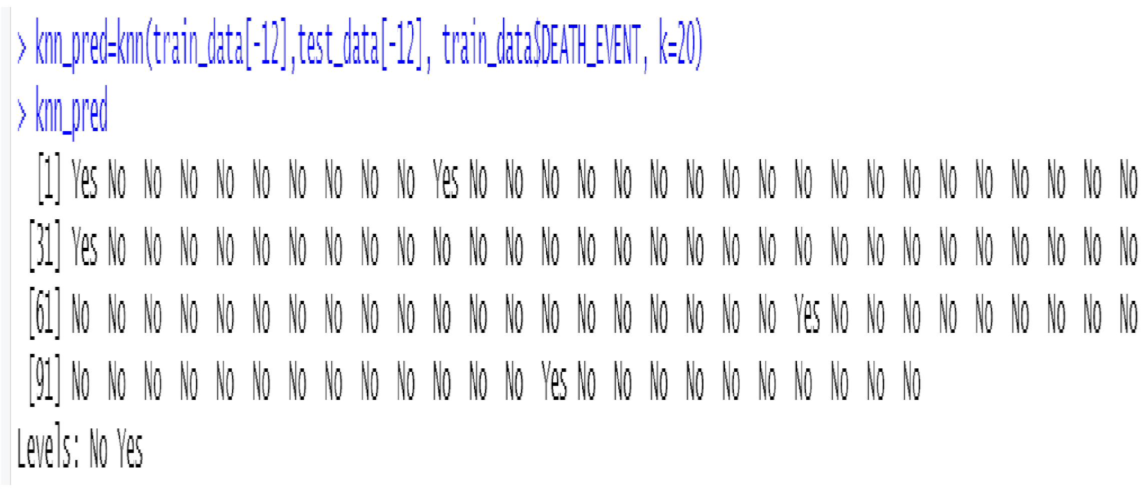
test_data=failure_data[split_data==2,]



Data		
failure_data	299 obs. of 12 variables	
test_data	114 obs. of 12 variables	
train_data	185 obs. of 12 variables	
Values		
confusion	'table' int [1:2, 1:2] 81 2 28 3	
knn_pred	Factor w/ 2 levels "No","Yes": 2…	
split_data	int [1:299] 1 2 2 1 2 1 1 1 2 1 …	

Applying the K-Nearest Neighbour's Algorithm (assuming k=20)

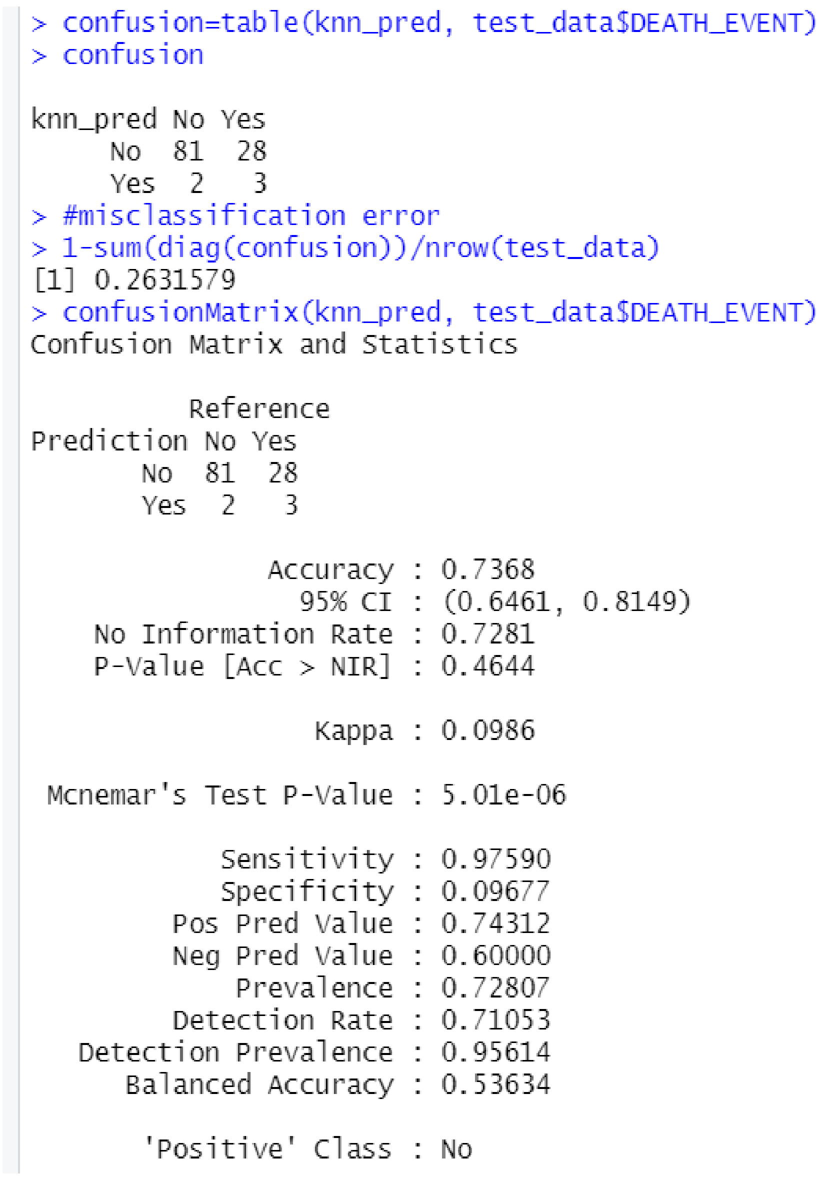
knn_pred=knn(train_data[-12],test_data[-12], train_data$DEATH_EVENT, k=20)
knn_pred



```
> knn_pred=knn(train_data[-12],test_data[-12], train_data$DEATH_EVENT, k=20)
> knn_pred
  [1] Yes No  No  No  No  No  No  No  No  No  Yes No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No 
 [31] Yes No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No 
 [61] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  Yes No  No  No  No  No  No  No  No  No 
 [91] No  No  No  No  No  No  No  No  No  No  No  No  No  Yes No  No  No  No  No  No  No  No  No  No 
Levels: No Yes
```

