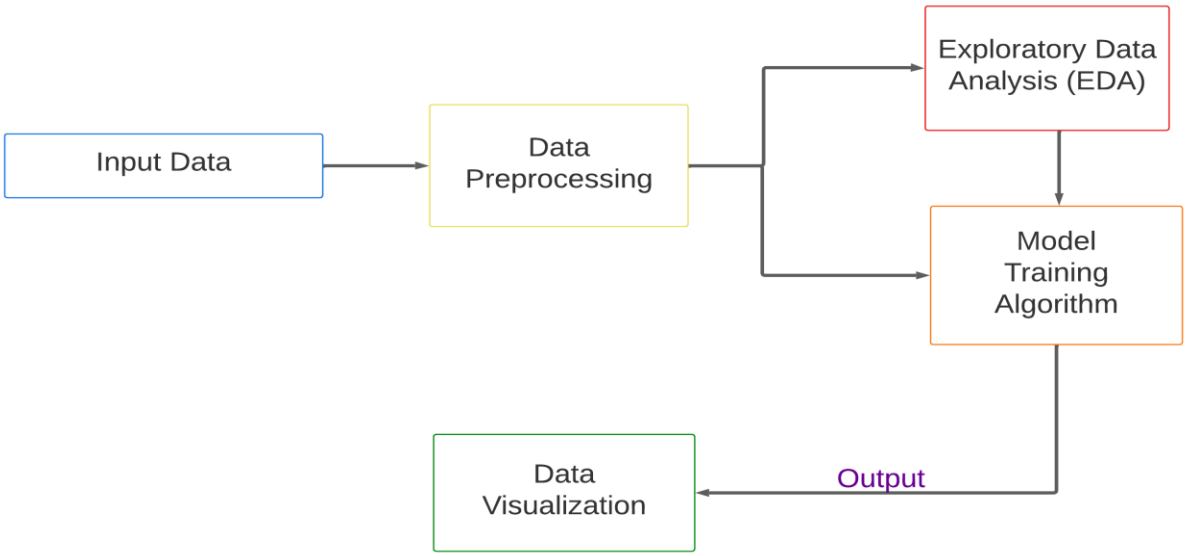
Subject: Data Intensive Computing
Subject Code: CSE 4/587
Professor: Eric Mikida
School of Engineering and Applied Sciences
University at Buffalo, Buffalo, NY 14260

Project Phase 2

PROJECT TITLE	**DATE**
Analysis and Prediction of stroke in Human body	11/14/2022

PROJECT TEAM MEMBERS	
Briyana Rana	UBIT Number: 50442498 UB Name: brana
Yashesh Pandya	UBIT Number: 50442549 UB Name: yasheshp



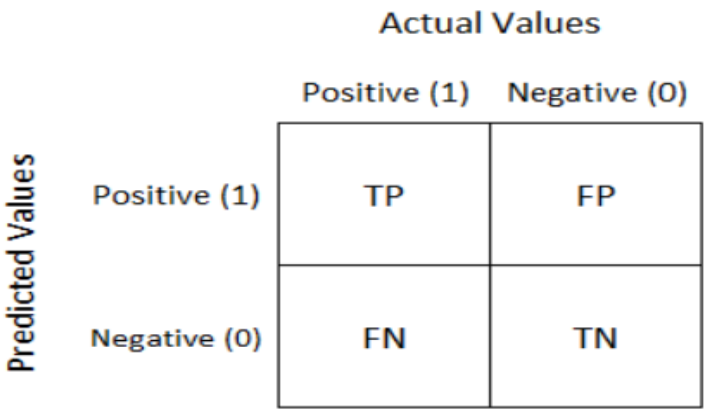
- Prediction of stroke in human body is a classification problem which will predict 1 if the person with certain dependent attributes can have stroke in near future and 0 if he/she is safe from stroke. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. Methods like Support Vector Machine, Logistic Regression, Random Forest Classifier, Decision Tree and K-Nearest Neighbors, Naïve Bayes, Gaussian Mixture Model etc are used for classification. Hence our aim is to create awareness among patients regarding the possibility of stroke in their body. To achieve our goal, we have implemented top 5 popular methods like Support Vector Machine, Logistic Regression, Random Forest Classifier, Decision Tree and K-Nearest Neighbors.

❖ Metrices in project:-

- ❖ Confusion matrix
- ❖ Recall score
- ❖ F1 score
- ❖ Precision score
- ❖ Accuracy score
- ❖ ROC AUC score

➢ Confusion matrix:-

- An N x N matrix called a confusion matrix is used to assess the effectiveness of a classification model, where N is the total number of target classes.
- The matrix contrasts the actual target values with the ones that the machine learning model anticipated.



➢ Recall Score:-

- Recall score is used to assess how well a model performs in terms of accurately counting true positives among all real positive values.
- When the classes are severely unbalanced, the precision-recall score is a good indicator of the accuracy of predictions.
- The formula to find the recall score is:

tp / (tp + fn)

- Here tp is the number of true positives and fn the number of false negatives.
- The best value is 1 and the worst value is 0.

➢ F1 Score:-

- One of the most crucial assessment measures in machine learning is the F1-score.
- By combining accuracy and recall, two measures that would normally be in competition, it elegantly summarizes the prediction ability of a model.
- The formula to find f1 score is:

$$F = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}}.$$

- Here tp is number of true positives, fp is number of false positives, fn is number of false negatives.

➢ Precision score:-

- A measure of precision counts how many correctly positive forecasts were made. Therefore, precision determines the accuracy for the minority class.
- The formula to find the precision score is:

Precision = TruePositives / (TruePositives + FalsePositives)

- It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that were predicted.

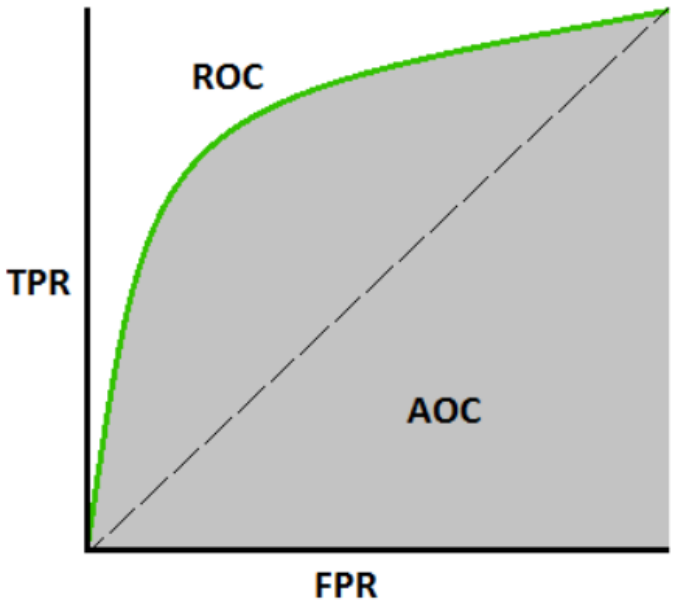
➢ Accuracy score:-

- The accuracy score is used to evaluate the model's effectiveness by calculating the ratio of total true positives to total true negatives across all made predictions.

