**Heart Disease Dataset Project**

**Introduction:**

**World Health Organization has estimated 12 million deaths occur worldwide; every year due to Heart diseases. Half the deaths in the United States and other developed countries are due to cardio vascular diseases. The early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications. This research intends to pinpoint the most relevant/risk factors of heart disease as well as predict the overall risk using logistic regression and other classification algorithms.**

This dataset gives a number of variables along with a target condition of having or not having heart disease. It contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date.

**Cause of Heart Disease:**

* Excess weight, especially around the stomach area, increases a woman's risk of developing cardiovascular disease and lack of physical activity makes it worse.
* Diabetes causes damage to blood vessels so diabetes is a major factor in developing cardiovascular disease.
* Unhealthy foods, lack of exercise, lead to heart disease. So can high blood pressure, infections, and birth defects.
* Smoking is one of the biggest causes of cardiovascular disease.
* Just a few cigarettes a day can damage the blood vessels and reduce the amount of oxygen available in our blood.

**Problem Statement:**

The given dataset tells the details of the patients who have heart disease or not, according to

the given features. We will try to use this data to predict whether a patient has heart disease or

not. There are 200 patient records in the dataset along with 14 columns.

**Dataset Description:**

The following are the descriptions of the features present in the dataset and they are as follows:

1.age - age in years

2.sex - (1 = male; 0 = female)

3.cp - chest pain type

4.trestbps - resting blood pressure (in mm Hg on admission to the hospital)

5.chol - serum cholesterol in mg/dl

6.fbs - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)

7.restecg - resting electrocardiographic results

8.thalach - maximum heart rate achieved

9.exang - exercise induced angina (1 = yes; 0 = no)

10.oldpeak - ST depression induced by exercise relative to rest

11.slope - the slope of the peak exercise ST segment

12.ca - number of major vessels (0-3) coloured by fluoroscopy

13.thal - 3 = normal; 6 = fixed defect; 7 = reversable defect

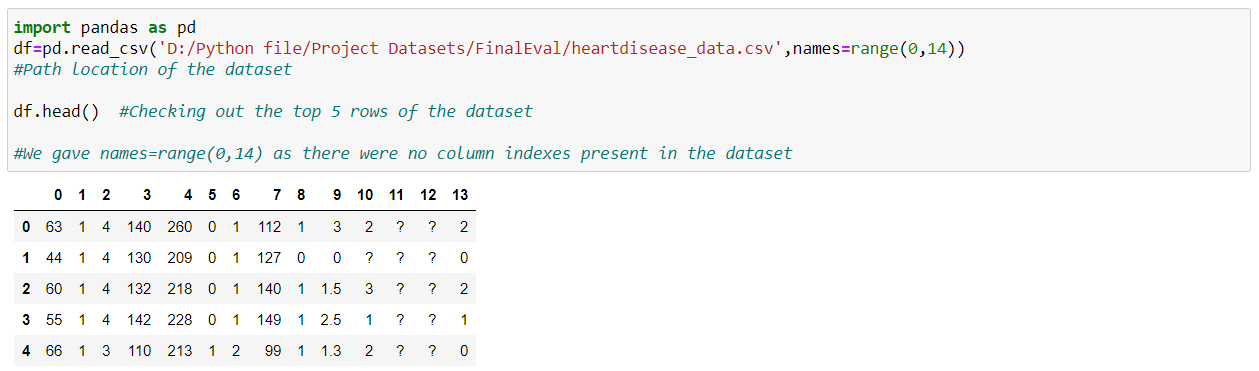
14.target - having disease or not (1=yes, 0=no)

We need to take target column as the target variable and predict whether a person has heart disease or not with the data given and since the target variable is of binary data, we need to approach by using classification analysis.

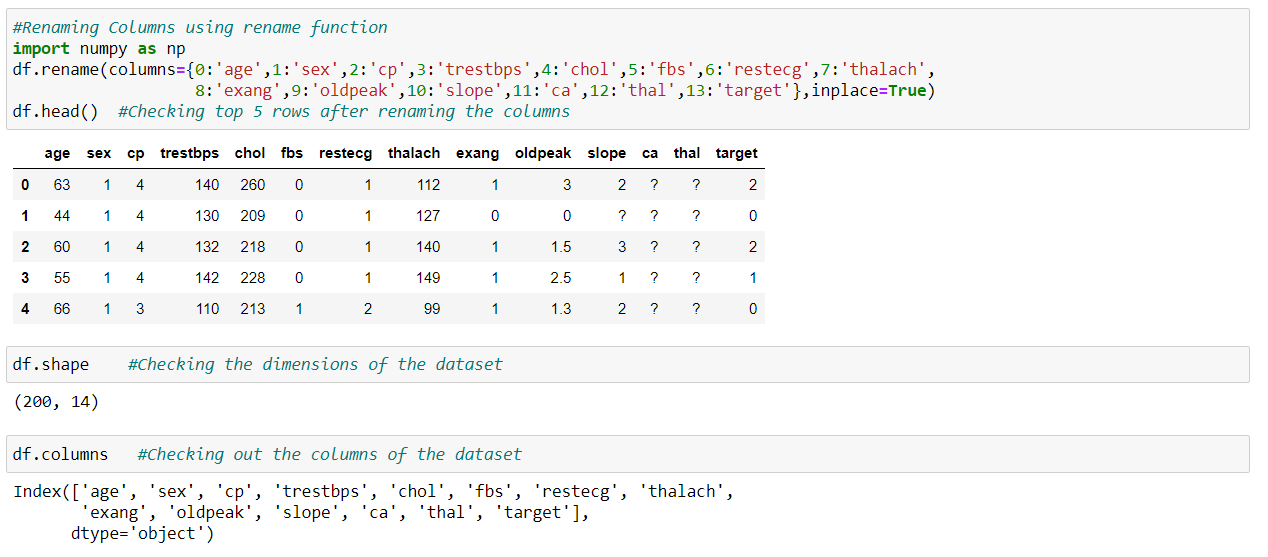
**Importing Warning Library:**



**Load the dataset and check out its features and dimensions:**



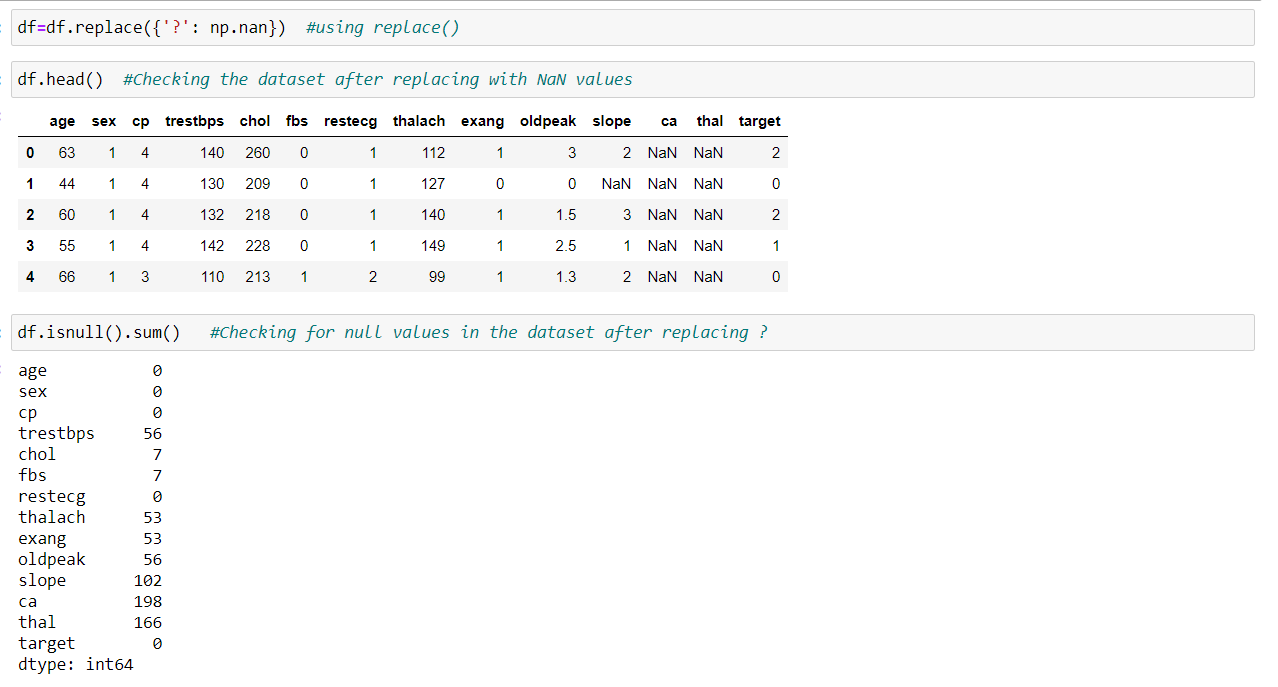
As there were no feature names in the dataset, we renamed the columns by using rename function



**Checking for any null values in the dataset:**

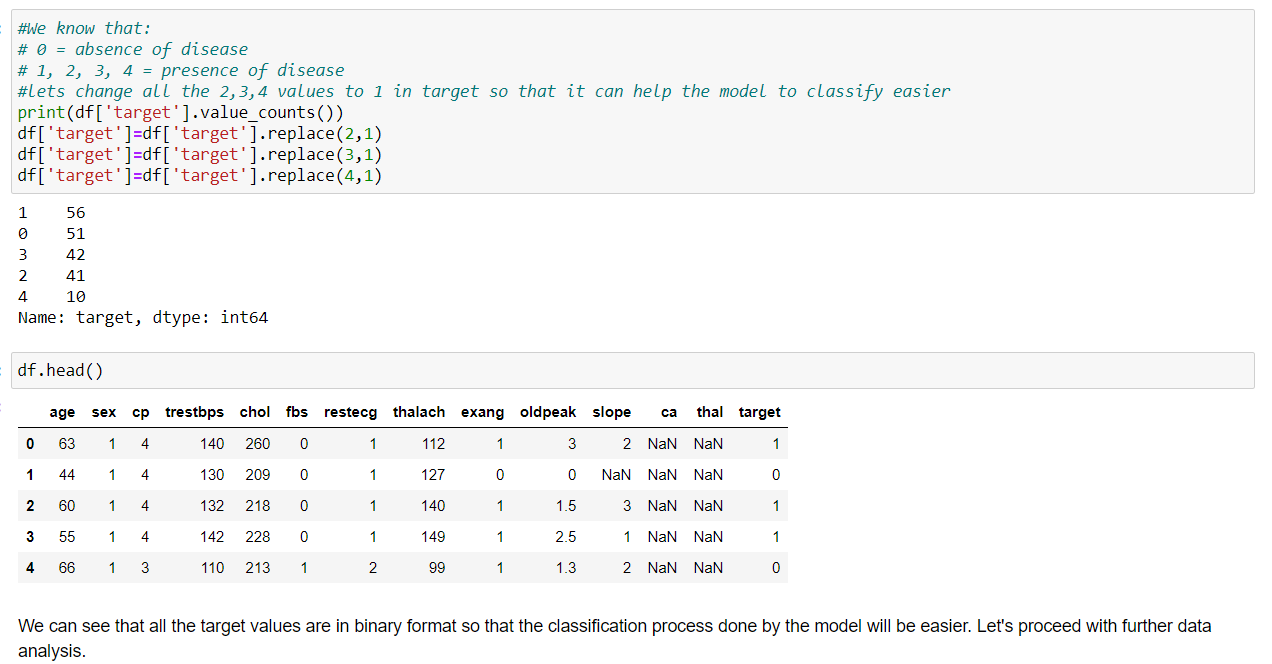


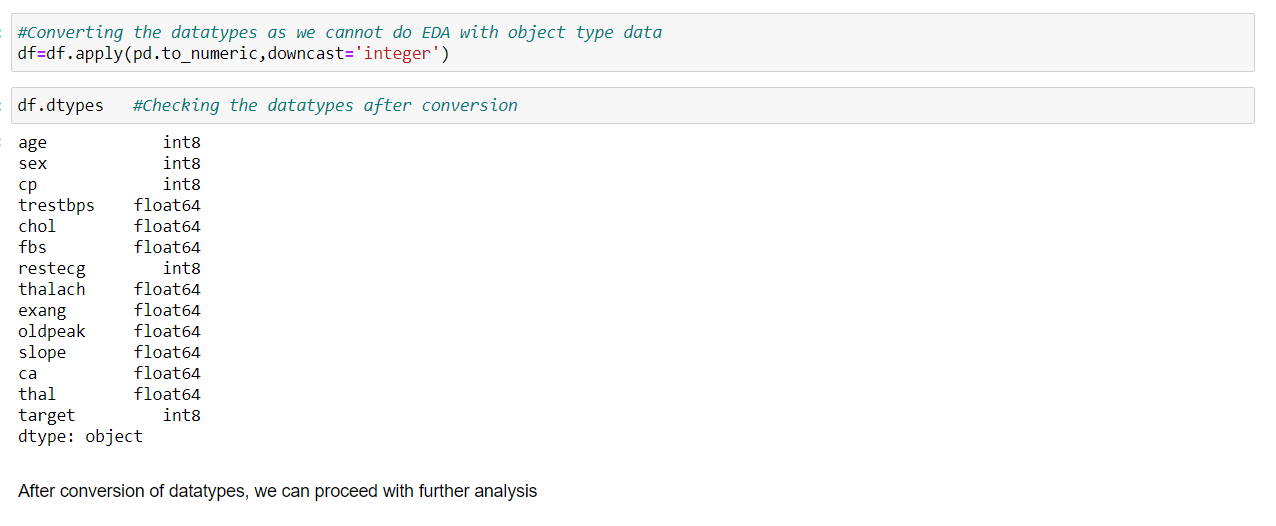
Even though there are no null values present, we can see that there are question marks '?' in many data and it means that there are missing values, although it is not considered as null values. We need to replace those values so that we can proceed with further analysis.



