

MINOR PROJECT REPORT
(SWE5302)
ON
PREDICTING HEART DISEASE USING BASE LEARNING
ALGORITHMS AND ENSEMBLE CLASSIFIER



Under the guidance of
Dr. Ruchika Malhotra
(HOD, SE, DTU)

Department of Computer Science
Delhi Technological University, Delhi

Submitted By:
Suman Nandi
(2k20/SWE/23)

TABLE OF CONTENTS

Sr. No	Chapters	Page No.
1.	Abstract	2
2.	Introduction	3
3.	Python Libraries Used	6
4.	Overview of Dataset Used	7
5.	Feature Selection	10
6.	Dataset Splitting	11
7.	Base learners Used	12
8.	Ensemble Classifiers Used	19
9.	Complete Source Code	27
10.	Conclusion	44
11.	References	45

ABSTRACT

In recent times heart related diseases are very common and are one of the main reasons for a huge number of deaths in the world over the last few decades and has emerged as the most life-threatening disease, all over the whole world. Several machine Learning algorithms and techniques have been applied to various medical datasets in order to automate the analysis of large and complex data. Advancement in the studies of Machine Learning algorithms and techniques have made possible to automate various medical datasets and analyze large and complex data. These algorithms identify the patterns very quickly and help the health care industry and professionals in the quick diagnosis of heart related diseases.

In this innovative project we will be depicting three main models namely **Logistic Regression**, **Decision tree** and **Multinomial Naïve Bayes** to analyze their performance. We will be also using ensemble learning techniques namely **bagging**, **Boosting** and **Stacking** in-order to boost the accuracy score of the machine learning models used.

INTRODUCTION

Advancement of machine learning helps in presenting remarkable features by the help of which patterns are identified very quickly which in turn gives the doctors ample time for planning and providing care to the patients.

In my project, I have employed three base learning algorithms namely Multinomial Naïve Bayes, Logistic Regression (LR), Decision Tree (DT) to predict the presence of heart disease. Along with this I have used ensemble classifiers i.e., bagging, boosting and stacking. We have taken various parameters into account such as Accuracy, Precision, Recall and F1-score in order to analyze the performance and a comparative study of the various models used.

MULTINOMIAL NAÏVE BAYES

Multinomial Naïve Bayes algorithm is based on probabilistic learning method that is mostly used in Natural Language Processing (NLP). Naive Bayes classifier consists of collection of several algorithms where all these algorithms share one common principle, and that is each feature being classified is not related to any other feature. It is based on the following formula:

$$P(A|B) = P(A) * P(B|A)/P(B)$$

Here we are finding the probability of class A when probability of B is already provided.

P(B) = It is probability of B

P(A) = It is probability of class A

P(B|A) = It is probability of occurrence of predictor B given class A probability

LOGISTIC REGRESSION

Logistic regression is basically a supervised learning classification algorithm by which we can predict the probability of a target variable. The nature of target or dependent variable is of binary type, which means there would be only two possible classes. Concisely, the dependent variable is binary in nature and can have two values either 1 (stands for success/yes) or 0 (stands for failure/no). In mathematical terms, a logistic regression model predicts $P(Y=1)$ as a function of X.

DECISION TREE

Decision Tree is a supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

Ensemble Learning:

Mendis Moreira et al. gave the definition of Ensemble learning as, "Ensemble learning is a process that uses a set of models, each of them obtained by applying a learning process to a given problem. This set of models (ensemble) is integrated in some way to obtain the final prediction." Ensemble methods are basically meta-algorithms which combines several algorithms in one model either to use in classification or regression problems or for both. In defect prediction we generally use the classification ability of ensemble approach. Ensemble techniques helps to reduce variance, bias factors (except noise, which is irreducible error).

There are 2 types of ensemble methods: -

- **Sequential** - Base learners are generated sequentially.
- **Parallel** - Base learners are generated simultaneously or in parallel manner.

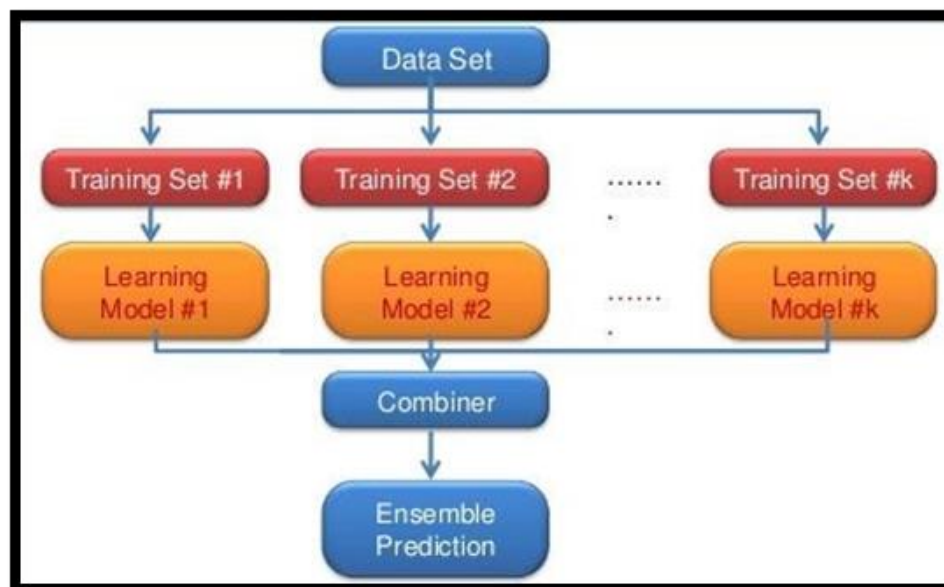


FIG 1. Ensemble Approach flowchart.

Some of the Ensemble techniques are: -

- **Bagging:** - It refers to the variance of the model. It is also known as Bootstrap Aggregation. Bagging is the application of the Bootstrap procedure, which improves the stability of a model by improving accuracy and reducing variance, thus reducing the problem of overfitting.
- **Boosting:** - It basically uses weighted averages in order to make weak learners into stronger learners. It is done on Homogenous models.
- **Stacking:** - In stacking we combine several different weak learners by training a meta-model to output predictions based on the multiple predictions returned by these weak models.

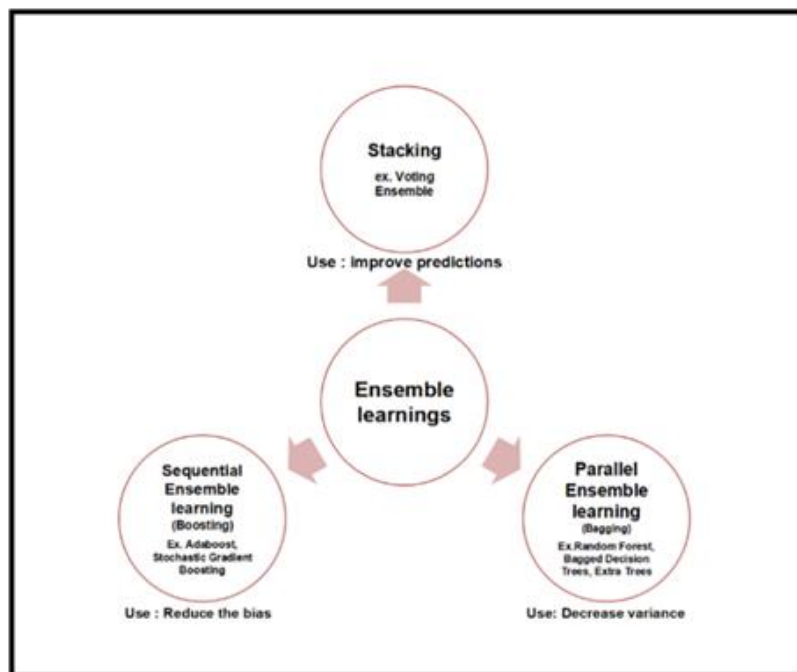


FIG 2. Types of Ensemble Techniques

PYTHON LIBRARIES USED

A Python library is a collection of functions and methods that make it possible for us to perform many actions without writing code.

In our project, we used a variety of python libraries:

- **NumPy:** NumPy is used for processing large multi-dimensional arrays and matrices, with the help of a large collection of high-level mathematical functions. It's a good idea to have a basic and scientific calculation engine in the learning of the room.
- **Pandas:** Panda is a popular Python library for data analysis. Pandas is designed to output the data, and preparation.
- **Matplotlib:** Matplotlib is a Python library used for data visualization. It provides a different type of graphics, and charts for data visualization: column chart, graph, bar, chat, etc
- **Scikit-learn:** Scikit-learn is one of the most popular ML libraries for ML algorithms. It supports many of the supervised and unsupervised learning algorithms. Scikit-learn can be applied to mining and the analysis of the to-do is a great tool to start with ML.
- **Seaborn:** Seaborn is based on matplotlib. It basically provides a high-level interface for drawing attractive and informative statistical tables.
- **train_test_split:** This is splitting of data in a training and a test.

OVERVIEW OF DATASET USED

The Heart Disease dataset has been taken from Kaggle. This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. It has a total number of 303 rows and 14 columns among which 165 have a heart disease.

The dataset comprises of the following attributes:

- age: age in years
- sex: (1 = male; 0 = female)
- cp: chest pain type
- trestbps: resting blood pressure (in mm Hg on admission to the hospital)
- chol: serum cholesterol in mg/dl
- fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg: resting electrocardiographic results
- thalach: maximum heart rate achieved
- exang: exercise induced angina (1 = yes; 0 = no)
- oldpeak: ST depression induced by exercise relative to rest
- slope: the slope of the peak exercise ST segment
- ca: number of major vessels (0-3) colored by fluoroscopy
- thal: thalassemia (1 = normal; 2 = fixed defect; 3 = reversible defect)
- target: (1= heart disease or 0= no heart disease)

GRAPH PLOTTING

The interrelationship between Age & Heart disease and Gender & Heart disease, are plotted in order to see the relation between the attributes. From the line graph, we can conclude that in the age group ranging from 41-60 years, the rate of having heart disease is the highest.