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

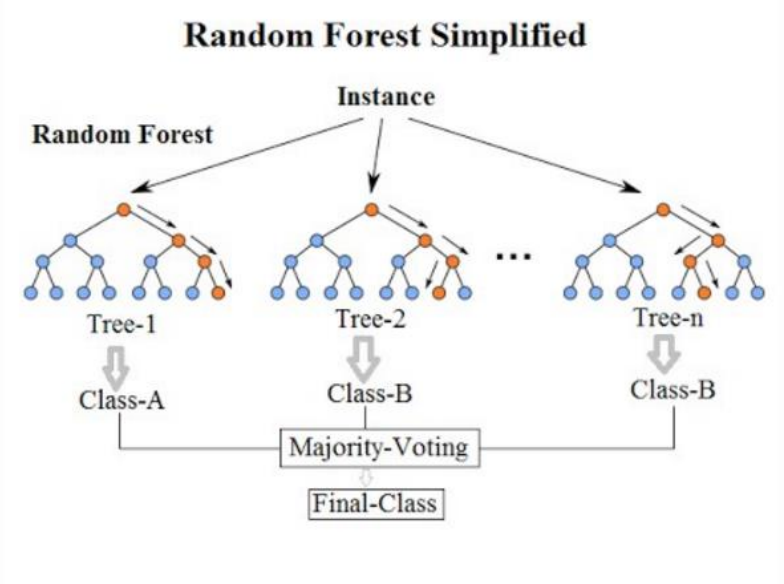
➢ ROC-AUC Score:-

- AUC - ROC curve is a performance measurement for the classification problems at various threshold settings.
- AUC stands for the level or measurement of separability, while ROC is a probability curve. It reveals how well the model can differentiate across classes.



1. Random Forest:-

- A classification system made up of several decision trees is called the random forest.
- When constructing each individual tree, it employs bagging and feature randomization in an effort to produce an uncorrelated forest of trees whose forecast by committee is more accurate than that of any individual tree.



- The mean squared error (MSE), which measures how your data branches from each node, is used when utilizing the Random Forest Algorithm to solve regression issues.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (fi - yi)^2$$

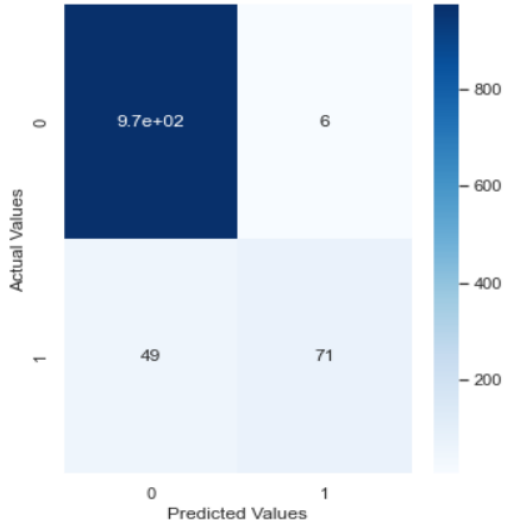
- Here N is the number of data points, fi is the value returned by the model and yi is the actual value for data point i.
- To determine which branch is the best choice for your forest, this algorithm estimates the distance between each node and the expected actual value. In this case, fi is the value the decision tree returned, and yi is the value of the data point you are testing at a particular node.

❖ Why this model:-

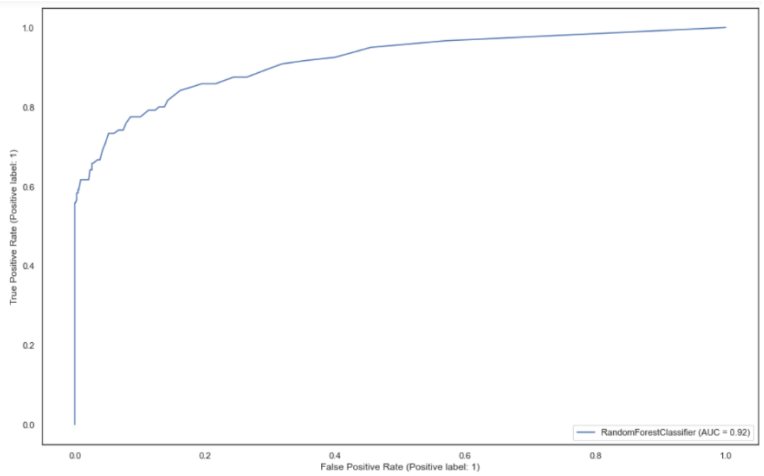
- The bias-variance trade-off is often well balanced by Random Forests, which also offer best accuracy.
- Both linear and non-linear connections can be handled successfully by Random Forests.
- Random Forests produce uncorrelated decision trees and execute feature selection implicitly. To do this, it constructs each decision tree using a random collection of characteristics. This makes it a fantastic model for working with data that has a lot of different properties.

❖ Interpretation:-

➢ Confusion matrix:-



➢ ROC- AUC curve plot:-



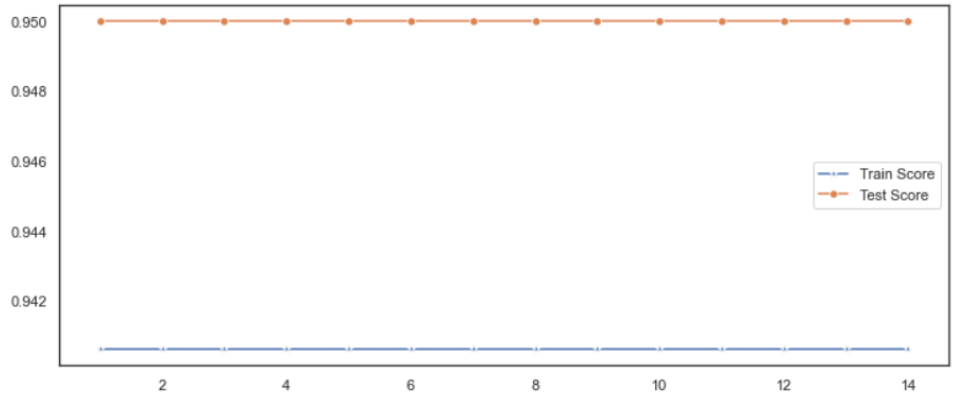
❖ Result:-

➢ Train test accuracy:-

```
y_pred_train_rf = rf.predict(X_train)
train_accuracy = accuracy_score(y_train, y_pred_train_rf)

y_pred_test_rf = rf.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred_test_rf)
print("Test accuracy_score: " + str(accuracy_score(y_test,y_pred_test_rf)))
print("Train accuracy_score: " + str(accuracy_score(y_train,y_pred_train_rf)))
```

```
Test accuracy_score: 0.95
Train accuracy_score: 1.0
```



