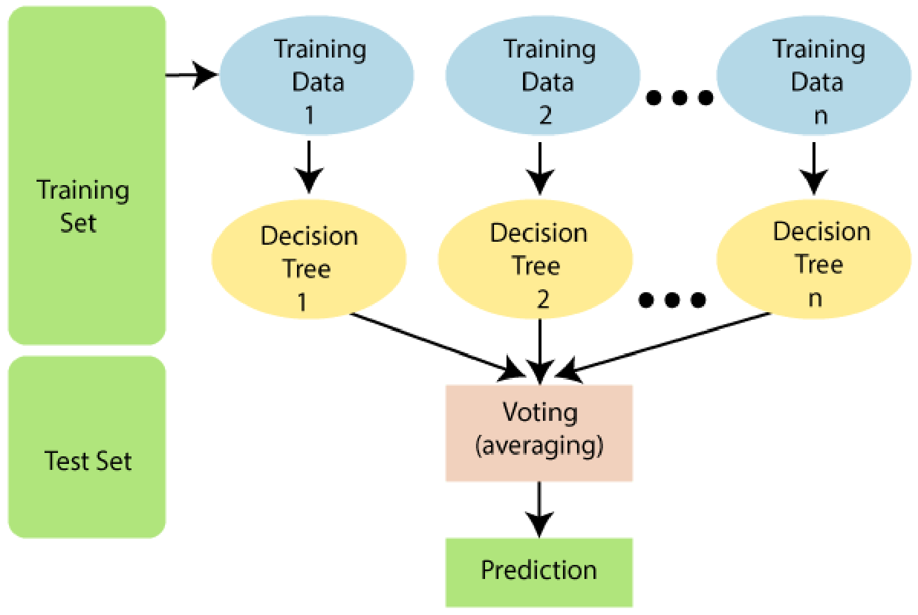
Manav Jawrani 19

EXPERIMENT NO. 8

Aim: To implement supervised learning algorithm like
i) Ada-Boosting
ii) Random forests

Theory:
Supervised learning is the type of machine learning in which machines are trained using well "labelled" training data, and on the basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.



As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

1. There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
2. The predictions from each tree must have very low correlations.

Random Forest works in two-phase first is to create the random forest by combining N decision trees, and second is to make predictions for each tree created in the first phase. The Working process can be explained in the below steps and diagram:

1. **Step 1**: Select random K data points from the training set.
2. **Step 2**: Build the decision trees associated with the selected data points (Subsets).
3. **Step 3**: Choose the number N for decision trees that you want to build.
4. **Step 4**: Repeat Step 1 & 2.
5. **Step 5**: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

AdaBoost:
Ada-boost or Adaptive Boosting is an ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get a high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and train the data sample in each iteration such that it ensures the accurate predictions of unusual

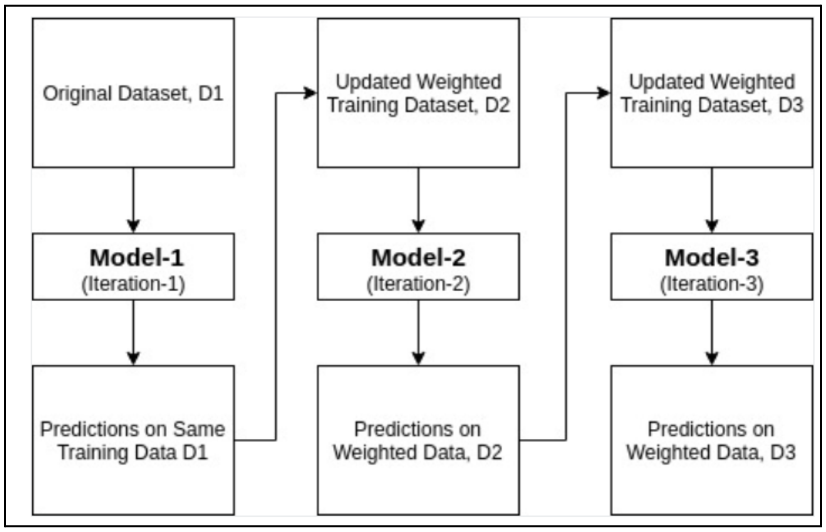
observations. Any machine learning algorithm can be used as a base classifier if it accepts weights on the training set. Adaboost should meet two conditions:

1. The classifier should be trained interactively on various weighted training examples.
2. In each iteration, it tries to provide an excellent fit for these examples by minimising training error.

