Assignment_3

September 20, 2023

1 Perform Data preprocessing on Titanic dataset

Anand Misra 21BAI1105

1.1 Importing the Libraries

[65]:
```python
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 LabelEncoder, StandardScaler
```

1.2 Importing the Dataset

[66]:
```python
data = pd.read_csv('Titanic-Dataset.csv')
```

1.3 Checking for Null Values

[67]:
```python
null_counts = data.isnull().sum()
null_counts
```

[67]:
```
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
```

1.3.1 Handling missing values

Age: filling the missing values with the mean of the 'Age' column.

[68]:
```
mean_age = data['Age'].mean()
data['Age'].fillna(mean_age, inplace=True)
```

Cabin: there are a large number of missing values in the 'Cabin' column; hence dropping it.

[69]:
```
data.drop('Cabin', axis=1, inplace=True)
```

Embarked: there are only two missing values, therefore filling this with the most frequent value (mode).

[70]:
```
mode_embarked = data['Embarked'].mode()[0]
data['Embarked'].fillna(mode_embarked, inplace=True)
```

1.3.2 Checking again

[71]:
```
null_counts = data.isnull().sum()
print(null_counts)
```

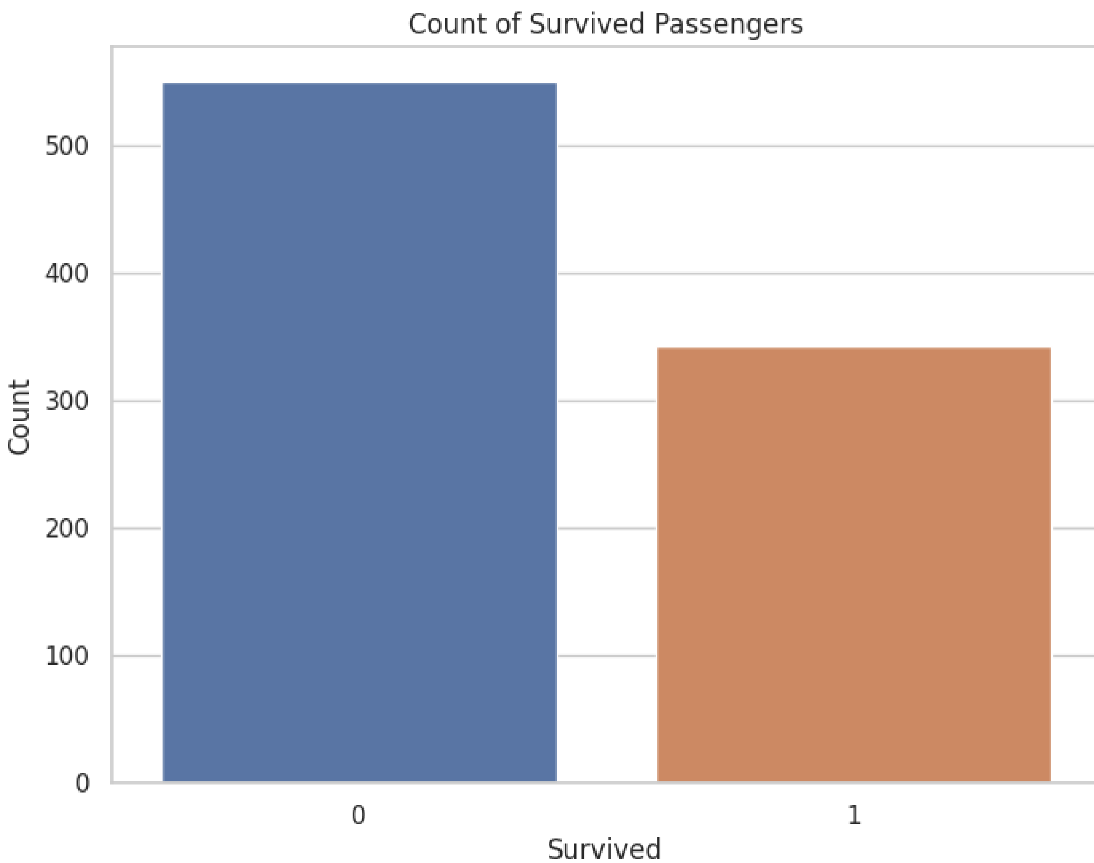
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