- **Precision score:-**

```
from sklearn.metrics import precision_score
precision_score(y_test,y_pred_test_rf)
```

```
0.922077922077922
```

- **Recall score:-**

```
from sklearn.metrics import recall_score
recall_score(y_test,y_pred_test_rf)
```

```
0.5916666666666667
```

- **F1 score:-**

```
f1_score(y_test,y_pred_test_rf)
```

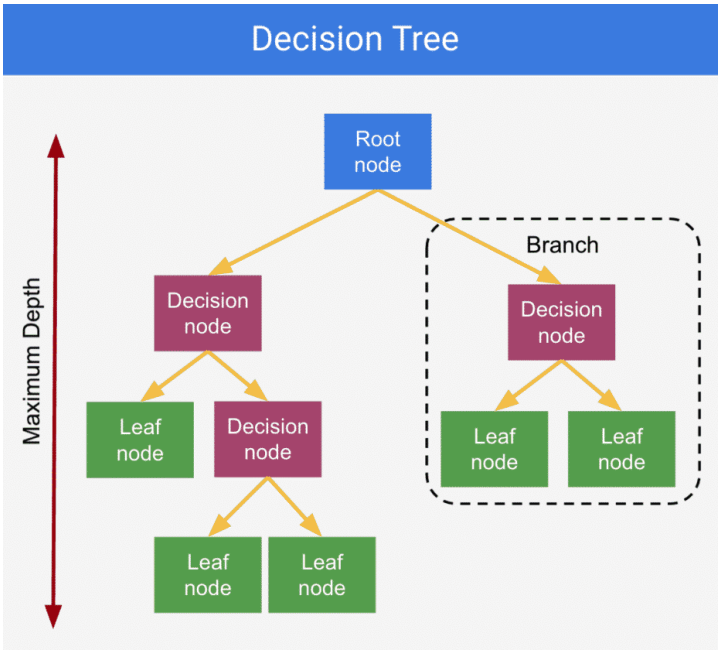
```
0.7208121827411168
```

- **ROC AUC score:-**

- As observed above, the True positives and True negatives are predicted with majority. And the model has very high precision score and AUC score which indicates good accuracy and high sensitivity and high specificity.

2. Decision Tree:-

- The Decision Tree is a Machine Learning Algorithm that has to ability to deal with both categorical and continuous data that can be used to predict regression and classification values. It is a binary tree with the root node and leaf nodes. The tree starts from the root node and moves down the tree based on the if-else condition.
- By the fig. we can see the root node, branches, internal nodes, and leaf nodes make up its hierarchical tree structure.



- There are various algorithms used to develop the decision tree namely.
  a. CART
  b. ID 3
  c. CHAID
  d. ID 4.5
- The main parameters that play an important role in developing the tree are Entropy and Gini Index. Entropy describes the amount of information needed to accurately describe data. So, if data is homogenous then entropy is 0 (that is pure), else if elements are equally divided then entropy move towards 1 (that is impure).

$$\text{Entropy} = -\sum_{i=1}^{n} p_i * \log(p_i)$$

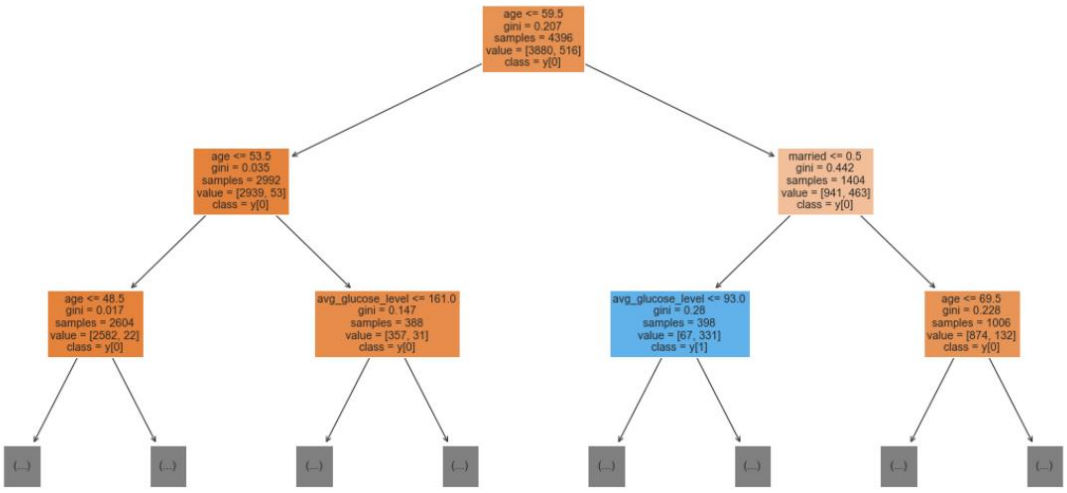
- The Gini index of value 0 means samples are perfectly homogeneous, Gini index of value 1 means maximal inequality among elements. It is the sum of the square of the probabilities of each class.

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} p_i^2$$

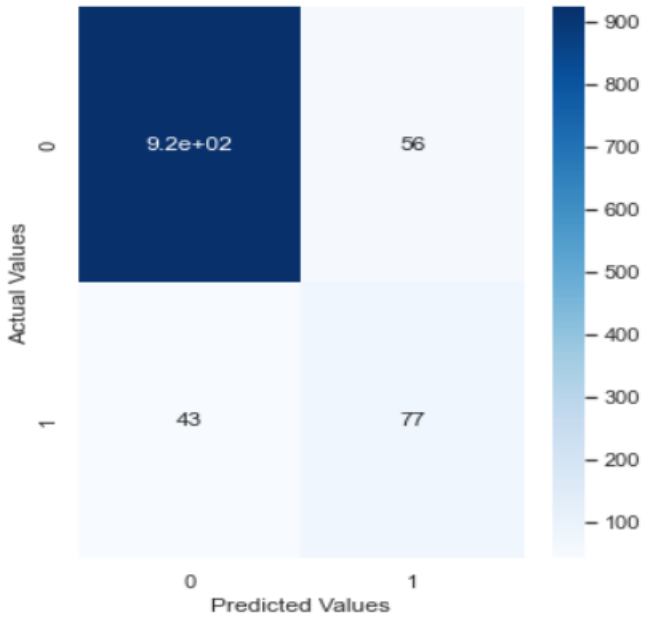
❖ **Why this model:-**

- Data normalization is not necessary for a decision tree.
- Scaling of data is not necessary when using a decision tree.
- Furthermore, the process of developing a decision tree is not significantly impacted by missing values in the data.

