Import Librarys

```
import pandas as pd # pandas use to manipulate the data
import matplotlib.pyplot as plt #use for visualization
import seaborn as sns ##use for advance visualization
```

About this file

Use the Titanic dataset to build a model that predicts whether a passenger on the Titanic survived or not. This is a classic beginner project with readily available dat

a. The dataset typically used for this project contains information about individual passengers, such as their age, gender, ticket class, fare, cabin, and whether or not they survived.

Data collection and processing

```
df = pd.read_csv("Titanic-Dataset.csv") #Load the dataset

df.head() #head use for starting 5 rows

   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
```

```
df.info() # info use for know the data type are present in our dataset
how many memory are use this dataset to store data and how many total
columns are present etc
```

```
<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
```

```
df.columns # columns use to know the names of the all columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age',
'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
df.shape # shape use for know about the total row and columns
```

```
(891, 12)
```

```
df.shape[0]*df.shape[1] #it show the total data present in the
datasets
```

```
10692
```

```
df.isnull().sum() # isnull use to know how many null values are
present in the dataset
```

```
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
```

Hendling missing data

```
# drop the "Cabin" column from the dataframe
df = df.drop(columns='Cabin', axis=1)

# replacing the missing values in "Age" column with mean value
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

```
C:\Users\nidhi kushwaha\AppData\Local\Temp\
ipykernel_2296\978008565.py:2: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


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

```
# finding the mode value of "Embarked" column
print(df['Embarked'].mode())
```

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

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

```
S
```

```
# replacing the missing values in "Embarked" column with mode value
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

```
C:\Users\nidhi kushwaha\AppData\Local\Temp\
ipykernel_2296\4224055363.py:2: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
```

```
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


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

```python
# check the number of missing values in each column
df.isnull().sum()
```

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

```python
# getting some statistical measures about the data
df.describe()
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   13.002015    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   22.000000    0.000000
50%     446.000000    0.000000    3.000000   29.699118    0.000000
75%     668.500000    1.000000    3.000000   35.000000    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
```

```
# value of count use for know the no of people Survived and not
Survived
df['Survived'].value_counts()
```

```
Survived
0    549
1    342
Name: count, dtype: int64
```

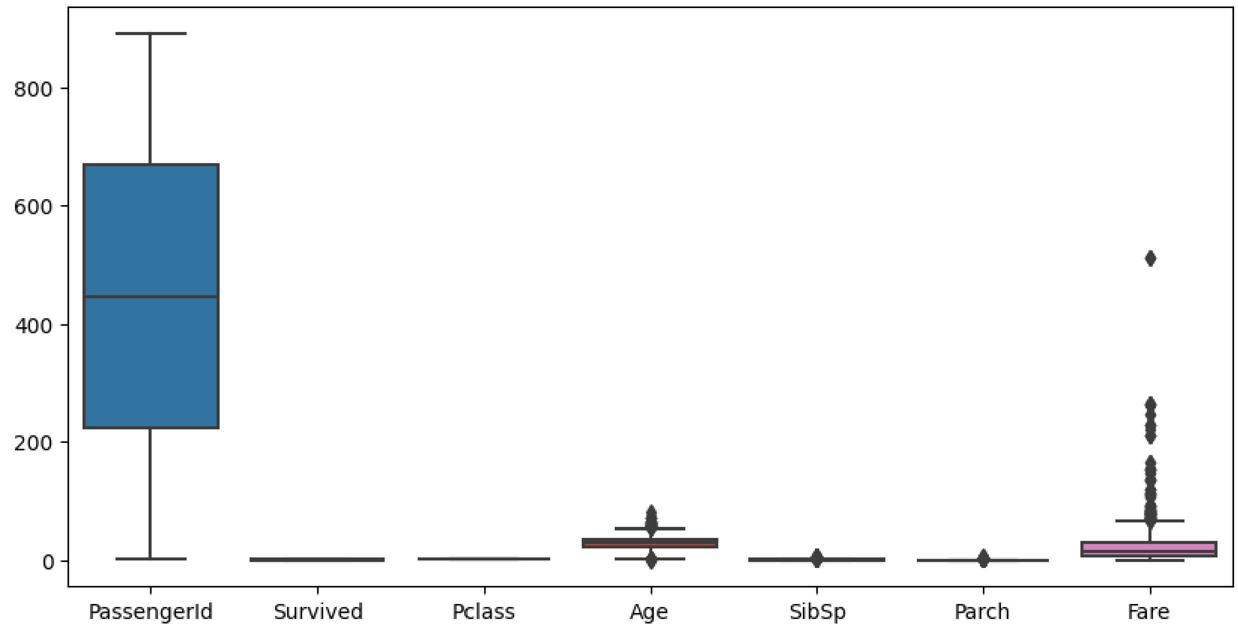
```
# value  count use for know the no of  Male and female
df['Sex'].value_counts()
```

```
Sex
male      577
female    314
Name: count, dtype: int64
```

