Machine Learning Algorithm using KNN

Step 1: Importing the Dataset

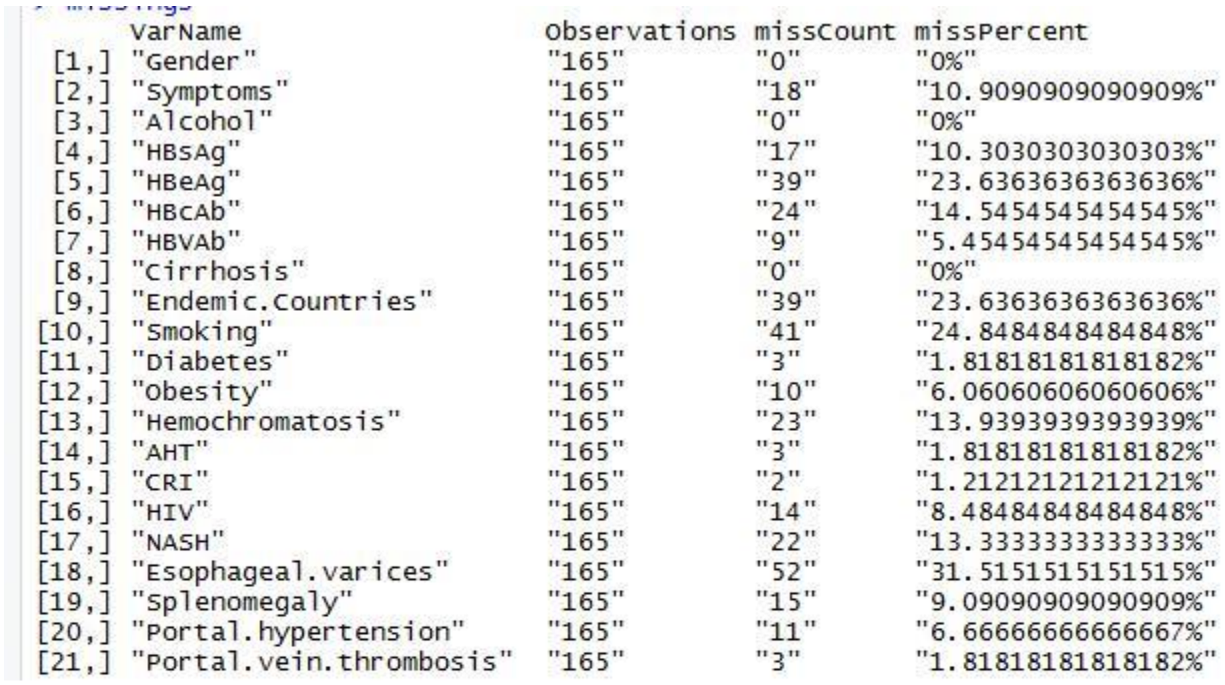
Step 2: Data Imputation

2.1 >>>Imputing '?' with 'NA'

2.2>>> Un factoring the variables for data imputation

2.3 >>>To check the missing values, and display a missing summary before data imputation.

Screenshot:



```
> missings
      VarName                     Observations missCount missPercent
 [1,] "Gender"                    "165"        "0"       "0%"
 [2,] "Symptoms"                  "165"        "18"      "10.9090909090909%"
 [3,] "Alcohol"                   "165"        "0"       "0%"
 [4,] "HBsAg"                     "165"        "17"      "10.3030303030303%"
 [5,] "HBeAg"                     "165"        "39"      "23.6363636363636%"
 [6,] "HBcAb"                     "165"        "24"      "14.5454545454545%"
 [7,] "HBVAb"                     "165"        "9"       "5.45454545454545%"
 [8,] "Cirrhosis"                 "165"        "0"       "0%"
 [9,] "Endemic.Countries"         "165"        "39"      "23.6363636363636%"
[10,] "Smoking"                   "165"        "41"      "24.8484848484848%"
[11,] "Diabetes"                  "165"        "3"       "1.81818181818182%"
[12,] "Obesity"                   "165"        "10"      "6.06060606060606%"
[13,] "Hemochromatosis"           "165"        "23"      "13.9393939393939%"
[14,] "AHT"                       "165"        "3"       "1.81818181818182%"
[15,] "CRI"                       "165"        "2"       "1.21212121212121%"
[16,] "HIV"                       "165"        "14"      "8.48484848484848%"
[17,] "NASH"                      "165"        "22"      "13.3333333333333%"
[18,] "Esophageal.varices"        "165"        "52"      "31.5151515151515%"
[19,] "Splenomegaly"              "165"        "15"      "9.09090909090909%"
[20,] "Portal.hypertension"       "165"        "11"      "6.66666666666667%"
[21,] "Portal.vein.thrombosis"    "165"        "3"       "1.81818181818182%"
```

