```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
customer= pd.read_csv('/content/Titanic.csv')
print(customer.describe())
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```python
customer.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
```

```python
print(customer.dtypes)
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

```
x= customer.describe([.25, .50, .75, .90])
print(x)
```

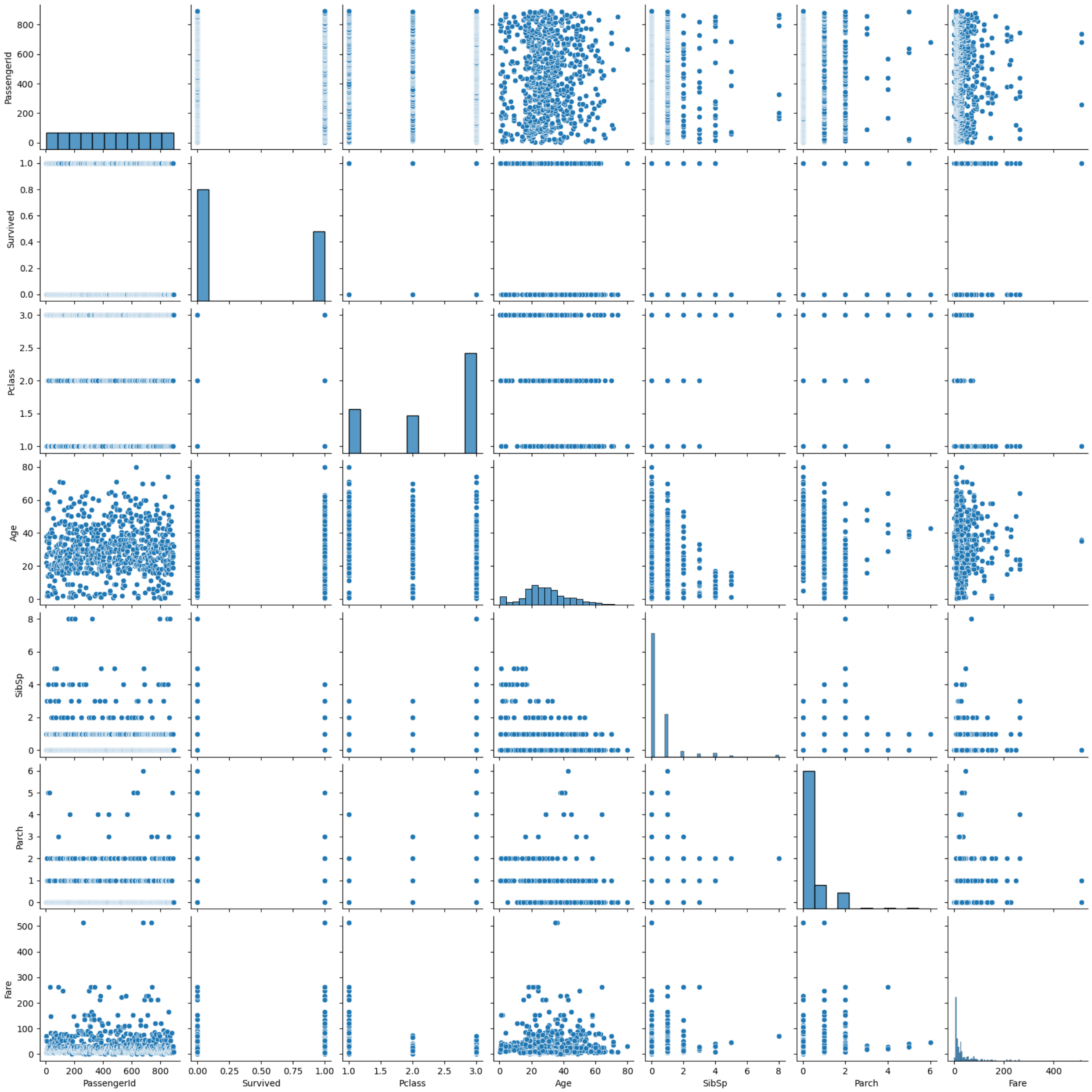
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
90%     802.000000    1.000000    3.000000   50.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
90%      2.000000   77.958300
max      6.000000  512.329200
```