After replacing the ‘?’ values with null values and checking again, we can see that there are more missing data and we should handle them.

**Handling target column:**

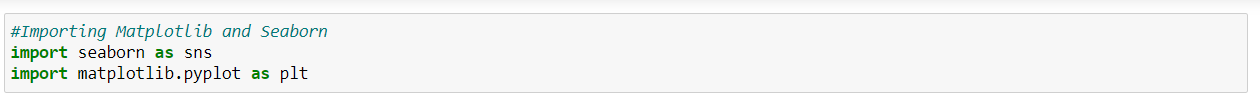




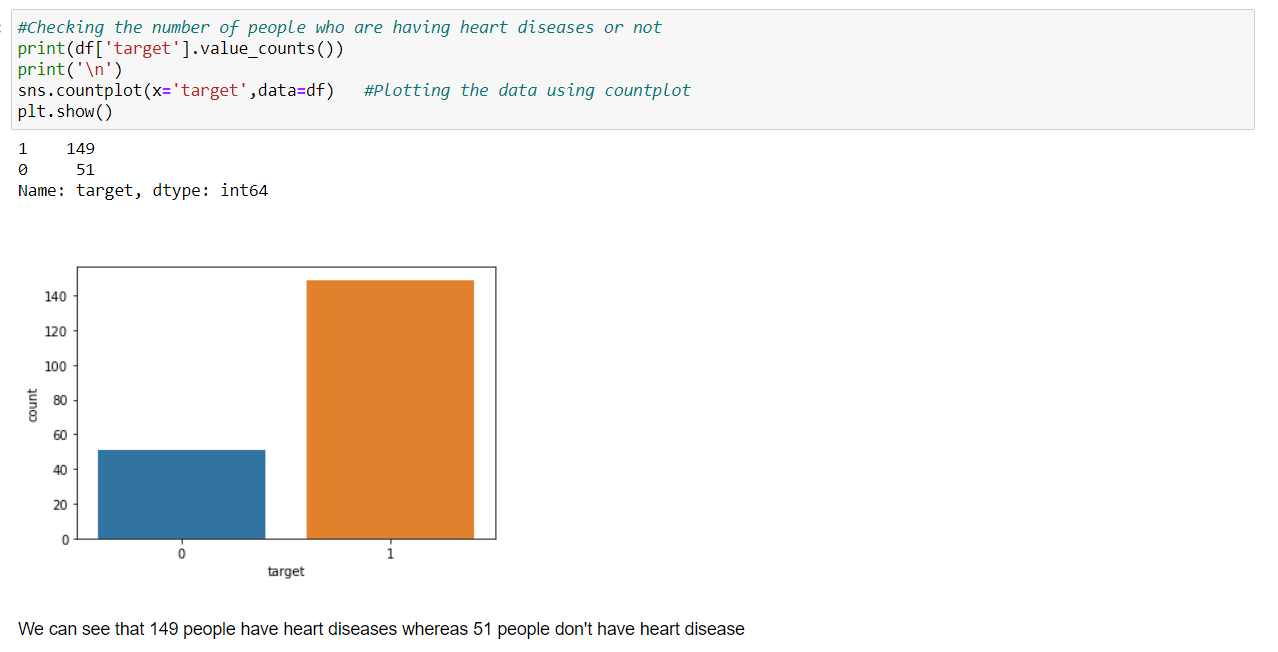
**Data Analysis:**

We will be using Exploratory Data Analysis (EDA) to check the relationship between features and also, we can understand the distribution of data in the dataset given. We plot graphs for the features by using data visualization libraries like matplotlib and seaborn.

**Importing matplotlib and seaborn library:**



**Checking the number of people who are having heart diseases or not:**



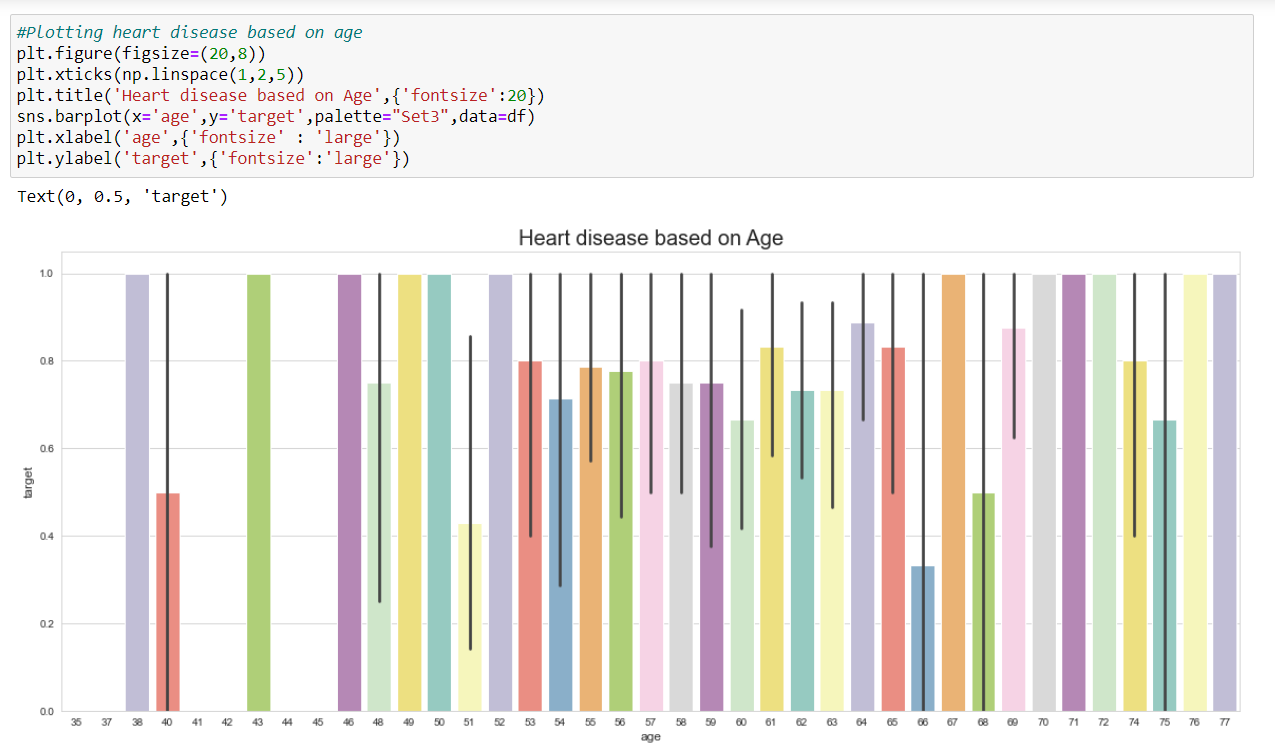
**Checking heart disease based on sex:**



**Plotting violin plot for checking the distribution of heart rate and cholesterol level:**



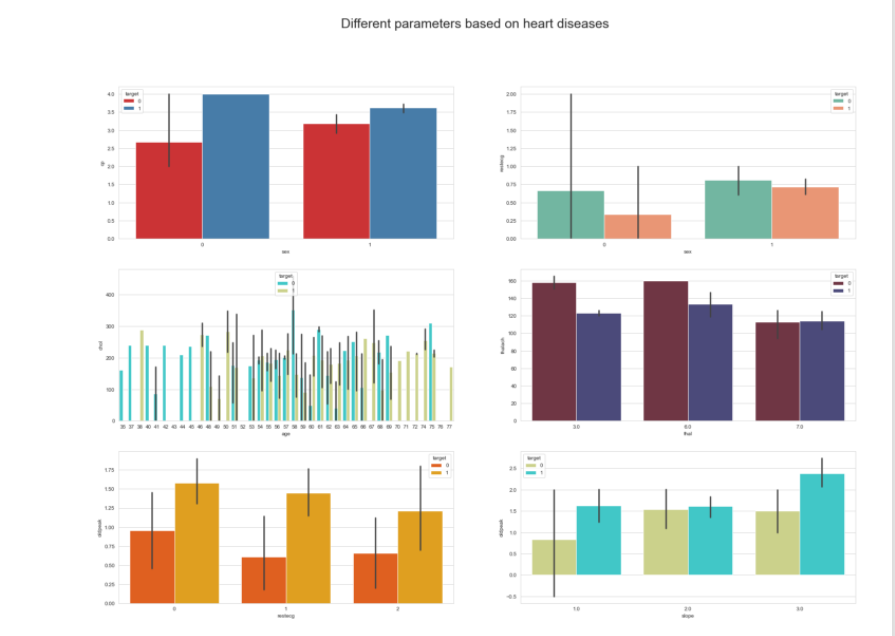
**Plotting heart disease based on age:**



We can see that the patients at age 38, 43, 46, 49, 50, 52, 67, 70, 71, 72, 76 and 77 are having the high risk of heart diseases. Age above 40+ are having high heart diseases rate.

**Plotting different parameters based on Target:**





Observations:

1.We can see that female patients are having more chest pain, when they have heart disease, compared to male patients. Male patients who don't have any heart diseases are having less chest pain.

2.The restecg results of male patients who don't have heart diseases are normal, whereas for female patients who have heart diseases are having less value, which means the risk is higher for them.

3.The cholesterol level should not be more than 200mg/dl and we can see that many patients have high cholesterol which is one of the main factors for getting heart diseases.

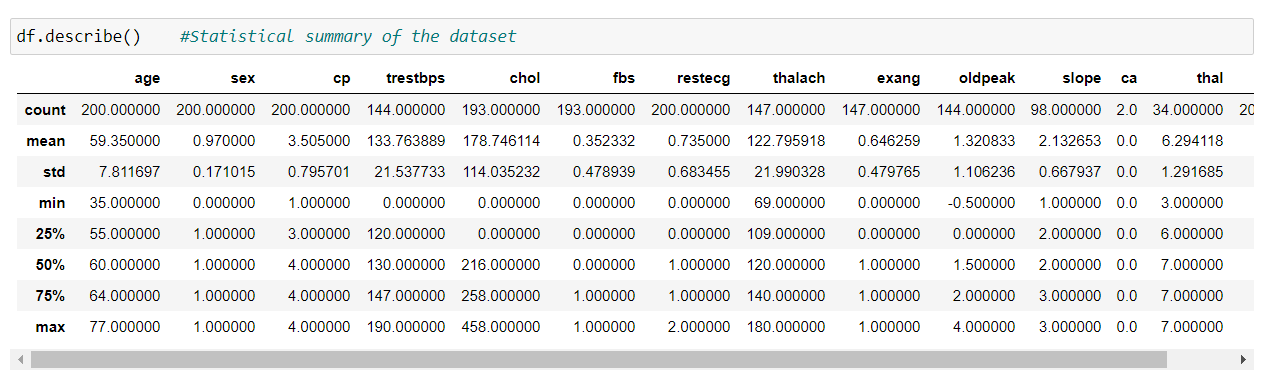
4.The heart rate for patients who are not having heart diseases is higher for thal=3.0 and 6.0, whereas for thal=7.0 both heart and non-heart disease patients is equal.

5.We can observe that if ST depression induced by exercise relative to rest in patients, we can observe more case of ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) in the patient.

6.We can see that the slope of the peak exercise ST segment is more for heart disease patients.

7.From the graphical representations, we can see that female patients are in high risk of getting heart diseases.

**Checking the Statistical summary of the dataset:**



Observations:

1.The minimum age of the person is 35 and the maximum age of the person is 77.