1. Graph Plot between Age and Heart Disease Count

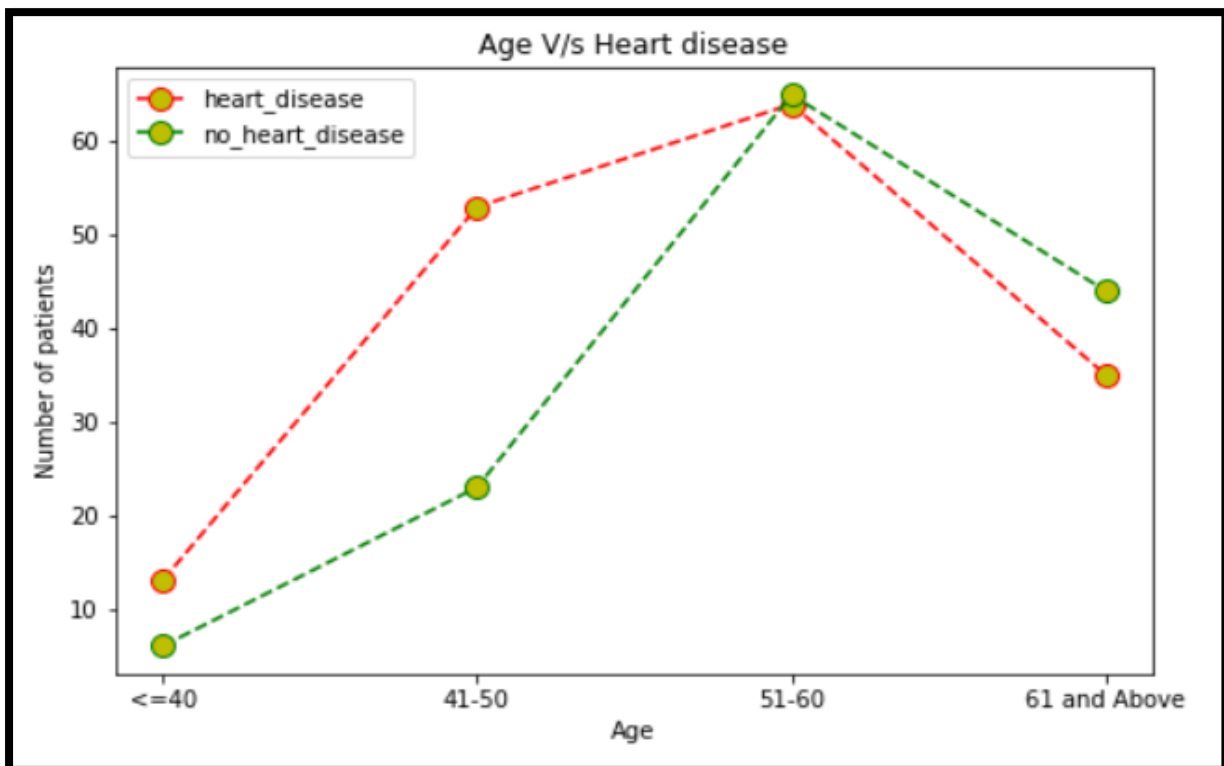
CODE:

```
# Plot a Line graph for Age V/s heart disease
plt.subplots(figsize =(8,5))
classifiers = ['<=40', '41-50', '51-60', '61 and Above']
heart_disease = [13, 53, 64, 35]
no_heart_disease = [6, 23, 65, 44]

l1 = plt.plot(classifiers, heart_disease , color='r', marker='o', linestyle ='dashed', markerfacecolor='y', markersize=10)
l2 = plt.plot(classifiers, no_heart_disease, color='g',marker='o', linestyle ='dashed', markerfacecolor='y', markersize=10 )

plt.xlabel('Age')
plt.ylabel('Number of patients')
plt.title('Age V/s Heart disease')
plt.legend((l1[0], l2[0]), ('heart_disease', 'no_heart_disease'))
plt.show()
```


OUTPUT:



2. Graph plot between gender and Target

CODE:

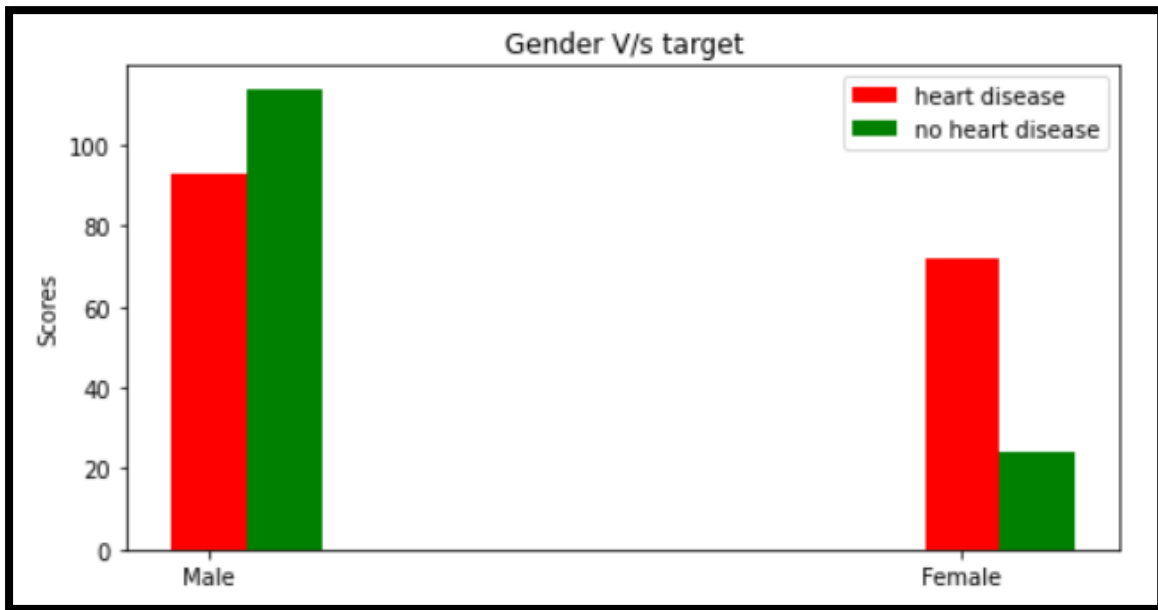
```
# Plot a bar graph for Gender V/s target
N = 2
ind = np.arange(N)
width = 0.1
fig, ax = plt.subplots(figsize =(8,4))

heart_disease = [93, 72]
rects1 = ax.bar(ind, heart_disease, width, color='r')
no_heart_disease = [114, 24]
rects2 = ax.bar(ind+width, no_heart_disease, width, color='g')

ax.set_ylabel('Scores')
ax.set_title('Gender V/s target')
ax.set_xticks(ind)
ax.set_xticklabels(('Male', 'Female'))
ax.legend((rects1[0], rects2[0]), ('heart disease', 'no heart disease'))

plt.show()
```

OUTPUT:



FEATURE SELECTION

Irrelevant or partially relevant features can negatively impact the model performance. So, in order to achieve better accuracy for our model we identified the highly important features from the dataset by implementing a feature selection method i.e., Correlation Matrix with Heat map. Correlation states how the features are related to each other or the target variable. Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable).

CORRELATION OF EACH FEATURE IN DATASET

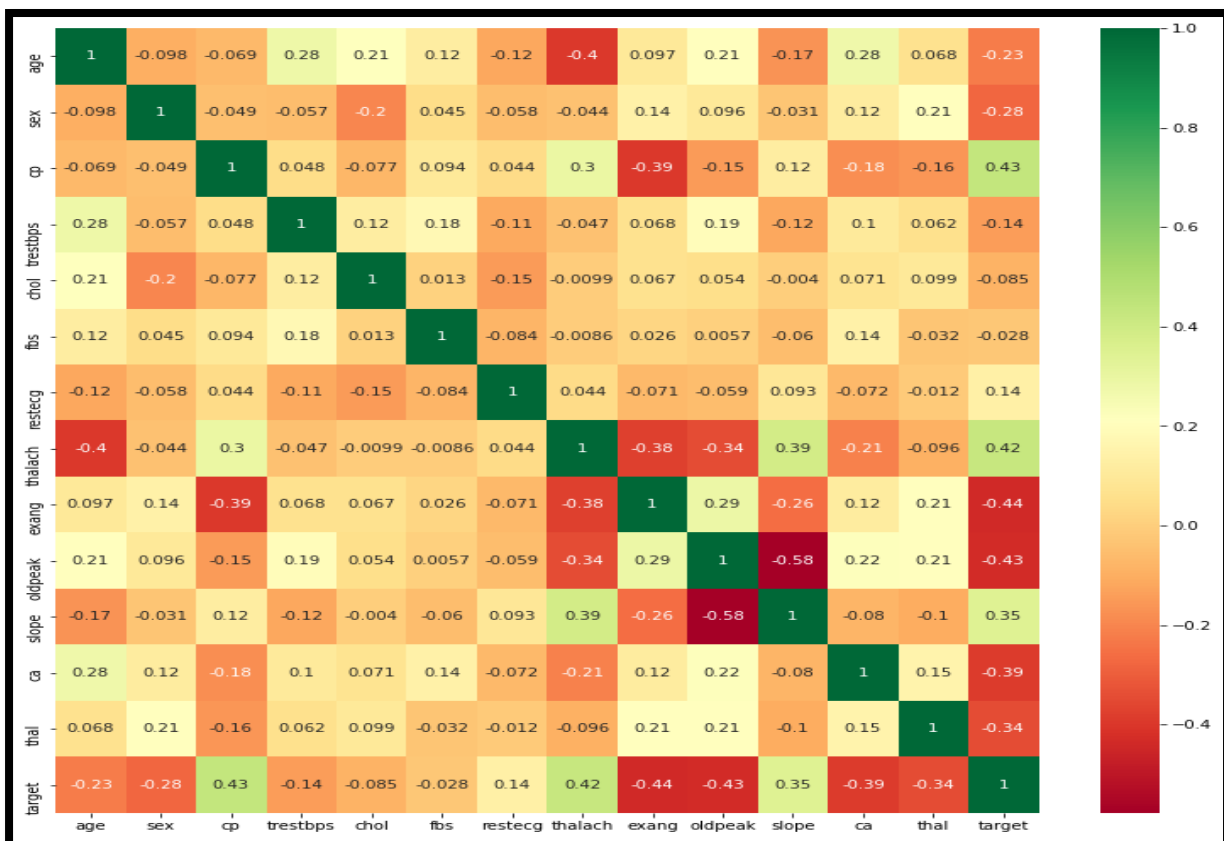
Code:

```
## Feature selection
#getting correlation of each feature in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(13,13))
#plotting heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")

data=data.drop(['sex', 'fbs', 'restecg', 'slope', 'chol', 'age', 'trestbps'], axis=1)

target=data['target']
data = data.drop(['target'],axis=1)
data.head()
```

Output:



DATASET SPLITTING:

From the heat map we can clearly observe that the features like Cp, thalach, exang, oldpeak, ca and thal are the highly correlated features with the target variable and we removed the less important features.

Here I am splitting the dataset in the ratio 70:30 in order to create training and testing sets.

Code:

```
# We split the data into training and testing set:  
x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.3, random_state=10)
```

BASE LEARNERS USED

Base learners are the normal machine learning algorithms. In this analysis Logistic Regression, Decision Tree and Random Forest are regarded as the base learners. Confusion matrix is created in order to show the performance of these classifiers.

1. Multinomial Naïve Bayes

Code:

```
## Base Learners
# 1. Multinomial Naive Bayes(NB)
classifierNB=MultinomialNB()
classifierNB.fit(x_train,y_train)
classifierNB.score(x_test, y_test)

y_preds = classifierNB.predict(x_test)|
print('MultinomialNB accuracy score: ',accuracy_score(y_test, y_preds))
```

Output:

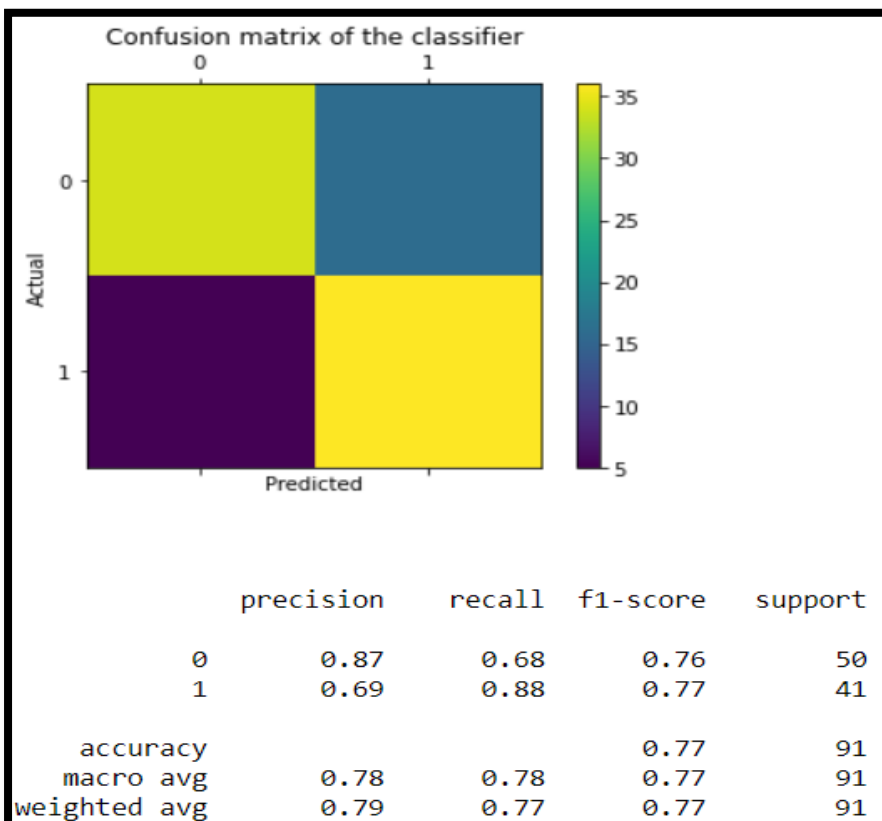
```
MultinomialNB accuracy score: 0.7692307692307693
```