1.4 Data Visualization

[72]:
```
sns.set(style="whitegrid")
```

1.4.1 Plot 1: Countplot of Survived Passengers

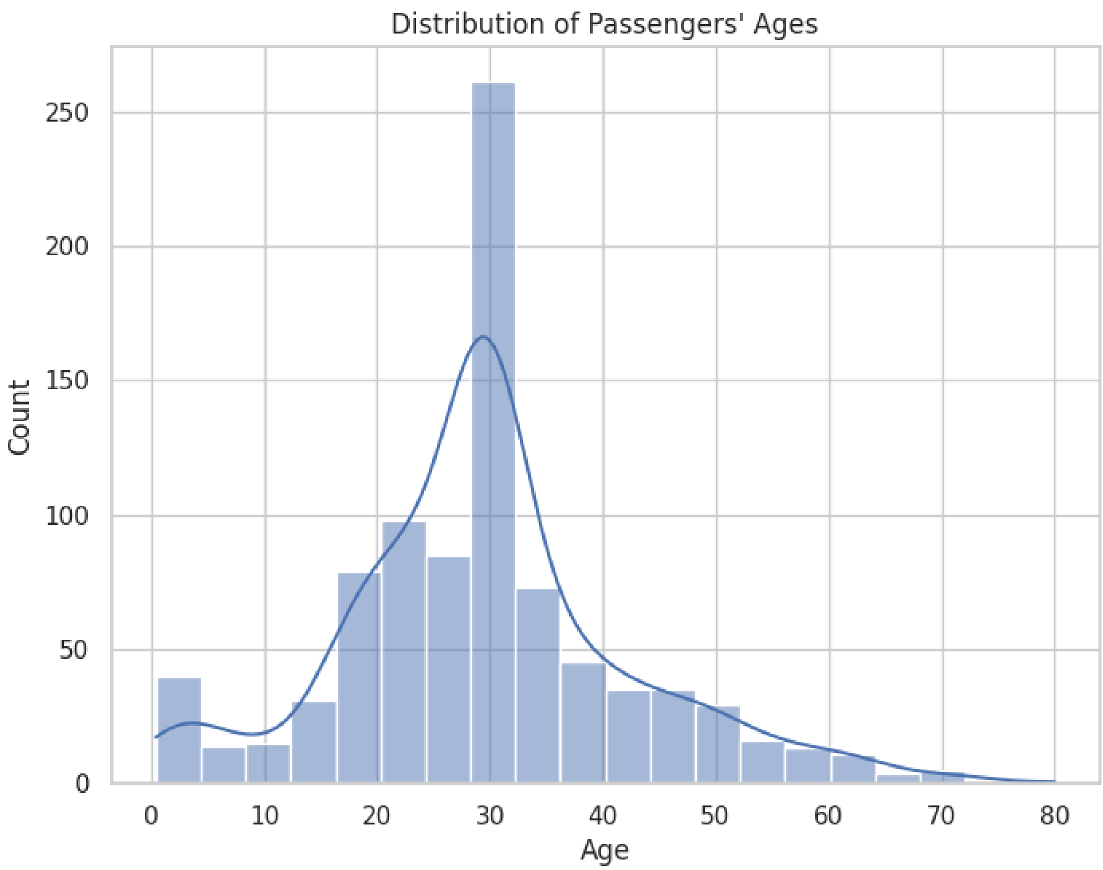
[73]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='Survived', data=data)
plt.title('Count of Survived Passengers')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.show()
```



The countplot shows that more passengers did not survive (Survived=0) than those who survived (Survived=1).

1.4.2 Plot 2: Distribution of Passengers' Ages

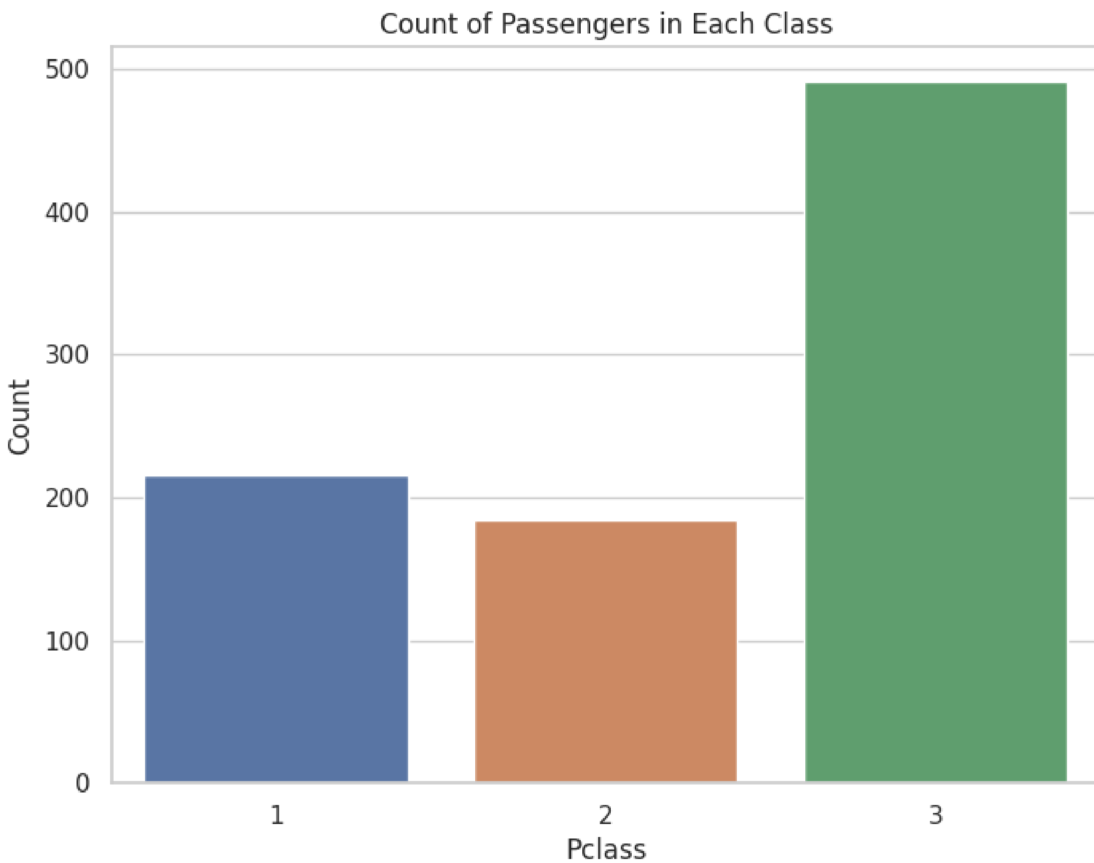
[74]:
```
plt.figure(figsize=(8, 6))
sns.histplot(data['Age'].dropna(), bins=20, kde=True)
plt.title('Distribution of Passengers\' Ages')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



