```
import numpy as np
import pandas as pd
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
```

Data collection and Preprocessing

```
titanic_data = pd.read_csv('/content/train (1).csv')
```

```
titanic_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450

Next steps: Generate code with titanic_data | View recommended plots | New interactive sheet

```
titanic_data.shape
```

```
(891, 12)
```

```
titanic_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
```

Checking for missing values

```
titanic_data.isnull().sum()
```

	0
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

Dropping Cabin axis and replacing Age missing values with their mean

```
titanic_data = titanic_data.drop(columns='Cabin', axis=1)
titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
```

```
/tmp/ipython-input-9-3693345471.py:2: FutureWarning: A value is trying to be s
The behavior will change in pandas 3.0. This inplace method will never work be

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.m


  titanic_data['Age'].fillna(titanic_data['Age'].mean(), inplace=True)
```

replacing the missing values in Embarked column with mode value

```
print(titanic_data['Embarked'].mode())
```

```
0    S
Name: Embarked, dtype: object
```

```
print(titanic_data['Embarked'].mode()[0])
```

```
S
```

```
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=True)
```

```
/tmp/ipython-input-12-3993763136.py:1: FutureWarning: A value is trying to be
The behavior will change in pandas 3.0. This inplace method will never work be

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.m


  titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0], inplace=
```

```
titanic_data.isnull().sum()
```

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	0
SibSp	0
Parch	0
Ticket	0
Fare	0
Embarked	0

dtype: int64

Data analysis

```
titanic_data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	F
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329

```
titanic_data['Survived'].value_counts()
```