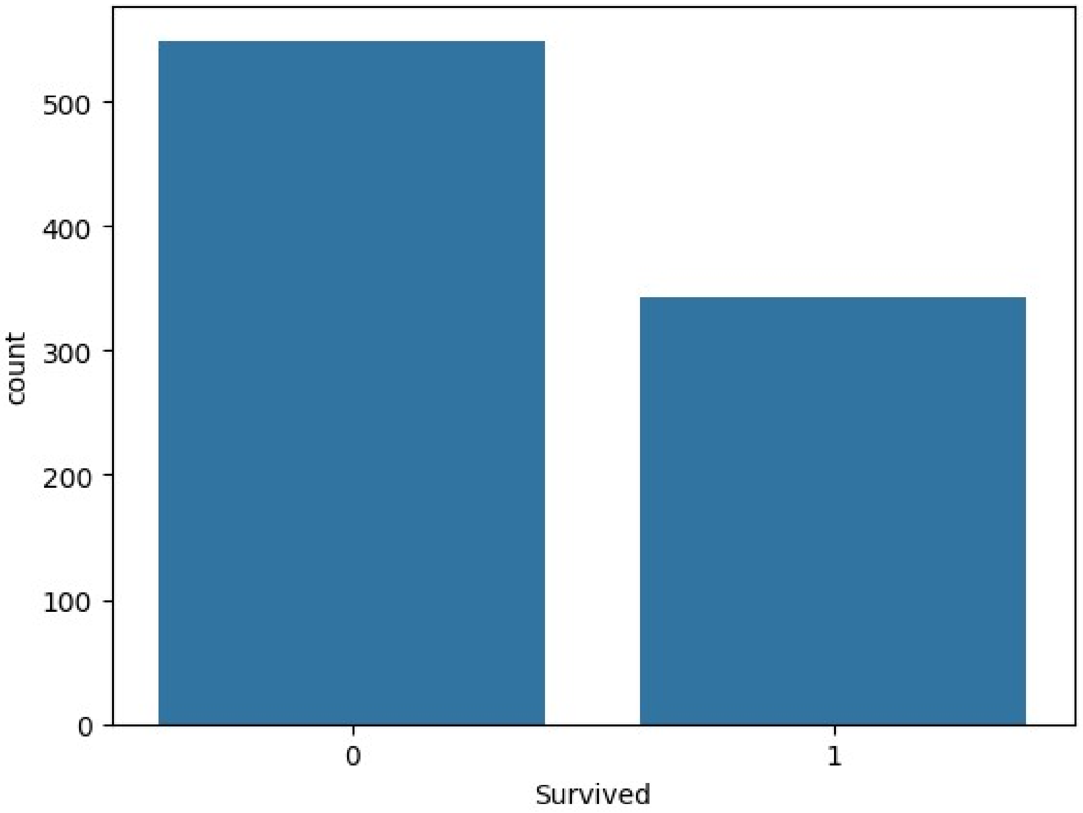
```
column= customer.columns.tolist()
print(column)
```

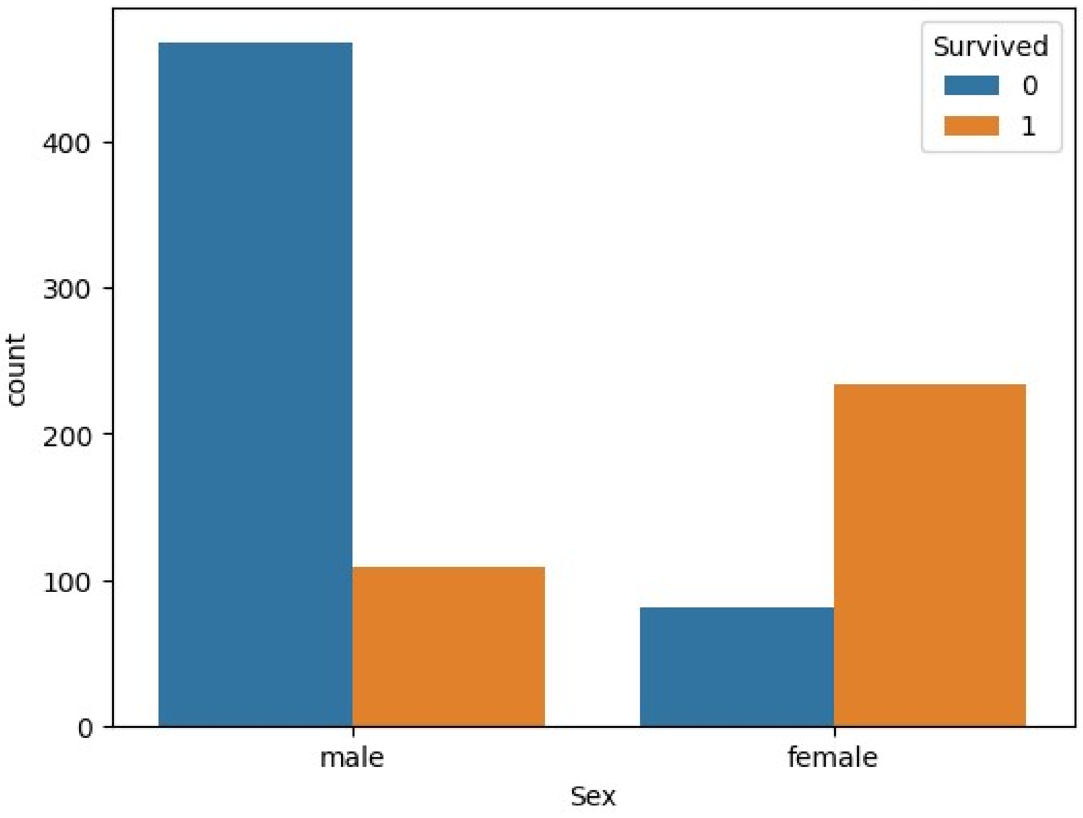
```
['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']
```