The distribution of passengers' ages is somewhat right-skewed, with a higher concentration of passengers in the younger age groups.

1.4.3 Plot 3: Countplot of Passengers' Classes (Pclass)

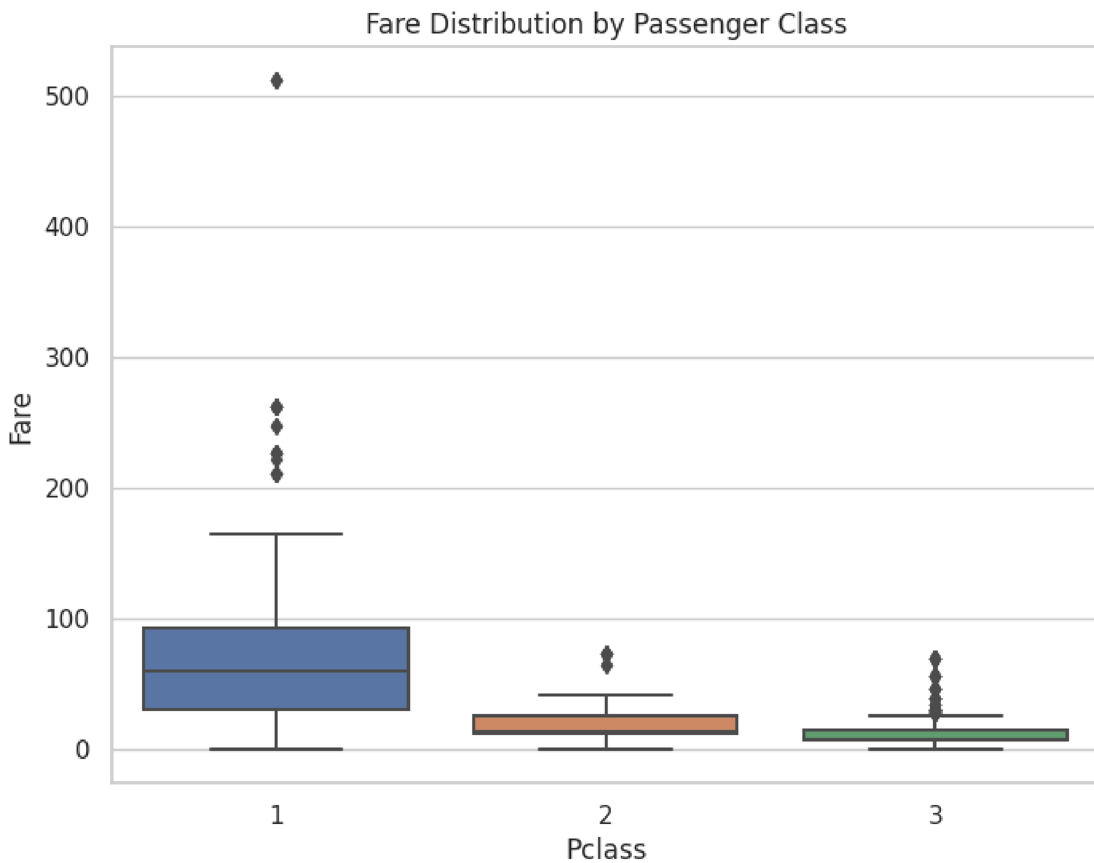
[75]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='Pclass', data=data)
plt.title('Count of Passengers in Each Class')
plt.xlabel('Pclass')
plt.ylabel('Count')
plt.show()
```