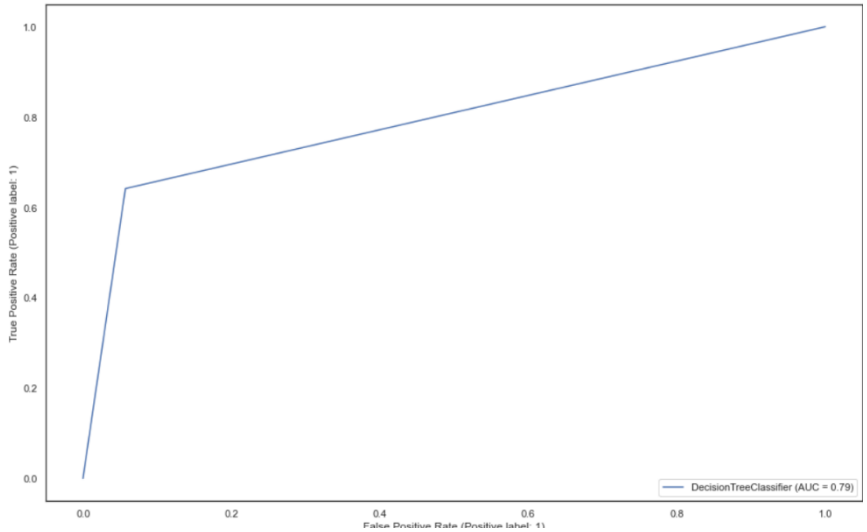
❖ **Interpretation:-**



➢ **Confusion Matrix:-**



➢ **ROC-AUC curve plot:-**

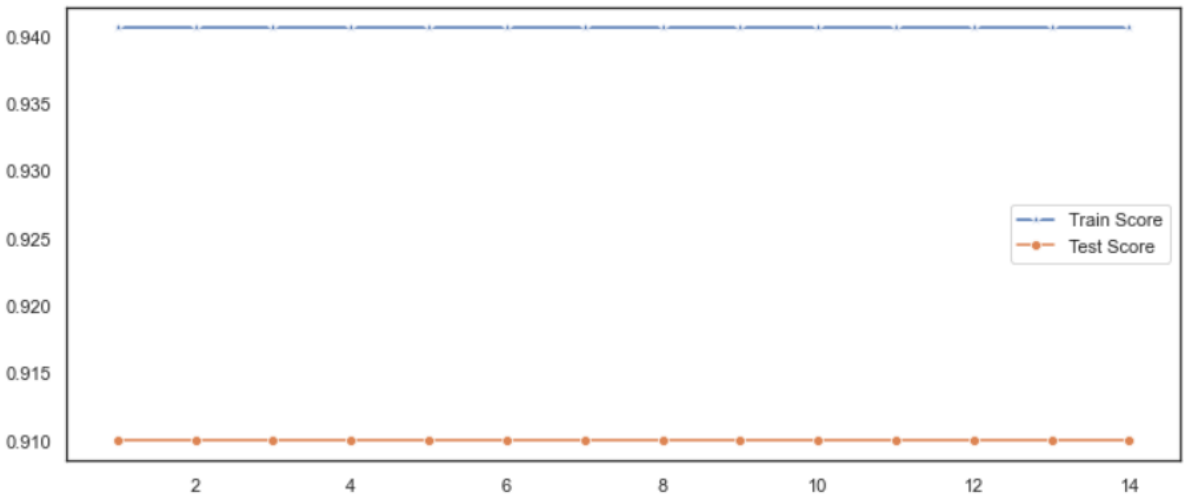


❖ Result:-

➢ Train test accuracy:-

```
model = DecisionTreeClassifier(max_depth= 20)
model.fit(X_train, y_train)
predsdtc = model.predict(X_test)
predsdtc1 = model.predict(X_train)
print("Test accuracy_score: " + str(accuracy_score(y_test, predsdtc)))
print("Train accuracy_score: " + str(accuracy_score(y_train, predsdtc1)))
```

```
Test accuracy_score: 0.91
Train accuracy_score: 1.0
```



➢ Precision score:-

```
precision_score(y_test,predsdtc)
```

```
0.5789473684210527
```

➢ Recall score:-

```
recall_score(y_test,predsdtc)
```

```
0.6416666666666667
```

- **F1 score:-**

```
print("f1_score: " + str(f1_score(y_test, predsdtc)))
```

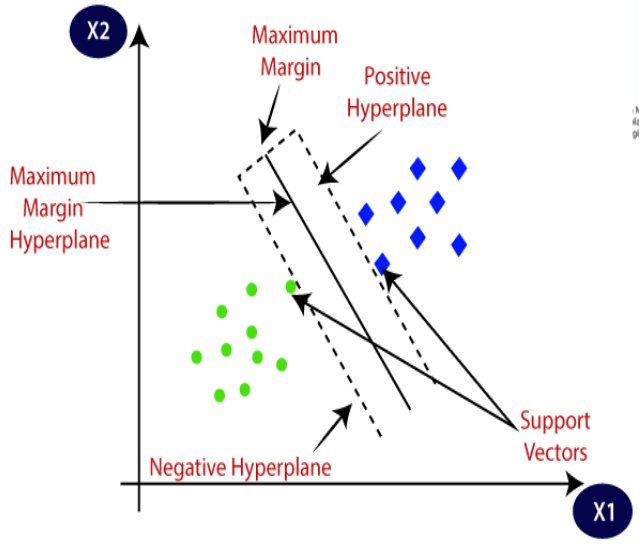
```
f1_score: 0.6086956521739131
```

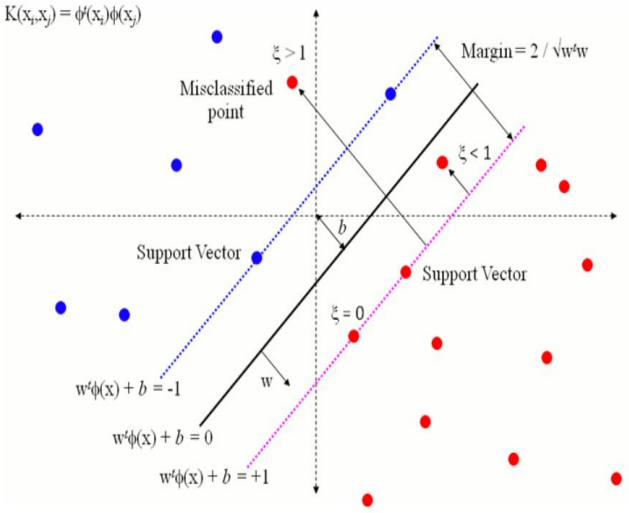
- **ROC AUC score:-**

- As observed above, the True positives and True negatives are predicted with majority. And the model has good precision score and AUC score is 0.79 which indicates it to be decent model. Also it has good recall and F1 score.