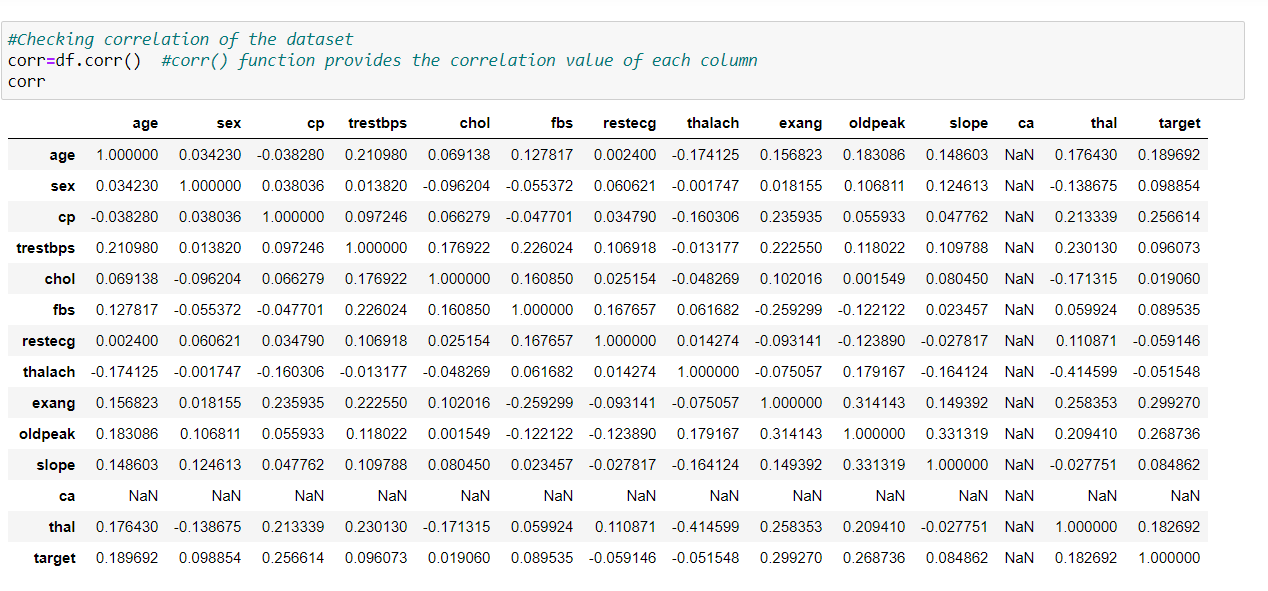
2.There is not much difference between the mean and median.

3.In columns like trestbps, chol and thalach the difference between 75th percentile and max is large and it is due to the presence of outliers.

4.The min value for most of the columns is 0.

**Correlation:**

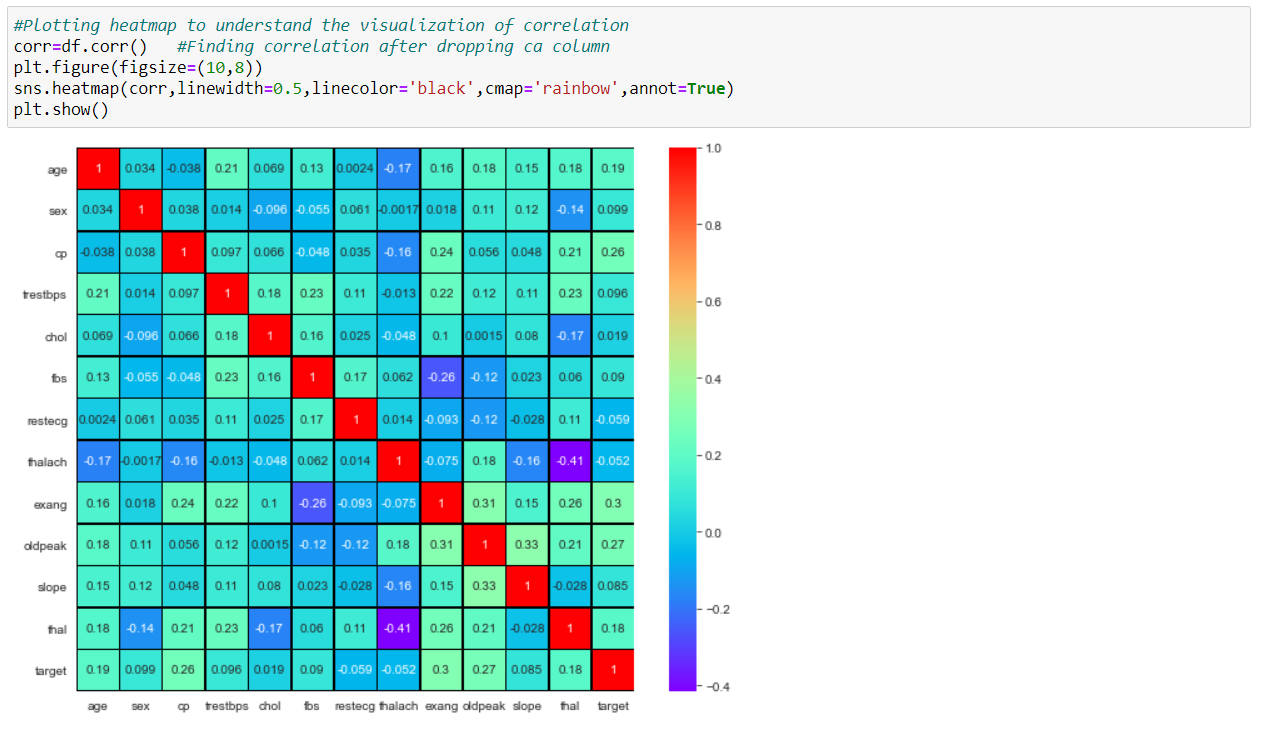
The statistical relationship between two variables is referred to as their correlation. The correlation factor represents the relation between columns in a given dataset. A correlation can be positive, meaning both variables are moving in the same direction or it can be negative, meaning that when one variable's value increasing, the other variable’s value is decreasing.



We can see that ca values are having more missing data and also it is not even correlated with target. Therefore, we can drop that column.

df. drop ('ca’, axis=1, inplace=True)

**Plotting heatmap for visualizing the correlation:**



Observations:

1.We can see that nearly 90% of the data is positively correlated with each other.

2.oldpeak and slope column have the highest positive correlation with each other at a value of 33%.

3.thal and thalach have the highest negative correlation with each other at a value of -41%.

4.Positive correlated columns have high impact with the target variable, whereas negatively correlated column has less or zero impact with the target variable.

**Data pre-processing:**

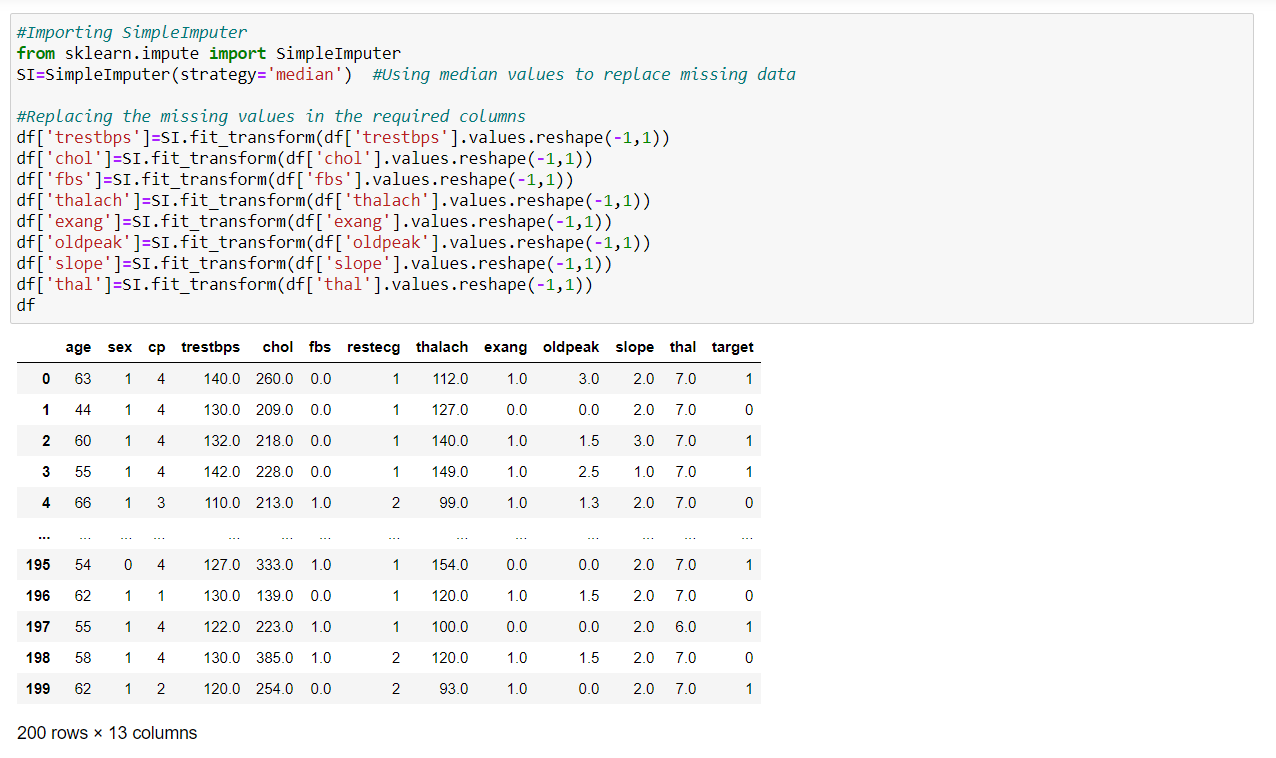
**Handling missing data by using Simple Imputer:**

**Simple Imputer** is a scikit-learn class which is helpful in handling the missing data in the predictive model dataset. It replaces the NaN values with a specified placeholder.  
It is implemented by the use of the **SimpleImputer()** method which takes the following arguments:

**-> missing\_values:** The missing\_values placeholder which has to be imputed. By default, is NaN

**-> strategy:** The data which will replace the NaN values from the dataset. The strategy argument can take the values – ‘mean'(default), ‘median’, ‘most\_frequent’ and ‘constant’.

**-> fill\_value:** The constant value to be given to the NaN data using the constant strategy.



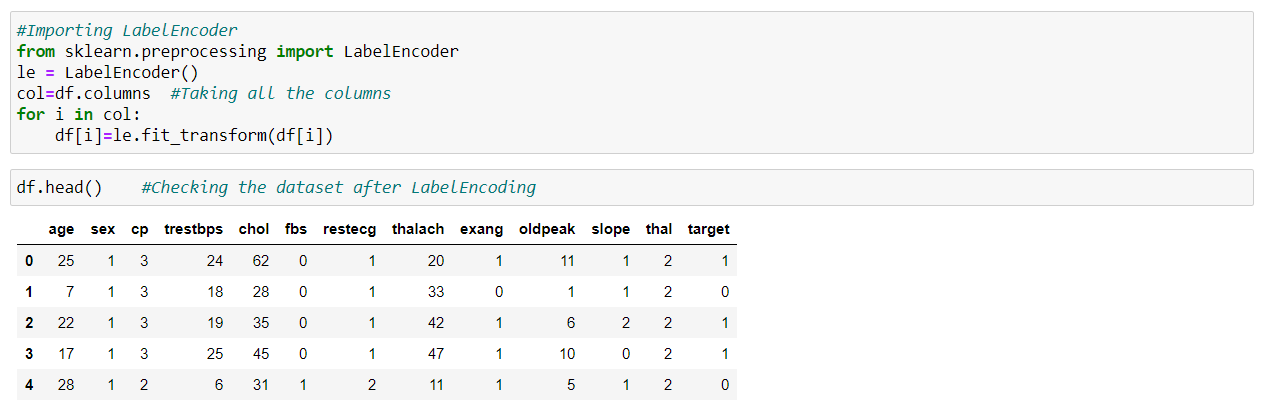
Here, we are using median as the strategy to fill up the missing data in the columns which have missing values.

