okenmljlt

January 23, 2025

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix,mean_squared_error, mean_absolute_error, r2_score
```

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

[3]:
```
data.head()
```

[3]:
```
   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
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Age     891 non-null    float64
 1   Pclass  891 non-null    int64
 2   Sex     891 non-null    int64
 3   SibSp   891 non-null    int64
 4   Parch   891 non-null    int64
 5   Fare    891 non-null    float64
dtypes: float64(2), int64(4)
memory usage: 41.9 KB
```

[23]:
```
data.describe()
```

[23]:

	Age	Pclass	Sex	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	29.699118	2.308642	0.352413	0.523008	0.381594	32.204208
std	13.002015	0.836071	0.477990	1.102743	0.806057	49.693429
min	0.420000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	22.000000	2.000000	0.000000	0.000000	0.000000	7.910400
50%	29.699118	3.000000	0.000000	0.000000	0.000000	14.454200
75%	35.000000	3.000000	1.000000	1.000000	0.000000	31.000000
max	80.000000	3.000000	1.000000	8.000000	6.000000	512.329200

[4]:
```
# Select relevant columns columns and convert categorical variables
data = data[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
```

[5]:
```
# Handle missing Age values
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
data['Age'] = data['Age'].fillna(data['Age'].mean())
```

[6]:
```
# Define target variable (y) and feature variables (X)
y = data['Survived']
X = data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
```

[7]:
```
# Scale features using StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[8]:
```
# Split data into training and testing sets
from sklearn.model_selection import train_test_split
```

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

[9]:
```python
# Create and train KNN Classifier model
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
```

[9]:
```
KNeighborsClassifier()
```

[10]:
```python
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

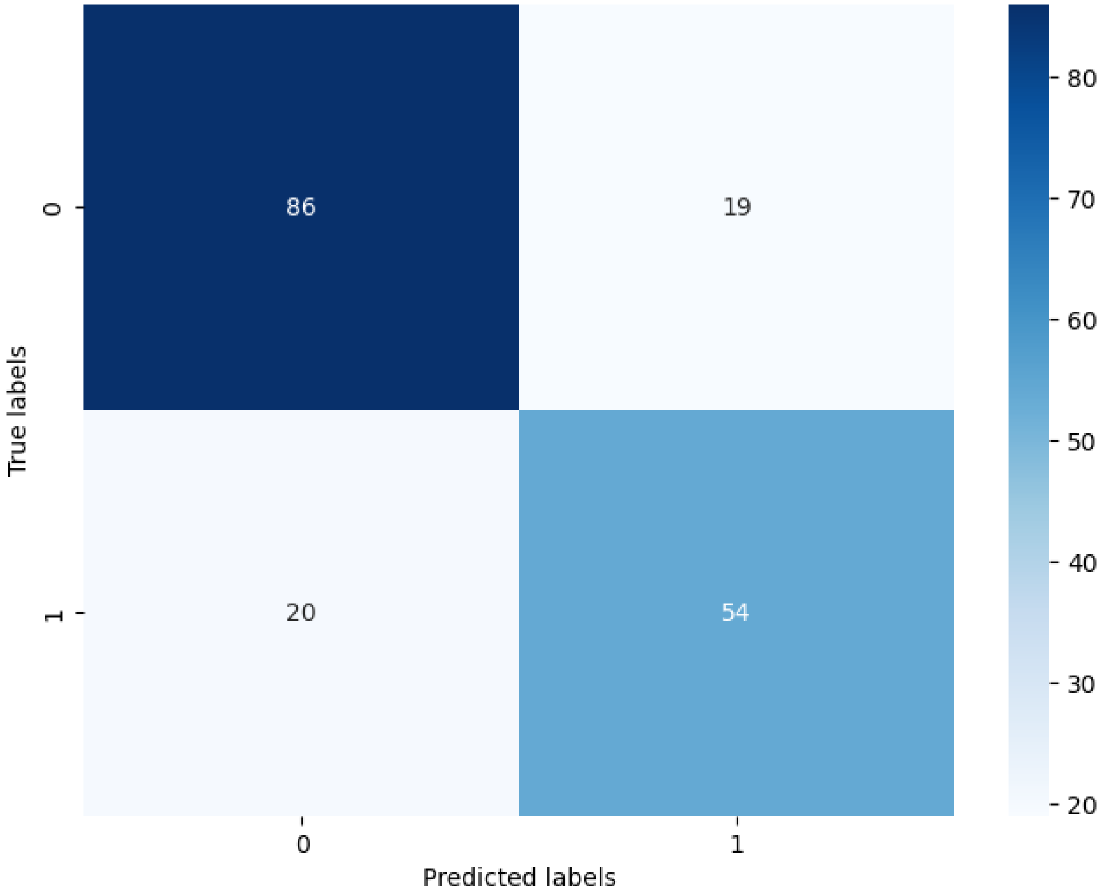
```
[0 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 1 0 0 0
 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 1 1
 0 0 1 1 1 1 0 1 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 1
 0 1 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 0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

