adaboost

January 23, 2025

[1]:
```
import pandas as pd
import numpy
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix,mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import HistGradientBoostingRegressor
```

[2]:
```
# Load Titanic dataset
data = pd.read_csv("C:/Users/91703/OneDrive/Desktop/TITANIC.csv")
```

[3]:
```
# Select relevant columns and convert categorical variables
data = data[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
```

[4]:
```
# Handle missing Age values
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
data['Age'] = data['Age'].fillna(data['Age'].mean())
```

[5]:
```
# Define target variable (y) and feature variables (X)
y = data['Survived']
X = data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
```

[6]:
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[7]:
```
# Create and train AdaBoost Classifier model
model = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1),␣
↪n_estimators=100, random_state=42, algorithm="SAMME")
model.fit(X_train, y_train)
```

[7]:
```
AdaBoostClassifier(algorithm='SAMME',
                   estimator=DecisionTreeClassifier(max_depth=1),
                   n_estimators=100, random_state=42)
```

[8]:
```python
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

```
[0 0 0 1 1 1 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0
 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 0 1 0 0 0 1 1 1 0 1
 0 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 0 0
 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 1 1]
```

[9]:
```python
# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:", confusion_matrix(y_test, y_pred))
```

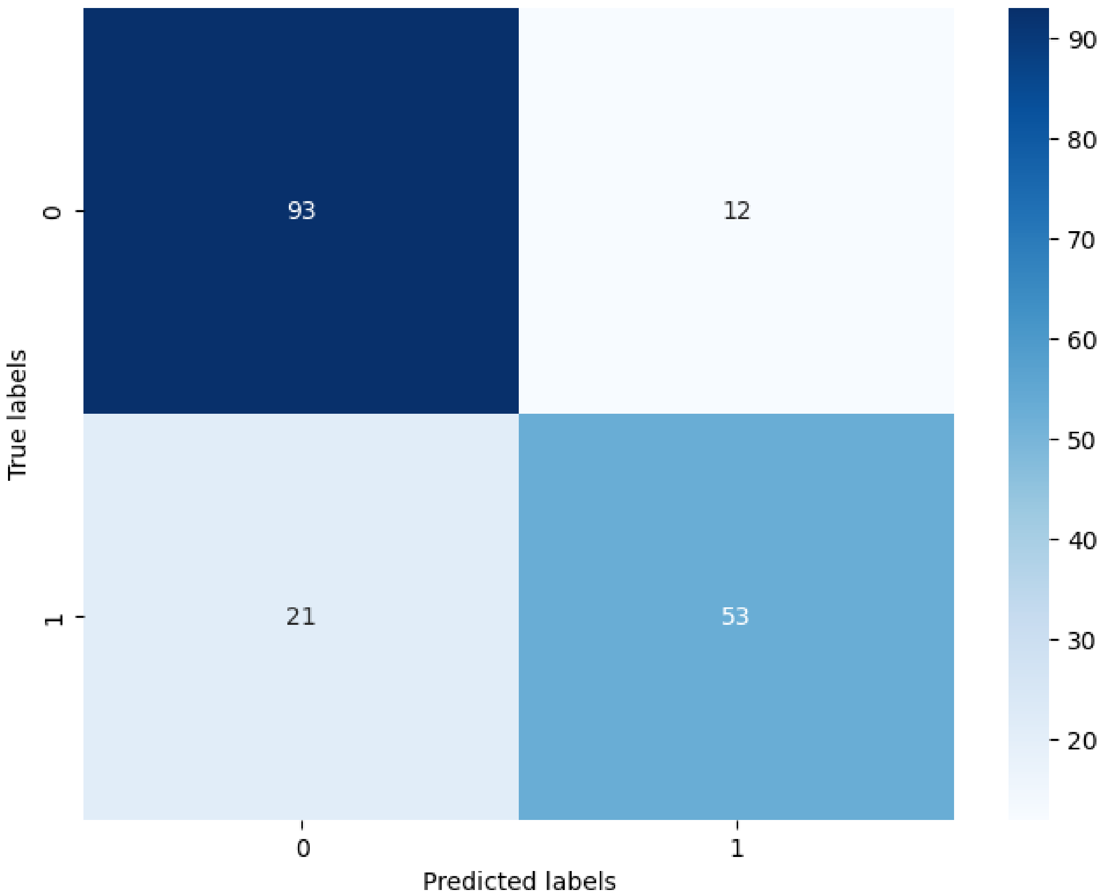
```
Accuracy: 0.8156424581005587
Classification Report:
               precision    recall  f1-score   support