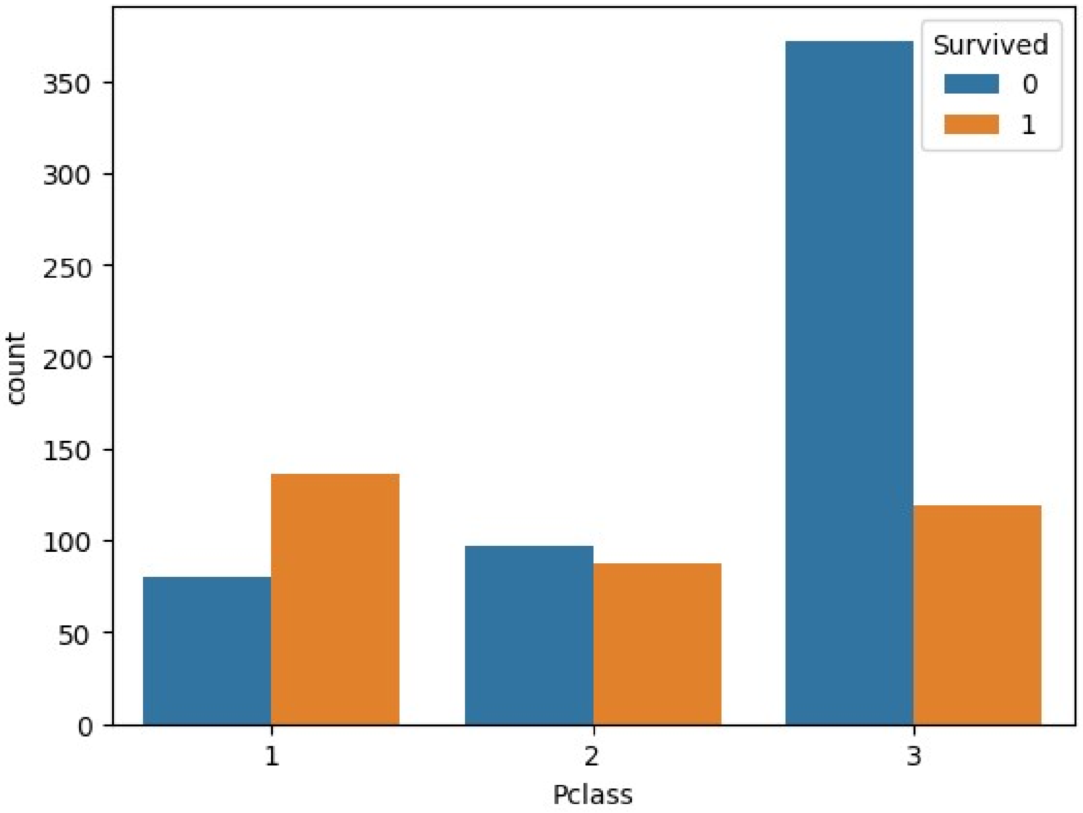
```
numeric_features = customer.select_dtypes(include=['int64',
'float64']).columns
sns.pairplot(customer[numeric_features])
plt.show()
```