[11]:
```python
# Evaluate model performance
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:", classification_report(y_test, y_pred))
print("Confusion Matrix:", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.7821229050279329
Classification Report:               precision    recall  f1-score   support

           0       0.81      0.82      0.82       105
           1       0.74      0.73      0.73        74

    accuracy                           0.78       179
   macro avg       0.78      0.77      0.77       179
weighted avg       0.78      0.78      0.78       179

Confusion Matrix: [[86 19]
 [20 54]]
```

[12]:
```python
# Plot Confusion Matrix
import matplotlib.pyplot as plt
import seaborn as sns
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()
```



[13]:
```
# Select relevant columns columns and convert categorical variables
data = data[['Age', 'Pclass', 'Sex', 'SibSp', 'Parch', 'Fare']]
```

[14]:
```
# Define target variable (y) and feature variables (X)
y = data['Age']
X = data[['Pclass', 'Sex', 'SibSp', 'Parch', 'Fare']]
```

[15]:
```
# Scale features using StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[16]:
```
# Split data into training and testing sets
from sklearn.model_selection import train_test_split
```

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

[17]:
```
# Create and train KNN Regressor model
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=5)
model.fit(X_train, y_train)
```

[17]:
```
KNeighborsRegressor()
```

[18]:
```
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

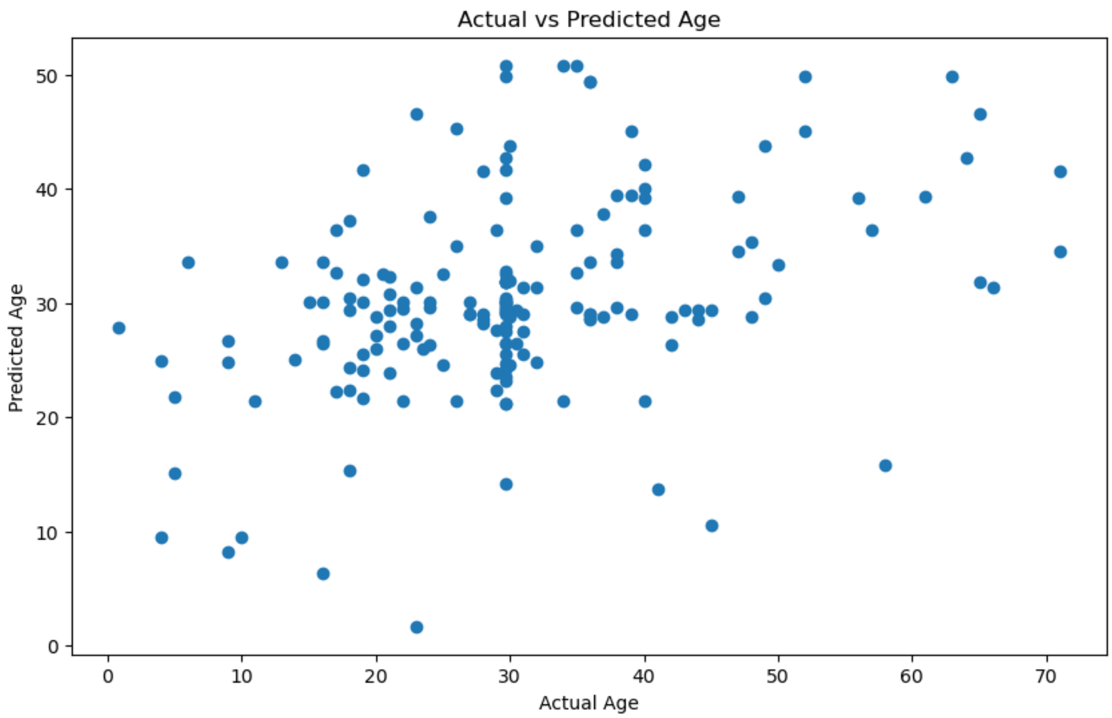
```
[21.2        31.4        28.8        33.6        25.03982353 34.93982353
 26.41947059 30.07964706 26.41947059 21.6        28.8        29.33982353
 14.22964706 31.91947059 29.         33.53982353 28.8        23.47964706
 29.         34.47964706 30.01947059 50.8        24.1        27.2
 27.55929412  9.53982353 39.97964706 29.         9.53982353 25.47964706
 30.01947059 26.41947059 39.13982353 24.27964706 29.33982353 29.07964706
 41.47964706 26.41947059 43.8        30.01947059 32.3        22.33982353
 29.33982353 31.85929412 10.6        26.63982353 32.07964706 27.2