Creating the Confusion Matrix

confusion=table (knn_pred, test_data$DEATH_EVENT)
confusion
confusionMatrix(knn_pred, test_data$DEATH_EVENT)



```
> confusion=table(knn_pred, test_data$DEATH_EVENT)
> confusion

knn_pred No Yes
     No  81  28
     Yes  2   3
> #misclassification error
> 1-sum(diag(confusion))/nrow(test_data)
[1] 0.2631579
> confusionMatrix(knn_pred, test_data$DEATH_EVENT)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  81  28
       Yes  2   3

               Accuracy : 0.7368
                 95% CI : (0.6461, 0.8149)
    No Information Rate : 0.7281
    P-Value [Acc > NIR] : 0.4644

                  Kappa : 0.0986

 Mcnemar's Test P-Value : 5.01e-06

            Sensitivity : 0.97590
            Specificity : 0.09677
         Pos Pred Value : 0.74312
         Neg Pred Value : 0.60000
             Prevalence : 0.72807
         Detection Rate : 0.71053
   Detection Prevalence : 0.95614
      Balanced Accuracy : 0.53634

       'Positive' Class : No
```

Fig-Confusion matrix for k=10

Misclassification error

> 1-sum(diag(confusion))/nrow(test_data)

[1] 0.263175

Plotting the Graph

plot(knn_pred~test_data$DEATH_EVENT)

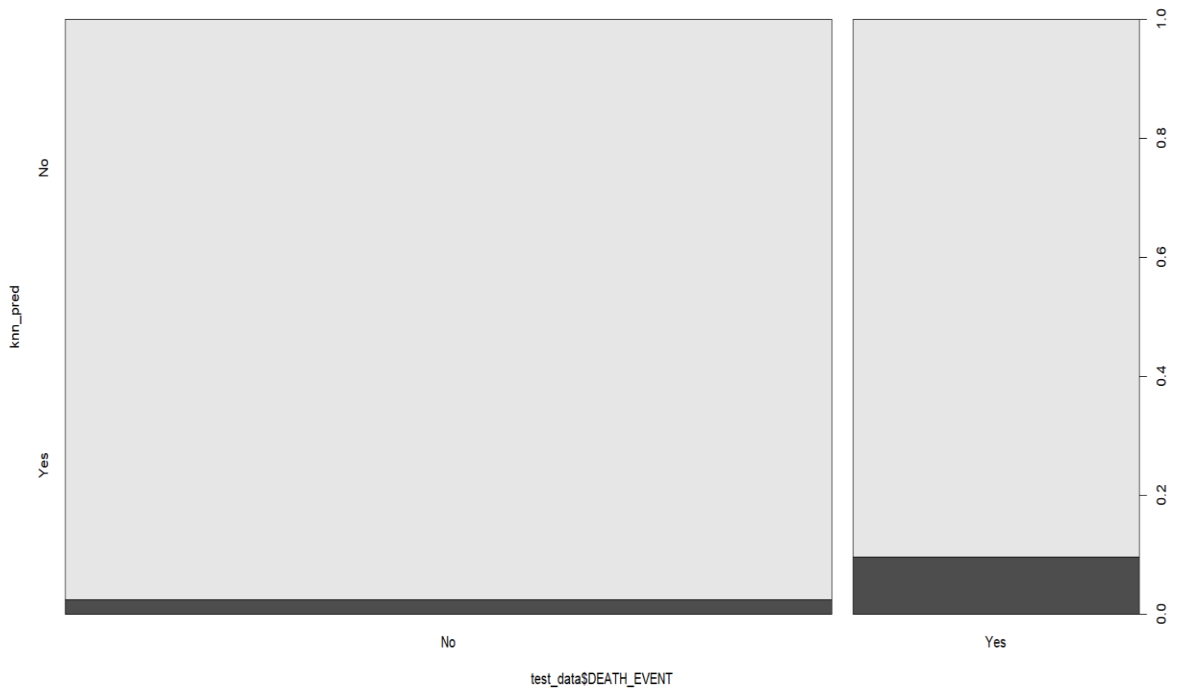


Fig- plot of knn pred vs death_event

➢ INFERENCE:

From this setup we can find that changing the value of k we can find the more accurate results. Out if total 114 observations, 30 were wrongly predicted and else were correctly predicted. So, we can say that we have improved the accuracy of our algorithm set.