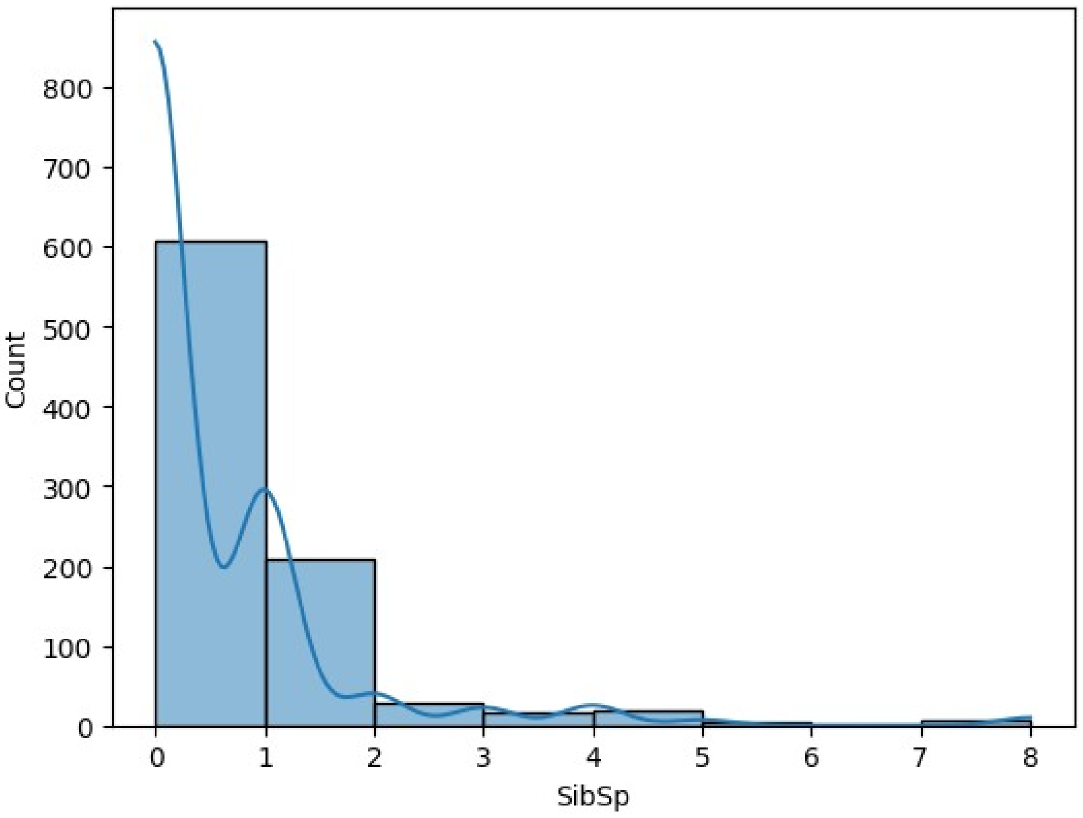


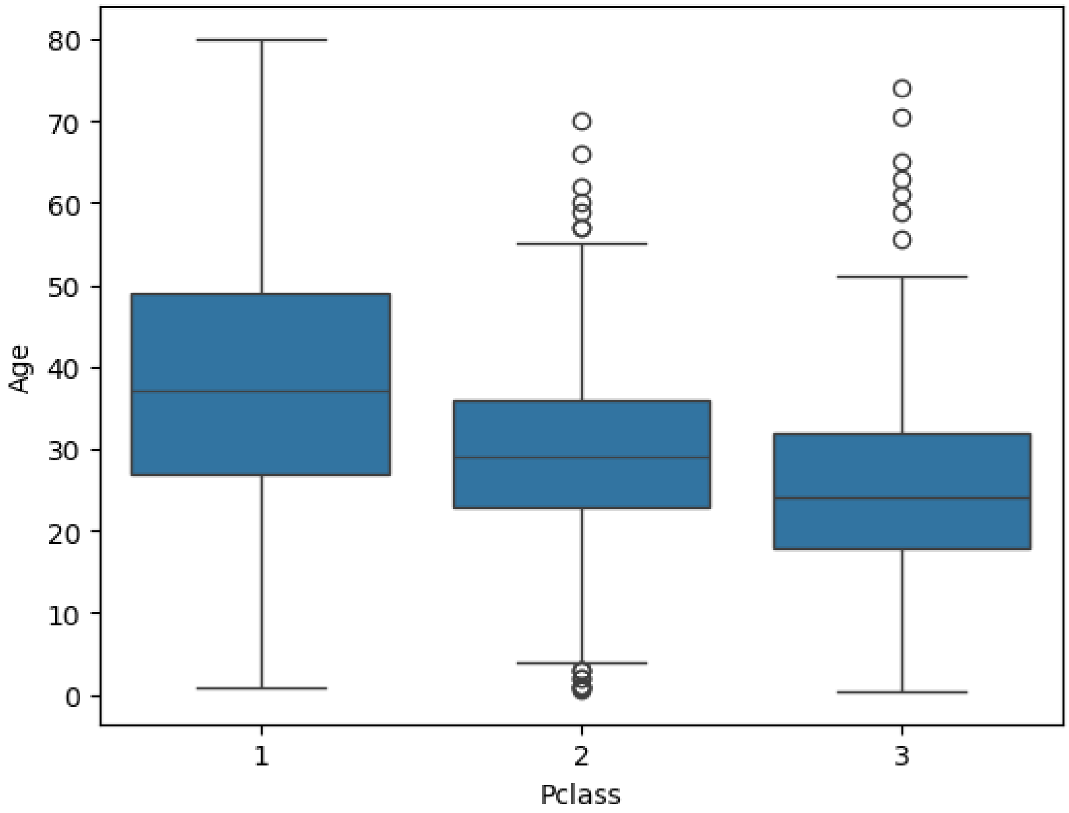
```
sns.countplot(x='Survived', data=customer)
plt.show()
sns.countplot(x='Sex', hue='Survived', data=customer)
plt.show()
sns.countplot(x='Pclass', hue='Survived', data=customer)
plt.show()
sns.histplot(x='SibSp', data=customer, bins=range(0, 9), kde=True)
plt.show()
sns.boxplot(x='Pclass', y='Age', data=customer)
plt.show()
```


500
400
300
200
100
0
count
0
1
Survived


Survived
0
1
400
300
200
100
0
count
male
female
Sex


Survived
0
1
count
0
50
100
150
200
250
300
350
1
2
3
Pclass


Count
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5
6
7
8
SibSp



```
customer['Age'].fillna(customer['Age'].median(), inplace=True)
# Recode categorical features to a class
customer['Sex'] = customer['Sex'].map({'male': 0, 'female': 1})
customer= pd.get_dummies(customer, columns=['Embarked'],
drop_first=True)
# Display the modified dataframe
print(customer.head())

   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
Parch  \
0                            Braund, Mr. Owen Harris    0  22.0      1
0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    1  38.0      1
0
2                             Heikkinen, Miss. Laina    1  26.0      0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    1  35.0      1
```

```
0
4                            Allen, Mr. William Henry    0  35.0      0
0

             Ticket     Fare Cabin  Embarked_Q  Embarked_S