How does the AdaBoost algorithm work?

It works in the following steps:

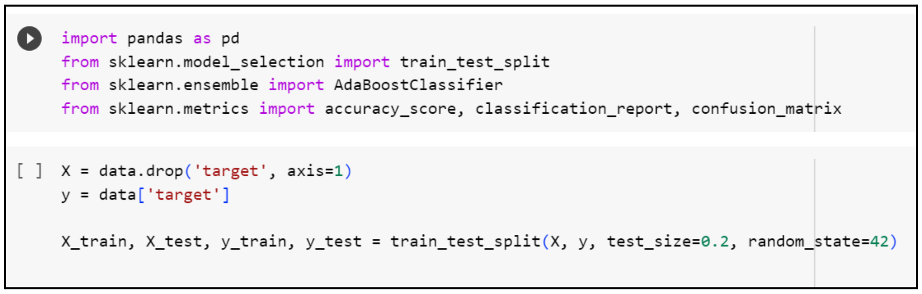
1. Initially, Adaboost selects a training subset randomly.
2. It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
3. It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
4. Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
5. This process iterates until the complete training data fits without any error or until reached the specified maximum number of estimators.
6. To classify, perform a "vote" across all of the learning algorithms you built.


Original Dataset, D1
Updated Weighted Training Dataset, D2
Updated Weighted Training Dataset, D3
Model-1 (Iteration-1)
Model-2 (Iteration-2)
Model-3 (Iteration-3)
Predictions on Same Training Data D1
Predictions on Weighted Data, D2
Predictions on Weighted Data, D3

Manav Jawrani 19

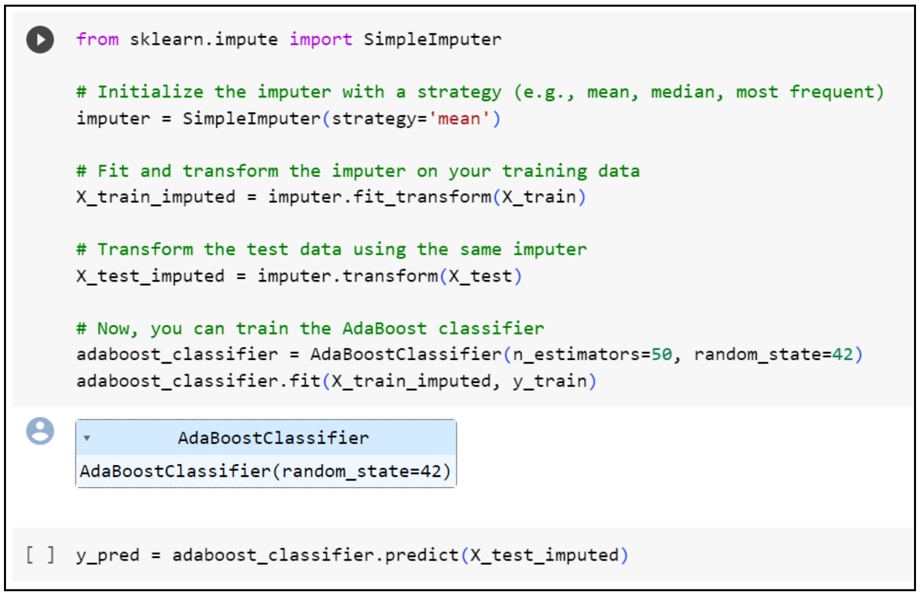
Code and Output:

Ada-Boosting:



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

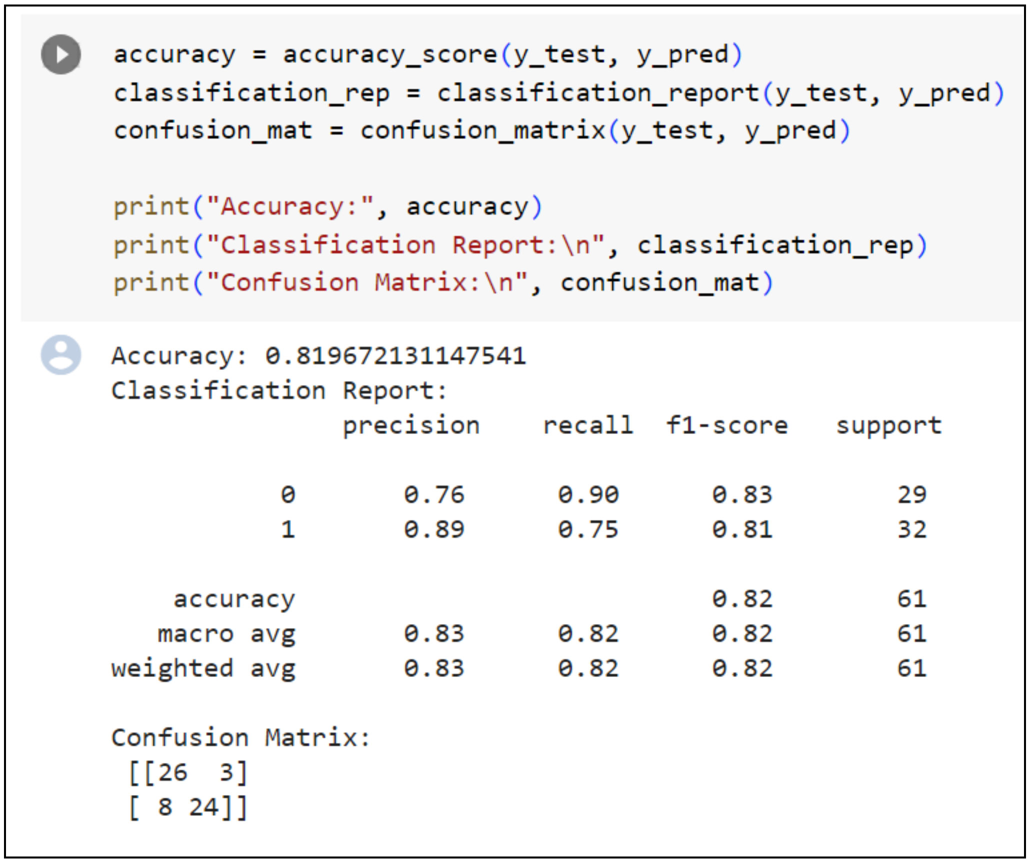
```
[ ] X = data.drop('target', axis=1)
    y = data['target']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