2.4 >>> Using Central Tendency imputing the data

2.5.>>> classifying " Class.1. Year. Survival. " into factors

Step 3: Splitting of the data (training, testing and validation)

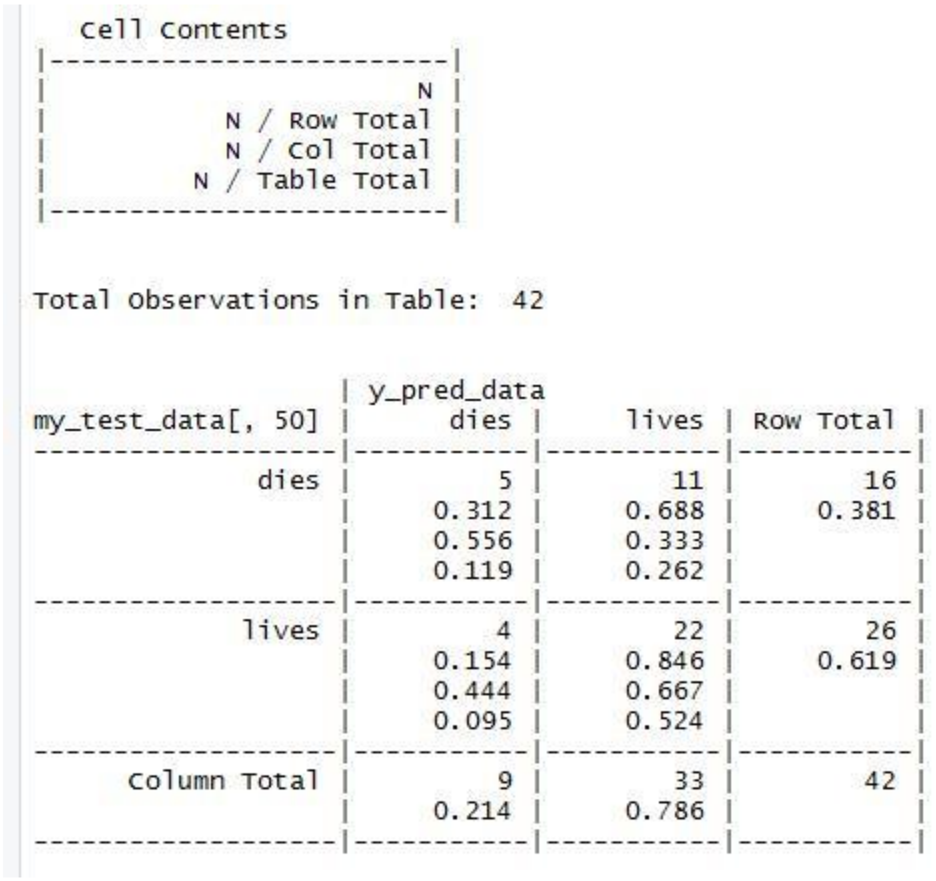
Step 4: We have used here KNN **classification algorithm** machine learning techniques for Model Development

Here The knn() function returns a factor vector of predicted labels for each of the examples in the test dataset, which has been assigned to **y_pred_data.**

Step 5 : - > Performance Measurement:-

Here in this step KNN evaluates how well the predicted classes in the y_pred_data vector match up with the known values in the **my_test_data[,50]** dataframe.

With the help of function **CrossTable()** we have created the confusion matrix.



```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  42


                  | y_pred_data
my_test_data[, 50] |      dies |     lives | Row Total |
-------------------|-----------|-----------|-----------|
              dies |         5 |        11 |        16 |
                   |     0.312 |     0.688 |     0.381 |
                   |     0.556 |     0.333 |           |
                   |     0.119 |     0.262 |           |
-------------------|-----------|-----------|-----------|
             lives |         4 |        22 |        26 |
                   |     0.154 |     0.846 |     0.619 |
                   |     0.444 |     0.667 |           |
                   |     0.095 |     0.524 |           |
-------------------|-----------|-----------|-----------|
      Column Total |         9 |        33 |        42 |
                   |     0.214 |     0.786 |           |
-------------------|-----------|-----------|-----------|
```

Above output explained below:

The cell percentages in the table indicate the proportion of values that fall into four Categories.
The top-left cell indicates the true negative results. These 5 of 42 values
are cases where the patient survival was died and the k-NN algorithm correctly identified it.

