lsu9nkynn

January 23, 2025

[51]:
```
#Desicion tree algorithim on both the  classification and regression models
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
from sklearn.metrics import mean_squared_error,mean_absolute_error, r2_score
import math
import matplotlib.pyplot as plt
import seaborn as sns
```

[12]:
```
# Load Titanic dataset
data = pd.read_csv(r"C:\Users\91703\OneDrive\Desktop\TITANIC.csv")
```

[13]:
```
data.head()
```

[13]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
```

```
4       0             373450   8.0500   NaN         S
```

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

[5]:
```
data.describe()
```

[5]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

[6]:
```
data.isnull().sum()
```

[6]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[15]:
```
data.dropna(subset=['Age','Cabin','Embarked'], inplace=True)
```

[17]:
```
data.isnull().sum()
```

[17]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

[20]:
```
# Handle missing Age values
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
data['Age'] = data['Age'].fillna(data['Age'].mean())
```

[21]:
```
# 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})
```

[22]:
```
data.head()
```

[22]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
1	1	1	NaN	38.0	1	0	71.2833
3	1	1	NaN	35.0	1	0	53.1000
6	0	1	NaN	54.0	0	0	51.8625
10	1	3	NaN	4.0	1	1	16.7000

```
11          1        1   NaN   58.0      0      0  26.5500
```

[23]:
```
# Define target variable (y) and feature variables (X)
y = data['Survived']
X = data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
```

[24]:
```
# 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)
```

[25]:
```
# Create and train Decision Tree Classifier model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

[25]:
```
DecisionTreeClassifier()
```

[27]:
```
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

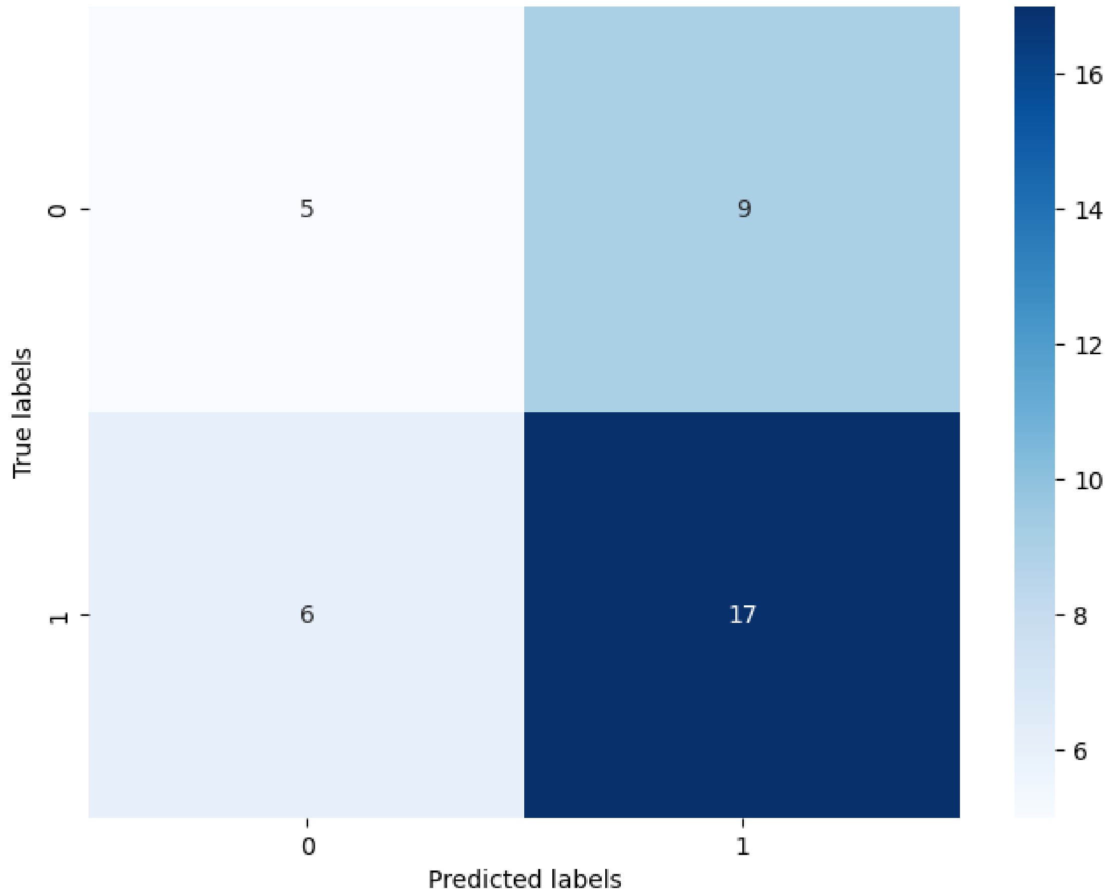
```
[1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 1 1]
```

[28]:
```
# 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: 0.5945945945945946
Classification Report:
               precision    recall  f1-score   support

