titanic-survival-prediction

November 20, 2023

[44]:
```
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.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[45]:
```
data=pd.read_csv("test.csv")
```

[46]:
```
data
```

[46]:
```
     PassengerId  Survived  Pclass  \
0            892         0       3
1            893         1       3
2            894         0       2
3            895         0       3
4            896         1       3
..           …       …     …
413         1305         0       3
414         1306         1       1
415         1307         0       3
416         1308         0       3
417         1309         0       3

                                             Name     Sex   Age  SibSp  Parch  \
0                                Kelly, Mr. James    male  34.5      0      0
1                Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                       Myles, Mr. Thomas Francis    male  62.0      0      0
3                                Wirz, Mr. Albert    male  27.0      0      0
4    Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1
..                                            …     …   …    …    …
413                            Spector, Mr. Woolf    male   NaN      0      0
414                  Oliva y Ocana, Dona. Fermina  female  39.0      0      0
415                  Saether, Mr. Simon Sivertsen    male  38.5      0      0
416                           Ware, Mr. Frederick    male   NaN      0      0
417                     Peter, Master. Michael J    male   NaN      1      1
```

```
                 Ticket      Fare Cabin Embarked
0                330911    7.8292   NaN        Q
1                363272    7.0000   NaN        S
2                240276    9.6875   NaN        Q
3                315154    8.6625   NaN        S
4               3101298   12.2875   NaN        S
..                  …        …    …        …
413           A.5. 3236    8.0500   NaN        S
414            PC 17758  108.9000  C105        C
415  SOTON/O.Q. 3101262    7.2500   NaN        S
416              359309    8.0500   NaN        S
417                2668   22.3583   NaN        C

[418 rows x 12 columns]
```

[47]:
```
data.shape
```

[47]:
```
(418, 12)
```

[48]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
 2   Pclass       418 non-null    int64
 3   Name         418 non-null    object
 4   Sex          418 non-null    object
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64
 7   Parch        418 non-null    int64
 8   Ticket       418 non-null    object
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object
 11  Embarked     418 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

[49]:
```
data.isnull().sum()
```

[49]:
```
PassengerId      0
Survived         0
Pclass           0
```