Outlire

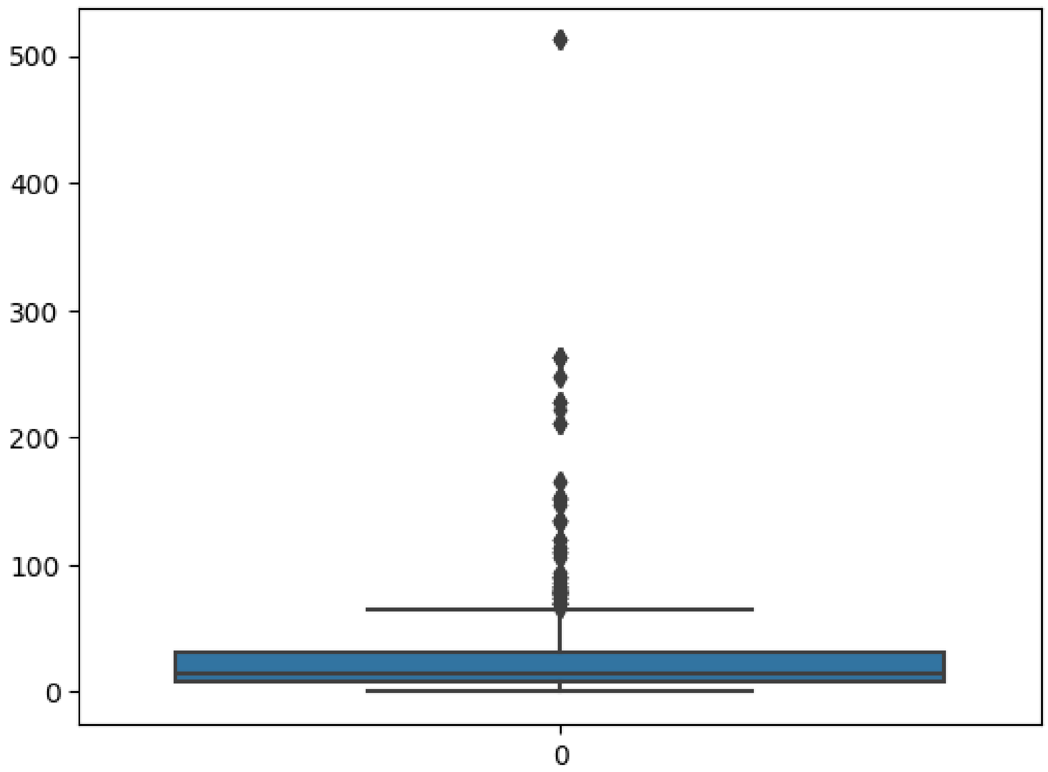
```
plt.figure(figsize=(10,5))
sns.boxplot(df)
```

```
<Axes: >
```



```
sns.boxplot(df["Fare"])
```

```
<Axes: >
```