Graph Plotting and Classification Report

Code:

```
#Graph Plotting and Classification Report
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print('\n')
print(classification_report(y_test, y_preds))
```

Output:



2. Logistic Regression

Code:

```
# 2. Logistic Regression(LR)
classifierLR=LogisticRegression()
classifierLR.fit(x_train,y_train)
classifierLR.score(x_test, y_test)

y_preds = classifierLR.predict(x_test)
print('Logistic Regression accuracy score: ',accuracy_score(y_test, y_preds))
```

Output:

```
Logistic Regression accuracy score:  0.7912087912087912
```

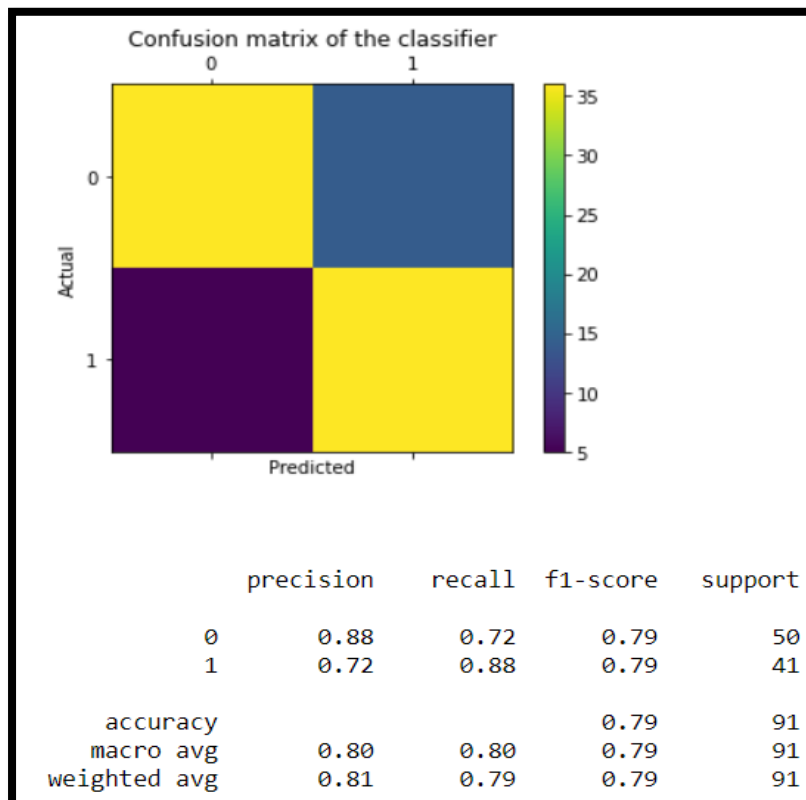
Graph Plotting Classification Report

Code:

```
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))
```

Classification Report Output:



3. DECISION TREE

Code:

```
# 3. Decision Tree (DT)
classifierDT=DecisionTreeClassifier(criterion="gini", random_state=50, max_depth=3, min_samples_leaf=5)
classifierDT.fit(x_train,y_train)
classifierDT.score(x_test, y_test)

y_preds = classifierDT.predict(x_test)
print('Decision Tree accuracy score: ',accuracy_score(y_test, y_preds))
```

Output:

```
Decision Tree accuracy score:  0.7582417582417582
```

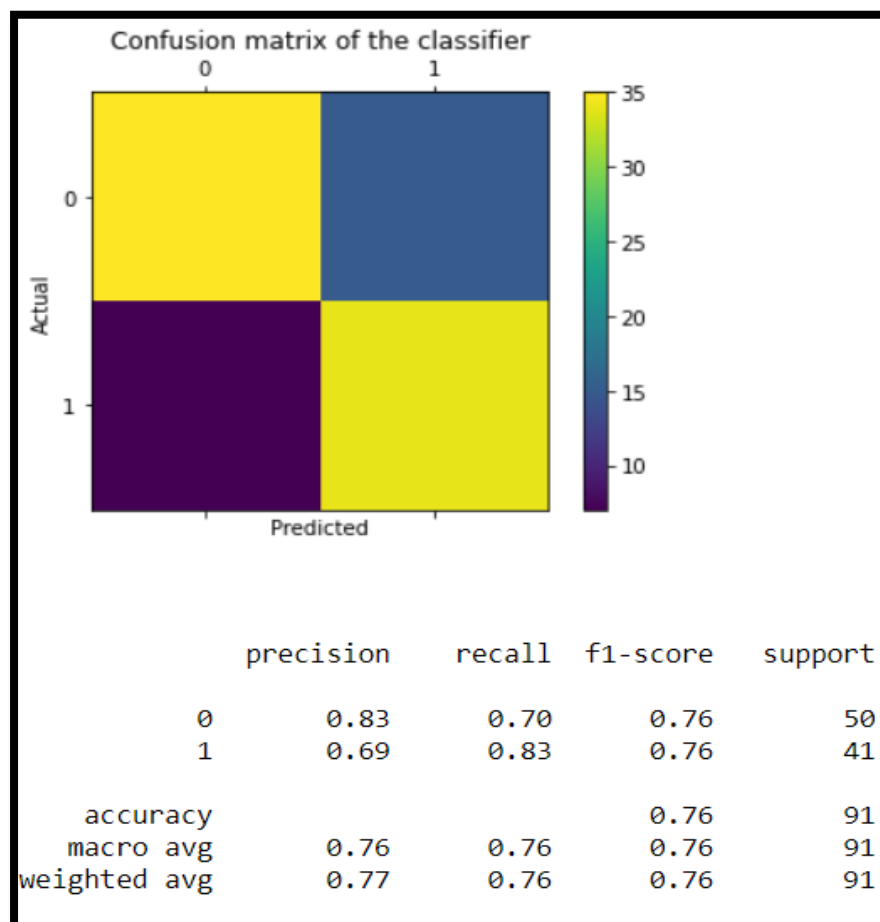
Classification Report

Code:

```
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))
```

Classification Report Output:



ACCURACY REPORT OF BASE LEARNERS USED

Code:

```
print('Accuracy Report of Base Learning Algorithms: ')\nprint('-----')\nprint('Accuracy of naive bayes: {}'.format(classifierNB.score(x_test, y_test)))\nprint('Accuracy of logistic regression: {}'.format(classifierLR.score(x_test, y_test)))\nprint('Accuracy of decision tree: {}'.format(classifierDT.score(x_test, y_test)))
```

Output:

```
Accuracy Report of Base Learning Algorithms:\n-----\nAccuracy of naive bayes: 0.7692307692307693\nAccuracy of logistic regression: 0.7912087912087912\nAccuracy of decision tree: 0.7582417582417582
```

ENSEMBLE CLASSIFIERS USED

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

- **Averaging methods:** The driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced. Example: Bagging methods.
- **Boosting methods:** The base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble. Examples: Gradient Tree Boosting.

In our Innovative project we have used Bagging ensemble classifier.

1. BAGGING

Bagging or Bootstrap aggregating creates several subsets of data from training sample chosen randomly with replacement. Each collection of subset data is used to train their decision trees. As a result, we get an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree classifier.

Code:

```
# 1. Bagging
classifierBa= BaggingClassifier(max_samples=0.5, max_features=1.0, n_estimators=50)
classifierBa.fit(x_train,y_train)
classifierBa.score(x_test, y_test)

y_preds = classifierBa.predict(x_test)
print('bagging_accuracy score: ',accuracy_score(y_test, y_preds))
```

Output:

```
bagging_accuracy score:  0.8131868131868132
```

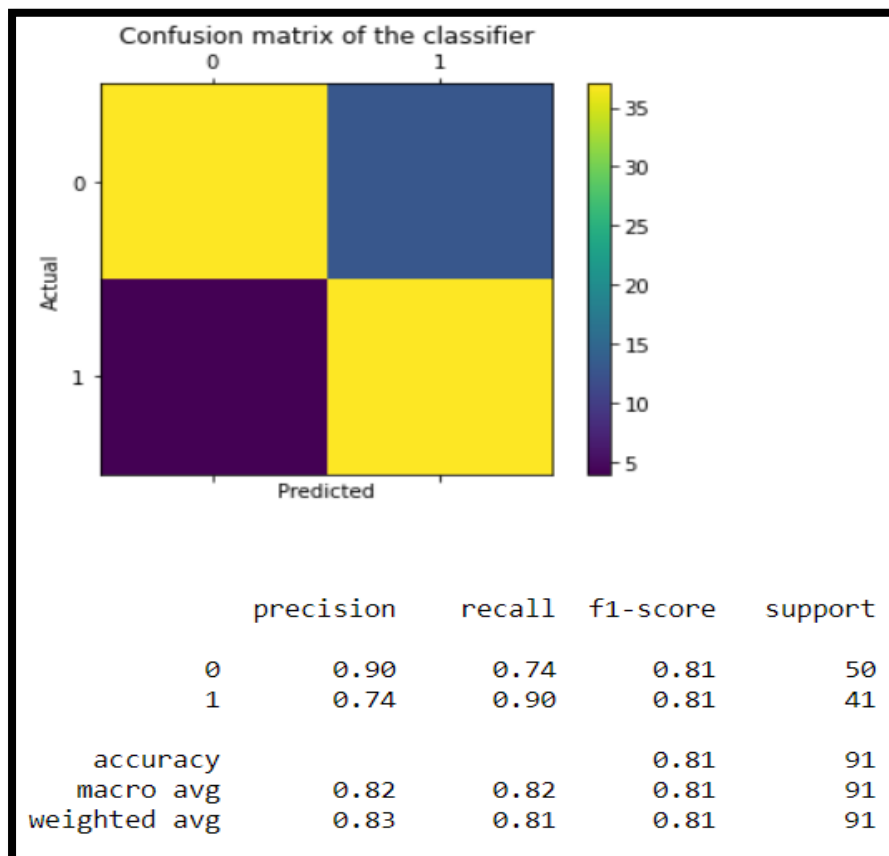
Graph Plotting and Classification Report

Code:

```
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))
```

Output:



2. BOOSTING:

2.1.ADA BOOST Classifier

CODE:

```
## 2. Boosting (Weight Based Boosting)
#1.AdaBoost Classifier
classifierAdaBoost= AdaBoostClassifier(n_estimators=500)
classifierAdaBoost.fit(x_train,y_train)
classifierAdaBoost.score(x_test, y_test)

y_preds = classifierAdaBoost.predict(x_test)
print('Ada_boost_accuracy score: ',accuracy_score(y_test, y_preds))
```