```
Name            0
Sex             0
Age            86
SibSp           0
Parch           0
Ticket          0
Fare            1
Cabin         327
Embarked        0
dtype: int64
```

[50]:
```
data=data.drop(columns='Cabin',axis=1)
```

[51]:
```
data['Age'].fillna(data['Age'].mean(),inplace=True)
```

[52]:
```
data['Embarked'].fillna(data['Embarked'].mode()[0],inplace=True)
```

[53]:
```
data['Fare'].fillna(data['Fare'].mode()[0],inplace=True)
```

[54]:
```
data.isnull().sum().sum()
```

[54]:
```
0
```

[55]:
```
data['Survived'].value_counts()
```

[55]:
```
0    266
1    152
Name: Survived, dtype: int64
```

[56]:
```
data.describe()
```

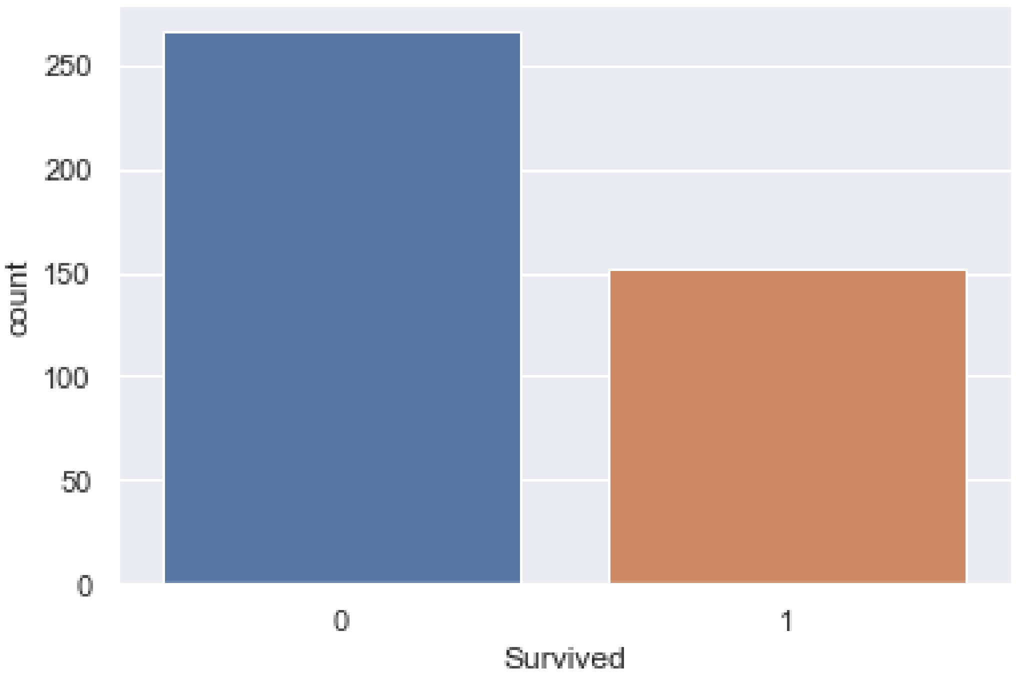
[56]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   418.000000  418.000000  418.000000  418.000000  418.000000
mean   1100.500000    0.363636    2.265550   30.272590    0.447368
std     120.810458    0.481622    0.841838   12.634534    0.896760
min     892.000000    0.000000    1.000000    0.170000    0.000000
25%     996.250000    0.000000    1.000000   23.000000    0.000000
50%    1100.500000    0.000000    3.000000   30.272590    0.000000
75%    1204.750000    1.000000    3.000000   35.750000    1.000000
max    1309.000000    1.000000    3.000000   76.000000    8.000000

            Parch        Fare
count  418.000000  418.000000
mean     0.392344   35.560497
std      0.981429   55.857145
min      0.000000    0.000000
25%      0.000000    7.895800
```

```
50%       0.000000   14.454200
75%       0.000000   31.471875
max       9.000000  512.329200
```

[57]:
```
sns.set()
```

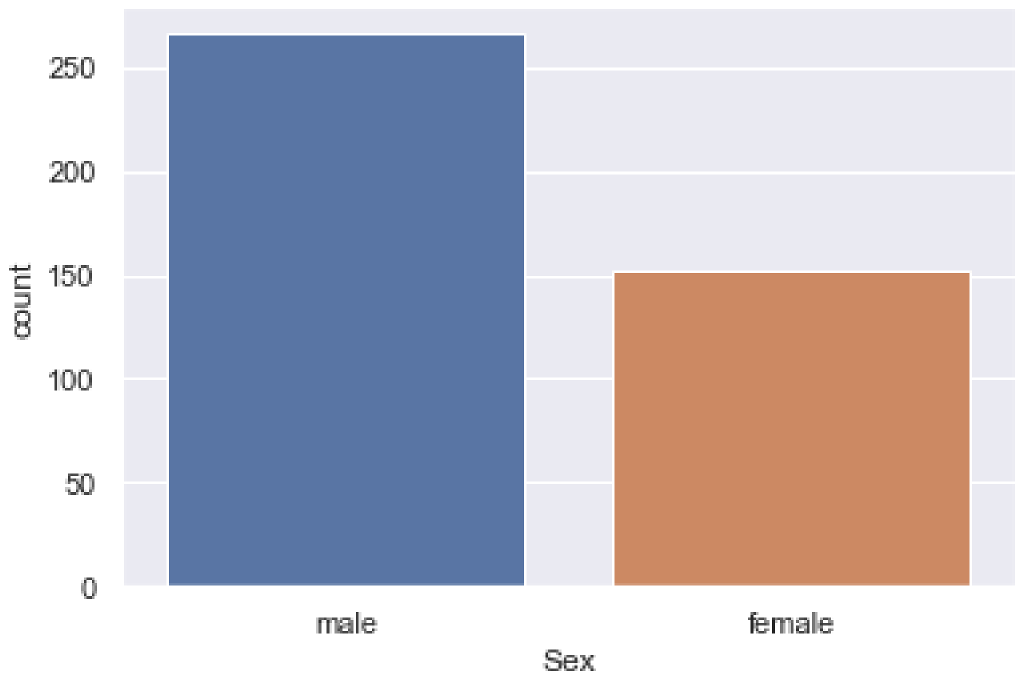
[58]:
```
sns.countplot(x='Survived',data=data)
```