The accuracy of the data is 73.68%. i.e., by revising the value of "k" we have optimized the algorithms by 6.18%.

More accurate results can be predicted by revising the data and finding the correct value of k.

NAÏVE BAIES CLASSIFIER

➢ INTRODUCTION

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object
- Since it's a probabilistic model we don't need to predict any "k" values for the observations.
- This technique will be used to find the probability of happening of the events using the training parameters and will be applied to the testing parameters.
- Bayes theorem can be classified as-

  Let A & B be two events of happening then, Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

  $$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

  Given that P(B) is not equal to zero.

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.
- P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Using 80% training and 20% testing data.

➢ RULES & RESULTS:

Importing library files and reading the csv data

library(ggplot2)

library(e1071)

library(caret)

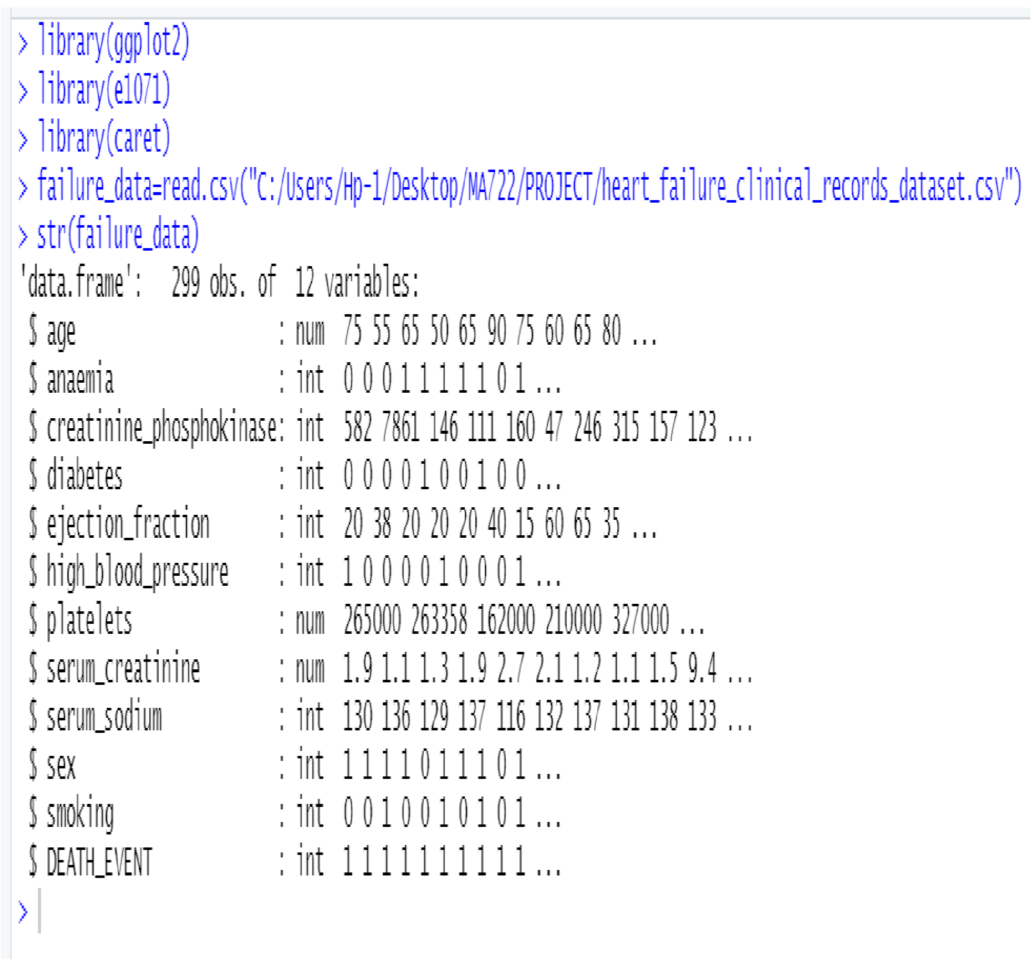
failure_data=read.csv("C:/Users/Hp-1/Desktop/MA722/PROJECT/heart_failure_clinical_records_dataset.csv")

str(failure_data) failure_data$DEATH_EVENT[failure_data$DEATH_EVENT ==0]<-'No'

failure_data$DEATH_EVENT[failure_data$DEATH_EVENT ==1]<-'Yes'

failure_data$DEATH_EVENT=factor(failure_data$DEATH_EVENT)

str(failure_data)