```
from sklearn.impute import SimpleImputer

# Initialize the imputer with a strategy (e.g., mean, median, most frequent)
imputer = SimpleImputer(strategy='mean')

# Fit and transform the imputer on your training data
X_train_imputed = imputer.fit_transform(X_train)

# Transform the test data using the same imputer
X_test_imputed = imputer.transform(X_test)

# Now, you can train the AdaBoost classifier
adaboost_classifier = AdaBoostClassifier(n_estimators=50, random_state=42)
adaboost_classifier.fit(X_train_imputed, y_train)
```

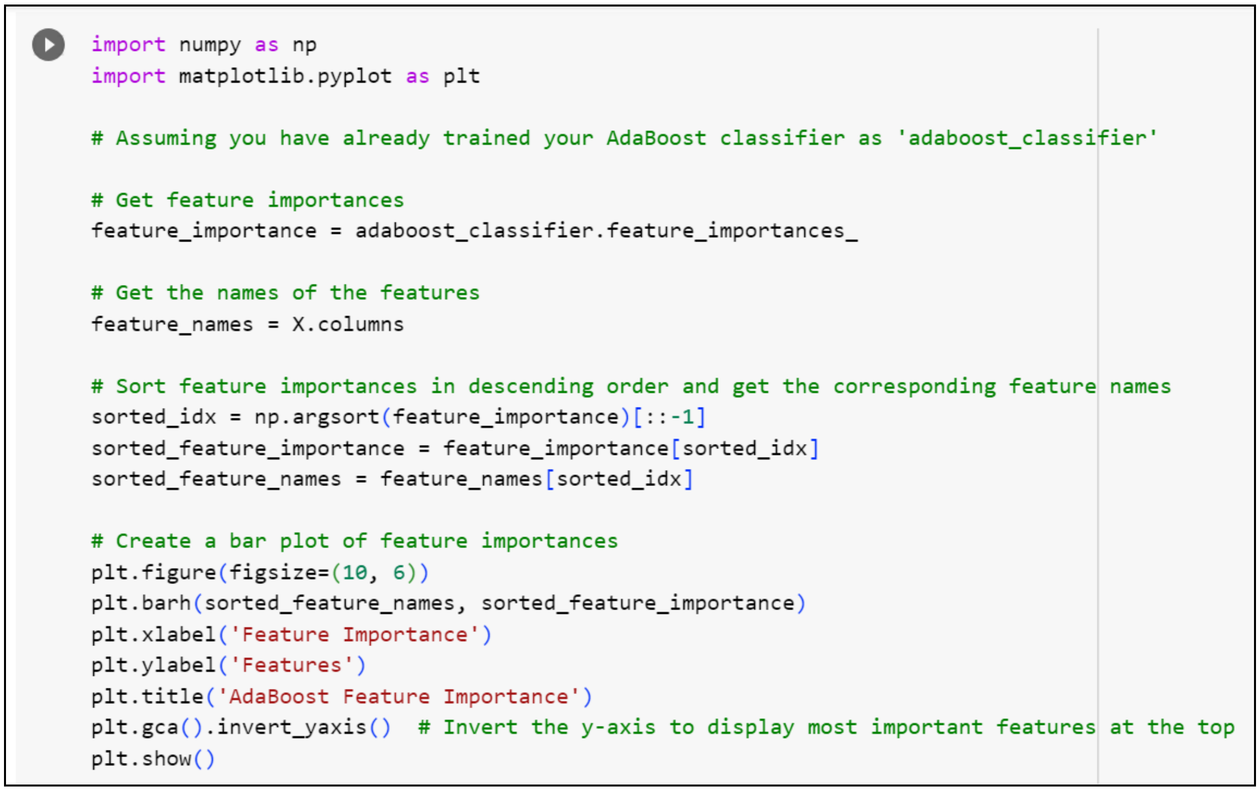
```
▾ AdaBoostClassifier
AdaBoostClassifier(random_state=42)
```

```
[ ] y_pred = adaboost_classifier.predict(X_test_imputed)
```

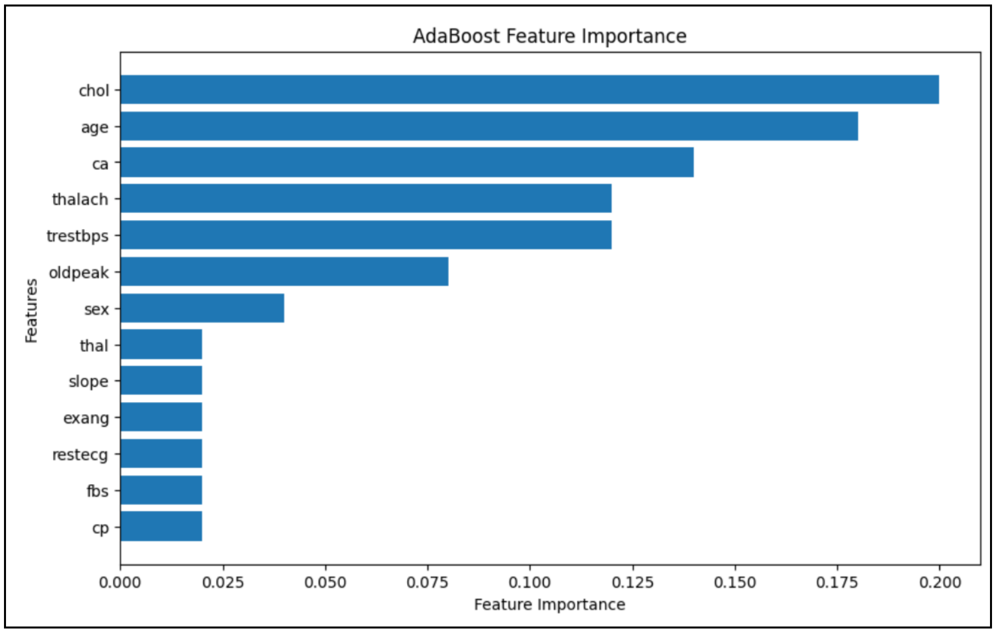