Most passengers were in the 3rd class (Pclass=3), followed by the 1st class (Pclass=1), and the 2nd class (Pclass=2).

1.4.4 Plot 4: Boxplot of Fare by Passenger Class (Pclass)

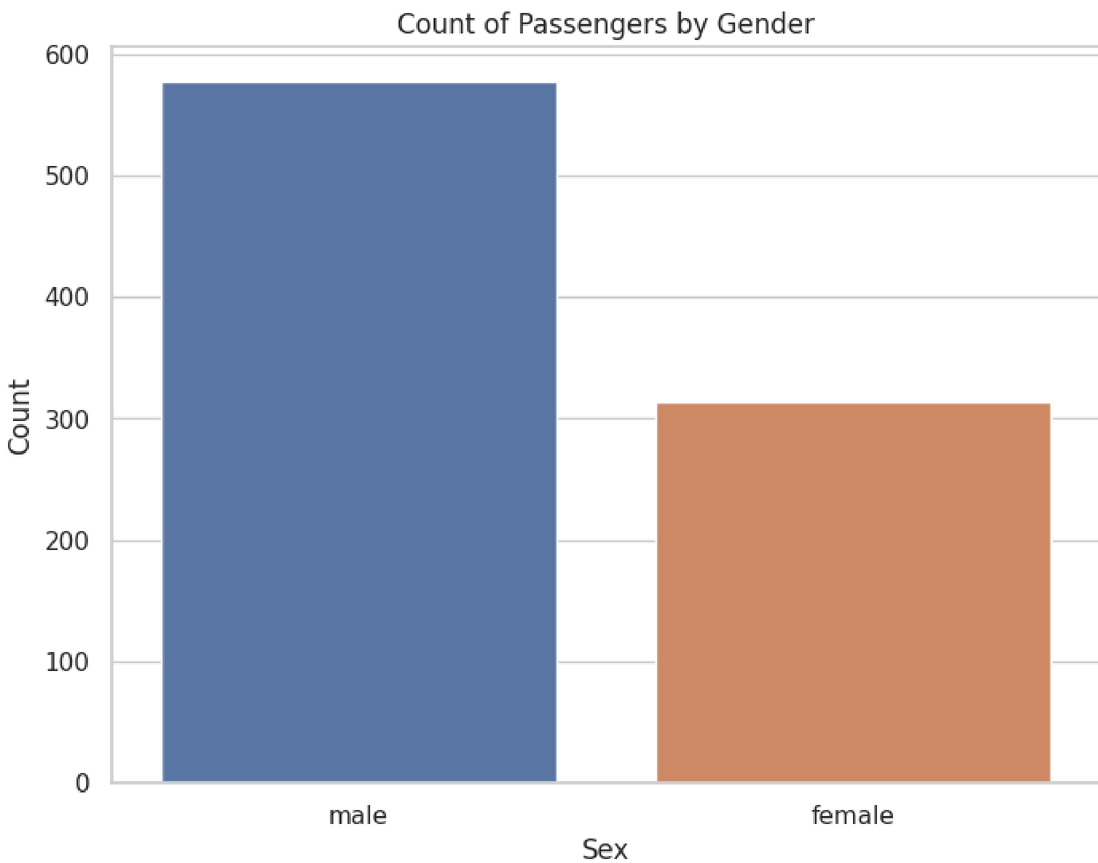
[76]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Pclass', y='Fare', data=data)
plt.title('Fare Distribution by Passenger Class')
plt.xlabel('Pclass')
plt.ylabel('Fare')
plt.show()
```