OUTPUT:

```
Ada_boost_accuracy score:  0.7362637362637363
```

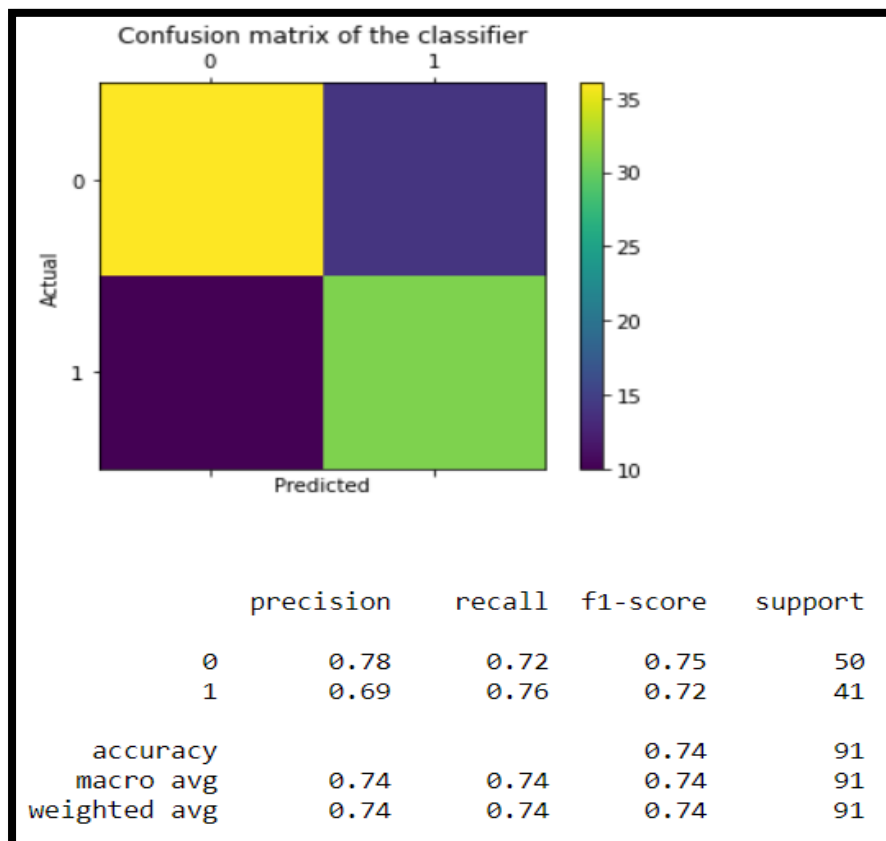
Graph Plotting and Classification Report

Code:

```
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))
```

Output:



2.2. GRADIENT BOOSTING CLASSIFIER

CODE:

```
## 2. Boosting (Residual Based Boosting)
#2. GradientBoosting Classifier
from sklearn.ensemble import GradientBoostingClassifier
classifierGBo = GradientBoostingClassifier(n_estimators=500, learning_rate=1.0, max_depth=1)
classifierGBo.fit(x_train,y_train)
classifierGBo.score(x_test, y_test)

y_preds = classifierGBo.predict(x_test)
print('Gradient_boosting_accuracy score: ',accuracy_score(y_test, y_preds))
```

OUTPUT:

```
Gradient_boosting_accuracy score:  0.8131868131868132
```

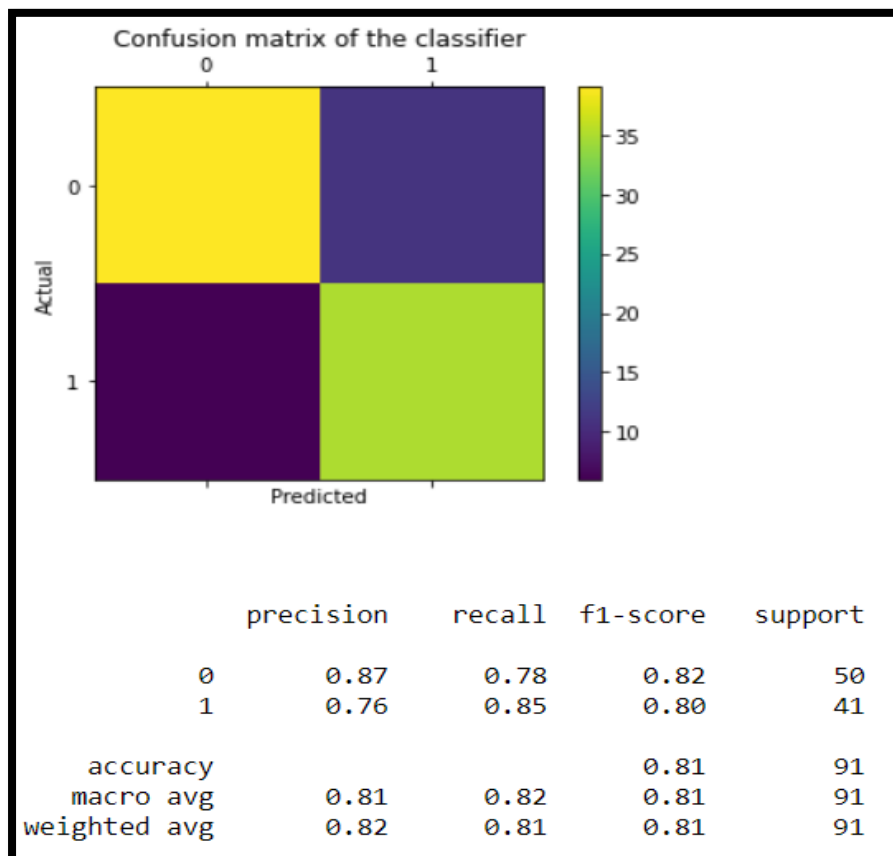

Graph Plotting and Classification Report

Code:

```
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))
```

Output:



3. STACKING

CODE:

```
## 3. Stacking
Estimator=[('dt',classifierDT),('nb',classifierNB)]
clf = StackingClassifier(estimators=Estimator, final_estimator=LogisticRegression())
clf.fit(x_train,y_train)
clf.score(x_test, y_test)

y_preds = clf.predict(x_test)
print('Stacking accuracy score: ',accuracy_score(y_test, y_preds))
```

OUTPUT:

```
Stacking accuracy score:  0.7692307692307693
```

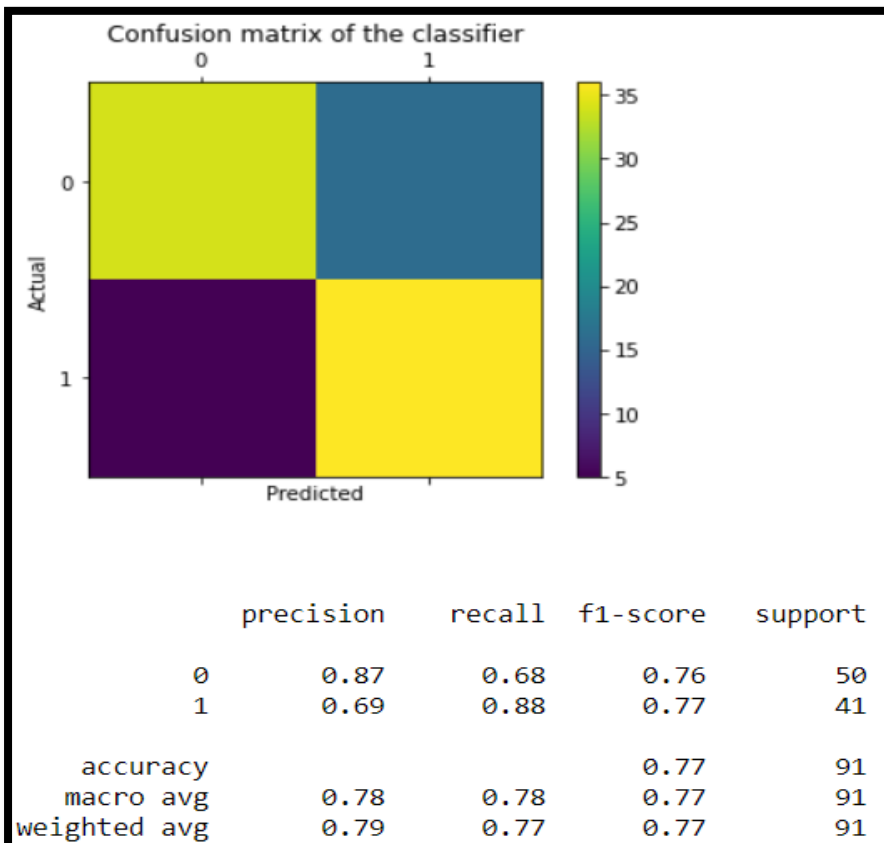
Graph Plotting and Classification Report

Code:

```
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))
```

Output:



COMPLETE SOURCE CODE

```
1. #Importing the libraries
2. import numpy as np
3. import pandas as pd
4. import matplotlib.pyplot as plt
5. import seaborn as sns
6. from sklearn.model_selection import train_test_split
7. #Importing Base Learners
8. from sklearn.naive_bayes import MultinomialNB
9. from sklearn.linear_model import LogisticRegression
10. from sklearn.tree import DecisionTreeClassifier
11. from sklearn.metrics import make_scorer, f1_score, recall_score, precision_score
12. from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
13. from sklearn.metrics import log_loss
14. import warnings
15. warnings.simplefilter(action = 'ignore', category= FutureWarning)
16. #Importing Ensemble Classifiers
17. from sklearn.ensemble import BaggingClassifier
18. from sklearn.ensemble import AdaBoostClassifier
19. from sklearn.ensemble import StackingClassifier
20. #Reading the csv dataset
21. data = pd.read_csv("C://Users/suman/OneDrive/Desktop/heart.csv",
    encoding='ANSI')
22. data.columns
23. data.head()
24. #Total number of rows and columns
25. data.shape
26. # Plotting a line graph for Age V/s heart disease
27. plt.subplots(figsize =(8,5))
28. classifiers = ['<=40', '41-50', '51-60','61 and Above']
29. heart_disease = [13, 53, 64, 35]
30. no_heart_disease = [6, 23, 65, 44]
31. l1 = plt.plot(classifiers, heart_disease , color='r', marker='o', linestyle ='dashed',
    markerfacecolor='y', markersize=10)
32. l2 = plt.plot(classifiers, no_heart_disease, color='g',marker='o', linestyle ='dashed',
    markerfacecolor='y', markersize=10 )
33. plt.xlabel('Age')
34. plt.ylabel('Number of patients')
35. plt.title('Age V/s Heart disease')
36. plt.legend((l1[0], l2[0]), ('heart_disease', 'no_heart_disease'))
37. plt.show()
38. # Plotting a bar graph for Gender V/s target
39. N = 2
40. ind = np.arange(N)
41. width = 0.1
42. fig, ax = plt.subplots(figsize =(8,4))
```

```

43. heart_disease = [93, 72]
44. rects1 = ax.bar(ind, heart_disease, width, color='r')
45. no_heart_disease = [114, 24]
46. rects2 = ax.bar(ind+width, no_heart_disease, width, color='g')
47. ax.set_ylabel('Scores')
48. ax.set_title('Gender V/s target')
49. ax.set_xticks(ind)
50. ax.set_xticklabels(('Male','Female'))
51. ax.legend((rects1[0], rects2[0]), ('heart disease', 'no heart disease'))
52. plt.show()
53. ## Feature selection
54. #get correlation of each feature in dataset
55. corrmatrix = data.corr()
56. top_corr_features = corrmatrix.index
57. plt.figure(figsize=(13,13))
58. #plot heat map
59. g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
60. data=data.drop(['sex', 'fbs', 'restecg', 'slope', 'chol', 'age', 'trestbps'], axis=1)
61. target=data['target']
62. data = data.drop(['target'],axis=1)
63. data.head()
64. # Splitting the data into training and testing set:
65. x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.3,
    random_state=10)
66. ## Base Learners
67. # 1. Multinomial Naive Bayes(NB)
68. classifierNB=MultinomialNB()
69. classifierNB.fit(x_train,y_train)
70. classifierNB.score(x_test, y_test)
71. y_preds = classifierNB.predict(x_test)
72. print('MultinomialNB accuracy score: ',accuracy_score(y_test, y_preds))
73. #Graph Plotting and Classification Report
74. import pylab as plt
75. labels=[0,1]
76. cmx=confusion_matrix(y_test,y_preds, labels)
77. print(cmx)
78. fig = plt.figure()
79. ax = fig.add_subplot(111)
80. cax = ax.matshow(cmx)
81. plt.title('Confusion matrix of the classifier')
82. fig.colorbar(cax)
83. ax.set_xticklabels([""] + labels)
84. ax.set_yticklabels([""] + labels)
85. plt.xlabel('Predicted')
86. plt.ylabel('Actual')
87. plt.show()