[58]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



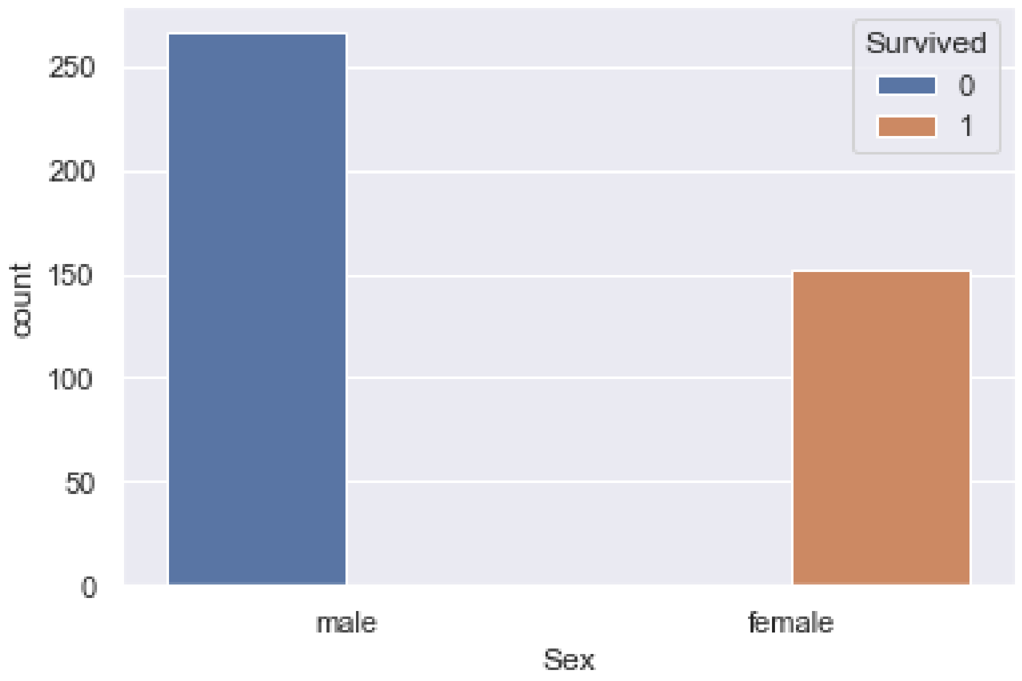
[59]:
```
sns.countplot(x='Sex',data=data)
```

[59]:
```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



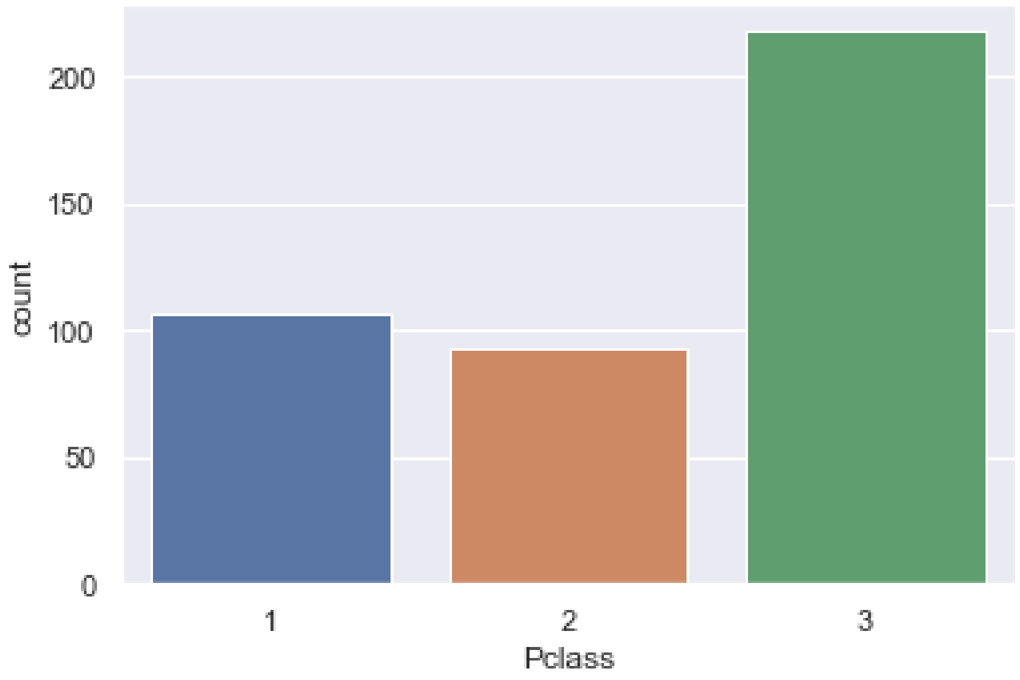
```
[60]: sns.countplot(x='Sex',hue='Survived',data=data)
```