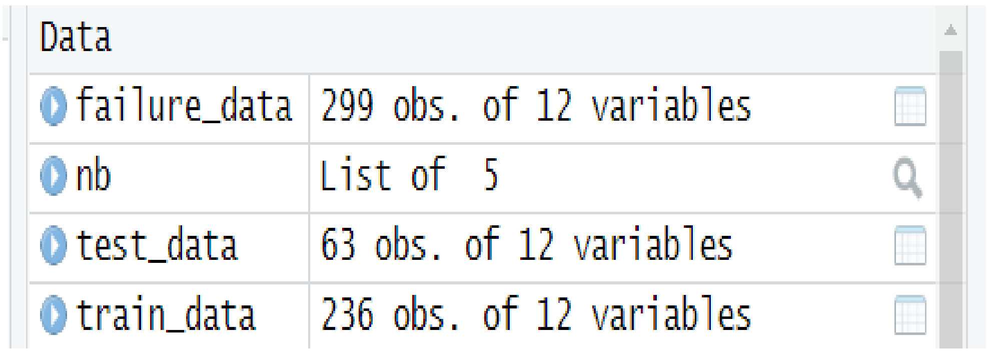
```
> library(ggplot2)
> library(e1071)
> library(caret)
> failure_data=read.csv("C:/Users/Hp-1/Desktop/MA722/PROJECT/heart_failure_clinical_records_dataset.csv")
> str(failure_data)
'data.frame':	299 obs. of  12 variables:
 $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
 $ anaemia                 : int  0 0 0 1 1 1 1 1 0 1 ...
 $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
 $ diabetes                : int  0 0 0 0 1 0 0 1 0 0 ...
 $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
 $ high_blood_pressure     : int  1 0 0 0 0 1 0 0 0 1 ...
 $ platelets               : num  265000 263358 162000 210000 327000 ...
 $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
 $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
 $ sex                     : int  1 1 1 1 0 1 1 1 0 1 ...
 $ smoking                 : int  0 0 1 0 0 1 0 1 0 1 ...
 $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
> 
```

Splitting the data into training and testing (splitting of 80% training and 20% testing)

data_part = sample(2, nrow(failure_data), replace = T, prob=c(0.8,0.2))

train_data=failure_data[data_part==1,]

test_data=failure_data[data_part==2,]



Data	
failure_data	299 obs. of 12 variables
nb	List of 5
test_data	63 obs. of 12 variables
train_data	236 obs. of 12 variables

Case A – Considering **All the Parameters**.

❖ **FOR TRAINING DATA**

➢ **RULES & RESULTS:**

```
nb=naiveBayes(DEATH_EVENT~ .,data = train_data)
nb
#Misclassification error on training data
nb_predict=predict(nb,train_data)
tab=table(nb_predict,train_data$DEATH_EVENT)
tab
1 - sum(diag(tab))/ sum(tab)
plot(nb_predict,train_data$DEATH_EVENT)
#Using confusion matrix function
confusionMatrix(nb_predict,train_data$DEATH_EVENT)
```

```
>nb=naiveBayes(DEATH_EVENT~.,data = train_data)
> nb

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)
```

```
A-priori probabilities:
Y
       No       Yes 
0.7028112 0.2971888 

Conditional probabilities:
     age
Y         [,1]     [,2]
  No  59.00381 10.49734
  Yes 64.70270 13.38035

     anaemia
Y          [,1]      [,2]
  No  0.4114286 0.4935046
  Yes 0.4729730 0.5026770

     creatinine_phosphokinase
Y         [,1]      [,2]
  No  540.3371  764.3488
  Yes 689.0676 1220.3929

     diabetes
Y          [,1]      [,2]
  No  0.4342857 0.4970851
  Yes 0.4459459 0.5004626

     ejection_fraction
Y         [,1]     [,2]
  No  40.38857 10.78675
  Yes 33.44595 12.64602

     high_blood_pressure
Y          [,1]      [,2]
  No  0.3314286 0.4720775
  Yes 0.4324324 0.4987953

     platelets
Y         [,1]     [,2]
  No  264323.9 94286.58
  Yes 252538.7 96085.55

     serum_creatinine
Y         [,1]      [,2]
  No  1.178229 0.6594266
  Yes 1.887973 1.6082987

     serum_sodium
Y         [,1]     [,2]
  No  137.4114 3.718838
  Yes 135.3784 5.122346
```

```
    sex
Y          [,1]      [,2]
  No  0.6685714 0.4720775
  Yes 0.5810811 0.4967499

    smoking
Y          [,1]      [,2]
  No  0.3428571 0.4760262
  Yes 0.3378378 0.4762015

> #Misclassification error on training data
> nb_predict=predict(nb,train_data)
> tab=table(nb_predict,train_data$DEATH_EVENT)
> tab

nb_predict  No Yes
       No  163  55
       Yes  12  19
> 1 - sum(diag(tab))/ sum(tab)
[1] 0.2690763
> plot(nb_predict,train_data$DEATH_EVENT)
```