```

```

88. print('\n')
89. print(classification_report(y_test, y_preds))
90. #Graph Plotting and Classification Report
91. import pylab as plt
92. labels=[0,1]
93. cmx=confusion_matrix(y_test,y_preds, labels)
94. print(cmx)
95. fig = plt.figure()
96. ax = fig.add_subplot(111)
97. cax = ax.matshow(cmx)
98. plt.title('Confusion matrix of the classifier')
99. fig.colorbar(cax)
100.     ax.set_xticklabels([""] + labels)
101.     ax.set_yticklabels([""] + labels)
102.     plt.xlabel('Predicted')
103.     plt.ylabel('Actual')
104.     plt.show()
105.     print('\n')
106.     print(classification_report(y_test, y_preds))
107. # 2. Logistic Regression(LR)
108.     classifierLR=LogisticRegression()
109.     classifierLR.fit(x_train,y_train)
110.     classifierLR.score(x_test, y_test)
111.
112.     y_preds = classifierLR.predict(x_test)
113.     print('Logistic Regression accuracy score: ',accuracy_score(y_test, y_preds))
114. #Graph Plotting and Classification Report
115.     import pylab as plt
116.     labels=[0,1]
117.     cmx=confusion_matrix(y_test,y_preds, labels)
118.     print(cmx)
119.     fig = plt.figure()
120.     ax = fig.add_subplot(111)
121.     cax = ax.matshow(cmx)
122.     plt.title('Confusion matrix of the classifier')
123.     fig.colorbar(cax)
124.     ax.set_xticklabels([""] + labels)
125.     ax.set_yticklabels([""] + labels)
126.     plt.xlabel('Predicted')
127.     plt.ylabel('Actual')
128.     plt.show()
129.     print('\n')
130.     print(classification_report(y_test, y_preds))
131. # 3. Decision Tree (DT)
132.     classifierDT=DecisionTreeClassifier(criterion="gini", random_state=50,
        max_depth=3, min_samples_leaf=5)

```

```

133. classifierDT.fit(x_train,y_train)
134. classifierDT.score(x_test, y_test)
135. y_preds = classifierDT.predict(x_test)
136. print('Decision Tree accuracy score: ',accuracy_score(y_test, y_preds))
137. #Graph Plotting and Classification Report
138. import pylab as plt
139. labels=[0,1]
140. cmx=confusion_matrix(y_test,y_preds, labels)
141. print(cmx)
142. fig = plt.figure()
143. ax = fig.add_subplot(111)
144. cax = ax.matshow(cmx)
145. plt.title('Confusion matrix of the classifier')
146. fig.colorbar(cax)
147. ax.set_xticklabels([""] + labels)
148. ax.set_yticklabels([""] + labels)
149. plt.xlabel('Predicted')
150. plt.ylabel('Actual')
151. plt.show()
152. print('\n')
153. print(classification_report(y_test, y_preds))
154. print('Accuracy Report of Base Learning Algorithms: ')
155. print('-----')
156. print('Accuracy of naive bayes: {}'.format(classifierNB.score(x_test, y_test)))
157. print('Accuracy of logistic regression: {}'.format(classifierLR.score(x_test,
    y_test)))
158. print('Accuracy of decision tree: {}'.format(classifierDT.score(x_test, y_test)))
159. # 1. Bagging
160. classifierBa= BaggingClassifier(max_samples=0.5, max_features=1.0,
    n_estimators=50)
161. classifierBa.fit(x_train,y_train)
162. classifierBa.score(x_test, y_test)
163. y_preds = classifierBa.predict(x_test)
164. print('bagging_accuracy score: ',accuracy_score(y_test, y_preds))
165. import pylab as plt
166. labels=[0,1]
167. cmx=confusion_matrix(y_test,y_preds, labels)
168. print(cmx)
169. fig = plt.figure()
170. ax = fig.add_subplot(111)
171. cax = ax.matshow(cmx)
172. plt.title('Confusion matrix of the classifier')
173. fig.colorbar(cax)
174. ax.set_xticklabels([""] + labels)
175. ax.set_yticklabels([""] + labels)
176. plt.xlabel('Predicted')

```

```

177. plt.ylabel('Actual')
178. plt.show()
179. print('\n')
180. print(classification_report(y_test, y_preds))
181. ## 2. Boosting (Weight Based Boosting)
182. #1.AdaBoost Classifier
183. classifierAdaBoost= AdaBoostClassifier(n_estimators=500)
184. classifierAdaBoost.fit(x_train,y_train)
185. classifierAdaBoost.score(x_test, y_test)
186. y_preds = classifierAdaBoost.predict(x_test)
187. print('Ada_boost_accuracy score: ',accuracy_score(y_test, y_preds))
188. #Graph Plotting and Classification Report
189. import pylab as plt
190. labels=[0,1]
191. cmx=confusion_matrix(y_test,y_preds, labels)
192. print(cmx)
193. fig = plt.figure()
194. ax = fig.add_subplot(111)
195. cax = ax.matshow(cmx)
196. plt.title('Confusion matrix of the classifier')
197. fig.colorbar(cax)
198. ax.set_xticklabels([""] + labels)
199. ax.set_yticklabels([""] + labels)
200. plt.xlabel('Predicted')
201. plt.ylabel('Actual')
202. plt.show()
203. print('\n')
204. print(classification_report(y_test, y_preds))
205. ## 2. Boosting (Residual Based Boosting)
206. #2. GradientBoosting Classifier
207. from sklearn.ensemble import GradientBoostingClassifier
208. classifierGBo= GradientBoostingClassifier(n_estimators=500,
    learning_rate=1.0, max_depth=1)
209. classifierGBo.fit(x_train,y_train)
210. classifierGBo.score(x_test, y_test)
211. y_preds = classifierGBo.predict(x_test)
212. print('Gradient_boosting_accuracy score: ',accuracy_score(y_test, y_preds))
213. #Graph Plotting and Classification Report
214. import pylab as plt
215. labels=[0,1]
216. cmx=confusion_matrix(y_test,y_preds, labels)
217. print(cmx)
218. fig = plt.figure()
219. ax = fig.add_subplot(111)
220. cax = ax.matshow(cmx)
221. plt.title('Confusion matrix of the classifier')

```



```

222.     fig.colorbar(cax)
223.     ax.set_xticklabels([""] + labels)
224.     ax.set_yticklabels([""] + labels)
225.     plt.xlabel('Predicted')
226.     plt.ylabel('Actual')
227.     plt.show()
228.     print('\n')
229.     print(classification_report(y_test, y_preds))
230.     ## 3. Stacking
231.     Estimator=[('dt',classifierDT),('NV',classifierNB)]
232.     clf = StackingClassifier(estimators=Estimator,
        final_estimator=LogisticRegression())
233.     clf.fit(x_train,y_train)
234.     clf.score(x_test, y_test)
235.     y_preds = clf.predict(x_test)
236.     print('Stacking accuracy score: ',accuracy_score(y_test, y_preds))
237.     #Graph Plotting and Classification Report
238.     import pylab as plt
239.     labels=[0,1]
240.     cmx=confusion_matrix(y_test,y_preds, labels)
241.     print(cmx)
242.     fig = plt.figure()
243.     ax = fig.add_subplot(111)
244.     cax = ax.matshow(cmx)
245.     plt.title('Confusion matrix of the classifier')
246.     fig.colorbar(cax)
247.     ax.set_xticklabels([""] + labels)
248.     ax.set_yticklabels([""] + labels)
249.     plt.xlabel('Predicted')
250.     plt.ylabel('Actual')
251.     plt.show()
252.     print('\n')
253.     print(classification_report(y_test, y_preds))
254.     print('CONCLUSION : ')
255.     print('-----')
256.     print('Accuracy Report of Base Learning Algorithms: ')
257.     print('-----')
258.     print('Accuracy of naive bayes: {}'.format(classifierNB.score(x_test, y_test)))
259.     print('Accuracy of logistic regression: {}'.format(classifierLR.score(x_test,
        y_test)))
260.     print('Accuracy of decision tree: {}'.format(classifierDT.score(x_test, y_test)))
261.     print('\n')
262.     print('Accuracy Report of Ensemble Classifiers: ')
263.     print('-----')
264.     print('Bagging Accuracy Score: ',classifierBa.score(x_test, y_test))
265.     print('Ada_boost Accuracy Score: ',classifierAdaBoost.score(x_test, y_test))

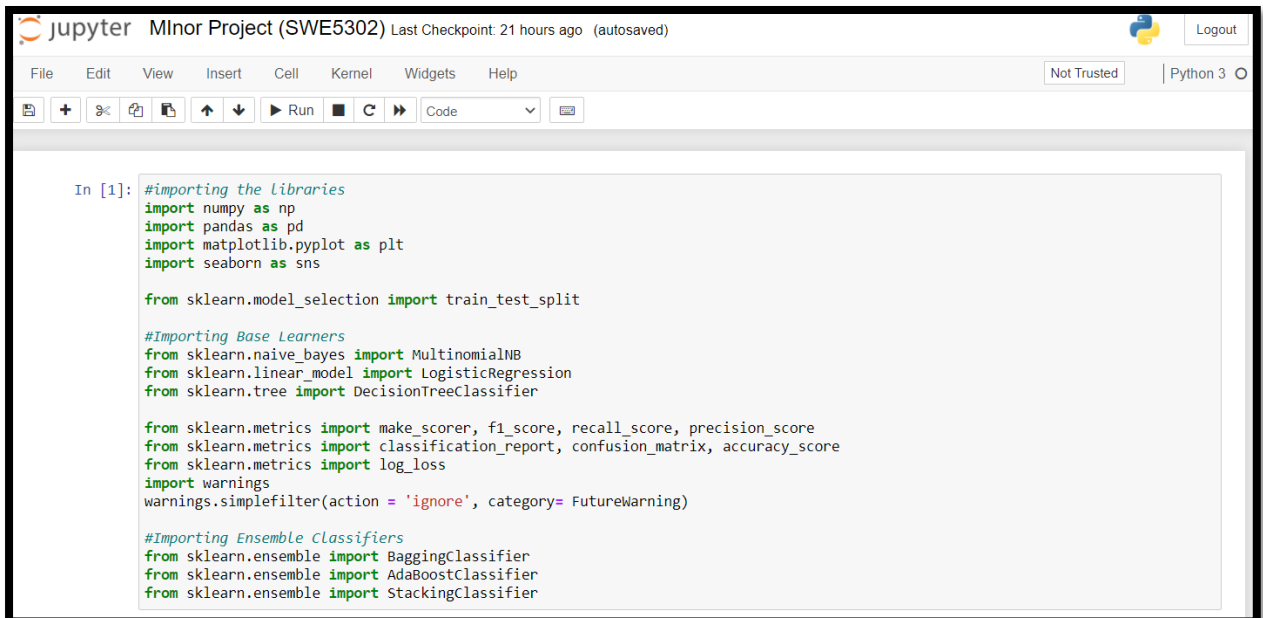
```

```

266.     print('Gradient Boosting Accuracy Score: ',classifierGBo.score(x_test, y_test))
267.     print('Stacking Accuracy Score: ',clf.score(x_test, y_test))