```
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
print("Confusion Matrix:\n", confusion_mat)
```

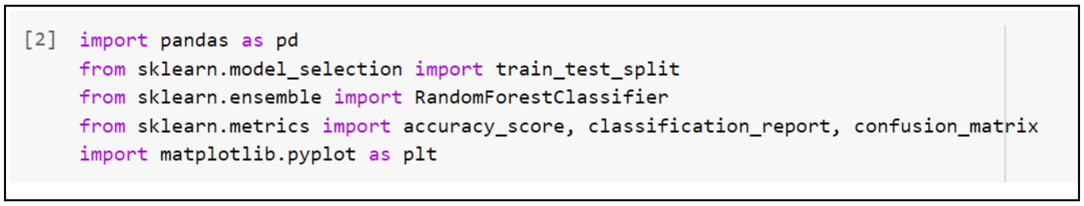
```
Accuracy: 0.819672131147541
Classification Report:
               precision    recall  f1-score   support

           0       0.76      0.90      0.83        29
           1       0.89      0.75      0.81        32

    accuracy                           0.82        61
   macro avg       0.83      0.82      0.82        61
weighted avg       0.83      0.82      0.82        61

Confusion Matrix:
 [[26  3]
 [ 8 24]]
```



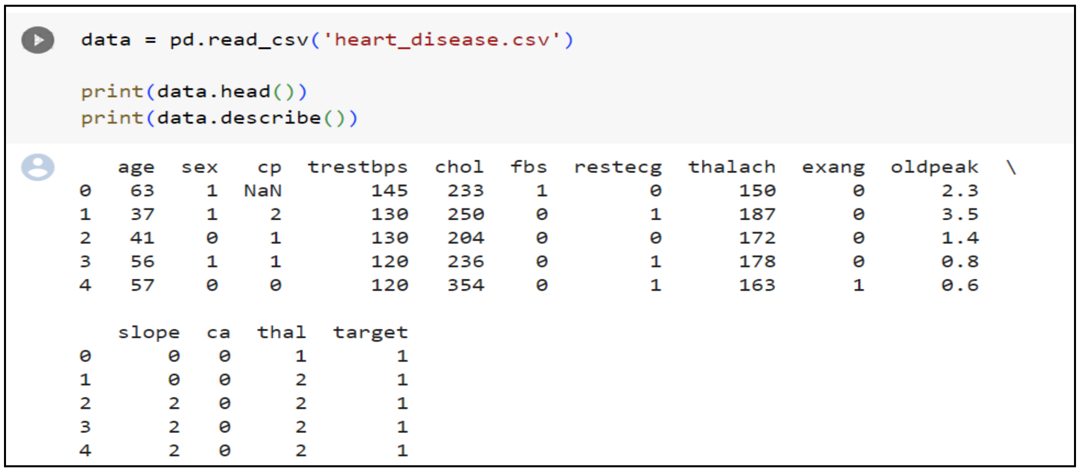
```python
import numpy as np
import matplotlib.pyplot as plt

# Assuming you have already trained your AdaBoost classifier as 'adaboost_classifier'

# Get feature importances
feature_importance = adaboost_classifier.feature_importances_

# Get the names of the features
feature_names = X.columns

# Sort feature importances in descending order and get the corresponding feature names
sorted_idx = np.argsort(feature_importance)[::-1]
sorted_feature_importance = feature_importance[sorted_idx]
sorted_feature_names = feature_names[sorted_idx]