The bottom-right cell indicates the true positive results, where the classifier
and the survival determined label agree that the patient lived. A total of 22
of 42 predictions were true positives.

The cells falling on the other diagonal contain counts of examples where the k-NN approach disagreed with the true label.

The examples in the
lower-left cell are false negative results, in this case, the predicted value was died, but the patient lived.
Here 4 cases happened out of 42 cases.
Although such errors are less dangerous as they
might lead a patient to believe that one is going to die, but in reality, the patient may
continue to live. Moreover this could lead to additional financial
burden on the health care system or additional stress for the patient as
additional tests or treatment may have to be provided.

The top-right cell contain the false positive results
there were 11 cases out of 42, when the model classifies a patient as living,
but in reality, it was that patient died.
Although such errors are very dangerous than a false negative result, they should also be avoided as
this causes the patient to die due to wrong prediction.

Performance:

Measure of Performance:

The **sensitivity** of a model (also called the true positive rate) measures the
proportion of positive examples that were correctly classified.
Sensitivity=TP/TP+FN
sensitivity ##0.8148148

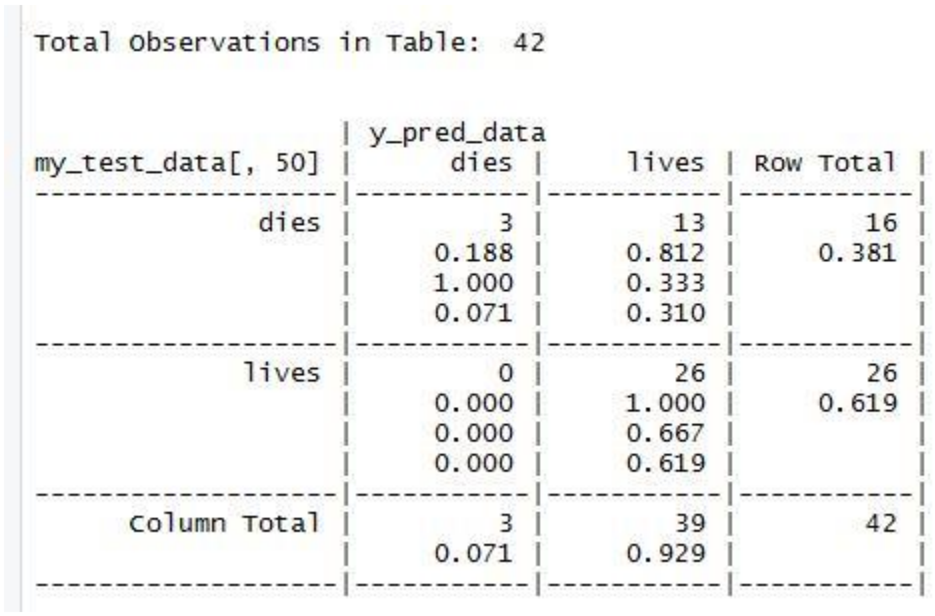
The **specificity** of a model (also called the true negative rate) measures
the proportion of negative examples that were correctly classified.
Specificity = TN/TN+FP
Specificity 0.2666667

The **Accuracy**:-
The accuracy is (TP + TN) / (TP + TN + FP + FN)
64 % is the accuracy

Step 5.1:- Now we have to decrease **false positive. So, performing the Z scale standardization**

Step 5.2 : Again did the data splitting after performing the Z scale standardization over the HCC dataset. Created the Model and checked the Model performance of KNN