```

CODE AND OUTPUT SCREENSHOTS



A screenshot of a Jupyter Notebook interface. The title bar shows 'jupyter Minor Project (SWE5302) Last Checkpoint: 21 hours ago (autosaved)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar has icons for file operations, running, and code execution. The code cell contains the following Python code:

```

In [1]: #importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

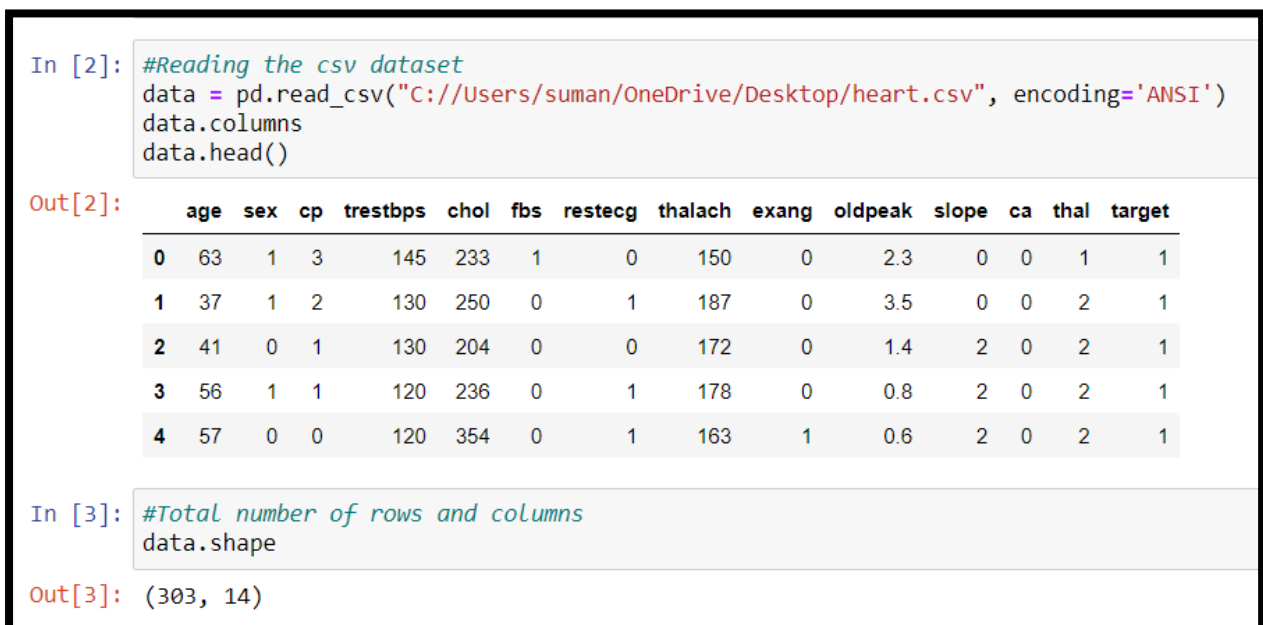
from sklearn.model_selection import train_test_split

#Importing Base Learners
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import make_scorer, f1_score, recall_score, precision_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.metrics import log_loss
import warnings
warnings.simplefilter(action = 'ignore', category= FutureWarning)

#Importing Ensemble Classifiers
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import StackingClassifier

```



A screenshot of a Jupyter Notebook interface showing the second cell. The code cell contains the following Python code:

```

In [2]: #Reading the csv dataset
data = pd.read_csv("C://Users/suman/OneDrive/Desktop/heart.csv", encoding='ANSI')
data.columns
data.head()

```

The output of the code is displayed below the code cell:

```

Out[2]:

```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

The code cell continues with:

```

In [3]: #Total number of rows and columns
data.shape

```

The output of the code is displayed below the code cell:

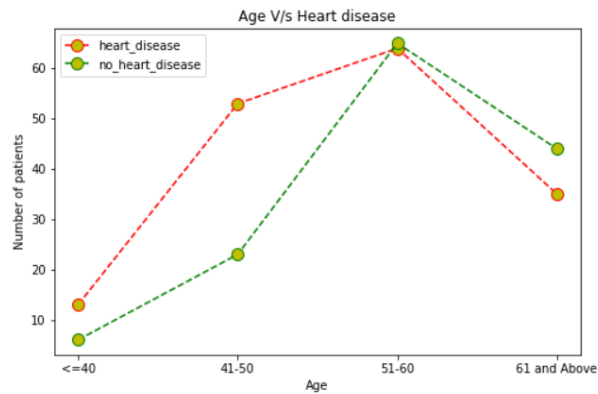
```

Out[3]: (303, 14)

```

```
In [4]: # Plotting a Line graph for Age V/s heart disease
plt.subplots(figsize=(8,5))
classifiers = ['<=40', '41-50', '51-60', '61 and Above']
heart_disease = [13, 53, 64, 35]
no_heart_disease = [6, 23, 65, 44]
l1 = plt.plot(classifiers, heart_disease, color='r', marker='o', linestyle='dashed', markerfacecolor='y', markersize=10)
l2 = plt.plot(classifiers, no_heart_disease, color='g', marker='o', linestyle='dashed', markerfacecolor='y', markersize=10)

plt.xlabel('Age')
plt.ylabel('Number of patients')
plt.title('Age V/s Heart disease')
plt.legend((l1[0], l2[0]), ('heart_disease', 'no_heart_disease'))
plt.show()
```

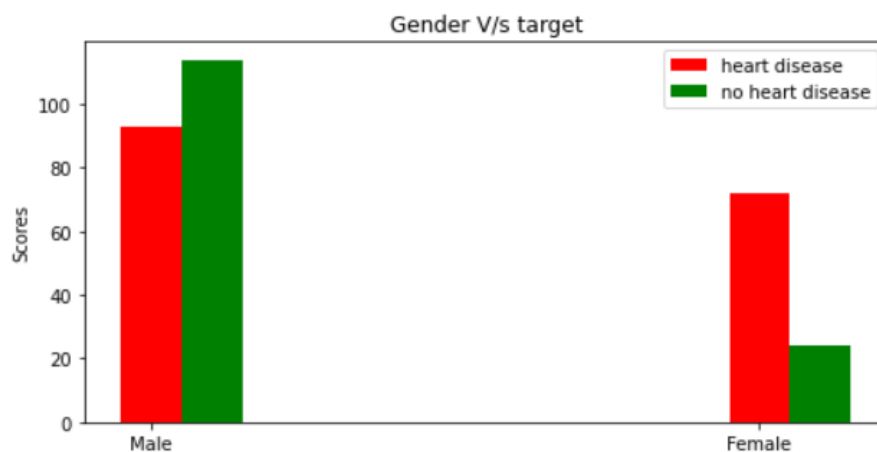


```
In [5]: # Plotting a bar graph for Gender V/s target
N = 2
ind = np.arange(N)
width = 0.1
fig, ax = plt.subplots(figsize=(8,4))

heart_disease = [93, 72]
rects1 = ax.bar(ind, heart_disease, width, color='r')
no_heart_disease = [114, 24]
rects2 = ax.bar(ind+width, no_heart_disease, width, color='g')

ax.set_ylabel('Scores')
ax.set_title('Gender V/s target')
ax.set_xticks(ind)
ax.set_xticklabels(('Male', 'Female'))
ax.legend((rects1[0], rects2[0]), ('heart disease', 'no heart disease'))

plt.show()
```



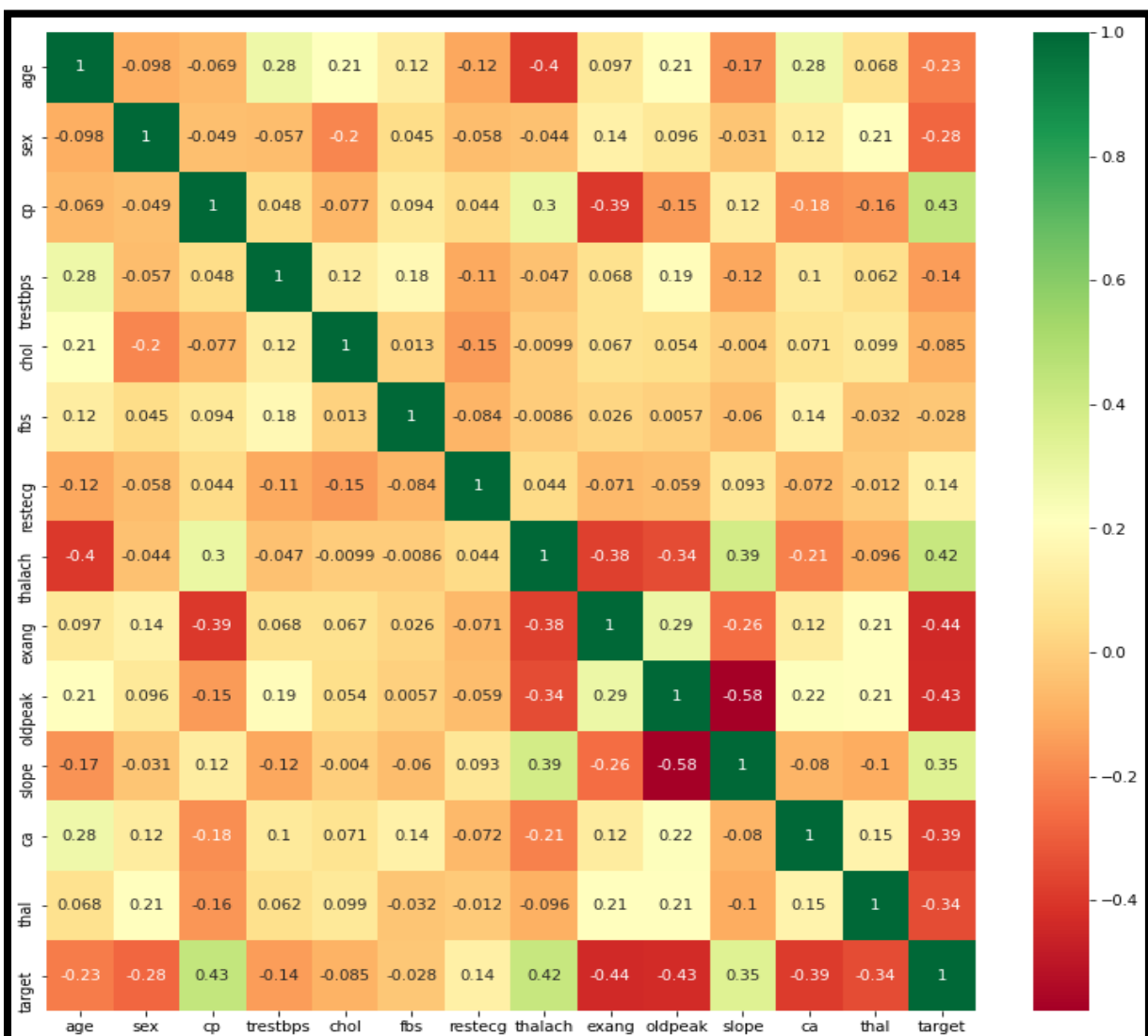
```
In [6]: ## Feature selection
# get correlation of each feature in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(13,13))
# plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")

data=data.drop(['sex', 'fbs', 'restecg', 'slope', 'chol', 'age', 'trestbps'], axis=1)

target=data['target']
data = data.drop(['target'],axis=1)
data.head()
```

```
Out[6]:
```