```
[60]: <AxesSubplot:xlabel='Sex', ylabel='count'>
```



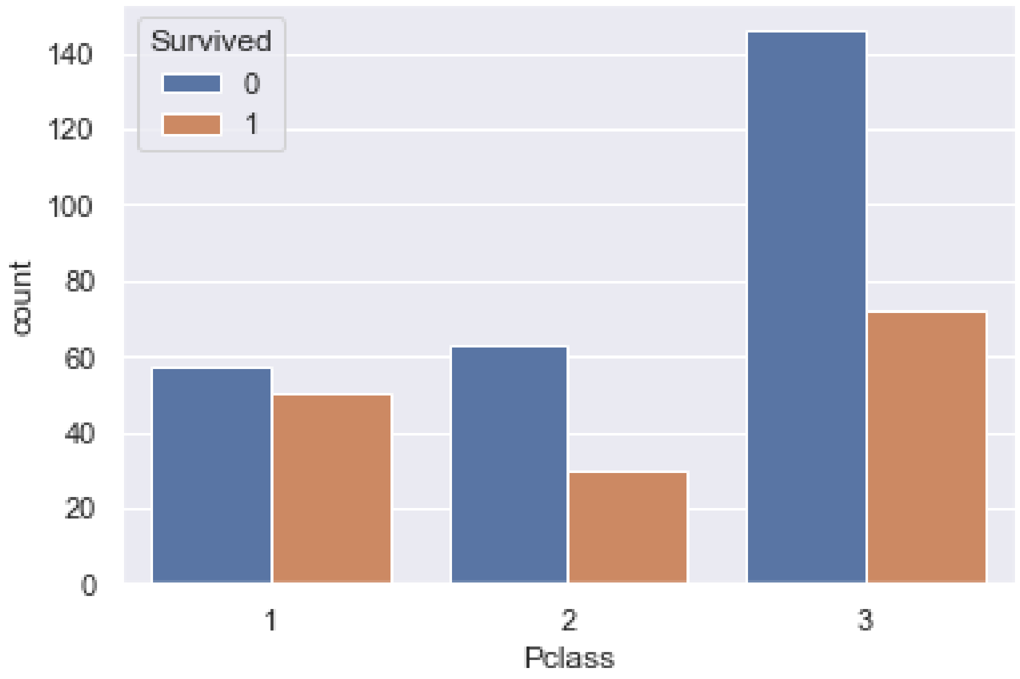
```
[61]: sns.countplot(x='Pclass',data=data)
```

```
[61]: <AxesSubplot:xlabel='Pclass', ylabel='count'>
```



```
[62]: sns.countplot(x='Pclass',hue='Survived',data=data)
```

```
[62]: <AxesSubplot:xlabel='Pclass', ylabel='count'>
```



[63]:
```
data['Sex'].value_counts()
```

[63]:
```
male      266
female    152
Name: Sex, dtype: int64
```

[64]:
```
data['Embarked'].value_counts()
```

[64]:
```
S    270
C    102
Q     46
Name: Embarked, dtype: int64
```

[65]:
```
data.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':
 ↪2}},inplace=True)
```

[66]:
```
data
```

[66]:
```
     PassengerId  Survived  Pclass  \
0            892         0       3
1            893         1       3
2            894         0       2
3            895         0       3
4            896         1       3
```

