mubx98vlp

January 23, 2025

[29]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
```

[30]:
```
# Load Titanic dataset
data = pd.read_csv(r"C:\Users\91703\OneDrive\Desktop\TITANIC.csv")
```

[31]:
```
data.head()
```

[31]:
```
   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
```

[32]:
```
# 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})
```

[33]:
```
# Handle missing Age values
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
data['Age'] = data['Age'].fillna(data['Age'].mean())
```

[34]:
```
# Define target variable (y) and feature variables (X)
y = data['Survived']
X = data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]
```

[35]:
```
# 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)
```

[36]:
```
# Create and train SVM Classifier model
model = SVC(kernel='rbf', C=1, random_state=42)
model.fit(X_train, y_train)
```

[36]:
```
SVC(C=1, random_state=42)
```

[37]:
```
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

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

[38]:
```
# 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))
```

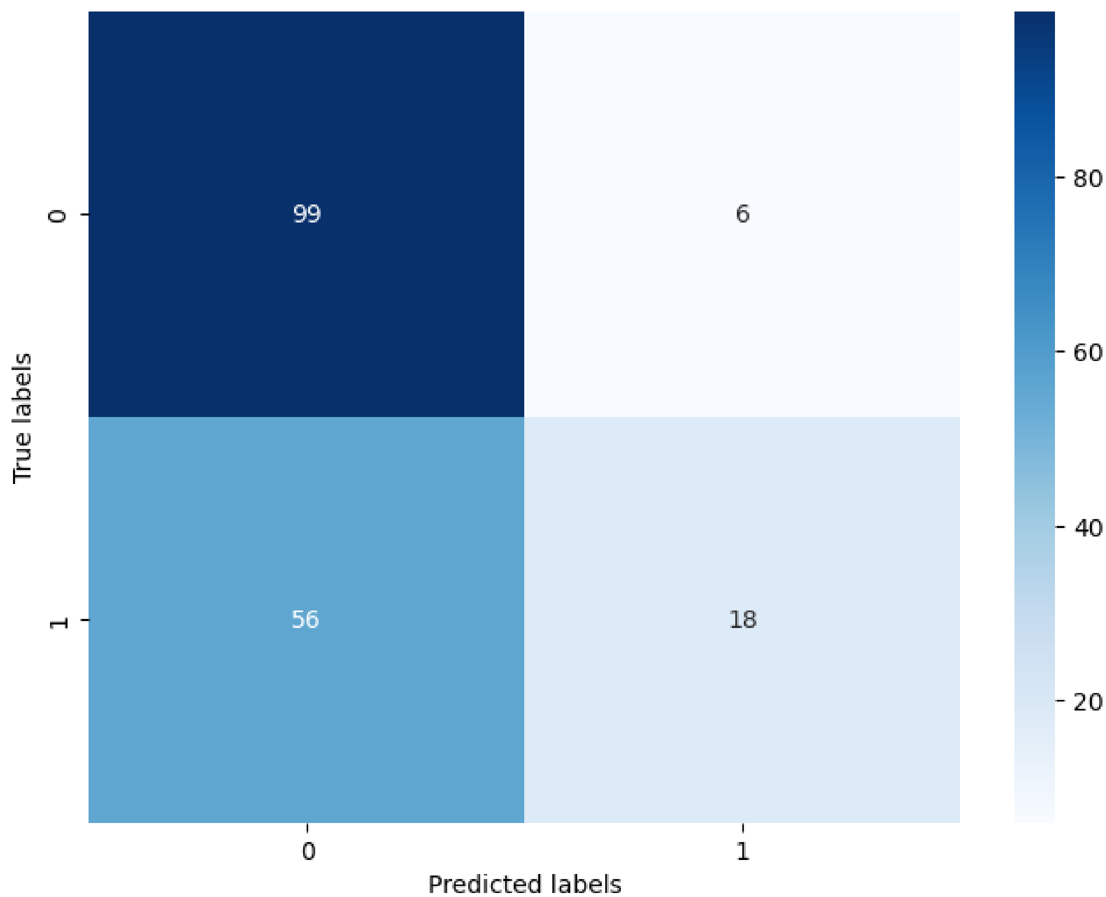
```
Accuracy: 0.6536312849162011
Classification Report:
               precision    recall  f1-score   support

           0       0.64      0.94      0.76       105
           1       0.75      0.24      0.37        74

    accuracy                           0.65       179
   macro avg       0.69      0.59      0.56       179
weighted avg       0.68      0.65      0.60       179
```

```
Confusion Matrix: [[99  6]
 [56 18]]
```