	cp	thalach	exang	oldpeak	ca	thal
0	3	150	0	2.3	0	1
1	2	187	0	3.5	0	2
2	1	172	0	1.4	0	2
3	1	178	0	0.8	0	2
4	0	163	1	0.6	0	2



```

In [7]: # Splitting the data into training and testing set:
x_train, x_test, y_train, y_test = train_test_split(data, target, test_size=0.3, random_state=10)

In [8]: ## Base Learners
# 1. Multinomial Naive Bayes(NB)
classifierNB=MultinomialNB()
classifierNB.fit(x_train,y_train)
classifierNB.score(x_test, y_test)

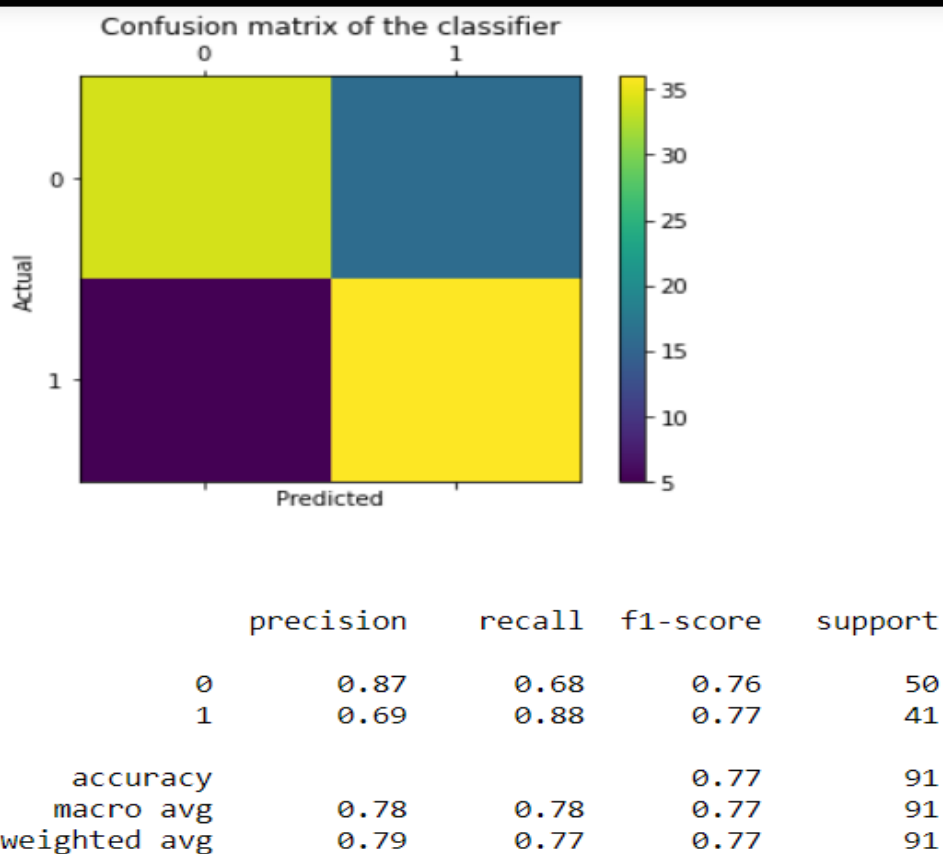
y_preds = classifierNB.predict(x_test)
print('MultinomialNB accuracy score: ',accuracy_score(y_test, y_preds))

MultinomialNB accuracy score:  0.7692307692307693

In [9]: #Graph Plotting and Classification Report
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print('\n')
print(classification_report(y_test, y_preds))

[[34 16]
 [ 5 36]]

```



```
In [10]: # 2. Logistic Regression(LR)
classifierLR=LogisticRegression()
classifierLR.fit(x_train,y_train)
classifierLR.score(x_test, y_test)

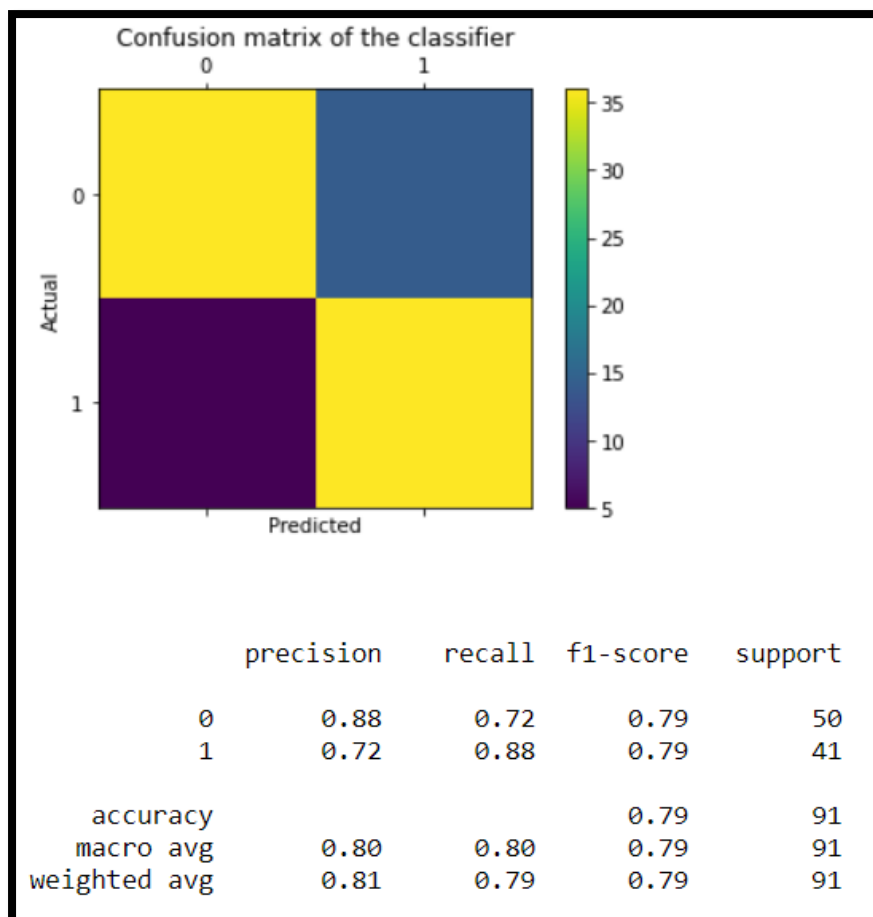
y_preds = classifierLR.predict(x_test)
print('Logistic Regression accuracy score: ',accuracy_score(y_test, y_preds))

Logistic Regression accuracy score:  0.7912087912087912
```

```
In [11]: #Graph Plotting and Classification Report
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))

[[36 14]
 [ 5 36]]
```



```
In [12]: # 3. Decision Tree (DT)
classifierDT=DecisionTreeClassifier(criterion="gini", random_state=50, max_depth=3, min_samples_leaf=5)
classifierDT.fit(x_train,y_train)
classifierDT.score(x_test, y_test)

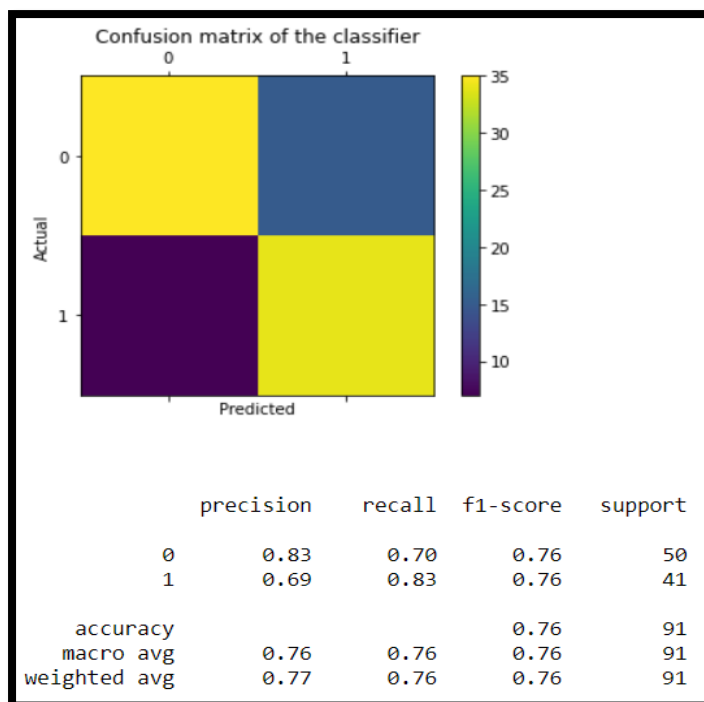
y_preds = classifierDT.predict(x_test)
print('Decision Tree accuracy score: ',accuracy_score(y_test, y_preds))

Decision Tree accuracy score: 0.7582417582417582

In [13]: import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))

[[35 15]
 [ 7 34]]
```



```
In [14]: print('Accuracy Report of Base Learning Algorithms: ')
print('-----')
print('Accuracy of naive bayes: {}'.format(classifierNB.score(x_test, y_test)))
print('Accuracy of logistic regression: {}'.format(classifierLR.score(x_test, y_test)))
print('Accuracy of decision tree: {}'.format(classifierDT.score(x_test, y_test)))
```

Accuracy Report of Base Learning Algorithms:

Accuracy of naive bayes: 0.7692307692307693
Accuracy of logistic regression: 0.7912087912087912
Accuracy of decision tree: 0.7582417582417582

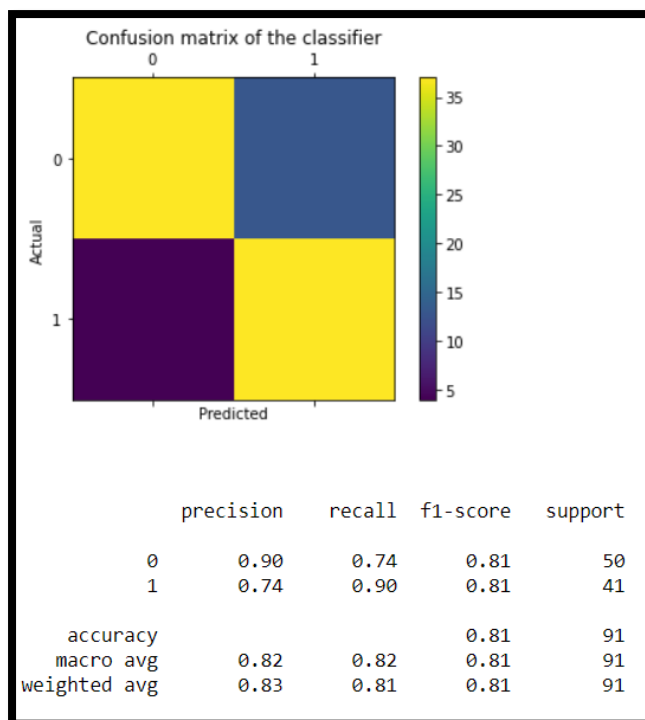
```
In [15]: # 1. Bagging
classifierBa = BaggingClassifier(max_samples=0.5, max_features=1.0, n_estimators=50)
classifierBa.fit(x_train, y_train)
classifierBa.score(x_test, y_test)

y_preds = classifierBa.predict(x_test)
print('bagging_accuracy score: ', accuracy_score(y_test, y_preds))

bagging_accuracy score: 0.8131868131868132
```

```
In [16]: import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print('\n')
print(classification_report(y_test, y_preds))

[[ 37  13]
 [  4  37]]
```

```
In [17]: ## 2. Boosting (Weight Based Boosting)
#1.AdaBoost Classifier
classifierAdaBoost= AdaBoostClassifier(n_estimators=500)
classifierAdaBoost.fit(x_train,y_train)
classifierAdaBoost.score(x_test, y_test)

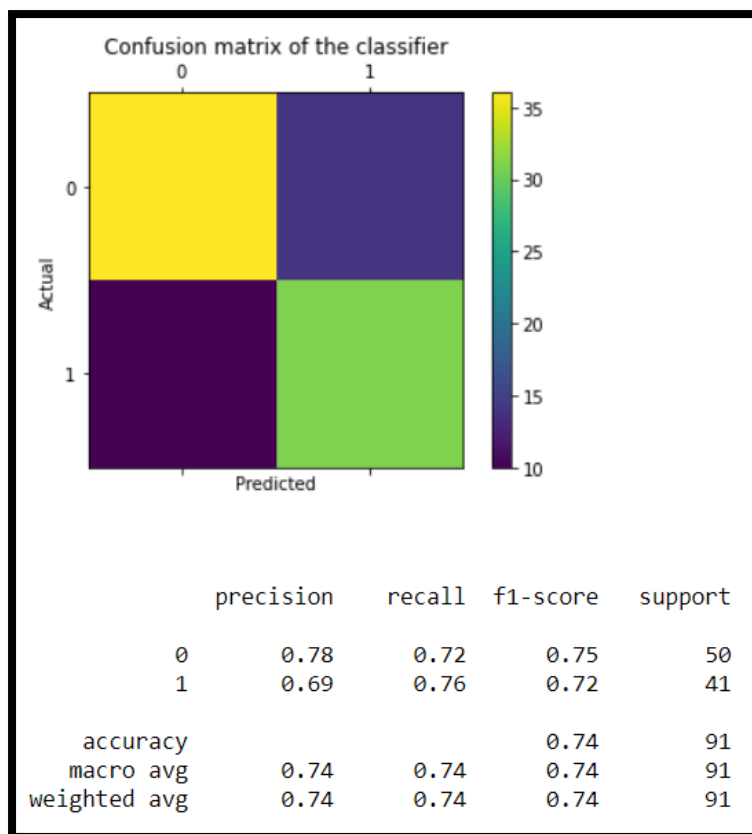
y_preds = classifierAdaBoost.predict(x_test)
print('Ada_boost_accuracy score: ',accuracy_score(y_test, y_preds))

Ada_boost_accuracy score:  0.7362637362637363
```

```
In [18]: import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print('\n')
print(classification_report(y_test, y_preds))

[[36 14]
 [10 31]]
```