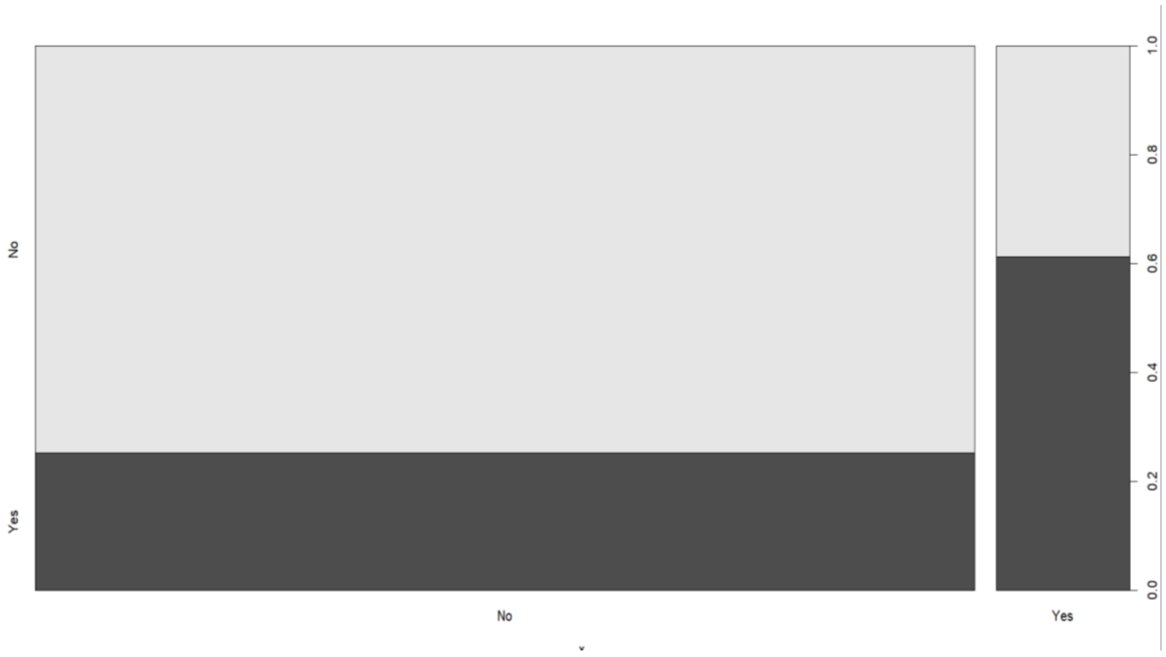


Fig-nb predict vs train data for death event

```
> #Using confusion matrix function
> confusionMatrix(nb_predict,train_data$DEATH_EVENT)
Confusion Matrix and Statistics
```

```
          Reference
Prediction  No Yes
       No  163  55
       Yes  12  19

               Accuracy : 0.7309
                 95% CI : (0.6713, 0.785)
    No Information Rate : 0.7028
    P-Value [Acc > NIR] : 0.1842

                  Kappa : 0.2261

 Mcnemar's Test P-Value : 2.88e-07

            Sensitivity : 0.9314
            Specificity : 0.2568
         Pos Pred Value : 0.7477
         Neg Pred Value : 0.6129
             Prevalence : 0.7028
         Detection Rate : 0.6546
   Detection Prevalence : 0.8755
      Balanced Accuracy : 0.5941

       'Positive' Class : No
```

❖ FOR TESTING DATA

```
nb_predict_test=predict(nb,test_data)
tab2=table(nb_predict_test,test_data$DEATH_EVENT)
tab2
#MISCLASSIFICATION ERROR
1 - sum(diag(tab2))/ sum(tab2)
plot(nb_predict_test~test_data$DEATH_EVENT)
confusionMatrix(nb_predict_test,test_data$DEATH_EVENT)
```

```
> nb_predict_test=predict(nb,test_data)
> tab2=table(nb_predict_test,test_data$DEATH_EVENT)
> tab2

nb_predict_test No Yes
           No  26  14
           Yes  2   8

> 1 - sum(diag(tab2))/ sum(tab2)
[1] 0.32

> plot(nb_predict_test~test_data$DEATH_EVENT)
```