0         A/5 21171   7.2500   NaN           0           1
1          PC 17599  71.2833   C85           0           0
2  STON/O2. 3101282   7.9250   NaN           0           1
3            113803  53.1000  C123           0           1
4            373450   8.0500   NaN           0           1
```

```
customer.drop(['Cabin', 'Ticket'], axis=1, inplace=True)

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
# Assuming 'df' is your DataFrame with the provided data
# Step 1: Split the data into X (features) and Y (target)
X =customer[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']]
Y = customer['Survived']
# Split the 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)

print(X_train)
```

```
     Pclass   Age  SibSp  Parch      Fare
331       1  45.5      0      0   28.5000
733       2  23.0      0      0   13.0000
382       3  32.0      0      0    7.9250
704       3  26.0      1      0    7.8542
813       3   6.0      4      2   31.2750
..      ...   ...    ...    ...       ...
106       3  21.0      0      0    7.6500
270       1  28.0      0      0   31.0000
860       3  41.0      2      0   14.1083
435       1  14.0      1      2  120.0000
102       1  21.0      0      1   77.2875

[712 rows x 5 columns]
```

```
print(Y_train)
```

```
331    0
733    0
382    0
704    0
813    0
```

```
      ..
106    1
270    0
860    0
435    1
102    0
Name: Survived, Length: 712, dtype: int64
```

```
print(X_test)
```