Handling Outlires

```
# handling Outlire withe the help of Z Score
for i in df[["Age","SibSp","Parch","Fare"]]:
    Upper_Boundary = df[i].mean() + 3 * df[i].std()
    Lower_Boundary = df[i].mean() - 3 * df[i].std()
df =df[(df['Fare']<Upper_Boundary) & (df['Fare']>Lower_Boundary)]
```

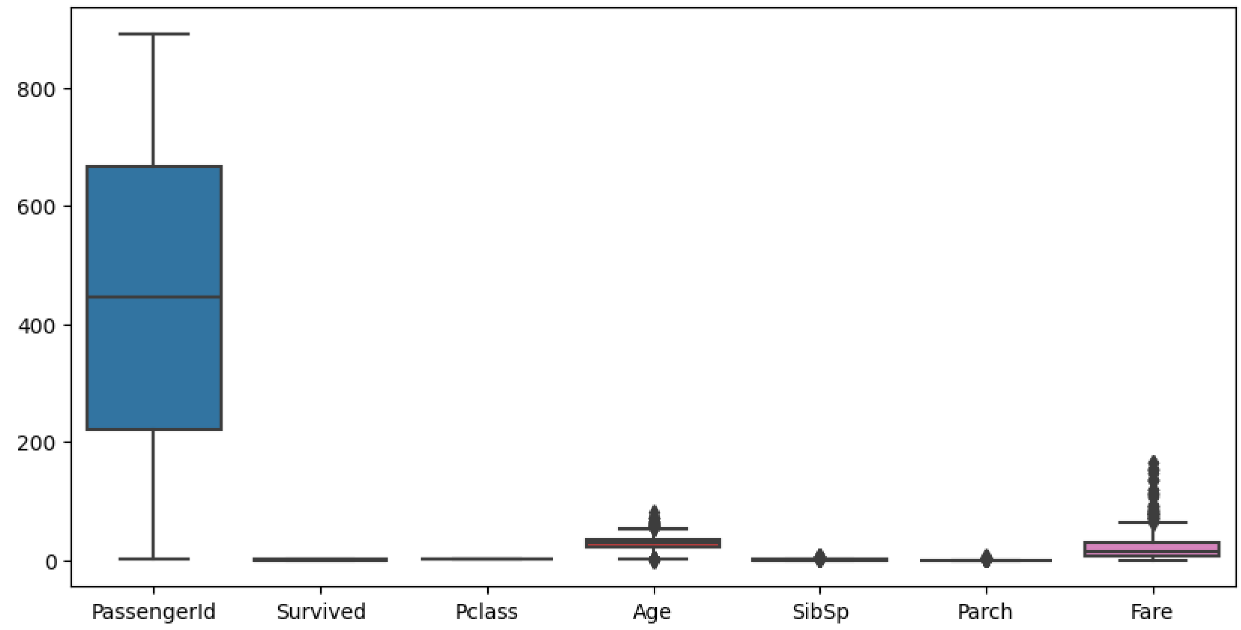
```
df.shape
```

```
(871, 11)
```

```
# we can see we have a 891 rows in dataset and after Handling outlire
we have 849 data
```

```
plt.figure(figsize=(10,5))
sns.boxplot(df)
```