# Create a bar plot of feature importances
plt.figure(figsize=(10, 6))
plt.barh(sorted_feature_names, sorted_feature_importance)
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('AdaBoost Feature Importance')
plt.gca().invert_yaxis()  # Invert the y-axis to display most important features at the top
plt.show()
```



Random Forest:



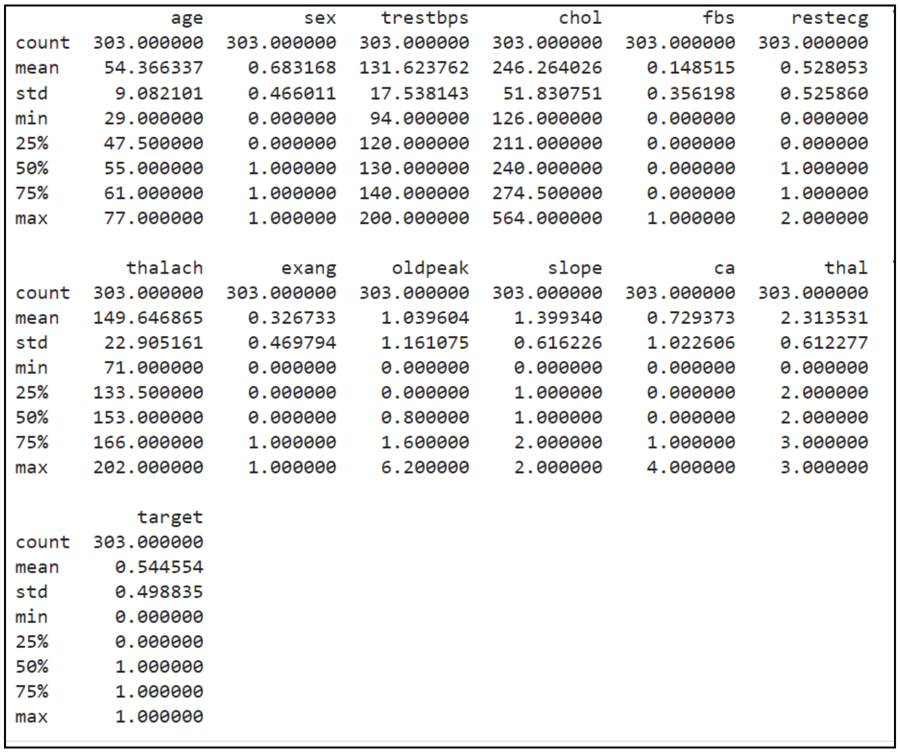
```
[2] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    import matplotlib.pyplot as plt
```



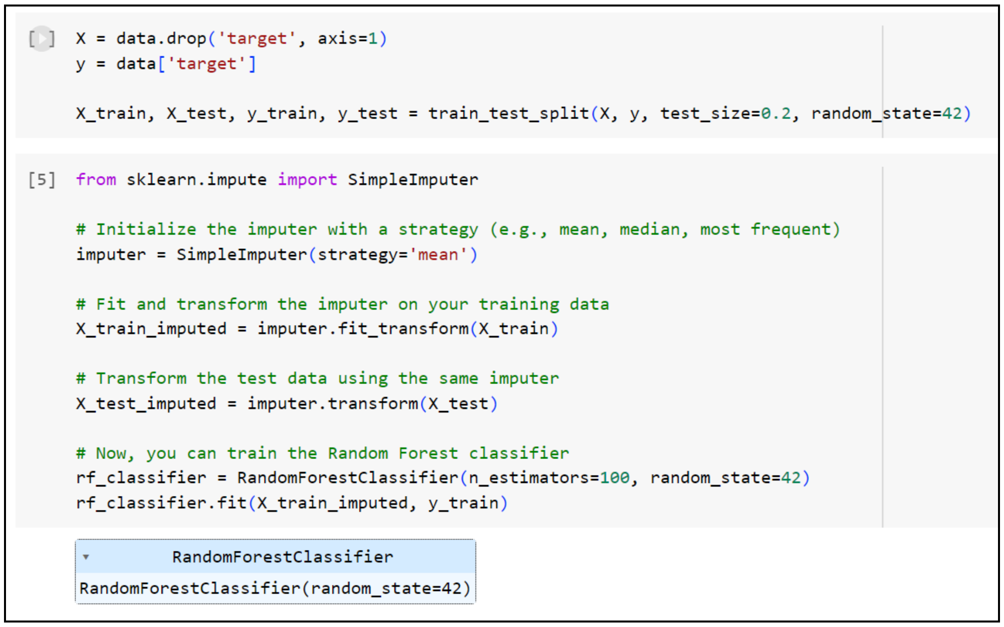
```
data = pd.read_csv('heart_disease.csv')

print(data.head())
print(data.describe())
```

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

   slope  ca  thal  target
0      0   0     1       1
1      0   0     2       1
2      2   0     2       1
3      2   0     2       1
4      2   0     2       1
```