3. Support Vector Machine (SVM):-

- A supervised machine learning approach called Support Vector Machine (SVM) is utilized for both classification and regression.
- The object behind SVM is to find the hyperplane which maximizes the margin and differentiates between the classes. If the margin is small, more are chance for points to be misclassified.





- The central plane wT x + b = 0 separates the data points. Above the plane wT x + b = 1 are the positive data points and below the plane wT x + b = -1 are the negative points. Assuming π(+) be positive points and π(-) be negative points. Let the distance between the π(+) and π(-) be 'r'

$$r = \frac{2}{\|w\|}$$

- Assuming $\pi(+)$ and $\pi(-)$ are equidistant from each other. If the data is completely linear separable then Hard Margin SVM comes into the picture, so the optimization problem now becomes

$$\max(w) \{\frac{2}{\|w\|}\}$$

- When the data is non-linearly separated then a Soft Margin comes into the picture, so the optimization problem now becomes:

$$\max(w) \{\frac{2}{\|w\|}\} \text{ such that } y_i * (w^T x_i + b) >= 1$$

- In short, A hard-margin can be utilized when classes can be completely separated linearly. If not, a soft-margin is necessary.
- If there are no points in the margin, then we can say no points are violating the margin then the above mathematical formulation applies for hard max but if the point is misclassified that is if the point is on the opposite side of the maximizing hyperplane or points lie inside the margin area then soft max mathematical formulation is used. If the data points are misclassified, then we calculate the loss term and minimize the error. So, the final optimal constrained equation becomes:

$$\text{Minimize } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i, \xi_i \geq 0$$

- The kernel in SVM has the ability to take the data points in X space in the d dimension and transform the points in Z space in the d T dimension. The kernels can be of different types Linear, Non-Linear, Polynomial, radial basis function, and sigmoid.

Name	**Formula**
Radial Basis Function	$K(x, x_i) = \exp(-\text{gamma} * \text{sum}((x - x_i^2)))$
Polynomial	$K(x, x_i) = 1 + \text{sum}(x + x_i)^d$
Sigmoid	$K(x, x_i) = \tanh(\alpha * x_i * x_{j+} \beta)$
Linear	$K(x, x_i) = \text{sum}(x * x_i)$

- Radial Basis Function Kernel: It is known to have good performance with certain parameters and the result of the training have a small error compared to another kernel.
- Polynomial Kernel: It is only used to solve classification problems on the normalized dataset. The 'd' parameter is the degree of the kernel. The greater the value of d the

less resulting accuracy will be fluctuating and be less stable (the higher the d value more the hyperplane curves).

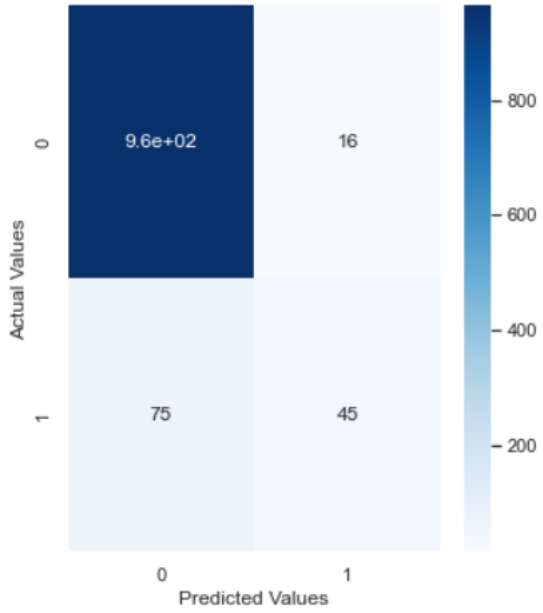
- Linear Kernel: This kernel normally uses the dot product of the observations.

❖ Why this model:-

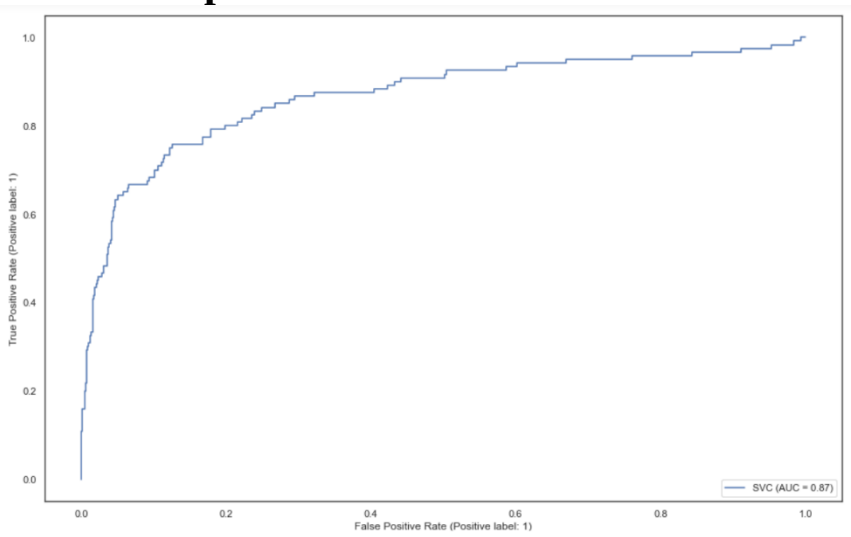
- In high dimensional spaces, it works well.
- In situations where there are more dimensions than samples, it is effective.
- It is also memory efficient since it only needs a small fraction of training points for the decision function (known as support vectors).