Survived	count
0	549
1	342

dtype: int64

```
titanic_data['Sex'].value_counts()
```

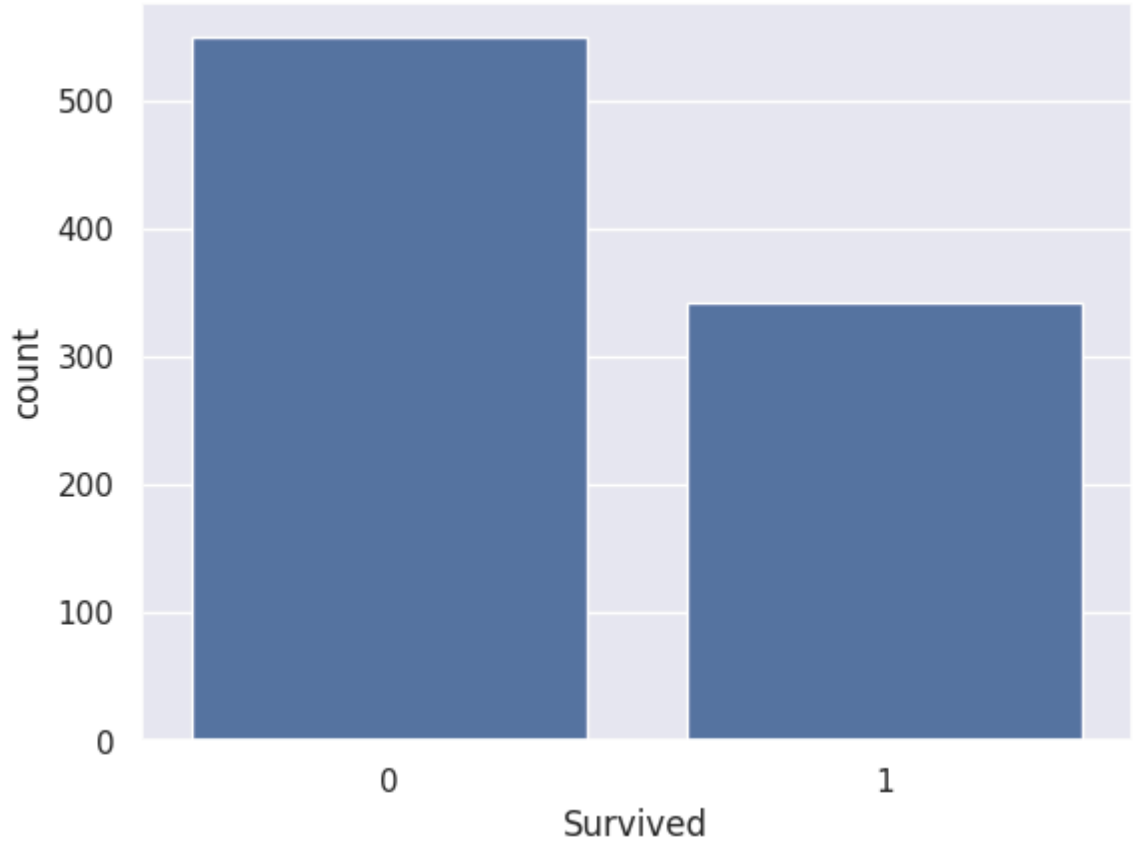
Sex	count
male	577
female	314

dtype: int64

```
sns.set()
```

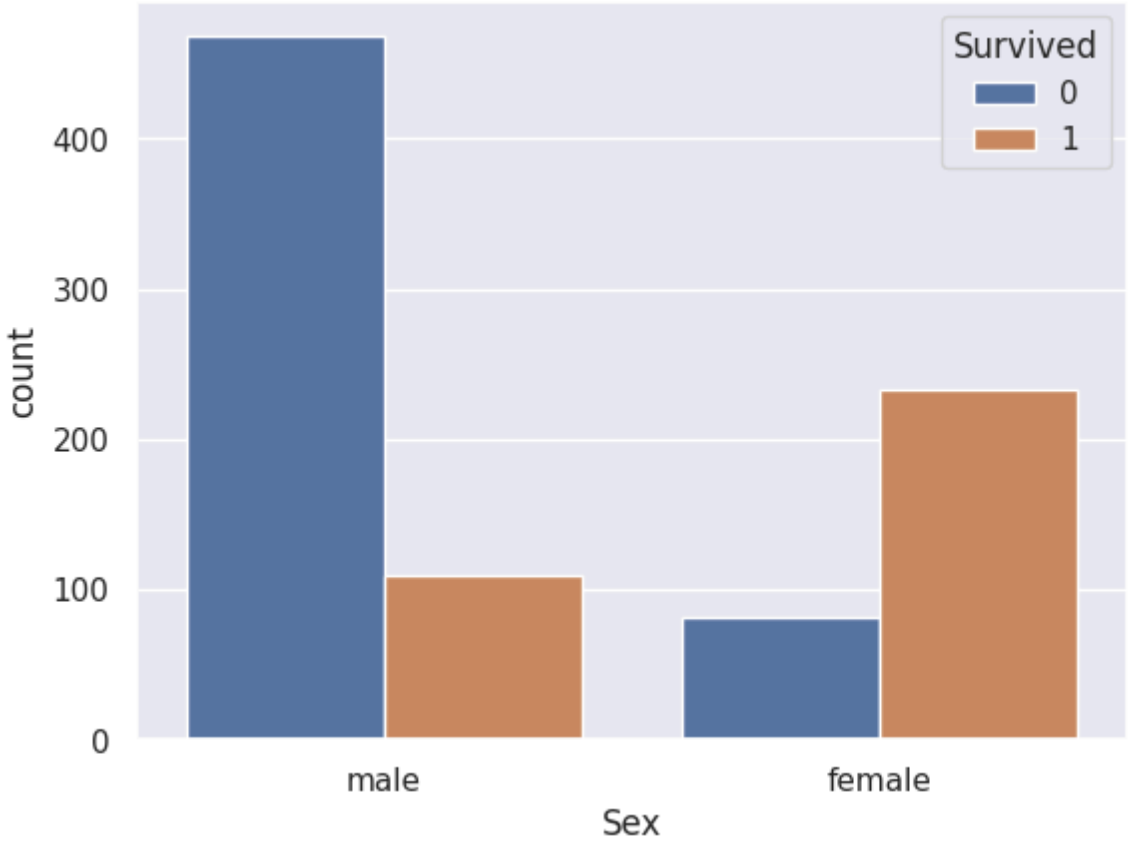
```
sns.countplot(x='Survived', data=titanic_data)
```

```
<Axes: xlabel='Survived', ylabel='count'>
```