The boxplot reveals that 1st class passengers (Pclass=1) generally paid higher fares compared to 2nd and 3rd class passengers. There are also some outliers in the 1st class fare distribution.

1.4.5 Plot 5: Countplot of Passengers' Gender (Sex)

[77]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='Sex', data=data)
plt.title('Count of Passengers by Gender')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()
```



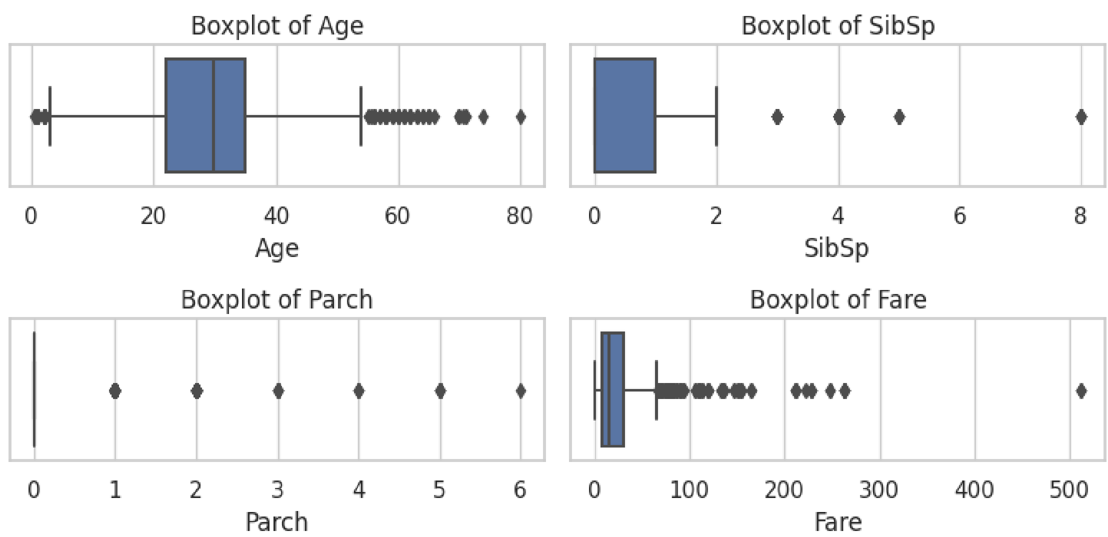
The countplot shows that there were more male passengers (Sex=Male) than female passengers (Sex=Female) on the Titanic.

1.5 Outlier Detection

[78]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(8,4))

sns.boxplot(x='Age', data=data, ax=axes[0, 0])
axes[0, 0].set_title('Boxplot of Age')
sns.boxplot(x='SibSp', data=data, ax=axes[0, 1])
axes[0, 1].set_title('Boxplot of SibSp')
sns.boxplot(x='Parch', data=data, ax=axes[1, 0])
axes[1, 0].set_title('Boxplot of Parch')
sns.boxplot(x='Fare', data=data, ax=axes[1, 1])
axes[1, 1].set_title('Boxplot of Fare')

plt.tight_layout()
plt.show()
```