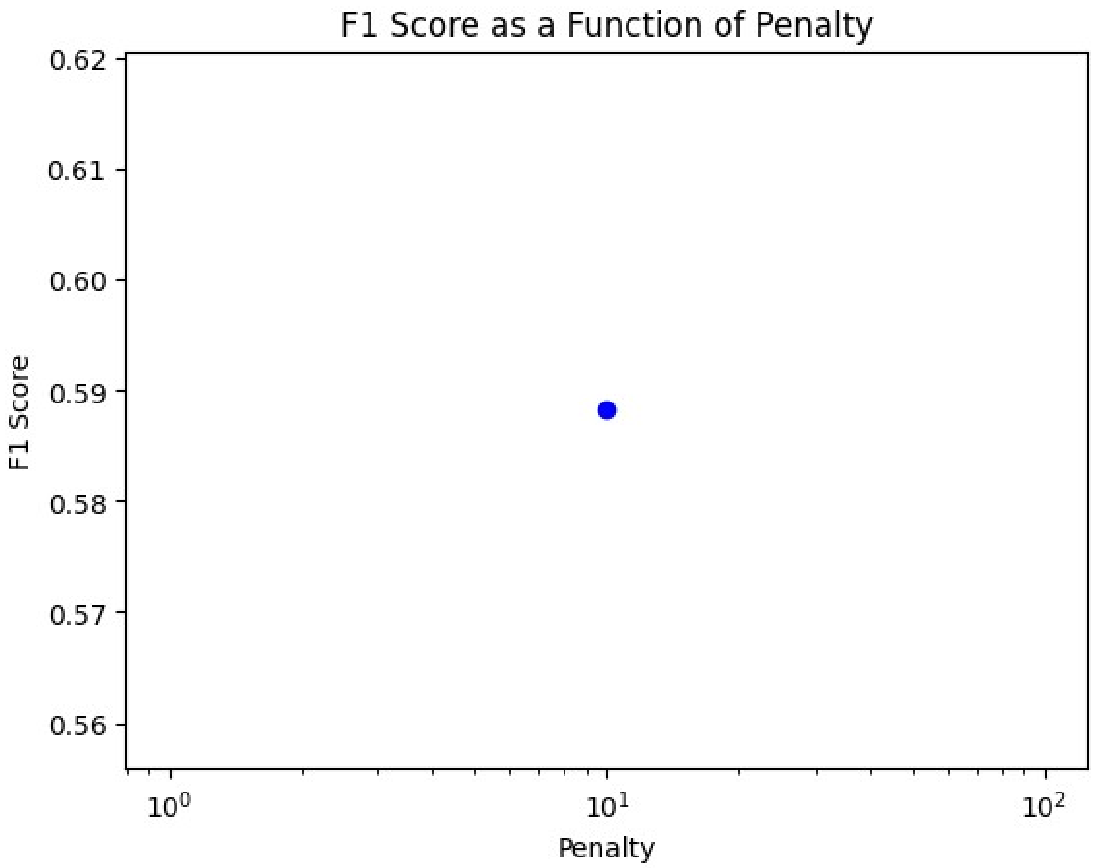
```
     Pclass   Age  SibSp  Parch     Fare
709       3  28.0      1      1  15.2458
439       2  31.0      0      0  10.5000
840       3  20.0      0      0   7.9250
720       2   6.0      0      1  33.0000
39        3  14.0      1      0  11.2417
..      ...   ...    ...    ...      ...
433       3  17.0      0      0   7.1250
773       3  28.0      0      0   7.2250
25        3  38.0      1      5  31.3875
84        2  17.0      0      0  10.5000
10        3   4.0      1      1  16.7000

[179 rows x 5 columns]
```

```
print(Y_test)
```

```
709    1
439    0
840    0
720    1
39     1
      ..
433    0
773    0
25     1
84     1
10     1
Name: Survived, Length: 179, dtype: int64
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
X = customer[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']]
y = customer['Survived']
X = X.fillna(X.mean())
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,
random_state=42)
model = LogisticRegression()
penalty_values = [0.1, 0.5, 1, 2, 5, 10]
f1_scores = []
penalties = []
for penalty in penalty_values:
  model.set_params(C=1/penalty)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
f1 = f1_score(y_test, y_pred)
f1_scores.append(f1)
penalties.append(penalty)
plt.scatter(penalties, f1_scores, color='blue')
plt.title('F1 Score as a Function of Penalty')
plt.xlabel('Penalty')
plt.ylabel('F1 Score')
plt.xscale('log')
```


F1 Score as a Function of Penalty
F1 Score
10^0
10^1
10^2
Penalty