**Label Encoding:**

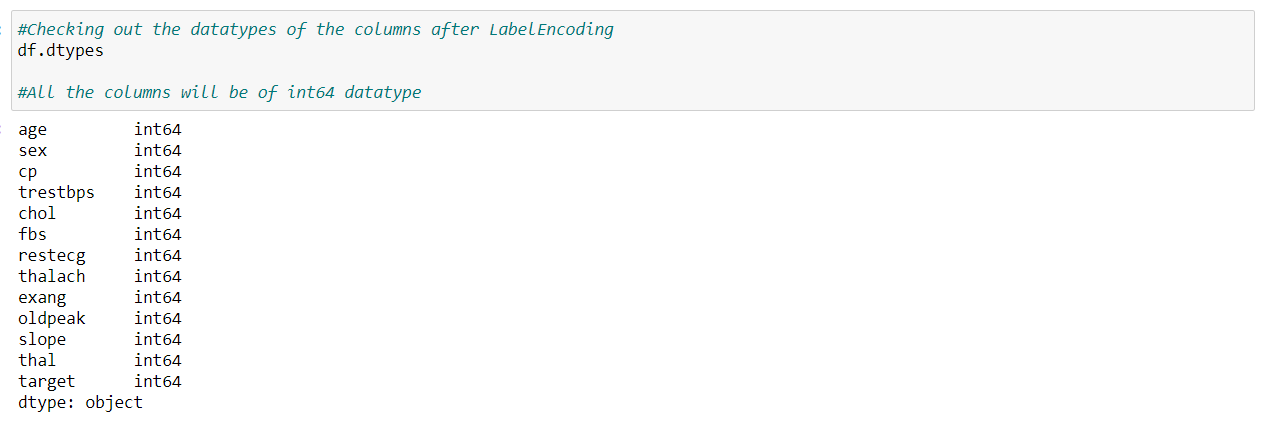
As class column has categorical data, we need to convert into numerical for proceeding further analysis and it can be done by using Label Encoder.

**Label Encoder** - refers to converts the labels into numeric form i.e., the columns which are in alphabetical or categorical values are assigned with numbers so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated during the process.

Import label encoder for converting categorical data (object) into numeric (int 64) data.



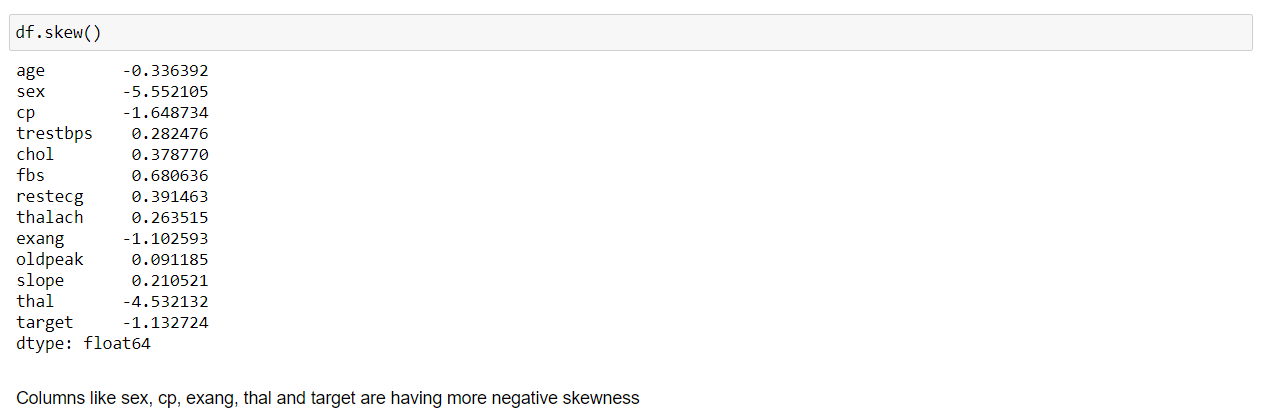
We see that all the columns are converted from categorical into numeric data by using Label Encoder.



**Skewness:**

Skewness is a measure of the symmetry of a distribution. The highest point of a distribution is its mode. The mode marks the response value on the x-axis that occurs with the highest probability. A distribution is skewed if the tail on one side of the mode is fatter or longer than on the other: it is asymmetrical.

In an asymmetrical distribution a **negative skew** indicates that the tail on the left side is longer than on the right side (left-skewed), conversely a **positive skew** indicates the tail on the right side is longer than on the left (right-skewed). Asymmetric distributions occur when extreme values lead to a distortion of the normal distribution.

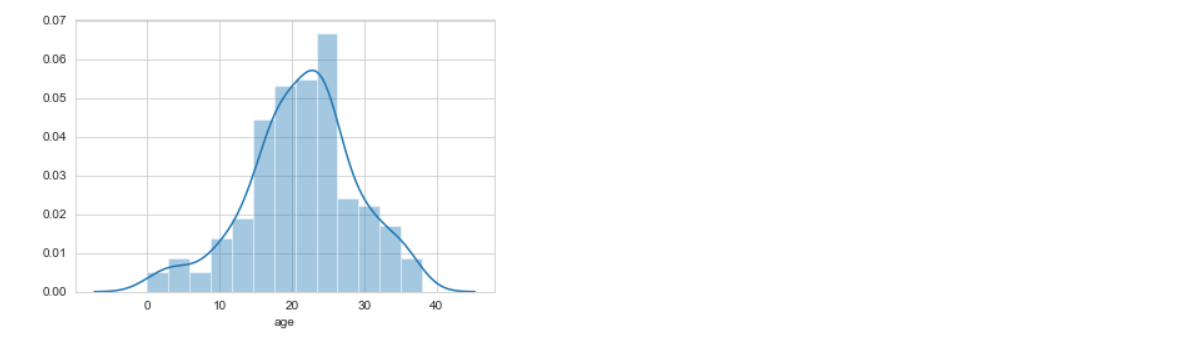


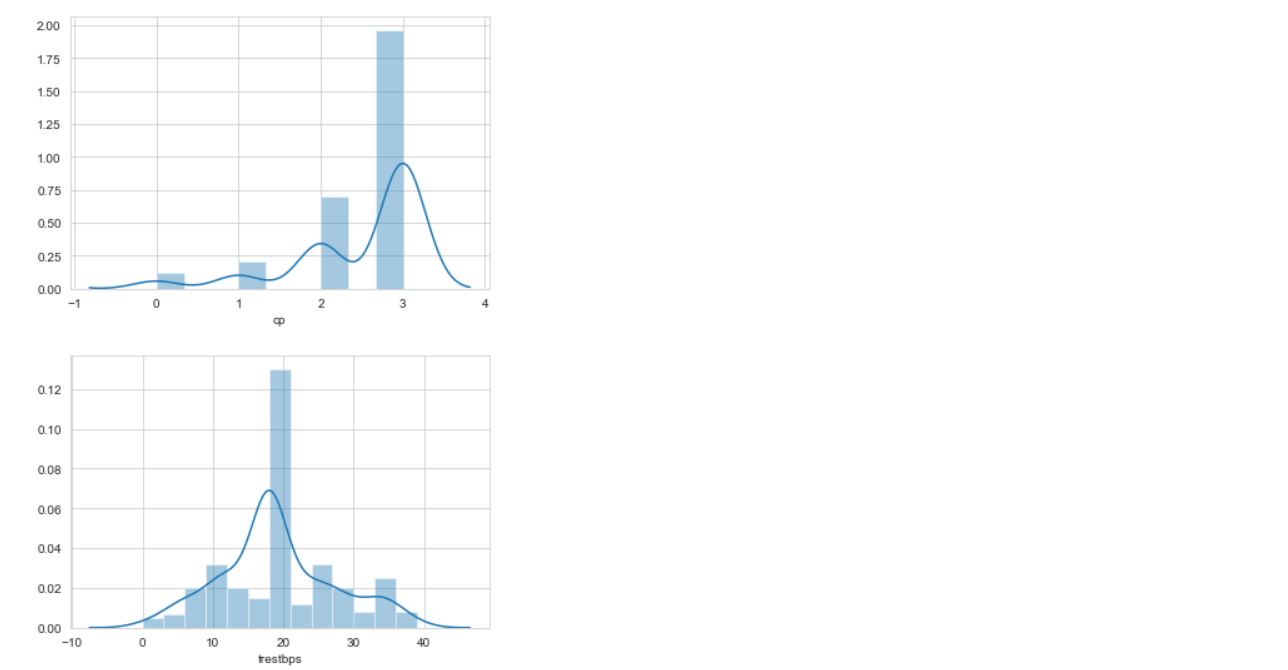
The range of skewed data is between -0.55<df<0.55 and if the value is obtained above the range, then it is said to be skewed. We can check the distribution of the features data using distplot.

**Plotting distplot:**

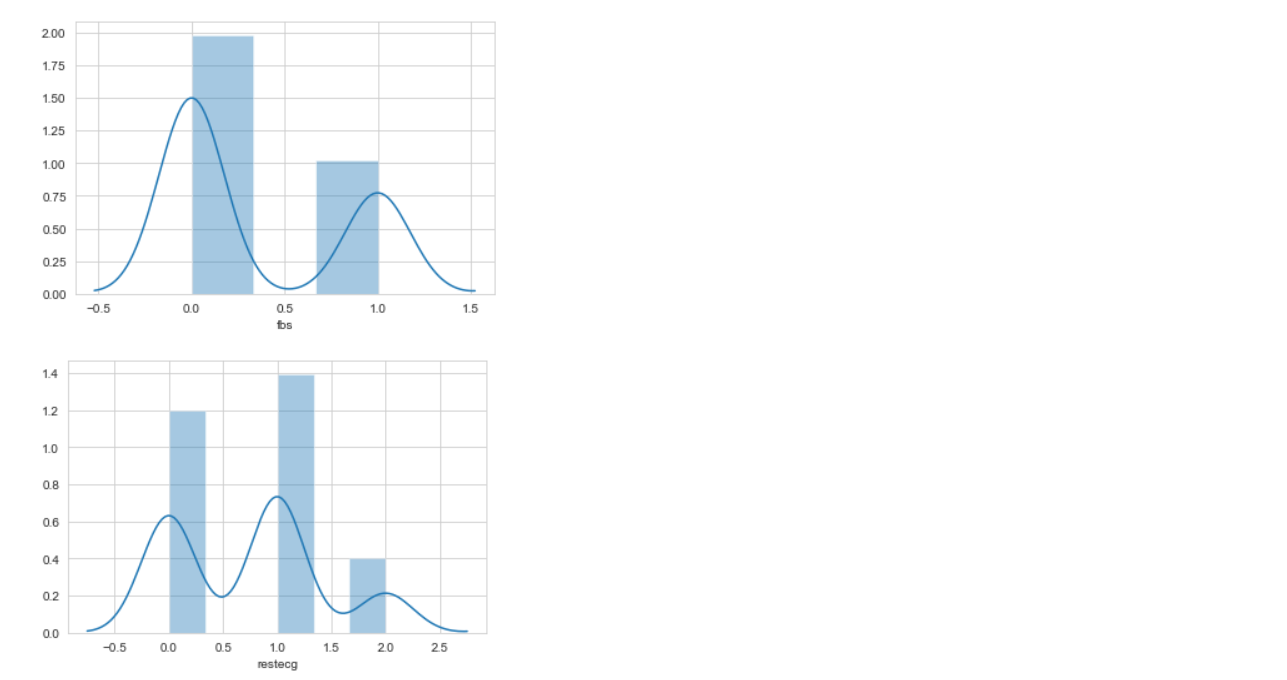


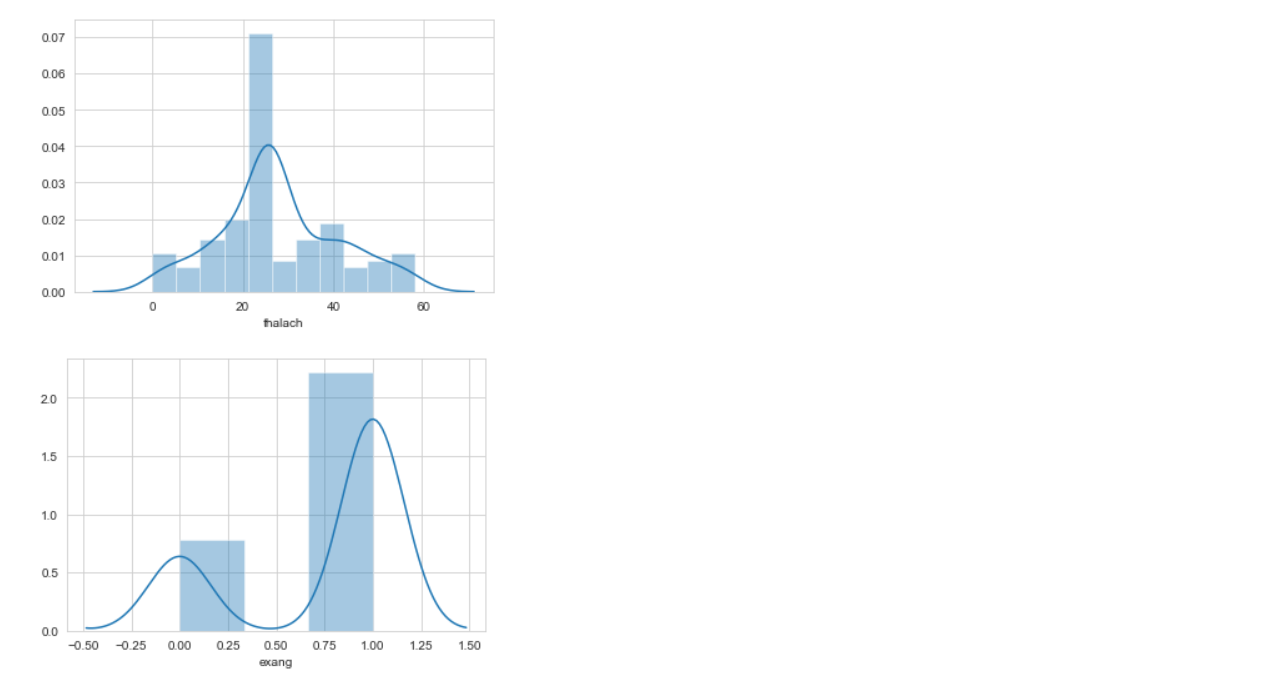
If we run a loop with all the columns for plotting the distplot, we can see that all the columns would be plotted simultaneously. Below is the graphical distribution of the columns data according to the skewness of each column:

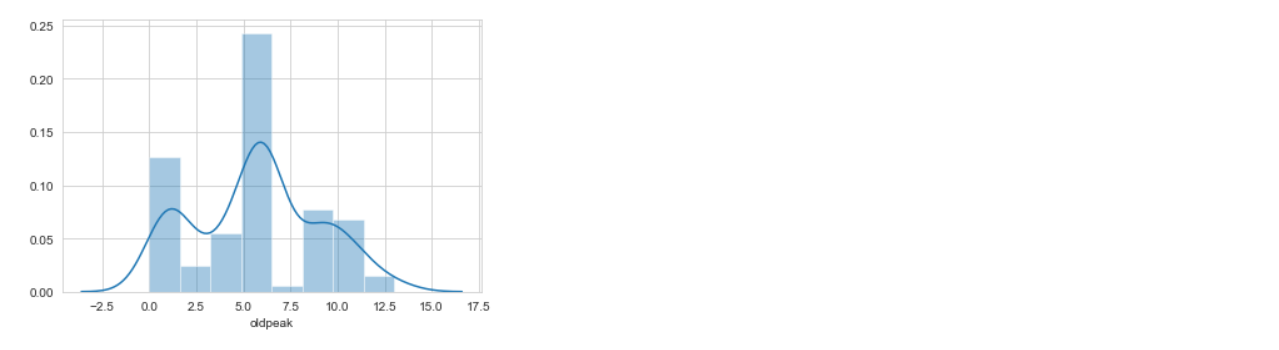


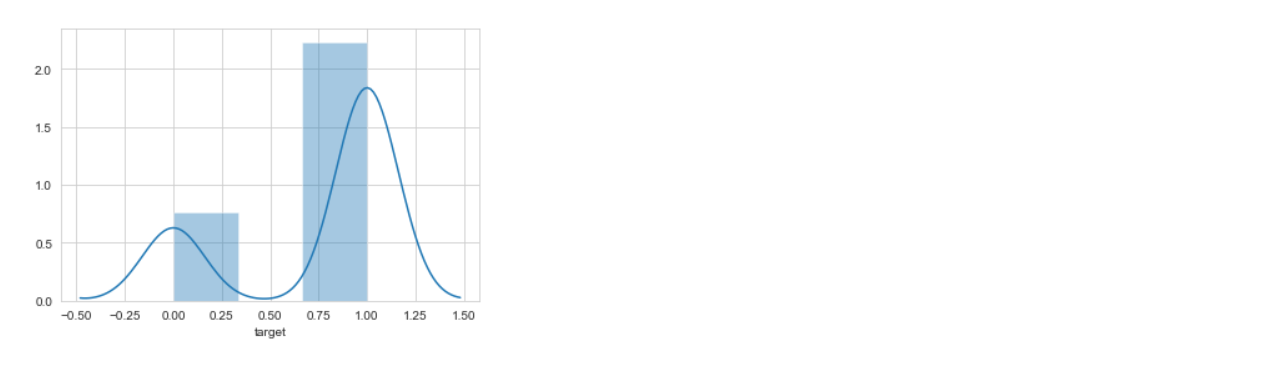










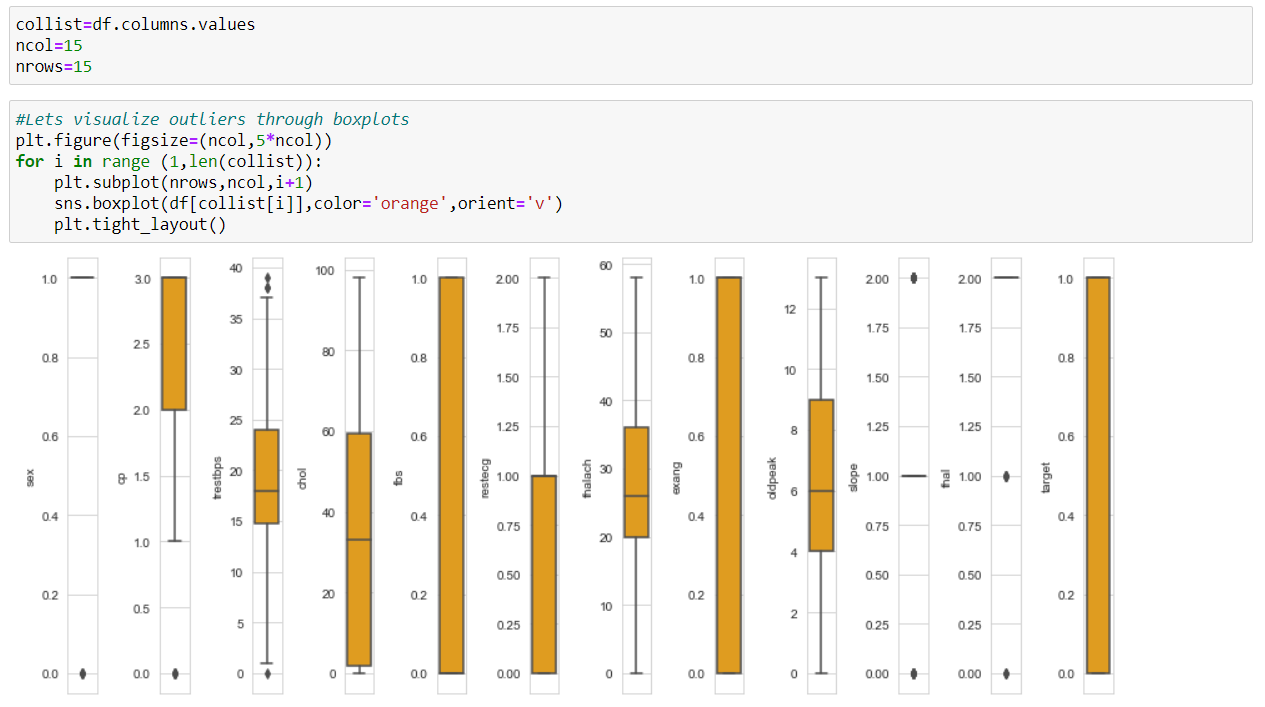


Most of the features are not normally distributed and it is due to the presence of outliers.

**Outliers:**

In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

We can check outliers present in the dataset by plotting boxplot.

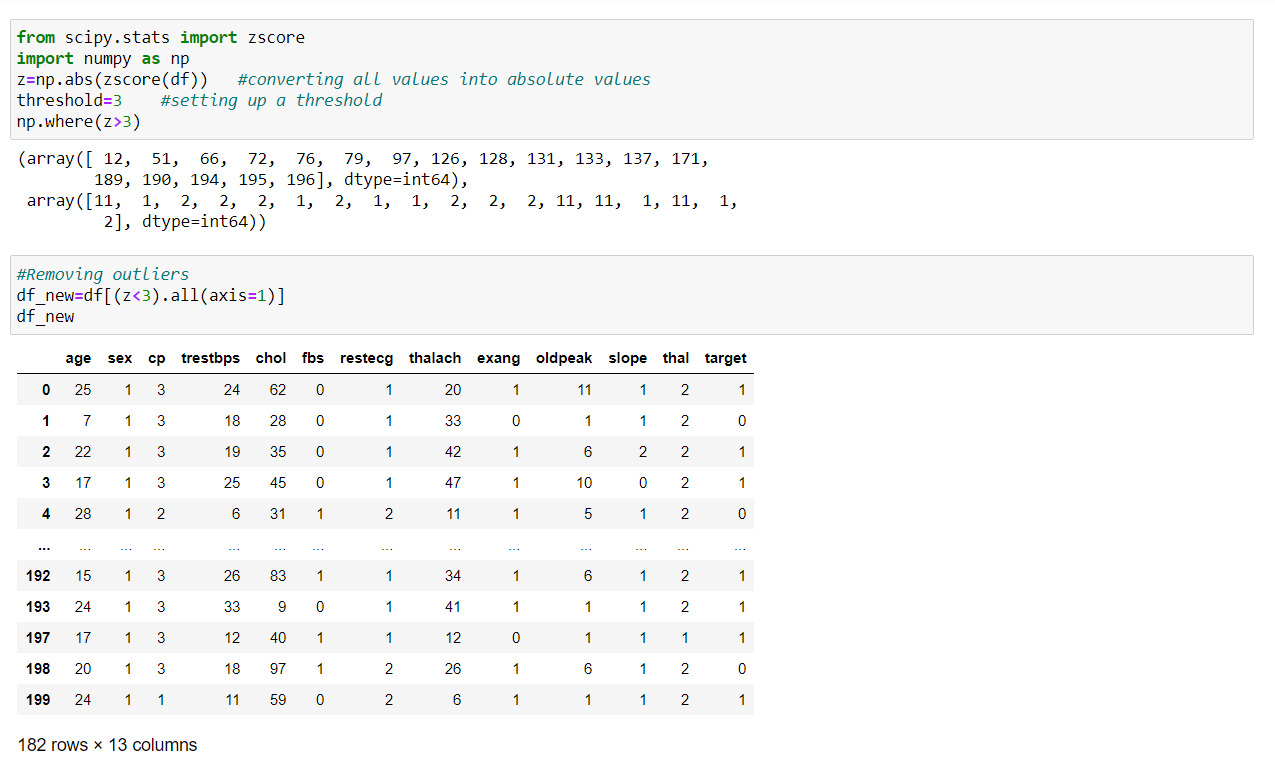


We can see that there are outliers present in the data and we can handle them by using z-score method.

**Z-score:**

A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. If a Z-score is 0, it indicates that the data point's score is identical to the mean score. A Z-score of 1.0 would indicate a value that is one standard deviation from the mean. Z-scores may be positive or negative, with a positive value indicating the score is above the mean and a negative score indicating it is below the mean.

First, we will convert all values to absolute values and then we will set a threshold value for identifying the outliers and with that value, we will be removing the outliers from the dataset and check the number of rows affected. Below is the procedure to remove outliers:



We can see that the original dataset had 200 rows of data and after removal of outliers, we can see that nearly 18 rows of data had outliers and they had been removed by using z-score method.

**Model Training:**

We need to split independent and dependent variables from the dataset in order to perform further analysis



**Treating skewness:**

We can treat the skewness by using methods like log-transform, square-root transform, cube-root transform, etc. Below is the way to treat skewness:

#We are removing skewness from independent variables by using log transform

for col in df\_x. columns:

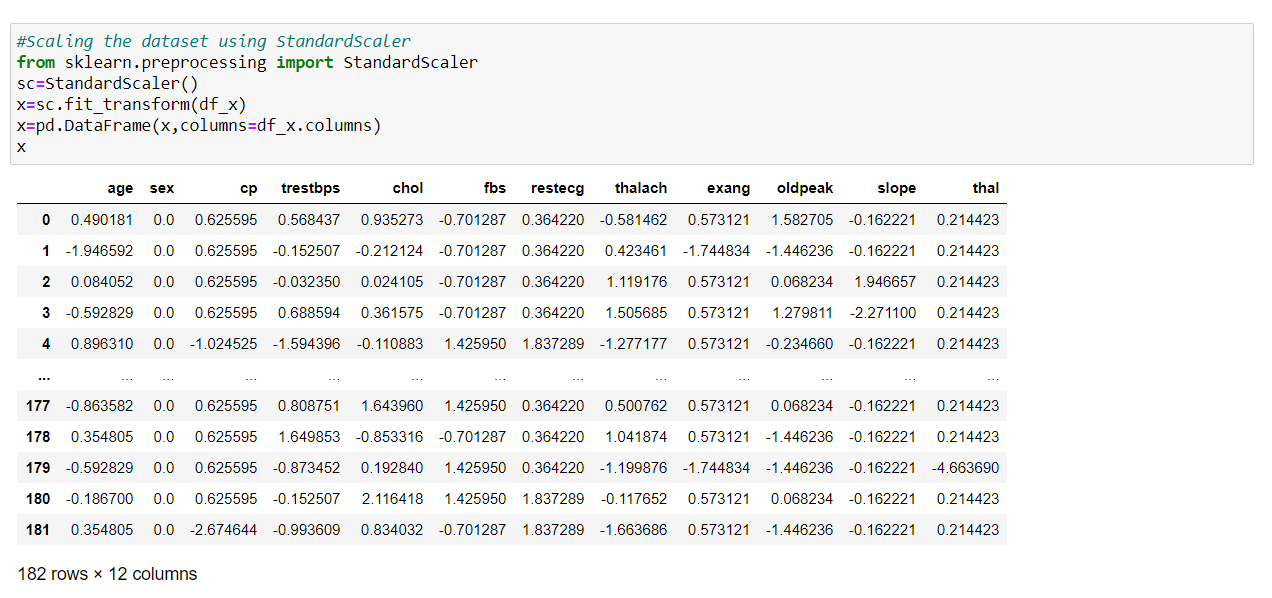
if df\_x. skew(). loc[col]>0.55:

df\_x[col]=np.log1p(df\_x[col])

**Scaling the data:**

Sometimes model can be biased to higher values in dataset, so it is better to scale the dataset so that we can bring all the columns in common range. We can use Standard Scaler here.