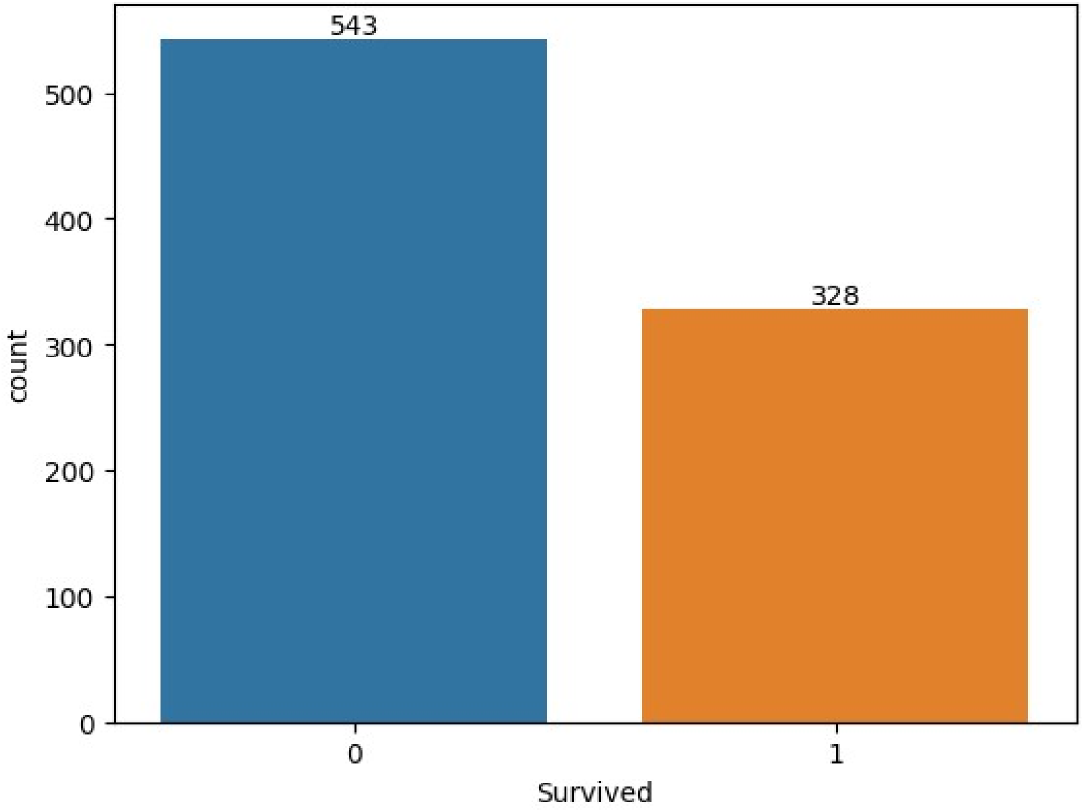
```
<Axes: >
```



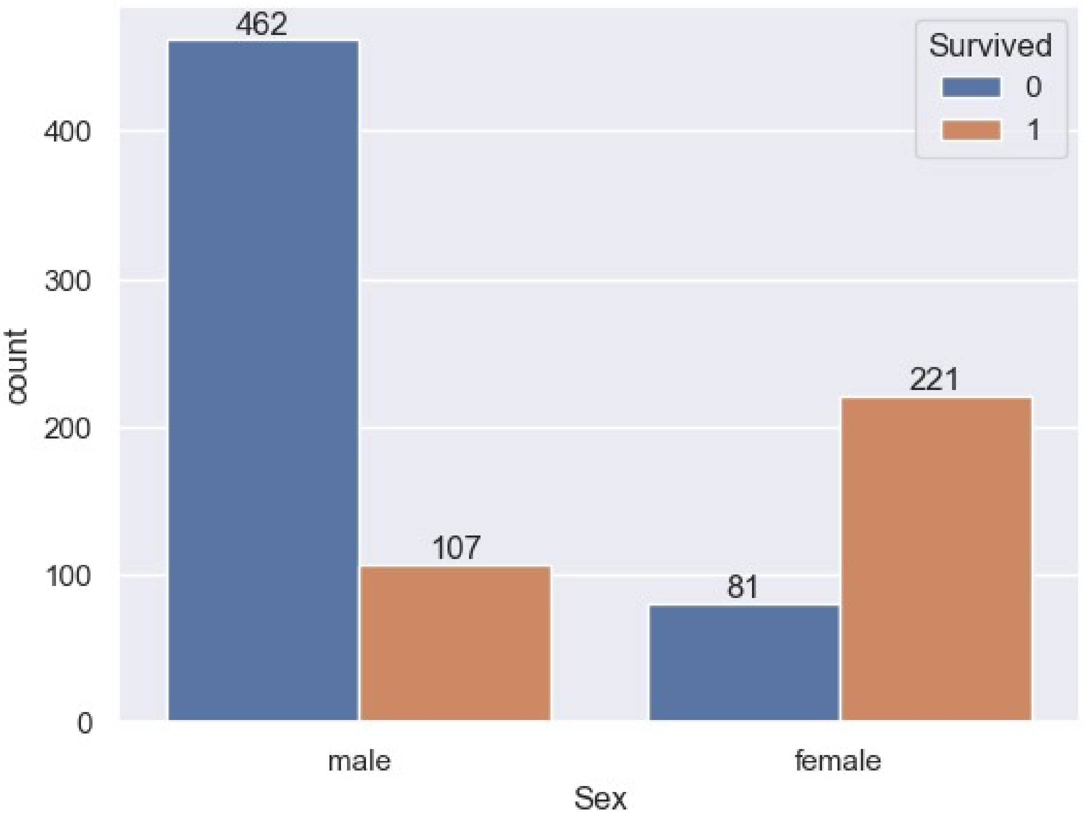
Data Visualization

```
#use a countplot to count the number for Survived peoples on non
survived by bar graph
ax = sns.countplot(x= "Survived",data = df)
ax.bar_label(ax.containers[0])
```