Manav Jawrani 19



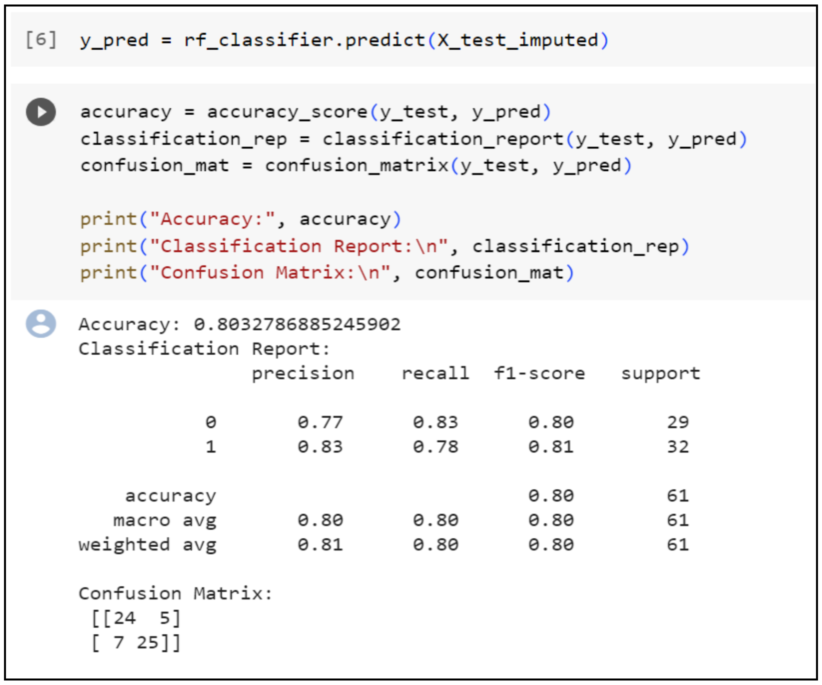
```
              age         sex    trestbps        chol         fbs     restecg
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168  131.623762  246.264026    0.148515    0.528053
std      9.082101    0.466011   17.538143   51.830751    0.356198    0.525860
min     29.000000    0.000000   94.000000  126.000000    0.000000    0.000000
25%     47.500000    0.000000  120.000000  211.000000    0.000000    0.000000
50%     55.000000    1.000000  130.000000  240.000000    0.000000    1.000000
75%     61.000000    1.000000  140.000000  274.500000    0.000000    1.000000
max     77.000000    1.000000  200.000000  564.000000    1.000000    2.000000

          thalach       exang     oldpeak       slope          ca        thal
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean   149.646865    0.326733    1.039604    1.399340    0.729373    2.313531
std     22.905161    0.469794    1.161075    0.616226    1.022606    0.612277
min     71.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%    133.500000    0.000000    0.000000    1.000000    0.000000    2.000000
50%    153.000000    0.000000    0.800000    1.000000    0.000000    2.000000
75%    166.000000    1.000000    1.600000    2.000000    1.000000    3.000000
max    202.000000    1.000000    6.200000    2.000000    4.000000    3.000000

           target
count  303.000000
mean     0.544554
std      0.498835
min      0.000000
25%      0.000000
50%      1.000000
75%      1.000000
max      1.000000
```



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

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[5] from sklearn.impute import SimpleImputer

    # Initialize the imputer with a strategy (e.g., mean, median, most frequent)
    imputer = SimpleImputer(strategy='mean')