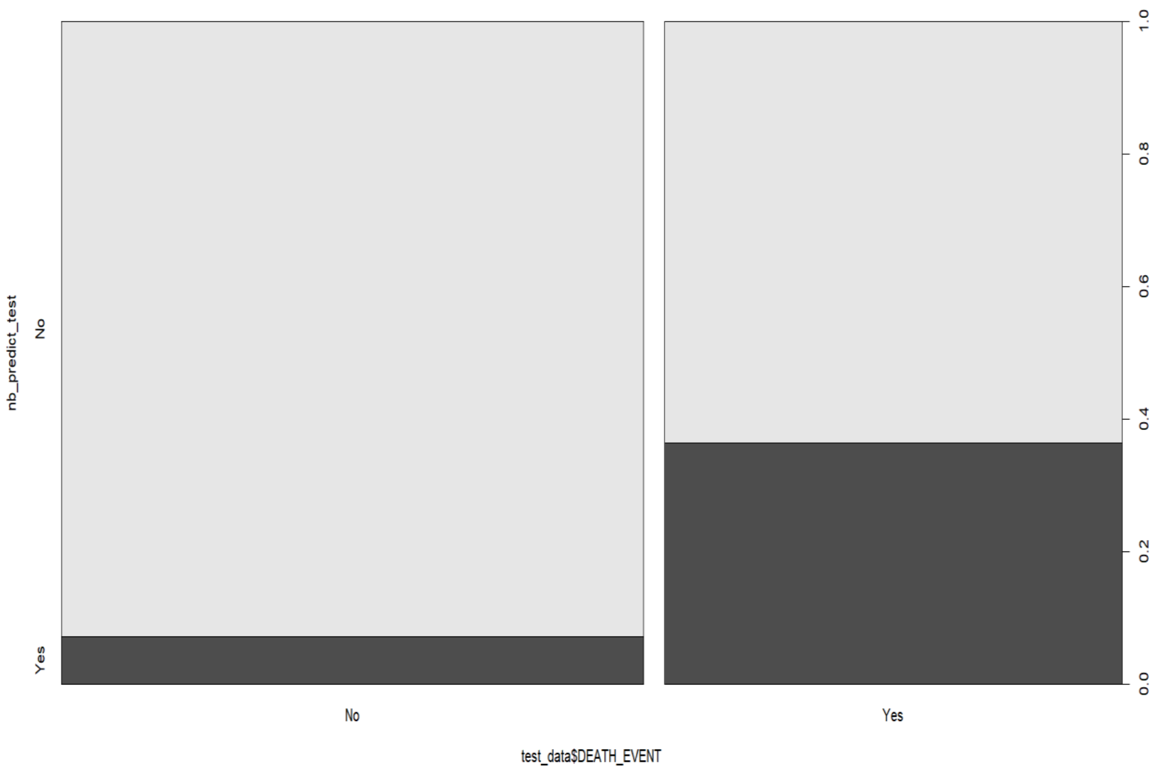


Fig- nb predict vs test data for death _event

```
> confusionMatrix(nb_predict_test,test_data$DEATH_EVENT)
Confusion Matrix and Statistics


         Reference
Prediction No Yes
       No  26  14
       Yes  2   8

             Accuracy : 0.68
               95% CI : (0.533, 0.8048)
   No Information Rate : 0.56
   P-Value [Acc > NIR] : 0.05710
```

```
                  Kappa : 0.3103

 Mcnemar's Test P-Value : 0.00596

            Sensitivity : 0.9286
            Specificity : 0.3636
         Pos Pred Value : 0.6500
         Neg Pred Value : 0.8000
             Prevalence : 0.5600
         Detection Rate : 0.5200
   Detection Prevalence : 0.8000
      Balanced Accuracy : 0.6461

       'Positive' Class : No
```

- **INFERENCE:**

From the above algorithm solution, we can conclude that naïve baies algorithm solution is successfully worked out. Following results were obtained,

- Naïve Baies classifier algorithm was successfully applied using the the R-studio software
- For this dataset out of total 236 observations of training data 189 observations were correctly predicted and the remaining data were misclassified, which gives the accuracy of 73.09%. For the testing data of total 50 observation 34 observations were correctly predicted which gives the total accuracy of 68%.
- The probability table of the each of the parameters were discussed and shown in the table above.

Case B – Considering **age+diabetes+high_blood_prressure.**

❖ FOR TRAINING DATA

```
nb=naiveBayes(DEATH_EVENT~age+diabetes+high_blood_pressure,data =
train_data)
nb

#Misclassification error on training data
nb_predict=predict(nb,train_data)
tab=table(nb_predict,train_data$DEATH_EVENT)
tab
1 - sum(diag(tab))/ sum(tab)
plot(nb_predict,train_data$DEATH_EVENT)
#Using confusion matrix function
confusionMatrix(nb_predict,train_data$DEATH_EVENT)
```

```
> nb=naiveBayes(DEATH_EVENT~age+diabetes+high_blood_pressure,data =
train_data)
> nb

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
       No       Yes 
0.7028112 0.2971888 

Conditional probabilities:
     age
Y         [,1]     [,2]
  No  59.00381 10.49734
  Yes 64.70270 13.38035

     diabetes
Y          [,1]      [,2]
  No  0.4342857 0.4970851
  Yes 0.4459459 0.5004626

     high_blood_pressure
Y          [,1]      [,2]
  No  0.3314286 0.4720775
  Yes 0.4324324 0.4987953
```

```
> #Misclassification error on training data
> nb_predict=predict(nb,train_data)
> tab=table(nb_predict,train_data$DEATH_EVENT)
> tab
```

```
nb_predict  No Yes
       No  164  56
       Yes  11  18
```

#MISCLASSIFICATION ERROR

```
> 1 - sum(diag(tab))/ sum(tab)
[1] 0.2690763

> plot(nb_predict,train_data$DEATH_EVENT)
```

