ML LAB ASSIGNMENT

SUPRATIM NAG -- CSE-AIML/22/057 -- GROUP-B

Q-8:Implementation of Ensemble Techniques

Write a python code to show ensemble technique using RandomForestClassifier

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
```

In [4]:
```
data = pd.read_csv(r"C:\Users\SUPRATIM NAG\OneDrive\Documents\ML\Personal_Datasets\Dataset.csv")
data.head(1)
```

Out[4]:

	Patient ID	Age	Blood Pressure	Cholesterol Levels	Heart Rate	BMI	Diagnosis	Treatment Plan	Recovery Status	Medication Type	Follow-up Requirement
0	101	65	130	250	72	28.0	Hypertension with high cholesterol.	Medication: Lisinopril (blood pressure), Stati...	Active Recovery	Lisinopril, Statins.	Quarterly.

In [5]:
```
data.shape
```

Out[5]:
```
(100, 11)
```

In [59]:
```
meddata=data[['Age','Blood Pressure','Cholesterol Levels','Heart Rate','BMI','Diagnosis']]
meddata.head(1)
```

Out[59]:

	Age	Blood Pressure	Cholesterol Levels	Heart Rate	BMI	Diagnosis
0	65	130	250	72	28.0	Hypertension with high cholesterol.

In [18]:
```
meddata['Diagnosis'] = meddata['Diagnosis'].apply(
    lambda x: 1 if any(condition in x for condition in ['Hypertension', 'Obesity', 'Overweight']) else 0
)
```

```
C:\Users\SUPRATIM NAG\AppData\Local\Temp\ipykernel_16388\3012899904.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  meddata['Diagnosis'] = meddata['Diagnosis'].apply(
```

In [19]:
```
print(meddata['Diagnosis'].value_counts())
```

```
Diagnosis
0    63
1    37
Name: count, dtype: int64
```

In [20]:
```
x = meddata.drop('Diagnosis', axis=1)
y = meddata['Diagnosis']
```

In [54]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20)
```

In [55]:
```
forest = RandomForestClassifier()
forest.fit(X_train, y_train)
```

Out[55]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [56]:
```
y_pred = forest.predict(X_test)
```

In [57]:
```
forest.score(X_test, y_test)
```

Out[57]:
```
0.8
```

In [58]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[12  2]
 [ 2  4]]
              precision    recall  f1-score   support

           0       0.86      0.86      0.86        14
           1       0.67      0.67      0.67         6

    accuracy                           0.80        20
   macro avg       0.76      0.76      0.76        20
weighted avg       0.80      0.80      0.80        20
```