    # Fit and transform the imputer on your training data
    X_train_imputed = imputer.fit_transform(X_train)

    # Transform the test data using the same imputer
    X_test_imputed = imputer.transform(X_test)

    # Now, you can train the Random Forest classifier
    rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_classifier.fit(X_train_imputed, y_train)
```

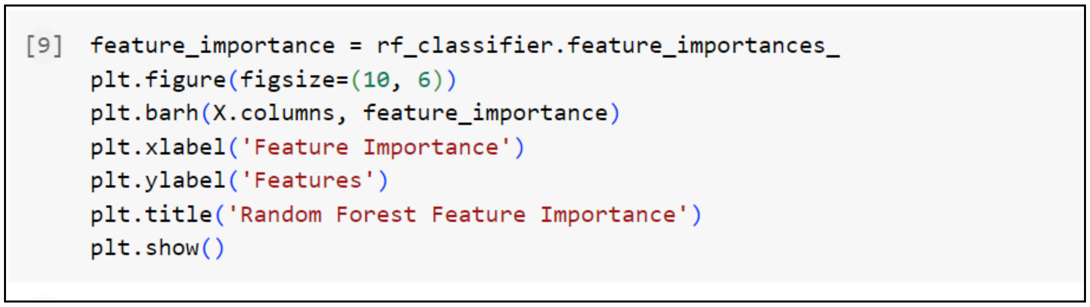
```
▾         RandomForestClassifier
RandomForestClassifier(random_state=42)
```



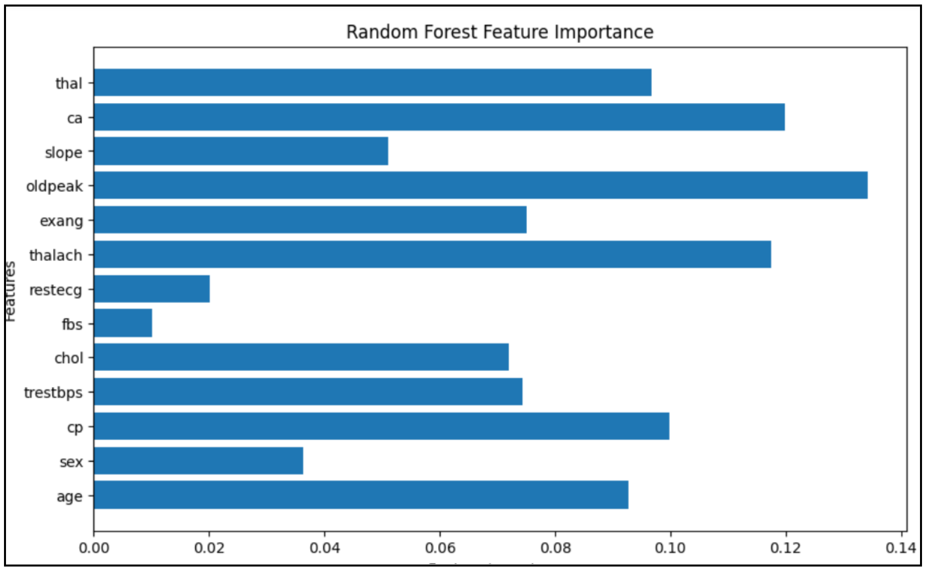
```
[6] y_pred = rf_classifier.predict(X_test_imputed)
```

```
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
print("Confusion Matrix:\n", confusion_mat)
```

```
Accuracy: 0.8032786885245902
Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.83      0.80        29
           1       0.83      0.78      0.81        32

    accuracy                           0.80        61
   macro avg       0.80      0.80      0.80        61
weighted avg       0.81      0.80      0.80        61

Confusion Matrix:
 [[24  5]
 [ 7 25]]
```



```
[9] feature_importance = rf_classifier.feature_importances_
    plt.figure(figsize=(10, 6))
    plt.barh(X.columns, feature_importance)
    plt.xlabel('Feature Importance')
    plt.ylabel('Features')
    plt.title('Random Forest Feature Importance')
    plt.show()
```



Conclusion - Thus we studied an overview of the supervised learning algorithms and implemented i) Ada-Boosting ii) Random forests on a Heart Disease Prediction dataset.