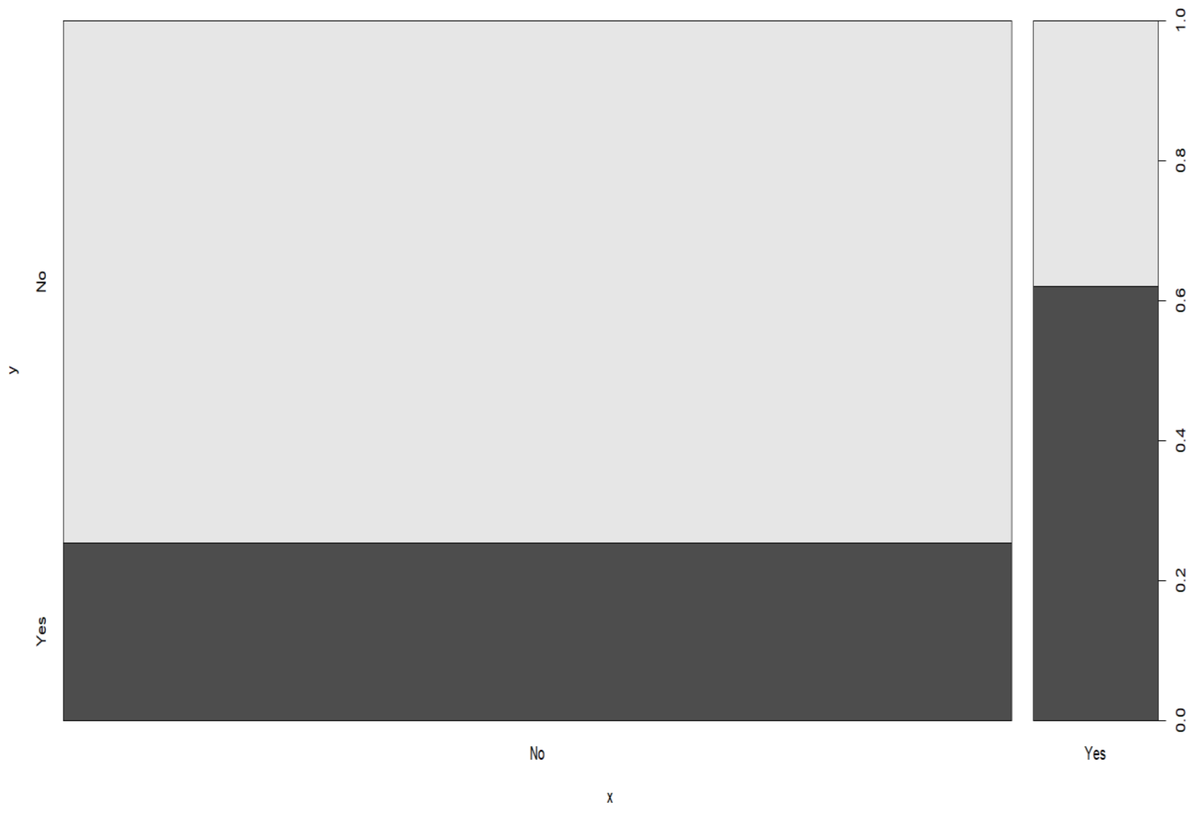


Fig-nb_predict vs the train data for death event.

```
> #Using confusion matrix function
> confusionMatrix(nb_predict,train_data$DEATH_EVENT)
Confusion Matrix and Statistics
```

```
          Reference
Prediction  No Yes
       No  164  56
       Yes  11  18

               Accuracy : 0.7309
                 95% CI : (0.6713, 0.785)
    No Information Rate : 0.7028
    P-Value [Acc > NIR] : 0.1842

                  Kappa : 0.2188

 Mcnemar's Test P-Value : 7.639e-08

            Sensitivity : 0.9371
            Specificity : 0.2432
         Pos Pred Value : 0.7455
         Neg Pred Value : 0.6207
             Prevalence : 0.7028
         Detection Rate : 0.6586
   Detection Prevalence : 0.8835
      Balanced Accuracy : 0.5902

       'Positive' Class : No
```

❖ FOR TESTING DATA

```
nb_predict_test=predict(nb,test_data)
tab2=table(nb_predict_test,test_data$DEATH_EVENT)
tab2
1 - sum(diag(tab2))/ sum(tab2)
plot(nb_predict_test~test_data$DEATH_EVENT)
confusionMatrix(nb_predict_test,test_data$DEATH_EVENT)
```

```
> nb_predict_test=predict(nb,test_data)
> tab2=table(nb_predict_test,test_data$DEATH_EVENT)
> tab2


nb_predict_test No Yes
            No  27  18
            Yes  1   4
#MISCLASSIFICATION ERROR

> 1 - sum(diag(tab2))/ sum(tab2)
[1] 0.38
```

```
> plot(nb_predict_test~test_data$DEATH_EVENT)
```