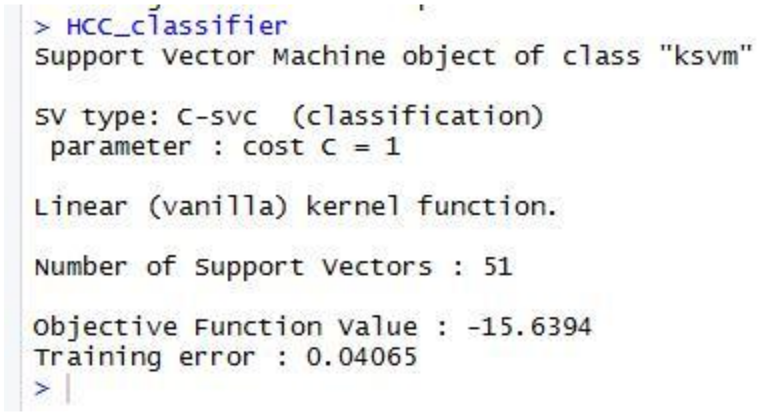
So, after this process Accuracy increased to 69%



```
Total Observations in Table:  42


                   | y_pred_data
my_test_data[, 50] |      dies |     lives | Row Total |
-------------------|-----------|-----------|-----------|
              dies |         3 |        13 |        16 |
                   |     0.188 |     0.812 |     0.381 |
                   |     1.000 |     0.333 |           |
                   |     0.071 |     0.310 |           |
-------------------|-----------|-----------|-----------|
             lives |         0 |        26 |        26 |
                   |     0.000 |     1.000 |     0.619 |
                   |     0.000 |     0.667 |           |
                   |     0.000 |     0.619 |           |
-------------------|-----------|-----------|-----------|
      Column Total |         3 |        39 |        42 |
                   |     0.071 |     0.929 |           |
-------------------|-----------|-----------|-----------|
```

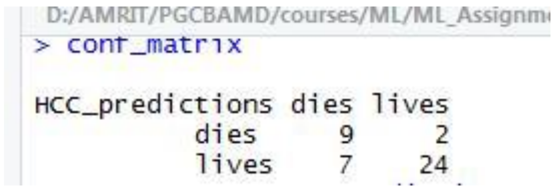
SVM:

Step 6:
SVMs use a boundary called a hyperplane to partition data into
groups of similar class values, In two dimensions, the task of the SVM algorithm is to identify a line that separates the two classes SVM uses Maximum Margin Hyperplane (MMH) creates the greatest separation between the two classes.



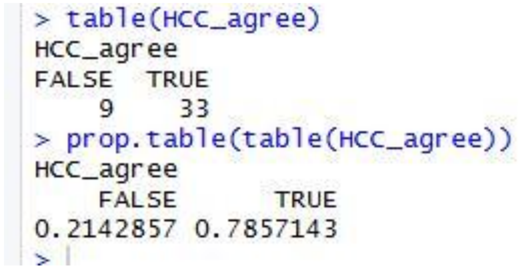
```
> HCC_classifier
Support Vector Machine object of class "ksvm" 

SV type: C-svc  (classification) 
 parameter : cost C = 1 

Linear (vanilla) kernel function. 

Number of Support Vectors : 51 

Objective Function Value : -15.6394 
Training error : 0.04065 
> 
```

Step 6.1: checking the Performance using Confusion Matrix:



```
D:/AMRIT/PGCBAMD/courses/ML/ML_Assignm
> conf_matrix

HCC_predictions dies lives
          dies     9     2
          lives    7    24
```

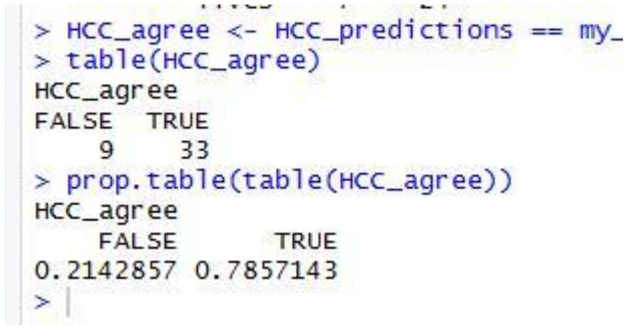
Now percentage of correction: 78 %



```
> table(HCC_agree)
HCC_agree
FALSE  TRUE
    9    33
> prop.table(table(HCC_agree))
HCC_agree
    FALSE      TRUE
0.2142857 0.7857143
>
```

Step 7:- Improving the Accuracy using kernel as Radial

So using Radial we are getting Accuracy of 78 %



```
> HCC_agree <- HCC_predictions == my_
> table(HCC_agree)
HCC_agree
FALSE  TRUE
    9    33
> prop.table(table(HCC_agree))
HCC_agree
    FALSE      TRUE
0.2142857 0.7857143
>
```