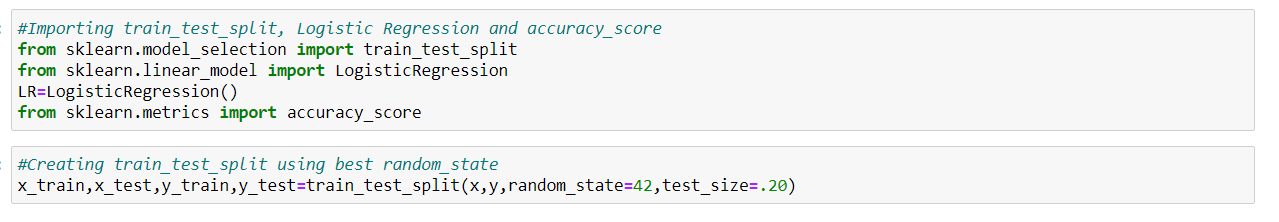
**Standard Scaler:** It performs the task of Standardization. Usually, a dataset contains variables that are different in scale. It will **normalize the feature,** i.e., each column of X, **individually,** so that each column/feature/variable will have μ = 0 and σ = 1.



We will scale the data first and then pass the data into a new Data Frame for further analysis.

**Building the Machine Learning models:**

After scaling the data, we can proceed with model building process. First, we will import the metrices and model required for finding the best random state:



The model performs well at random\_state= 42 so that we are creating our train\_test\_split with that.

**Finding out the best model:**

We will be using classification models like Logistic Regression, SVC, GaussianNB, DecisionTreeClassifier and KNeighborsClassifier algorithms for finding out the best model among those. Below are the definitions of the models mentioned:

**1. Logistic regression** - is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

**2.** **SVC** - “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. It is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving a SVM model sets of labelled training data for each category, they're able to categorize new text.

**3.** **Gaussian Naive Bayes** - algorithm is a special type of NB algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a gaussian distribution i.e., normal distribution. A Gaussian classifier is a generative approach in the sense that it attempts to model class posterior as well as input class-conditional distribution. Therefore, we can generate new samples in input space with a Gaussian classifier.

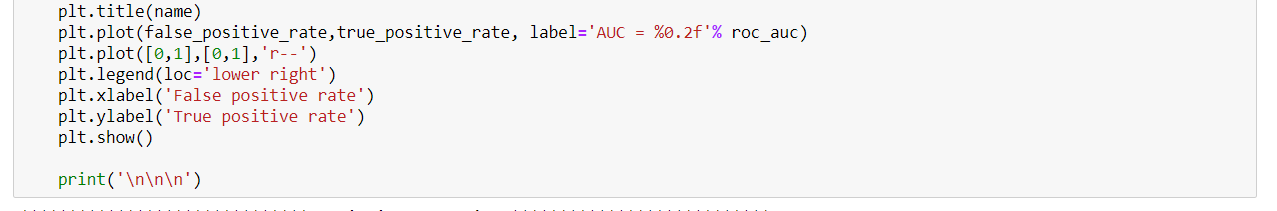
**4.** **Decision Tree Classifier** - Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

**5. KNeighbors** **Classifier** - It is a method based on k-nearest neighbors. In the KNeighbors model target is predicted by local interpolation of the targets which associated to the nearest neighbors in the training set. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification).

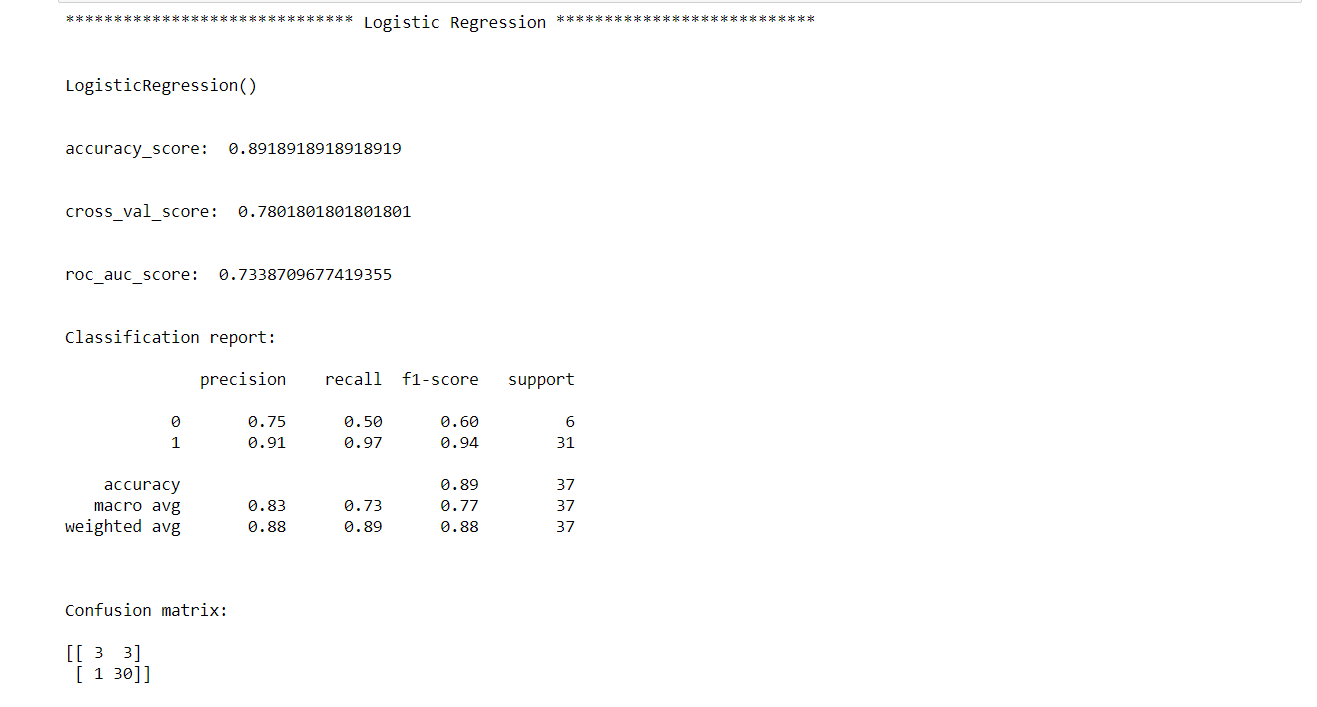
Importing all necessary libraries and metrices required:

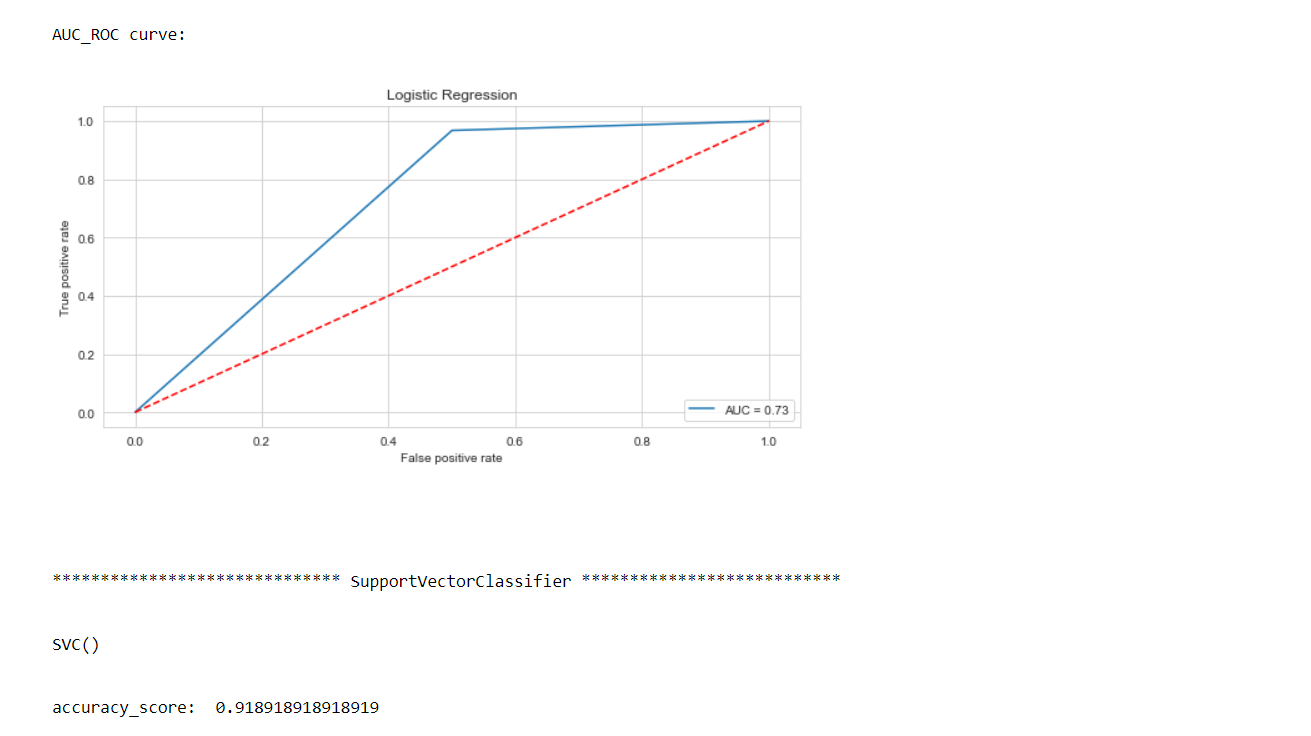


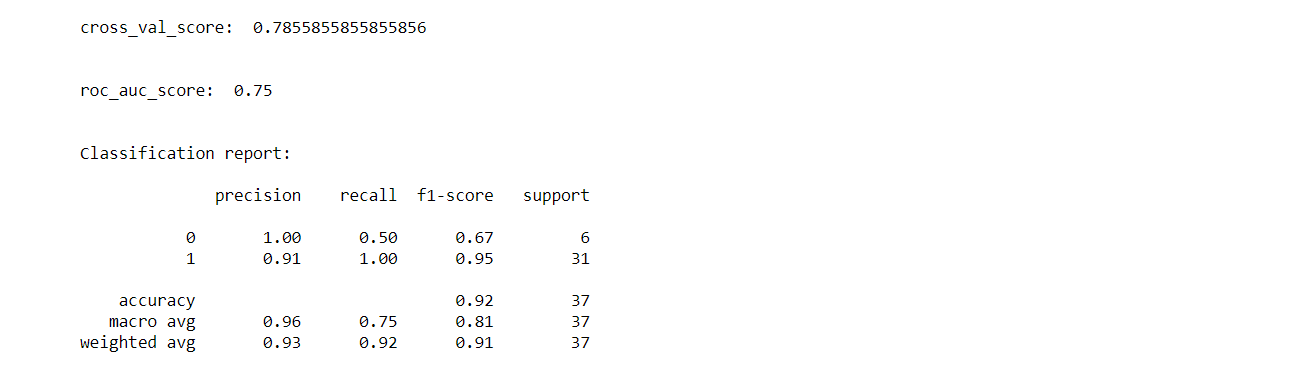
We will be using a for loop for checking the performance metrics of all algorithms together