❖ Interpretation:-

➢ Confusion matrics:-



➢ ROC-AUC curve plot:-

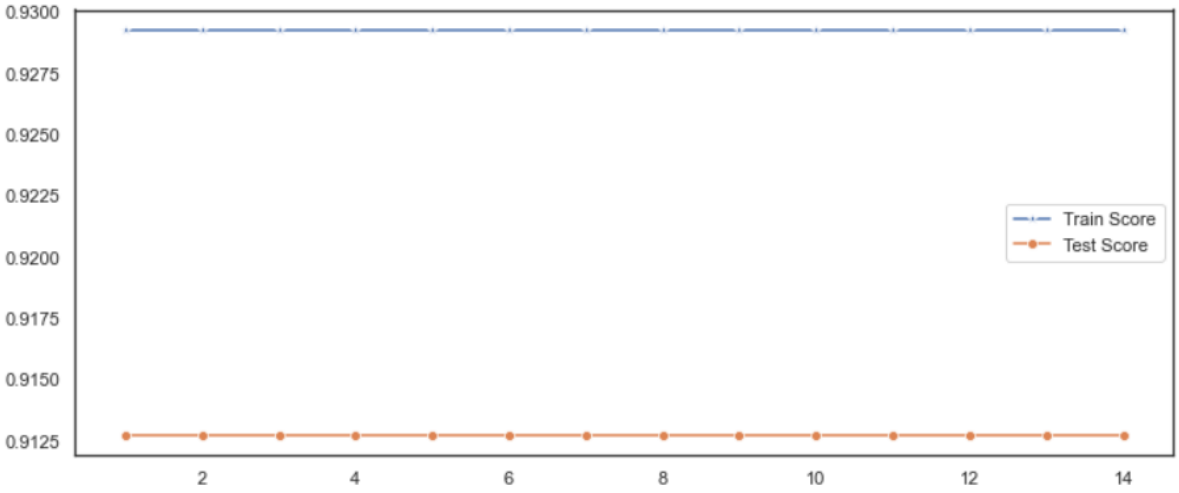


❖ Result:-

- **Train test accuracy:-**

```
print("Test accuracy_score: " + str(accuracy_score(y_test, y_pred)))
print("Train accuracy_score: " + str(accuracy_score(y_train, y_pred_train)))
```

```
Test accuracy_score: 0.9172727272727272
Train accuracy score: 0.9360782529572339
```



- **Precision score:-**

```
precision_score(y_test,y_pred)
```

```
0.7377049180327869
```

- **Recall score:-**

```
recall_score(y_test,y_pred)
```

```
0.375
```

- **F1 score:-**

```
f1_score(y_test,y_pred)
```

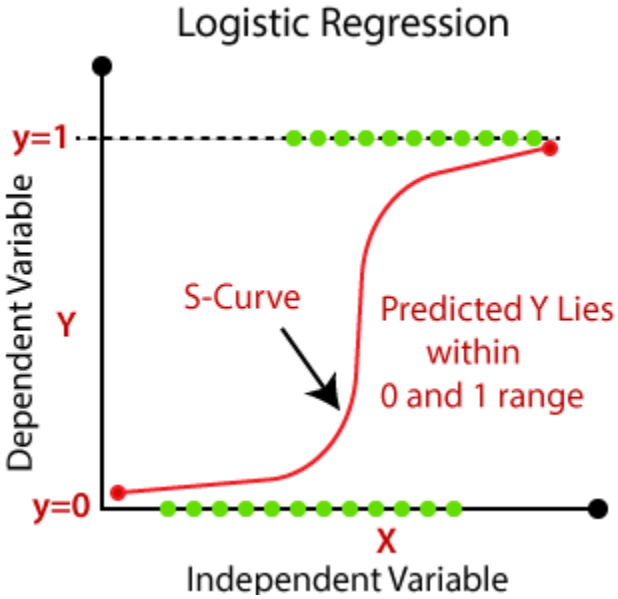
```
0.4972375690607734
```

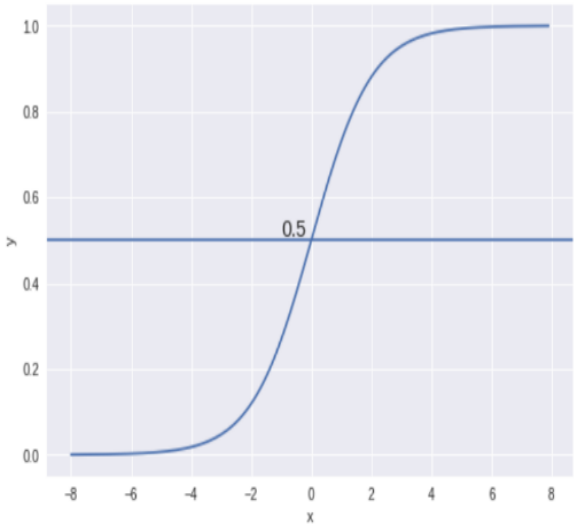
- **ROC AUC score:-**

- As observed above, the True positives and True negatives are predicted with majority. And the model has good precision score, and AUC score is 0.87 which indicates it to be good model. But it has low F1 score and Recall Score.

4. Logistic Regression:-

- Logistic Regression is a Machine Learning Algorithm used basically to deal with classification problems. It makes predictions based on the concept of probability. The cost function used by Logistic regression is termed the Sigmoid function. The hypothesis of the algorithms limits the sigmoid function between 0 and 1 hence its predictions are based on probability.
- Logistic regression is a kind of regression analysis and a frequently employed approach for dealing with binary classification issues.





- Logistic Regression hypothesis expectation: $0 <= h_\vartheta(x) <= 1$.
- Sigmoid Function: $f(x) = 1/(1+e^{(-x)})$
- As we expect our hypothesis to be between 0 and 1 we use the following formula:

$$Z = B_0 + B_1 X$$
$$h_\vartheta(x) = \text{sigmoid}(Z)$$
$$h_\vartheta(x) = 1 / (1 + e^{-(B_0\{ + B_1 x)})$$

- Logistic Regression uses decision boundaries to classify the data into different categories. In this case, the decision boundary is a linear line that differentiates between the classes. Suppose we have to categorize between Cat and a Dog, we take the boundary condition as p = 0.5, and if the p >= 0.5 we can say it is Cat and if p<= 0.5 we can categorize it as a Dog.
- The cost function is an important criterion to get minimize the error and get a better fit. So the cost function for Logistic regression is as follows:

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h\theta(x)) \text{ if } y = 1 \\ -\log(1 - h\theta(x)) \text{ if } y = 1 \end{cases}$$

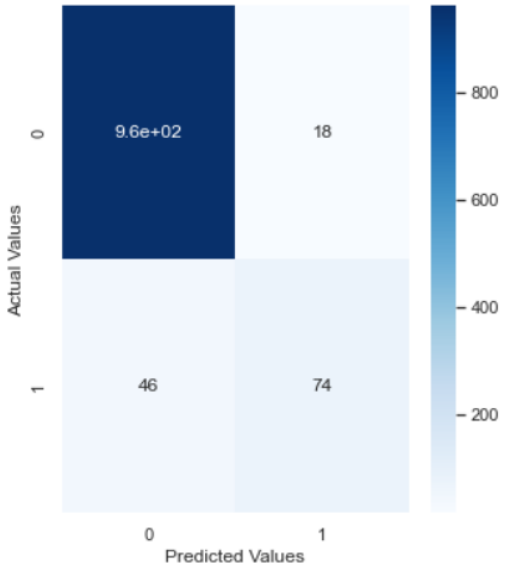
- We cannot apply the cost function of linear regression to logistic regression because we will end with a non-convex graph with multiple local minimums, and it would be difficult to minimize them all and get a maximum value.