- Outliers in 'Age' are not very apparent, but there are a few older passengers with ages significantly higher than the majority.
- 'SibSp' and 'Parch' have some outliers, which indicate passengers with an unusually large number of siblings/spouses or parents/children aboard.
- 'Fare' has visible outliers with fares much higher than the majority of passengers.

1.6 Splitting Dependent and Independent variables

- Before splitting, we have to drop 'Name' and 'Ticket' columns as it is not helpful for further process.

[79]:
```
data = data.drop(["Name", "Ticket"], axis=1)
```

[80]:
```
X = data.drop("Survived", axis=1)  # Attributes
y = data["Survived"]   # Target variable
```

[81]:
```
X.head()
```

[81]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	3	male	22.0	1	0	7.2500	S
1	2	1	female	38.0	1	0	71.2833	C
2	3	3	female	26.0	0	0	7.9250	S
3	4	1	female	35.0	1	0	53.1000	S
4	5	3	male	35.0	0	0	8.0500	S

[82]:
```
y.head()
```

[82]:
```
0    0
1    1
2    1
```

```
3    1
4    0
Name: Survived, dtype: int64
```

1.7 Encoding the required attributes

[83]:
```
label_encoder = LabelEncoder()

# Encoding the "Sex" column
X['Sex'] = label_encoder.fit_transform(X['Sex'])

# Encoding the "Embarked" column
X['Embarked'] = label_encoder.fit_transform(X['Embarked'])
```

[84]:
```
X.head()
```

[84]:
```
   PassengerId  Pclass  Sex   Age  SibSp  Parch     Fare  Embarked
0            1       3    1  22.0      1      0   7.2500         2
1            2       1    0  38.0      1      0  71.2833         0
2            3       3    0  26.0      0      0   7.9250         2
3            4       1    0  35.0      1      0  53.1000         2
4            5       3    1  35.0      0      0   8.0500         2
```

1.8 Feature Scaling

[89]:
```
scaler = StandardScaler()
cols = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X[cols] = scaler.fit_transform(X[cols])
```

[90]:
```
X.head()
```

[90]:
```
   PassengerId    Pclass  Sex       Age     SibSp     Parch      Fare  \
0            1  0.827377    1 -0.592481  0.432793 -0.473674 -0.502445
1            2 -1.566107    0  0.638789  0.432793 -0.473674  0.786845
2            3  0.827377    0 -0.284663 -0.474545 -0.473674 -0.488854
3            4 -1.566107    0  0.407926  0.432793 -0.473674  0.420730
4            5  0.827377    1  0.407926 -0.474545 -0.473674 -0.486337

   Embarked
0  0.585954
1 -1.942303
2  0.585954
3  0.585954
4  0.585954
```

1.9 Splitting Data into Train and Test

[91]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
```

[92]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (712, 8)
X_test shape: (179, 8)
y_train shape: (712,)
y_test shape: (179,)
```

[ ]:
```
```