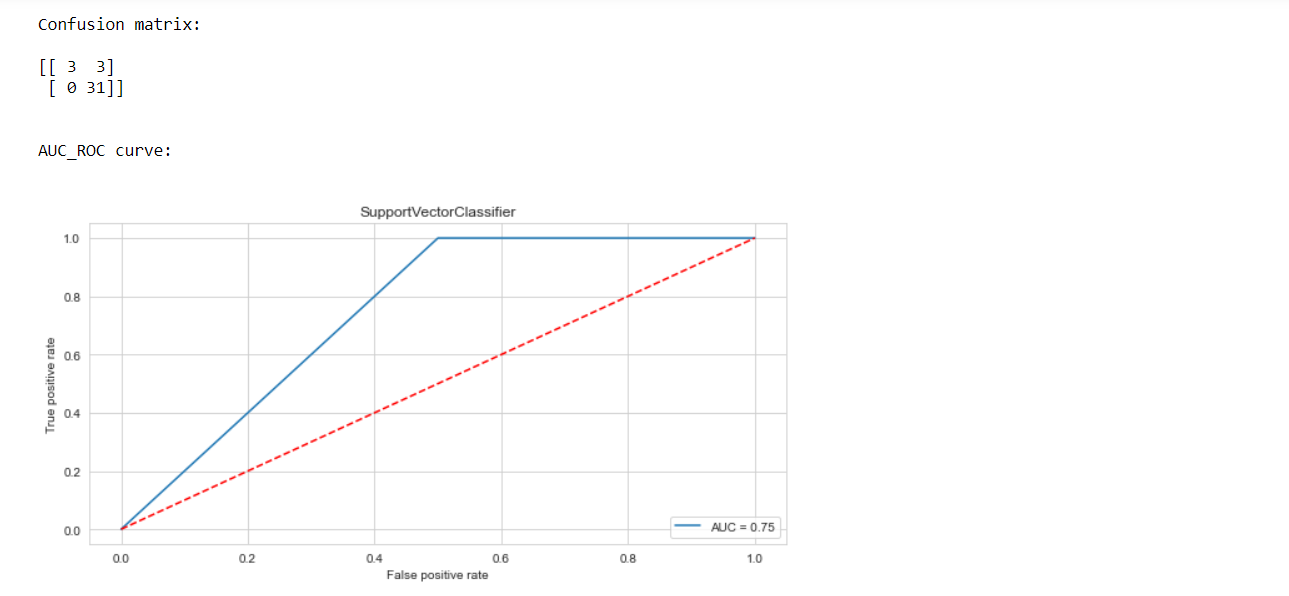


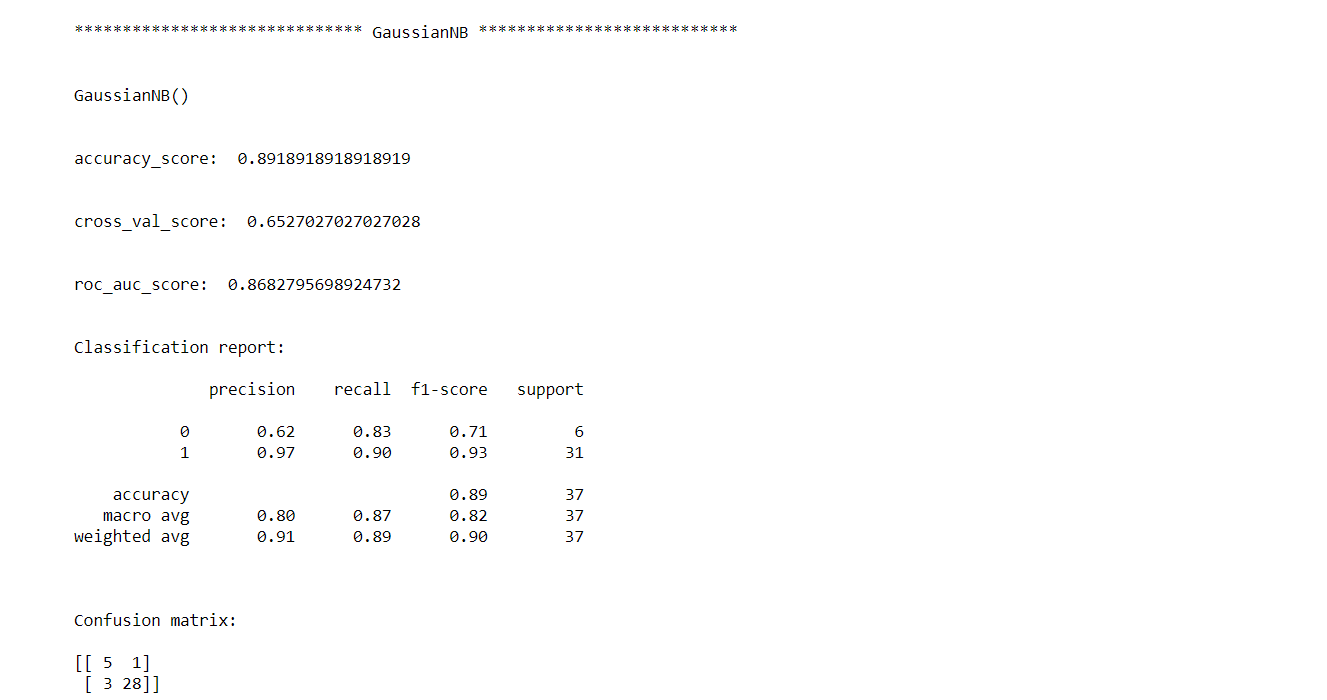
After running this loop, the following outputs will be obtained:

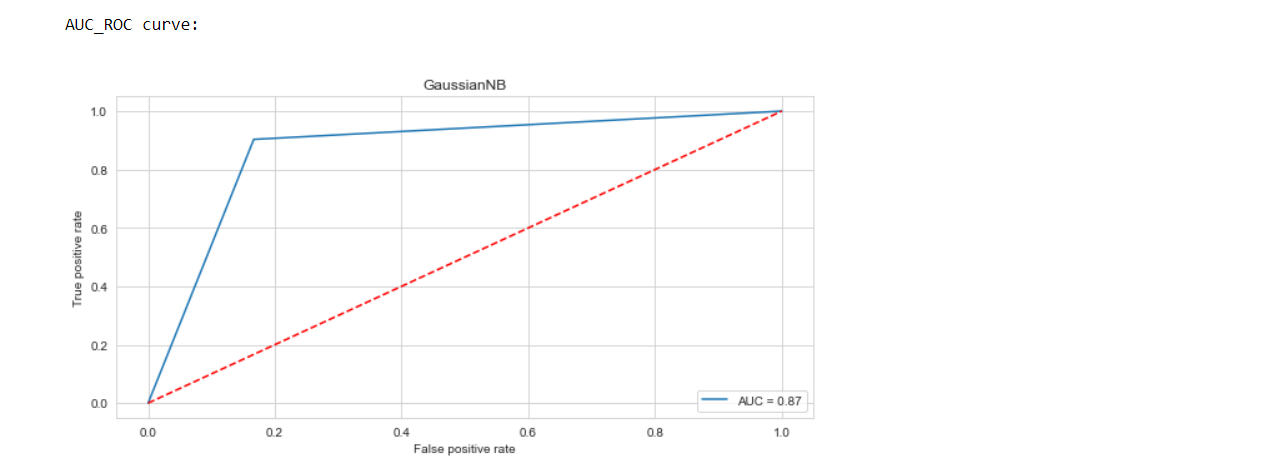


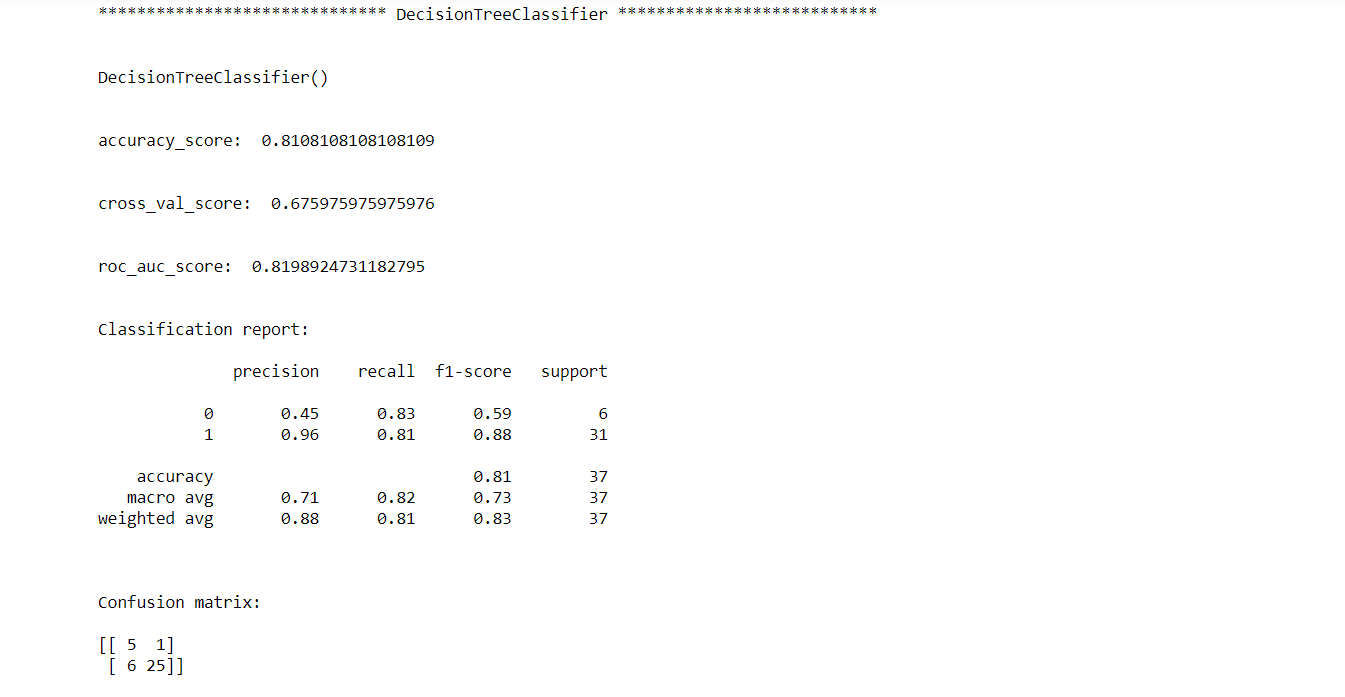


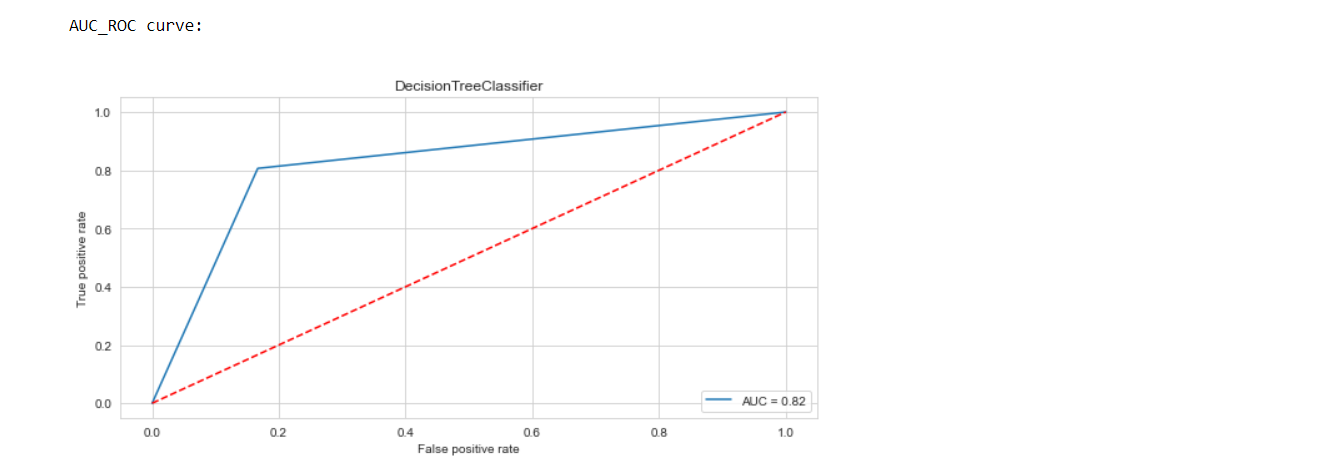


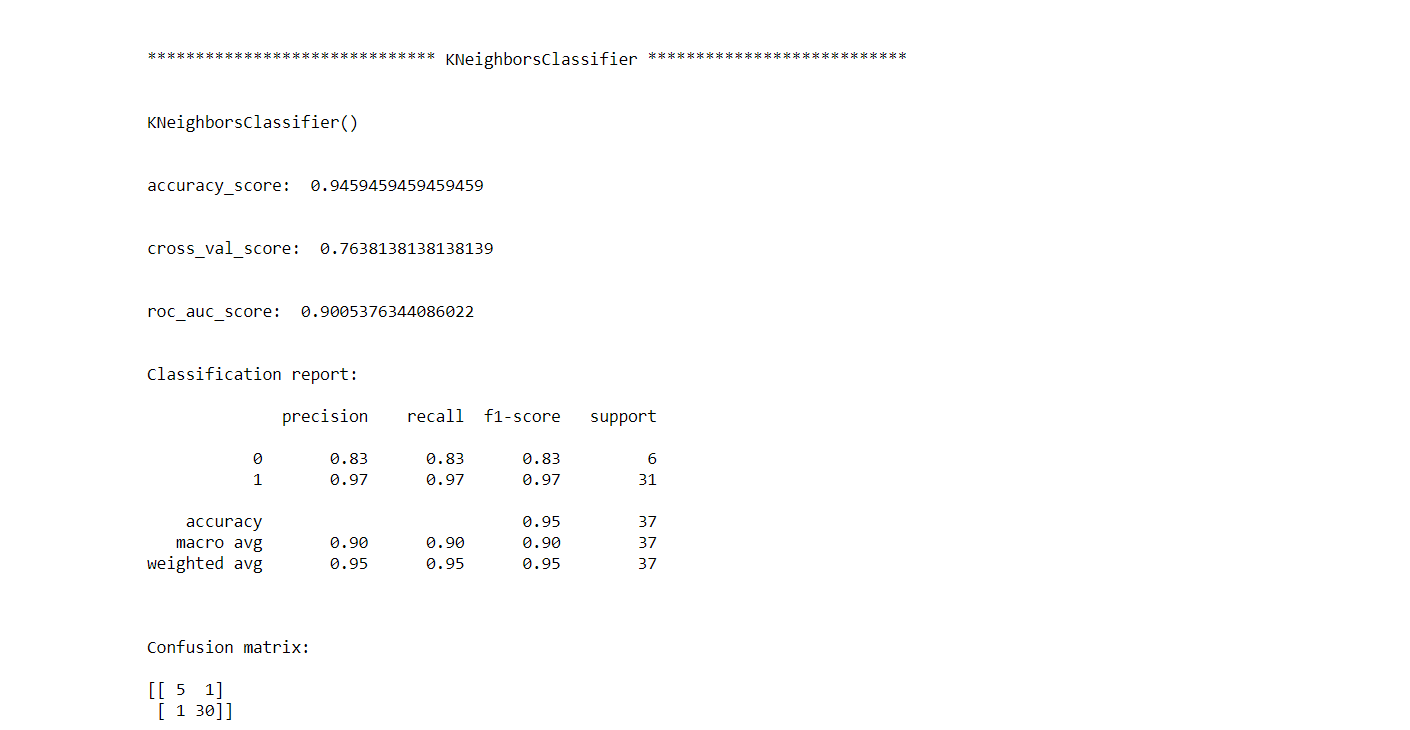


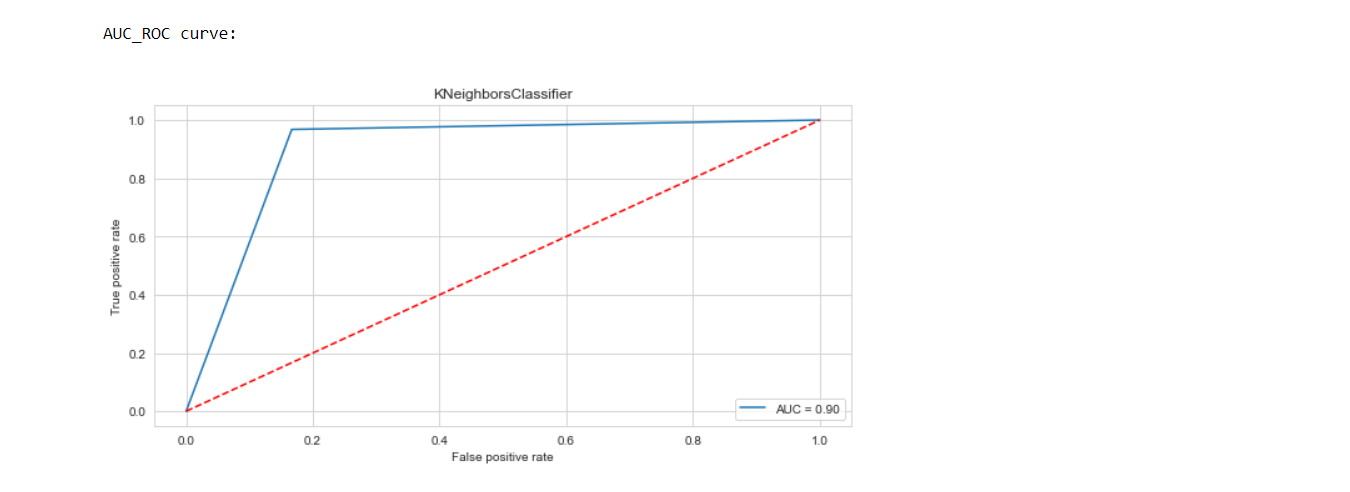












**Definitions of the metrices:**

**1. Cross Validation:**

Cross-validation helps to find out the over fitting and under fitting of the model.In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

**2. Confusion Matrix:**

 A **confusion matrix**, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a **matching matrix**). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabeling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

**3. Classification Report:**

The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative**: the case was negative and predicted negative
2. **TP / True Positive**: the case was positive and predicted positive
3. **FN / False Negative**: the case was positive but predicted negative
4. **FP / False Positive**: the case was negative but predicted positive

**Precision:** Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below:

Precision = TP/ (TP + FP)

**Recall:** Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below:

Recall = TP/(TP+FN)

**F1 score:** The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is:

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

**Support:** Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn’t change between models but instead diagnoses the evaluation process.

**4. AUC-ROC Curve and score:**

AUC (**Area Under the Curve**) - ROC (**Receiver Operating Characteristics**) curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease. The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

Score is the area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

From the above algorithms, KNeighbors Classifier algorithm is working well as it is giving good accuracy score and cross validation score. Now we will try to find out the best parameters and improve our scores by using Hyperparameter Tuning technique.

**Hyperparameter Tuning:**

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as **Hyperparameters.**These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as**Hyperparameter Tuning. We can do tuning by using GridSearchCV.**

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.



We can see that our accuracy score got decreased after Hyperparameter Tuning. For boosting the accuracy score, we can use Ensemble Techniques like RandomForestClassifier and AdaBoostClassifier.

**Ensemble Techniques:**

**Ensemble methods** is a machine learning technique that combines several base models in order to produce one optimal predictive model. Ensemble Methods allow us to take a sample of Decision Trees into account, calculate which features to use or questions to ask at each split, and make a final predictor based on the aggregated results of the sampled Decision Trees.

We used Random Forest Classifier and AdaBoost Classifier algorithms to boost up our scores and finding the best model.

**1. Random Forest:**

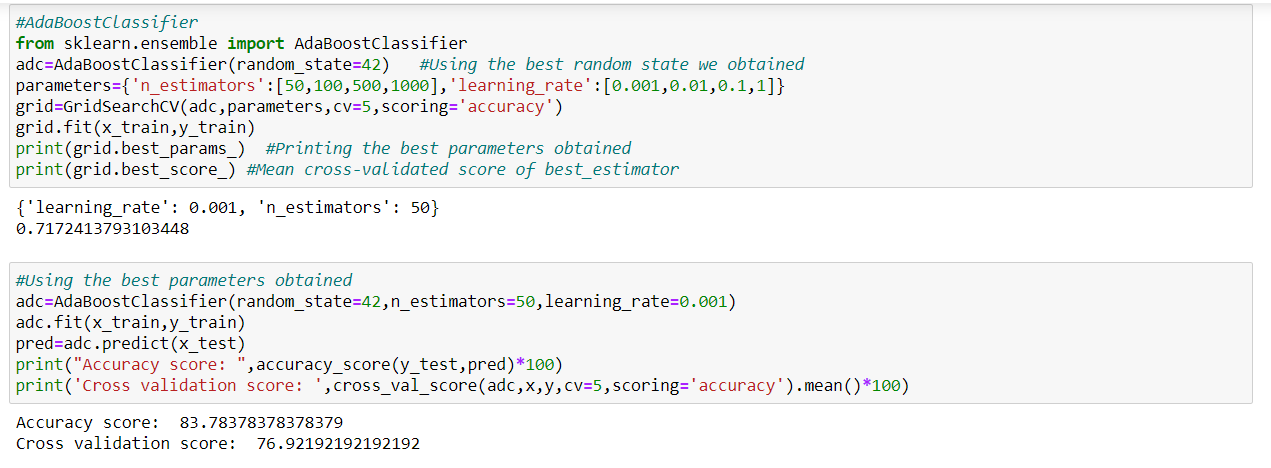
In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size max\_features.

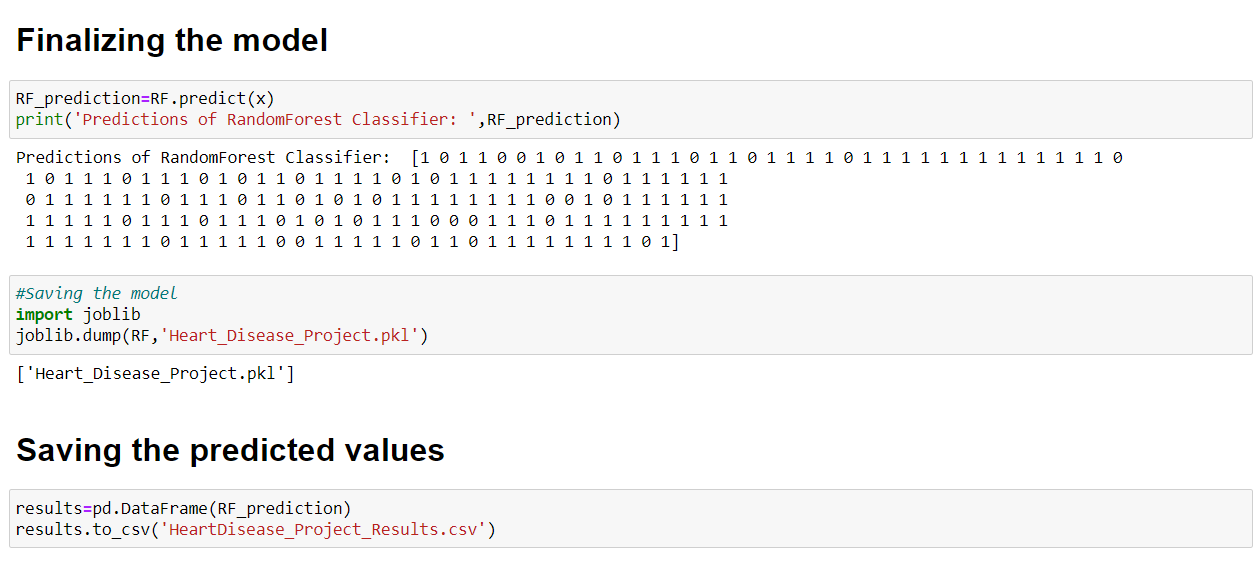


**2. AdaBoost:**

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.



After using ensemble techniques, we can see that RandomForestClassifier is the best algorithm among all as it is giving an accuracy score of 94.59 and cross validation score of 76.90, after using the best parameters obtained. We will now finalize the model by saving the predictions we got, saving the best model and also creating a new Data Frame for saving the predictions:



We can see the prediction values after using the best model obtained and we save the model in pickle format or obj format.

**Concluding Remarks:**

-> After getting an insight of this dataset, we came to see that the people are diagnosed with heart diseases on the basis of various parameters.

-> After analysing the dataset, we plotted graphs based on various parameters and from the graph we saw that female patients with high age were more severe than the male patients.

-> We also observed that the people aged above 60 is on high risk for getting heart diseases and we also observed that the people who has cholesterol level above 200 are at high risk of getting diagnosed with heart disease.

->We saw that the people who have normal defect are getting maximum heart rates and being diagnosed.

->We tried to reduced skewness, worked on missing values and removed the outliers for getting the better result of model.

->We applied lots of Classification models and came to a conclusion that the KNeighbors was working well and we tried to improve the accuracy of the model by finding the best parameter using Hyperparameter Tuning technique.

->After applying GridSearchCV, our scores were not increased and because of that we applied various ensemble techniques to boost our scores.

->By applying Ensemble Techniques, we got RandomForestClassifier as the best model among all as it gave an accuracy score of 94.5 and cross validation score of 76.9, which is quite good for this dataset.

->We finalized the model by finding the predicted values and saving the model in a pickle file. Then we saved the predicted results in a csv file.

->But we can improve the dataset by adding less skewed data, less outliers, more positive correlated data with the target variable, having fewer missing data, etc in order to achieve good scores.

->Overall, this dataset helped us to know about the reasons for getting heart diseases and also it was a good dataset to predict the presence of heart diseases.

**GitHub Link:** <https://github.com/Jerish7/ProjectsM20/blob/main/HeartDisease_Project.ipynb>