           0       0.82      0.89      0.85       105
           1       0.82      0.72      0.76        74

    accuracy                           0.82       179
   macro avg       0.82      0.80      0.81       179
weighted avg       0.82      0.82      0.81       179

Confusion Matrix: [[93 12]
 [21 53]]
```

[10]:
```python
# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues')
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.show()
```



[11]:
```
# Select relevant columns and convert categorical variables
data = data[['Age', 'Pclass', 'Sex', 'SibSp', 'Parch', 'Fare']]
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
```

[12]:
```
# Handle missing values
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
data['Age'] = data['Age'].fillna(data['Age'].mean())
```

[13]:
```
# Define target variable (y) and feature variables (X)
y = data['Age']
X = data[['Pclass', 'Sex', 'SibSp', 'Parch', 'Fare']]
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
data[['Pclass', 'SibSp', 'Parch', 'Fare']] = imputer.
 ↪fit_transform(data[['Pclass', 'SibSp', 'Parch', 'Fare']])
```

[14]:
```
# Split data into training and testing sets
```

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

[15]:
```
# Create and train AdaBoost Regressor model
model = AdaBoostRegressor(DecisionTreeRegressor(max_depth=1), n_estimators=100,␣
↪random_state=42)
model = HistGradientBoostingRegressor()
model.fit(X_train, y_train)
```

[15]:
```
HistGradientBoostingRegressor()
```

[16]:
```
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

```
[20.37925076 33.03821441 29.79172696 27.6594605  16.49529091 34.31836463
 32.24422894 24.67750644 32.24422894 33.51014658 32.93673764 29.68526189
 23.38691519 28.9880119  32.1189735  34.09219567 38.02548303 26.38266897
 32.1189735  32.86371629 29.34245504 49.00648239 24.08712432 26.77258367
 28.22043433 13.29237003 38.58744356 32.1189735  13.29237003 26.38266897
 29.34245504 32.24422894 39.12397318 27.05475416 29.68526189 27.60067185
 40.68438883 32.24422894 40.14277115 29.34245504 30.16371881 23.57538821
 29.68526189 32.24422894 20.65461021 29.80447322 29.68526189 26.77258367
 27.35841301 25.68450682  7.33325263 40.54374157 31.78761847 28.48274527
 32.24422894 36.39211919 33.03821441 35.07391897 34.20581694 28.48425678
 28.12148944 25.9826291  30.70754768 44.37444736 32.24422894 33.03821441
 29.8548642  28.48425678 14.08292954 45.03929936 22.48865568 27.78672487
 39.58705861 41.12819225 26.38266897 31.1238285  32.24422894 27.31506529
 32.1189735  26.28073497  7.80596643 33.03821441 35.68660461 27.23891341
 39.12397318 32.65912497 31.96873732 29.84631218 36.72753822 29.88810389
 26.28073497  5.79743052 42.56761717 26.34420585 29.34245504 26.38266897
 38.02548303 29.68526189 32.1189735  29.34245504 39.27928969 31.13661898
 37.77370518 27.35841301 29.68526189 25.52749219 25.38872435 35.01198676
 27.39620134 40.14277115 38.02548303 29.88810389 37.91562222 43.954191
 34.70012099 24.01577529 40.9989712  32.1189735  25.79212535 29.80447322
  5.24908989 34.31836463 45.03929936 24.64093805 29.88810389 49.00648239
 44.37444736 36.10509183 28.79078656 29.68526189 28.48425678 20.37925076
 31.1872183  25.58920046 35.49006808 27.60067185 30.57716594 28.12148944
 27.05475416 25.55773694 32.82653263 30.70754768 32.1189735  30.70754768
 30.62050272 26.59979237 32.1189735  33.03821441 34.51763963 29.68526189
 26.9640565  27.05475416 25.20495097 32.08883328 29.68526189 27.35841301
 27.87706076 25.20495097 30.70754768 20.37925076 27.5537224  49.00648239
 32.1189735  32.41152836 32.82653263 39.79139979 33.03821441 43.20984628
 29.34245504 27.35841301 39.46216674 21.96621512 43.954191   20.37925076
 31.13661898 28.9880119  32.49929883 33.03821441 18.62755246]
```

[17]:

```python
# Evaluate model performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared:", r2)
```

```
Mean Squared Error: 127.75959541084642
Mean Absolute Error: 8.550550712237309
R-squared: 0.24541436874490408
```

[18]:

```python
# Plot predicted vs actual values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Age")
plt.ylabel("Predicted Age")
plt.show()
```