❖ Why this model:-

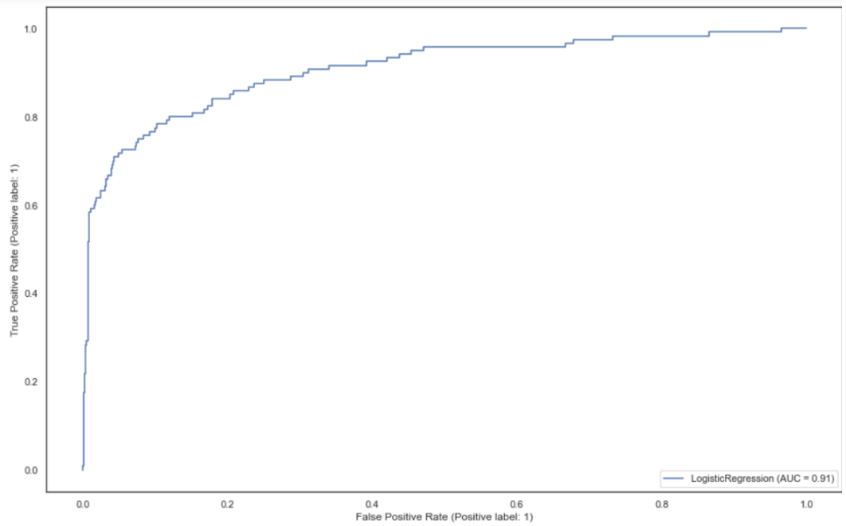
- The training of logistic regression is very effective and easy to execute and analyze.
- It classifies unfamiliar records fairly quickly.
- With multinomial regression, it is simple to add more classes, and class predictions are naturally seen from a probabilistic perspective.

❖ Interpretation:-

➢ Confusion matrix:-



➢ ROC-AUC curve plot:-

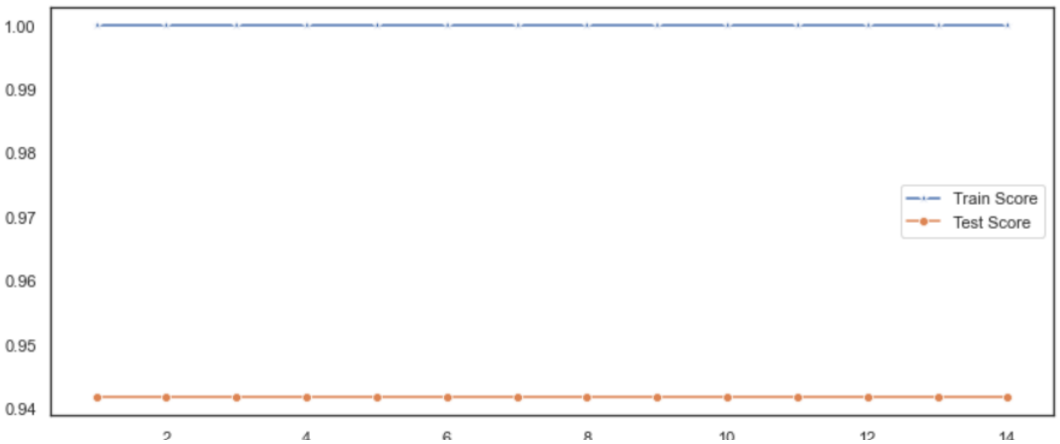


❖ Result:-

- **Train test accuracy:-**

```
print("Test accuracy_score: " + str(accuracy_score(y_test, y1_pred)))
print("Train accuracy_score: " + str(accuracy_score(y_train, y_train_pred1)))
```

```
Test accuracy_score: 0.9418181818181818
Train accuracy_score: 1.0
```



- **Precision score:-**

```
from sklearn.metrics import precision_score
precision_score(y_test,y1_pred)
```

```
0.8043478260869565
```

- **Recall score:-**

```
from sklearn.metrics import recall_score
recall_score(y_test,y1_pred)
```

```
0.6166666666666667
```

- **F1 score:-**

```
f1_score(y_test,y1_pred)
```

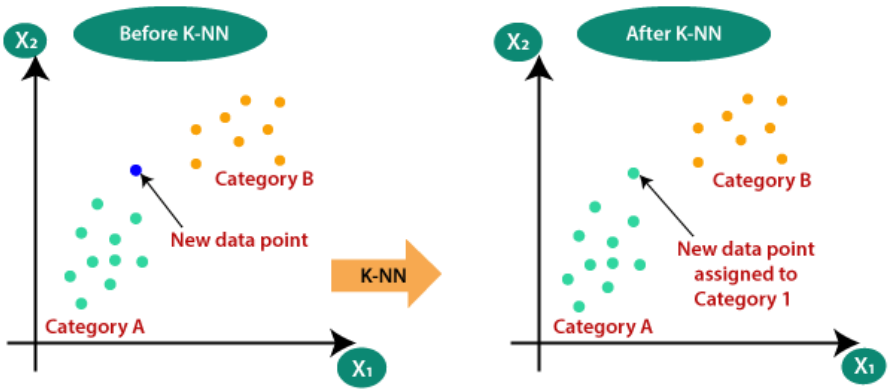
```
0.6981132075471699
```

- **ROC AUC score:-**

- As observed above, the True positives and True negatives are predicted with majority. And the model has good precision score, and AUC score is 0.91 which indicates it to be so good model. Also, it has good recall and F1 score.

5. KNN:-

- K-Nearest Neighbors, or KNN, is an acronym. It is an algorithm for supervised learning. In other words, we train it under close observation.
- Because it produces precise predictions, the KNN algorithm can compete with the most accurate models. The distance measurement affects how accurate the projections are.



- To calculate the KNN we use the 4 different method to find the distance:
  1. Euclidean distance
  2. Manhattan distance
  3. Jaccard distance
  4. Hamming distance

❖ Euclidean distance:-

- Euclidean distance the most commonly used method to find the distance. The formula is given below:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

- It actually measures the straight line between two points.