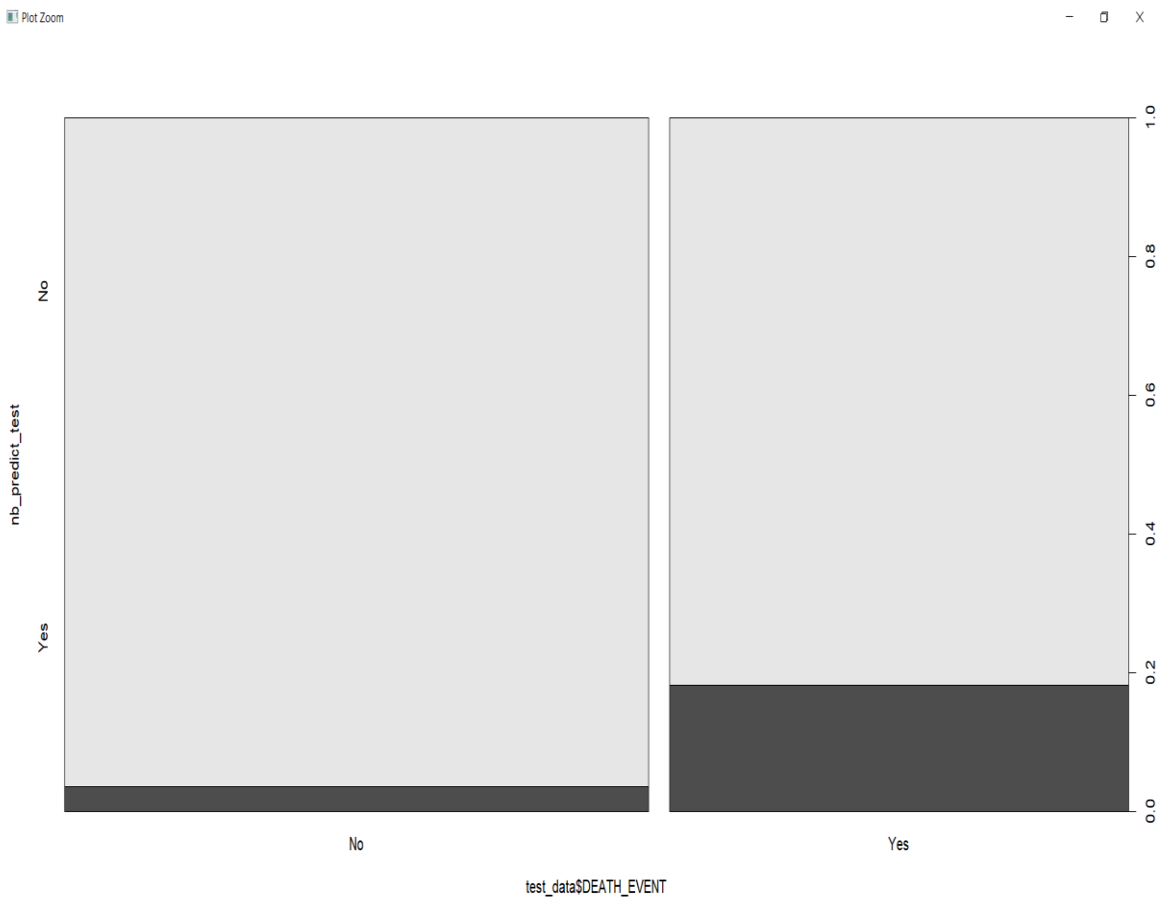


Fig- nb_predict vs test data for the death_event

```
> confusionMatrix(nb_predict_test,test_data$DEATH_EVENT)
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  27  18
       Yes  1   4

               Accuracy : 0.62
                 95% CI : (0.4717, 0.7535)
    No Information Rate : 0.56
    P-Value [Acc > NIR] : 0.2392312

                  Kappa : 0.1593

 Mcnemar's Test P-Value : 0.0002419
```

```
         Sensitivity : 0.9643
         Specificity : 0.1818
      Pos Pred Value : 0.6000
      Neg Pred Value : 0.8000
          Prevalence : 0.5600
      Detection Rate : 0.5400
Detection Prevalence : 0.9000
   Balanced Accuracy : 0.5731

    'Positive' Class : No
```

- **INFERENCE:**

From the above algorithm solution, we can conclude that naïve baies algorithm solution is successfully worked out. Following results were obtained,

- Through this, the main observation we can find is, even though there are 11 variable parameters but *age, diabetes, high_blood_pressure* plays a vital role in the decision making.
- Even though, we reduce the variables to 3, we can see that there is not much change in the probability. Hence, we can say that, one of our theoretical observations was found to be correct.
- All the other parameters bring the change in uplifting the value of the algorithms

VI. CONCLUSION

- Learning of the dataset and their implementation on the real-life problem was studied successfully.
- Various types of Algorithms and their real-world applications were studied successfully using R-studio.
- 2 types of algorithms (namely KNN and Naïve Bayes Algorithm) were successfully implemented and the outcomes were successfully studied.
- The maximum accuracy of the KNN algorithm was found to be 73.68% and that for the Naïve Baies is 73.09%.
- For the given dataset, we can say that Naïve Baies algorithm and KNN gives approximately the same accuracy for the data.
- More accurate data could be predicted good if we have chosen the best possible case for k.
- Also, for Naïve Baies the available dataset was randomly chosen, more accurate results can be found was symmetrically chosen which would have increased the accuracy.
- The prime reason that led to such low accuracy is that the no of available/past records were less, if the number of records is more, than the accuracy of the model will be increased.
- One can observe that *age*, *smoking, high_blood_pressure, diabetes* plays as a dominating role in the outcome of the result.
- Other parameters such as *creatine, sodium* a group also contribute in the outcome of the result.

--THANK YOU--