 29.6        15.784      15.1        45.         36.33982353 21.384
 31.85929412 37.2        31.4        33.4        36.4        25.47964706
 23.87964706 33.6        28.6        42.73982353 31.85929412 36.4
 30.8        27.61947059 22.2        49.8        21.4        22.4
 45.27964706 45.        28.2        28.67964706 26.41947059 29.53982353
 33.6        13.73982353  6.4        36.4        42.13982353 29.17964706
 39.13982353 34.47964706 27.53982353 24.53982353 34.27964706 32.53982353
 26.73982353  8.2        39.33982353 29.17964706 30.01947059 28.2
 28.8        29.33982353 33.6        30.01947059 42.73982353 32.6
 41.6        29.6        29.33982353 24.53982353 21.4        39.27964706
 28.73982353 39.13982353 31.8        32.53982353 41.6        49.4
 30.33982353 25.93982353 41.47964706 29.        21.4        35.33982353
  1.7        34.93982353 49.8        27.8        32.53982353 50.8
 43.8        37.53982353 30.11947059 29.33982353 25.47964706 21.2
 29.69911765 21.8        30.4        30.11947059 32.8        23.87964706
 26.33982353 30.47964706 27.93982353 28.6        29.         37.8
 39.4        21.4        29.         31.4        29.93982353 29.33982353
 24.73982353 26.33982353 24.4        30.47964706 29.33982353 29.6
 29.07964706 23.13982353 28.6        24.8        25.97964706 50.8
 29.         49.8        27.93982353 46.6        31.4        31.8
 30.01947059 29.6        46.6        15.4        49.4        24.8
 32.6        31.91947059 39.4        36.4        24.93982353]
```

[19]:
```
# Evaluate model performance
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```
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: 151.8969616344165
Mean Absolute Error: 9.10054682878738
R-squared: 0.1028520064418863
```

[20]:
```
# Plot actual vs predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Age")
plt.ylabel("Predicted Age")
plt.title("Actual vs Predicted Age")
plt.show()
```



[21]:
```
# Plot residuals
residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
plt.scatter(y_test, residuals)
plt.xlabel("Actual Age")
```

```
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```