```
..           …        …      …
413          1305        0       3
414          1306        1       1
415          1307        0       3
416          1308        0       3
417          1309        0       3

                                              Name  Sex       Age  SibSp  \
0                                 Kelly, Mr. James    0  34.50000      0
1                 Wilkes, Mrs. James (Ellen Needs)    1  47.00000      1
2                        Myles, Mr. Thomas Francis    0  62.00000      0
3                                 Wirz, Mr. Albert    0  27.00000      0
4     Hirvonen, Mrs. Alexander (Helga E Lindqvist)    1  22.00000      1
..                                             …   …       …    …
413                             Spector, Mr. Woolf    0  30.27259      0
414                   Oliva y Ocana, Dona. Fermina    1  39.00000      0
415                   Saether, Mr. Simon Sivertsen    0  38.50000      0
416                            Ware, Mr. Frederick    0  30.27259      0
417                      Peter, Master. Michael J     0  30.27259      1

     Parch              Ticket      Fare  Embarked
0        0              330911    7.8292         2
1        0              363272    7.0000         0
2        0              240276    9.6875         2
3        0              315154    8.6625         0
4        1             3101298   12.2875         0
..     …                 …       …        …
413      0           A.5. 3236    8.0500         0
414      0            PC 17758  108.9000         1
415      0  SOTON/O.Q. 3101262    7.2500         0
416      0              359309    8.0500         0
417      1                2668   22.3583         1

[418 rows x 11 columns]
```

[67]:
```
X=data.drop(columns=['PassengerId','Name','Ticket'],axis=1)
```

[68]:
```
Y=data['Survived']
```

[69]:
```
print(X)
```

```
     Survived  Pclass  Sex       Age  SibSp  Parch      Fare  Embarked
0           0       3    0  34.50000      0      0    7.8292         2
1           1       3    1  47.00000      1      0    7.0000         0
2           0       2    0  62.00000      0      0    9.6875         2
3           0       3    0  27.00000      0      0    8.6625         0
4           1       3    1  22.00000      1      1   12.2875         0
```

```
..        ...      ...  ...       ...      ...      ...       ...       ...
413         0        3    0  30.27259        0        0    8.0500            0
414         1        1    1  39.00000        0        0  108.9000            1
415         0        3    0  38.50000        0        0    7.2500            0
416         0        3    0  30.27259        0        0    8.0500            0
417         0        3    0  30.27259        1        1   22.3583            1

[418 rows x 8 columns]
```

[70]:
```python
print(Y)
```

```
0      0
1      1
2      0
3      0
4      1
      ..
413    0
414    1
415    0
416    0
417    0
Name: Survived, Length: 418, dtype: int64
```

[71]:
```python
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

[72]:
```python
print(X.shape,X_train.shape,X_test.shape)
```

```
(418, 8) (334, 8) (84, 8)
```

[73]:
```python
model=LogisticRegression(max_iter=1000)
```

[74]:
```python
model.fit(X_train,Y_train)
```

[74]:
```
LogisticRegression(max_iter=1000)
```

[75]:
```python
X_train_prediction=model.predict(X_train)
```

[76]:
```python
print(X_train_prediction)
```

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

[77]:
```
train_data_accuracy=accuracy_score(Y_train,X_train_prediction)
```

[78]:
```
print("Accuracy Score of training data: ",train_data_accuracy)
```

```
Accuracy Score of training data:  1.0
```

[79]:
```
X_test_prediction=model.predict(X_test)
```

[80]:
```
print(X_test_prediction)
```

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

[81]:
```
test_data_accuracy=accuracy_score(Y_test,X_test_prediction)
```

[82]:
```
print("Accuracy score of testing data:",test_data_accuracy)
```

```
Accuracy score of testing data: 1.0
```