❖ Manhattan distance:-

- This is yet another widely used metric for measuring distance since it calculates the absolute value between two places.
- The formula is:

$$\text{Manhattan Distance} = d(x,y) = \left(\sum_{i=1}^{m} |x_i - y_i|\right)$$

❖ Jaccard distance:-

- Jaccard can produce inaccurate findings and is quite sensitive to small sample sizes, especially when there are few or no observations in the data set.
- The formula for Jaccard index is:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

- By deducting the Jaccard Index from 100%, one may find the complement of the Jaccard Index, or the Jaccard distance, which has the following formula:

$$D(A,B) = 1 - J(A,B)$$

❖ Hamming distance:-

- This method helps to locate the locations where two vectors do not match and is commonly used with string or Boolean vectors. Because of this, it is also known as the overlap metric. This may be modeled using the formula below:

$$\text{Hamming Distance} = D_H = \left(\sum_{i=1}^{k} |x_i - y_i|\right)$$

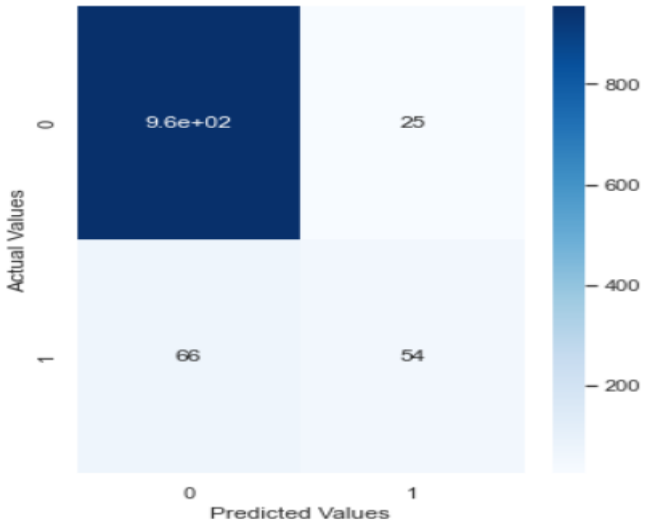
$$x = y \qquad D = 0$$
$$x \neq y \qquad D \neq 1$$

❖ Why this model:-

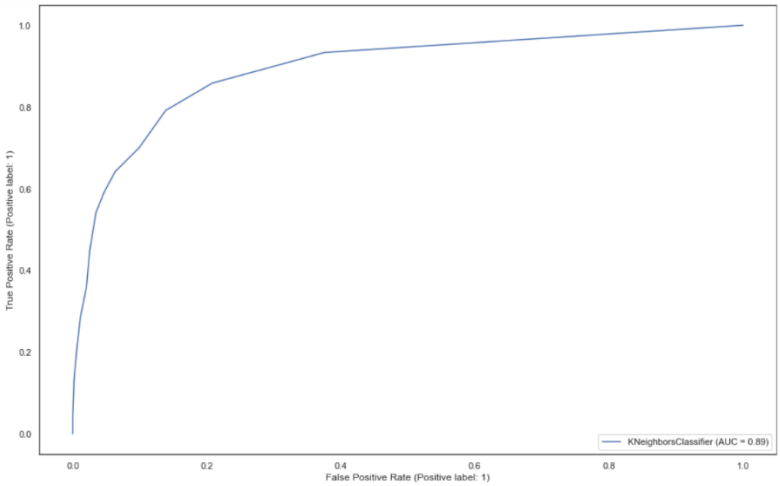
- Since the KNN algorithm doesn't need to be trained before producing predictions, fresh data may be supplied without disrupting the system's accuracy.
- A memory-based method is k-NN. As we add more training data, the classifier adjusts right away. In real-time application, it enables the algorithm to react swiftly to changes in the input.

❖ Interpretation:-

- **Confusion matrix:-**



- **ROC-AUC curve plot:-**

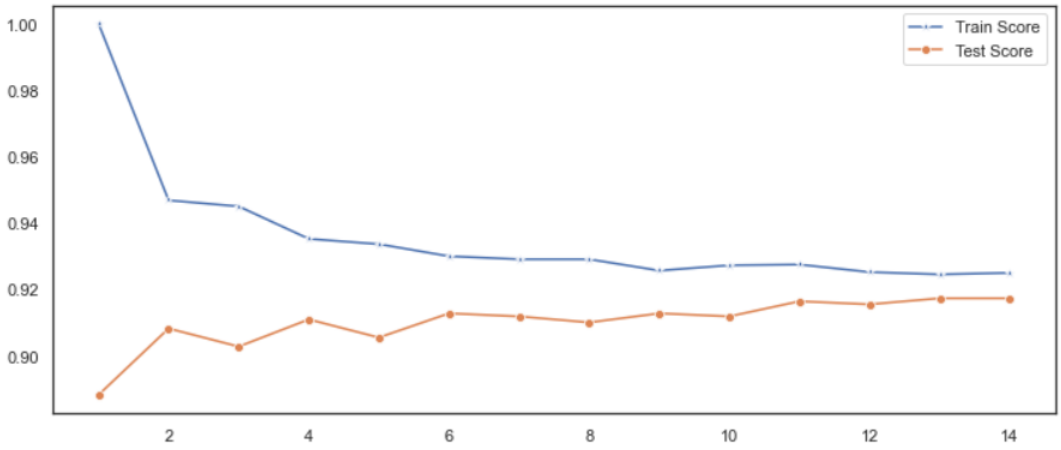


❖ Result:-

- **Train test accuracy:-**

```
train_accuracy = max(train_scores)
train_scores_ind = [i for i, v in enumerate(train_scores) if v == train_accuracy]
print('Train Accuracy {} % and k = {}'.format(train_accuracy*100,list(map(lambda x: x+1, train_scores_ind))))
max_test_score = max(test_scores)
test_scores_ind = [i for i, v in enumerate(test_scores) if v == max_test_score]
print('Test Accuracy {} % and k = {}'.format(max_test_score*100,list(map(lambda x: x+1, test_scores_ind))))
```

```
Train Accuracy 100.0 % and k = [1]
Test Accuracy 91.72727272727272 % and k = [13, 14]
```



- **Precision score:-**

```
precision_score(y_train,Y_pred_train)
```

```
0.6910112359550562
```

- **Recall score:-**

```
recall_score(y_test,Y_pred)
```

```
: 0.45
```

- **F1 score:-**

```
f1_score(y_test,Y_pred)
```

```
]: 0.5427135678391959
```

- **ROC AUC score:-**

- As observed above, the True positives and True negatives are predicted with majority. And the model has good precision score, and AUC score is 0.89 which indicates it to be so good model. But it has poor recall and F1 score.

❖ **Conclusion:-**

Result model	Accuracy score	Precession score	Recall score	F1 score	ROC-AUC score
Random forest classifier	0.95	0.92	0.59	0.72	0.92
Decision Tree	0.91	0.57	0.64	0.60	0.79
Support Vector Machine	0.91	0.73	0.37	0.49	0.87
Logistic Regression	0.94	0.80	0.61	0.69	0.91
KNN classifier	0.91	0.69	0.45	0.54	0.89

- As derived from the above table, Random Forest Classifier is giving highest accuracy score and AUC score which other metrics as compared to other classifiers. Hence we deployed Random Forest Classifier into our model to predict whether target attribute i.e if the patient will have stroke or not.