|  |  |
| --- | --- |
| **EX.NO:6 BAGGING , BOOSTING APPLICATION USING REGRESSIO TREES** | |
| IN [1] | import pandas as pd  import matplotlib.pyplot as plt  %matplotlib inline  from sklearn.tree import DecisionTreeClassifier |
| IN [2] | df=pd.read\_csv("diabetes.csv")  df.head() |
| IN [3] | df.isnull().sum() |
| IN [4] | df.describe() |
| IN [5] | pd.set\_option('display.float\_format', '{:.2f}'.format)  df.describe() |
| IN [6] | df.Outcome.value\_counts() |
| IN [7] | 268/500 |
| IN [8] | x= df.drop('Outcome',axis='columns')  y = df.Outcome |
| IN [9] | from sklearn.model\_selection import train\_test\_split  X = df.drop('Outcome',axis='columns')  y = df.Outcome  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42) |
| IN [10] | from sklearn.ensemble import BaggingClassifier  bagging\_clf = BaggingClassifier(base\_estimator=DecisionTreeClassifier(), n\_estimators=100, max\_samples=0.8,oob\_score=True, random\_state=42)  bagging\_clf.fit(X\_train, y\_train)  bagging\_clf.oob\_score\_ |
| IN [11] | bagging\_clf.score(X\_test, y\_test) |
| IN [12] | from sklearn.metrics import accuracy\_score  scores = {  'Bagging Classifier': {  'Train': accuracy\_score(y\_train, bagging\_clf.predict(X\_train)),  'Test': accuracy\_score(y\_test, bagging\_clf.predict(X\_test)),  },  } |
| IN [13] | from sklearn.ensemble import RandomForestClassifier  rf\_clf = RandomForestClassifier(random\_state=42, n\_estimators=1000)  rf\_clf.fit(X\_train, y\_train) |
| IN [14] | scores['Random Forest'] = {  'Train': accuracy\_score(y\_train,  'Test': accuracy\_score(y\_test,  rf\_clf.predict(X\_test)), } |
| IN [15] | from sklearn.ensemble import AdaBoostClassifier  ada\_boost\_clf = AdaBoostClassifier(n\_estimators=30)  ada\_boost\_clf.fit(X\_train, y\_train) |
| IN [16] | scores['AdaBoost'] = {  'Train': accuracy\_score(y\_train,  ada\_boost\_clf.predict(X\_train)),  'Test': accuracy\_score(y\_test,  ada\_boost\_clf.predict(X\_test)), } |
| IN [17] | from sklearn.ensemble import GradientBoostingClassifier  grad\_boost\_clf = GradientBoostingClassifier(n\_estimators=100, random\_state=42)  grad\_boost\_clf.fit(X\_train, y\_train) |
| IN [18] | scores['Gradient Boosting'] = {  'Train': accuracy\_score(y\_train,  grad\_boost\_clf.predict(X\_train)),  'Test': accuracy\_score(y\_test,  grad\_boost\_clf.predict(X\_test)), } |
| IN [19] | scores\_df = pd.DataFrame(scores)  scores\_df.plot(kind='barh', figsize=(15, 8)) |

**OUTPUT**

|  |  |
| --- | --- |
| OUT[3] | Pregnancies 0  Glucose 0  BloodPressure 0  SkinThickness 0  Insulin 0  BMI 0  DiabetesPedigreeFunction 0  Age 0  Outcome 0  dtype: int64 |
| OUT[13] | RandomForestClassifier(n\_estimators=1000, random\_state=42) |
| OUT[15] | AdaBoostClassifier(n\_estimators=30) |
| OUT[17] | GradientBoostingClassifier(random\_state=42) |
| OUT[19] | <AxesSubplot:> |