           0       0.45      0.36      0.40        14
           1       0.65      0.74      0.69        23

    accuracy                           0.59        37
   macro avg       0.55      0.55      0.55        37
weighted avg       0.58      0.59      0.58        37

Confusion Matrix: [[ 5  9]
 [ 6 17]]
```

[29]:
```
# 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()
```



[37]:
```
#Now the desicion tree regression model
# Select relevant columns and convert categorical variables
data = data[['Age', 'Pclass', 'Sex', 'SibSp', 'Parch', 'Fare']]
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
```

[38]:
```
# Handle missing Age values (already present in some rows, ignoring for␣
↪prediction)
data.dropna(subset=['Age'], inplace=True)
```

[39]:
```
# Define target variable (y) and feature variables (X)
y = data['Age']
X = data[['Pclass', 'Sex', 'SibSp', 'Parch', 'Fare']]
```

[40]:
```
# 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)
```

[41]:
```python
# Create and train Decision Tree Regressor model
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
```

[41]:
```
DecisionTreeRegressor()
```

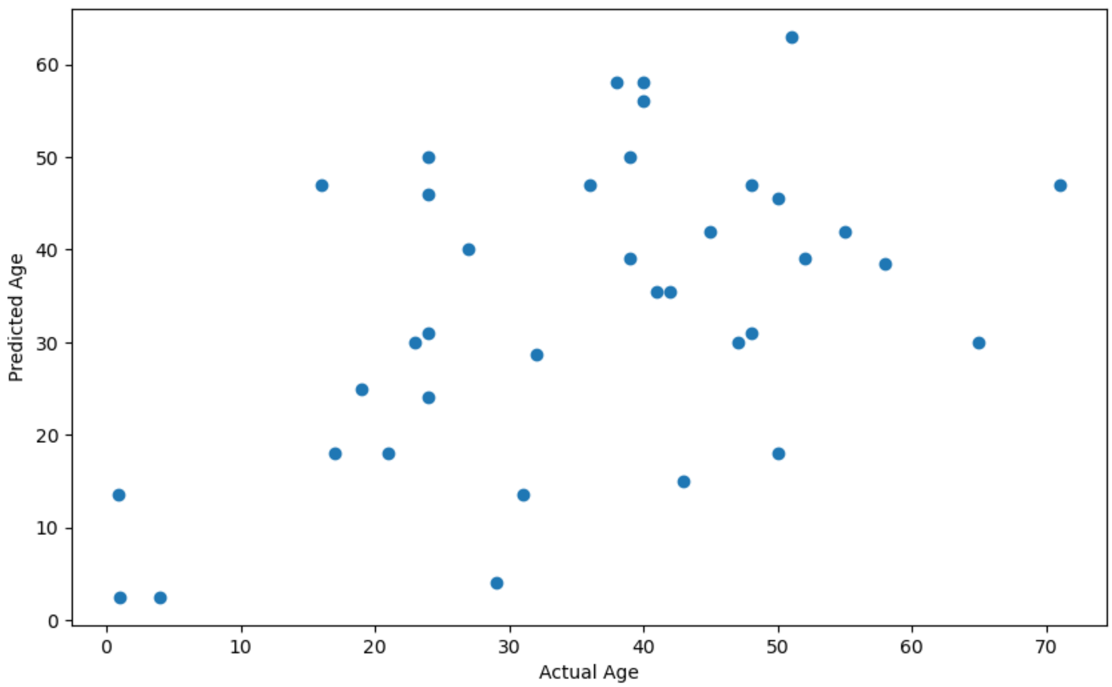
[45]:
```python
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

```
[50.          4.         18.         18.         31.         47.
 46.         35.5        50.         42.         38.5        63.
 42.         39.         39.         30.         25.         56.
 28.66666667 18.         15.         58.         35.5         2.5
 30.          2.5        13.5        30.         40.         47.
 24.         13.5        47.         31.         58.         45.5
 47.        ]
```

[52]:
```python
# Evaluate model performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = math.sqrt(mse)
r2 = r2_score(y_test, y_pred)
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
mdae = np.median(np.abs(y_test - y_pred))
print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
print("Coefficient of Determination (R-squared):", r2)
print("Mean Absolute Percentage Error (MAPE):", mape)
print("Median Absolute Error (MdAE):", mdae)
```

```
Mean Squared Error (MSE): 267.9220948948949
Mean Absolute Error (MAE): 13.092252252252251
Root Mean Squared Error (RMSE): 16.368325964951175
Coefficient of Determination (R-squared): 0.018573364695982142
Mean Absolute Percentage Error (MAPE): 78.33472362500122
Median Absolute Error (MdAE): 12.58
```

[44]:
```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()
```



Predicted Age
Actual Age
0
10
20
30
40
50
60
70