[39]:
```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()
```



[40]:
```python
#Handle missing Age values
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
data['Age'] = data['Age'].fillna(data['Age'].mean())
```

[41]:
```python
# Define target variable (y) and feature variables (X)
y = data['Age']
X = data[['Pclass', 'Sex', 'SibSp', 'Parch', 'Fare']]
```

[42]:
```python
# 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, y, test_size=0.2,␣
↪random_state=42)
```

[43]:
```python
# Create and train SVM Regressor model
from sklearn.svm import SVR
model = SVR(kernel='linear', C=1, epsilon=0.1) #we can use all these␣
↪("linear","rdf",'poly','sigmoid')
model.fit(X_train, y_train)
```

[43]:
```
SVR(C=1, kernel='linear')
```

[44]:
```python
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

```
[25.92974974 32.23856492 29.59949925 29.7741793  26.44052654 33.99114561
 28.60763877 25.27021458 28.60763877 30.88964316 32.7688916  29.59969898
 20.60911733 29.59838079 32.24255945 32.41935659 32.76801952 28.60776531
 32.24255945 34.93982531 29.5994526  34.89916107 26.43511396 29.6015764
 29.61000485 20.09583069 34.90103179 32.24255945 20.09583069 28.60780526
 29.5994526  28.60763877 34.90578527 28.60731921 29.59969898 27.43724037
 34.91346146 28.60763877 33.91469045 29.5994526  27.99384429 27.42540316
 29.59969898 29.59921964 27.10959534 22.94280364 29.59987202 29.6015764
 29.59810117 32.02022014 22.77188879 30.33310676 21.42370238 29.85827411
 29.59921964 30.97491295 32.23856492 32.75189387 31.26376105 28.6078452
 29.59925958 29.75260888 29.0990588  34.89816244 29.59921964 31.24698405
 32.33922689 29.59941936 25.91694075 31.81752231 27.59030334 26.92148999
 34.90467351 30.3165827  29.59938613 27.43899125 28.60763877 30.92673901
 31.25097858 25.61004056 19.45072902 31.24698405 30.39861693 29.59919966
 34.90578527 34.9358375  31.87345888 33.95612036 34.85673924 29.59842073
 26.60212074 16.93695113 34.90838171 29.5991864  29.5994526  29.59938613
 32.76713401 29.59969898 31.25097858 29.5994526  34.89828228 29.59822101
 31.83799424 29.59810117 29.59969898 27.42656828 27.61107486 34.91825489
 29.6020158  34.90627132 31.77555314 29.59842073 31.83483856 34.89874165
 35.21110003 29.60256832 34.91211003 32.24235972 27.59509677 21.95122277
 26.38602469 33.98705793 31.81702955 29.25061348 29.59842073 34.89916107
 33.90658828 34.98328571 26.44566605 29.59969898 28.6078452  25.92974974
 32.22178793 28.61518841 32.77500993 26.4456595  34.0423287  29.59925958
 29.59890007 26.45100545 29.60031142 29.0990588  32.24255945 30.09063967
 21.09801467 27.6006891  32.24255945 32.23856492 31.92657932 29.59969898
 29.60075753 29.59881347 29.59929282 26.46115154 29.59969898 29.59810117
 29.67710611 28.60769868 29.0990588  24.93857504 29.5983875  34.89916107
 32.24255945 31.82376032 29.60035136 33.44701471 32.23856492 31.77533344
 29.5994526  29.59810117 33.44921825 28.23346767 34.89890143 24.93816887
 29.59822101 29.59838079 18.92893841 31.24698405 24.94049241]
```

[46]:

```
# 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: 144.6838854003675
Mean Absolute Error: 8.772704271747479
R-squared: 0.14545454964701998
```

[45]:

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