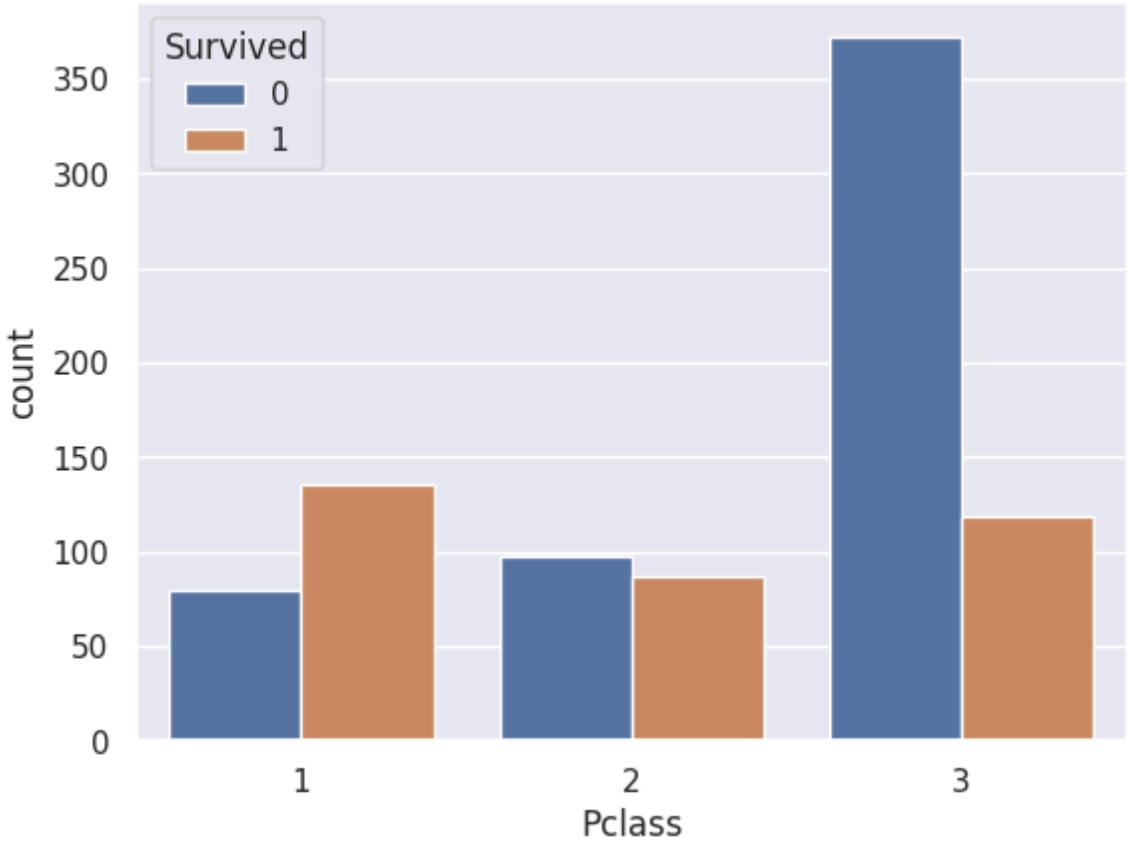

```
sns.countplot(x='Sex', hue='Survived', data=titanic_data)
```

```
<Axes: xlabel='Sex', ylabel='count'>
```




```
sns.countplot(x='Pclass', hue='Survived', data=titanic_data)
```

```
<Axes: xlabel='Pclass', ylabel='count'>
```



Label Encoding

```
titanic_data['Sex'].value_counts()
```

Sex	count
male	577
female	314

dtype: int64

```
titanic_data['Embarked'].value_counts()
```

Embarked	count
S	646
C	168
Q	77

dtype: int64

```
titanic_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}
```

```
/tmp/ipython-input-28-4126089538.py:1: FutureWarning: Downcasting behavior in
  titanic_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'
```

```
titanic_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.

Next steps: Generate code with titanic_data | View recommended plots | New interactive sheet

Split data and target

```
x = titanic_data.drop(columns = ['PassengerId','Name','Ticket','Survived'],axis=1
y = titanic_data['Survived']
```

Split Test and train data

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_sta
```

```
print(x.shape, x_train.shape, x_test.shape)
```

```
(891, 7) (712, 7) (179, 7)
```

Model Tarining(Logistic Regression)

```
regressor = LogisticRegression()
regressor.fit(x_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465:
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression

```
LogisticRegression()
```

Model Evaluation

on train data

```
x_train_prediction = regressor.predict(x_train)
training_data_accuracy = accuracy_score(y_train, x_train_prediction)
print('Accuracy score of training data : ', training_data_accuracy)
```

```
Accuracy score of training data :  0.8075842696629213
```

on test data

```
x_test_prediction = regressor.predict(x_test)
test_data_accuracy = accuracy_score(y_test, x_test_prediction)
print('Accuracy score of test data : ', test_data_accuracy)
```

```
Accuracy score of test data :  0.7821229050279329
```