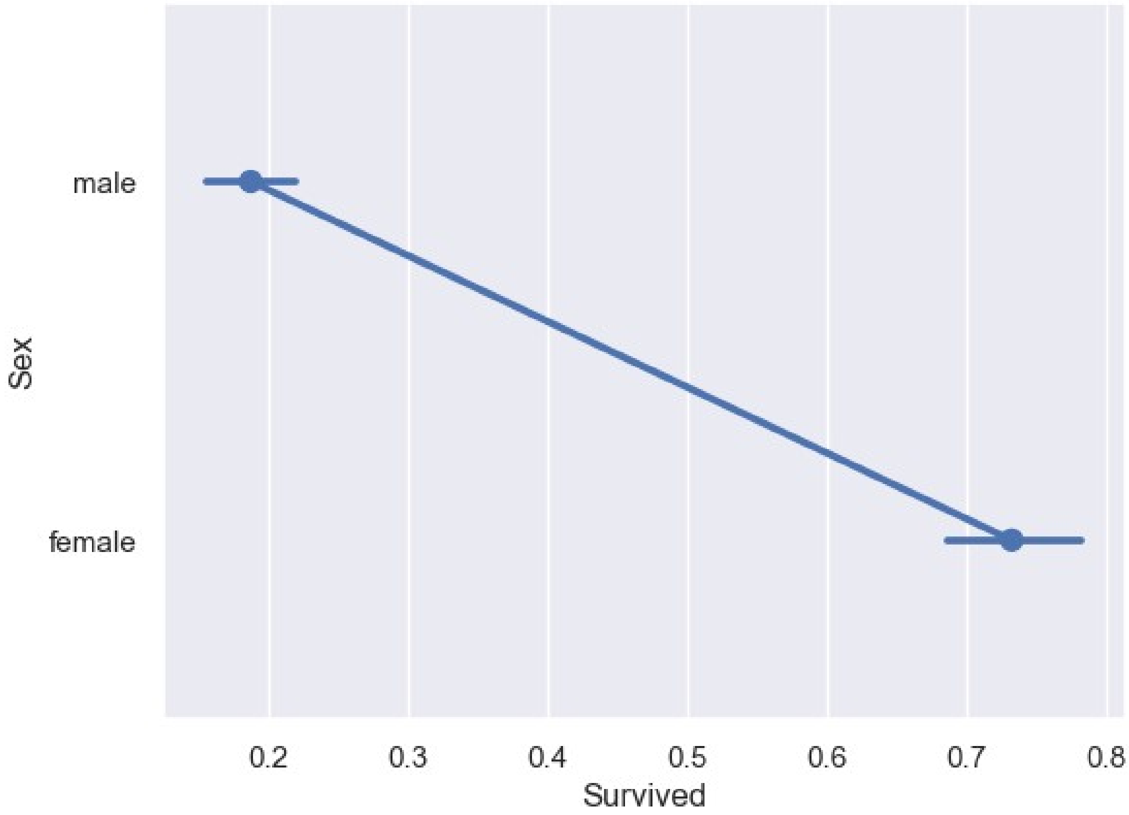
```
[Text(0, 0, '543'), Text(0, 0, '328')]
```