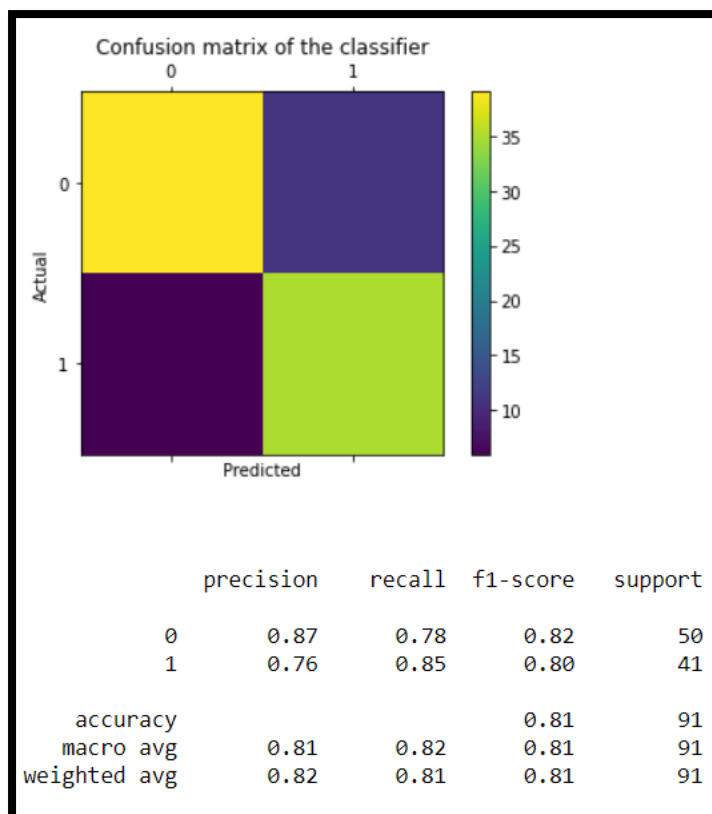
```
In [19]: ## 2. Boosting (Residual Based Boosting)
#2. GradientBoosting Classifier
from sklearn.ensemble import GradientBoostingClassifier
classifierGBo = GradientBoostingClassifier(n_estimators=500, learning_rate=1.0, max_depth=1)
classifierGBo.fit(x_train,y_train)
classifierGBo.score(x_test, y_test)

y_preds = classifierGBo.predict(x_test)
print('Gradient_boosting_accuracy score: ',accuracy_score(y_test, y_preds))

Gradient_boosting_accuracy score:  0.8131868131868132
```

```
In [20]: #Graph Plotting and Classification Report
import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print('\n')
print(classification_report(y_test, y_preds))

[[39 11]
 [ 6 35]]
```



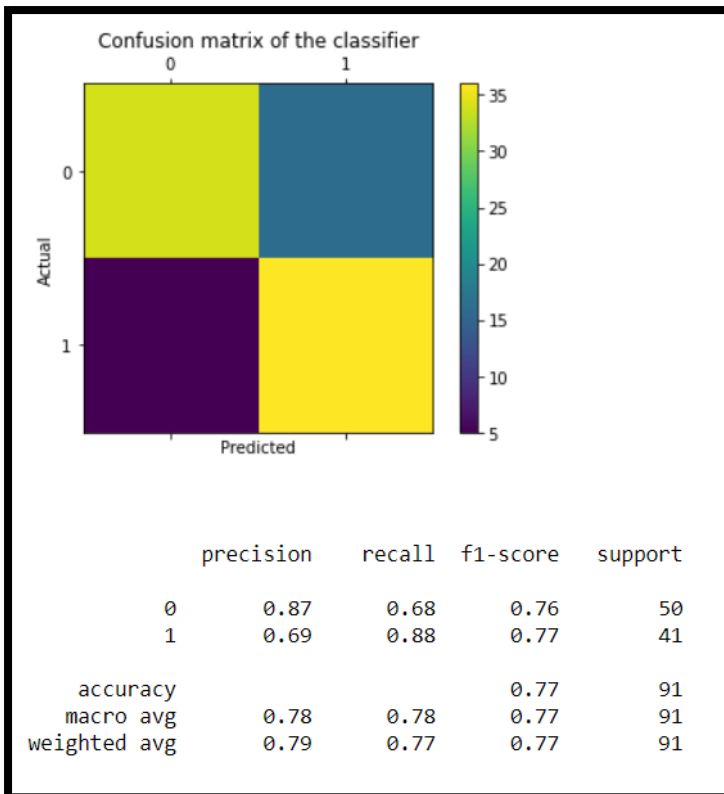
```
In [21]: ## 3. Stacking
Estimator=[('dt',classifierDT),('nb',classifierNB)]
clf = StackingClassifier(estimators=Estimator, final_estimator=LogisticRegression())
clf.fit(x_train,y_train)
clf.score(x_test, y_test)
```

```
y_preds = clf.predict(x_test)
print('Stacking accuracy score: ',accuracy_score(y_test, y_preds))
```

Stacking accuracy score: 0.7692307692307693

```
In [22]: import pylab as plt
labels=[0,1]
cmx=confusion_matrix(y_test,y_preds, labels)
print(cmx)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cmx)
plt.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print('\n')
print(classification_report(y_test, y_preds))
```

```
[[34 16]
 [ 5 36]]
```



```
In [23]: print('CONCLUSION : ')
print('-----')
print('Accuracy Report of Base Learning Algorithms: ')
print('-----')
print('Accuracy of naive bayes: {}'.format(classifierNB.score(x_test, y_test)))
print('Accuracy of logistic regression: {}'.format(classifierLR.score(x_test, y_test)))
print('Accuracy of decision tree: {}'.format(classifierDT.score(x_test, y_test)))
print('\n')
print('Accuracy Report of Ensemble Classifiers: ')
print('-----')
print('Bagging Accuracy Score: ',classifierBa.score(x_test, y_test))
print('Ada_boost Accuracy Score: ',classifierAdaBoost.score(x_test, y_test))
print('Gradient Boosting Accuracy Score: ',classifierGBo.score(x_test, y_test))
print('Stacking Accuracy Score: ',clf.score(x_test, y_test))
```

CONCLUSION :

Accuracy Report of Base Learning Algorithms:

Accuracy of naive bayes: 0.7692307692307693
Accuracy of logistic regression: 0.7912087912087912
Accuracy of decision tree: 0.7582417582417582

Accuracy Report of Ensemble Classifiers:

Bagging Accuracy Score: 0.8131868131868132
Ada_boost Accuracy Score: 0.7362637362637363
Gradient Boosting Accuracy Score: 0.8131868131868132
Stacking Accuracy Score: 0.7692307692307693

CONCLUSION

This innovative project attempts to predict the presence of heart disease by utilizing Base learners, Ensemble combination rules and Ensemble classifiers. The accuracy report of all the algorithms used are shown below.

It is observed that the accuracy has been increased by the using of Ensemble techniques namely **Bagging** and **Gradient Boosting** both having an accuracy of 81.3%.

CODE:

```
print('CONCLUSION : ')
print('-----')
print('Accuracy Report of Base Learning Algorithms: ')
print('-----')
print('Accuracy of naive bayes: {}'.format(classifierNB.score(x_test, y_test)))
print('Accuracy of logistic regression: {}'.format(classifierLR.score(x_test, y_test)))
print('Accuracy of decision tree: {}'.format(classifierDT.score(x_test, y_test)))
print('\n')
print('Accuracy Report of Ensemble Classifiers: ')
print('-----')
print('Bagging Accuracy Score: ',classifierBa.score(x_test, y_test))
print('Ada_boost Accuracy Score: ',classifierAdaBoost.score(x_test, y_test))
print('Gradient Boosting Accuracy Score: ',classifierGBo.score(x_test, y_test))
print('Stacking Accuracy Score: ',clf.score(x_test, y_test))
```

OUTPUT:

```
CONCLUSION :
-----
Accuracy Report of Base Learning Algorithms:
-----
Accuracy of naive bayes: 0.7692307692307693
Accuracy of logistic regression: 0.7912087912087912
Accuracy of decision tree: 0.7582417582417582

Accuracy Report of Ensemble Classifiers:
-----
Bagging Accuracy Score: 0.8131868131868132
Ada_boost Accuracy Score: 0.7362637362637363
Gradient Boosting Accuracy Score: 0.8131868131868132
Stacking Accuracy Score: 0.7692307692307693
```

REFERENCES

- Ghotra, B., McIntosh, S., & Hassan, A. E. (2015, may). Revisiting the impact of classification techniques on the performance of defect prediction models. In proceedings of the 37th international conference on software engineering-volume 1 (pp. 789-800). IEEE press.
- Panichella, a., Oliveto, R., & De lucia, A. (2014, february). Cross-project defect prediction models: l'union fait la force. In 2014 software evolution week-ieee conference on software maintenance, reengineering, and reverse engineering (CSMR-WCRE) (pp. 164-173). Ieee.
- Mendes-moreira, j., Soares, C., Jorge, A. M., & Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey. *Acm computing surveys (csur)* , 45 (1), 10.
- Tavish srivastava “ basics of ensemble learning explained in simple english ” [analyticsvidhya.Com](http://analyticsvidhya.com).
- Di Nucci, D., Palomba, F., Oliveto, R., & De Lucia, A. (2017). Dynamic selection of classifiers in bug prediction: An adaptive method. *IEEE Transactions on Emerging Topics in Computational Intelligence* , 1 (3), 202-212