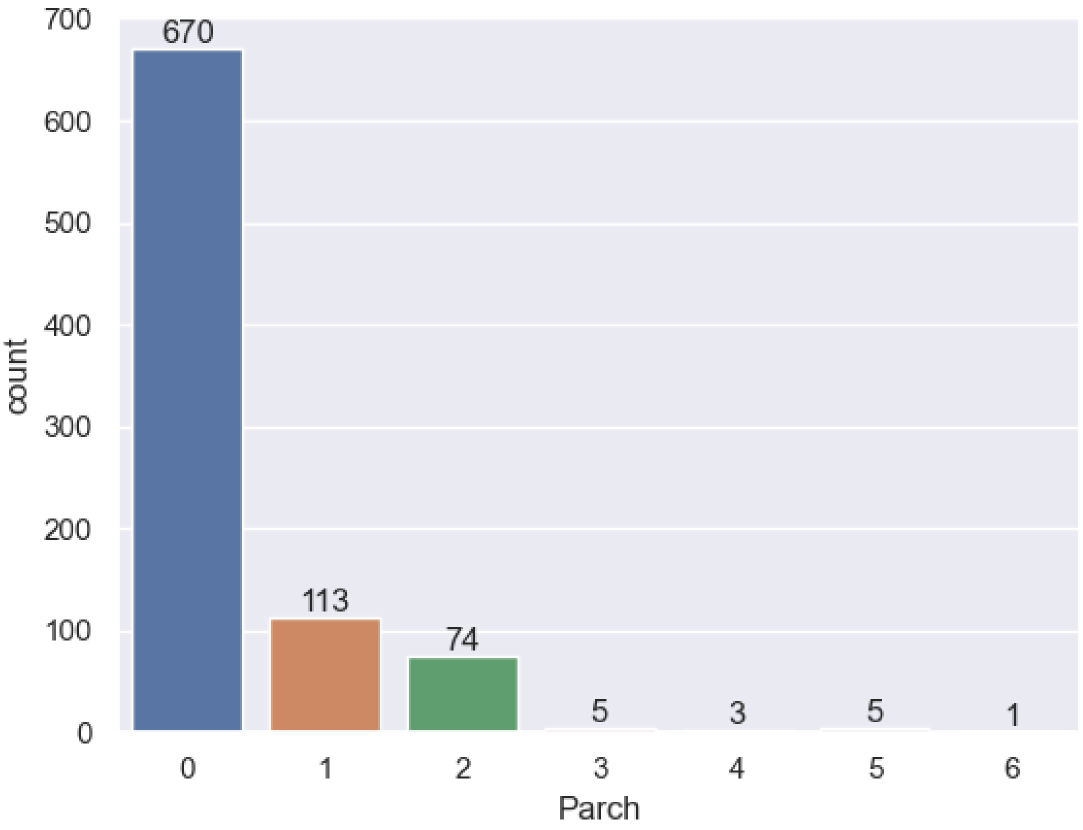
```
sns.set()

#use a countplot to count the number for Survived peoples on non
survived by by gender

ax = sns.countplot(data = df, x = 'Sex', hue = 'Survived')

for bars in ax.containers:
    ax.bar_label(bars)
```



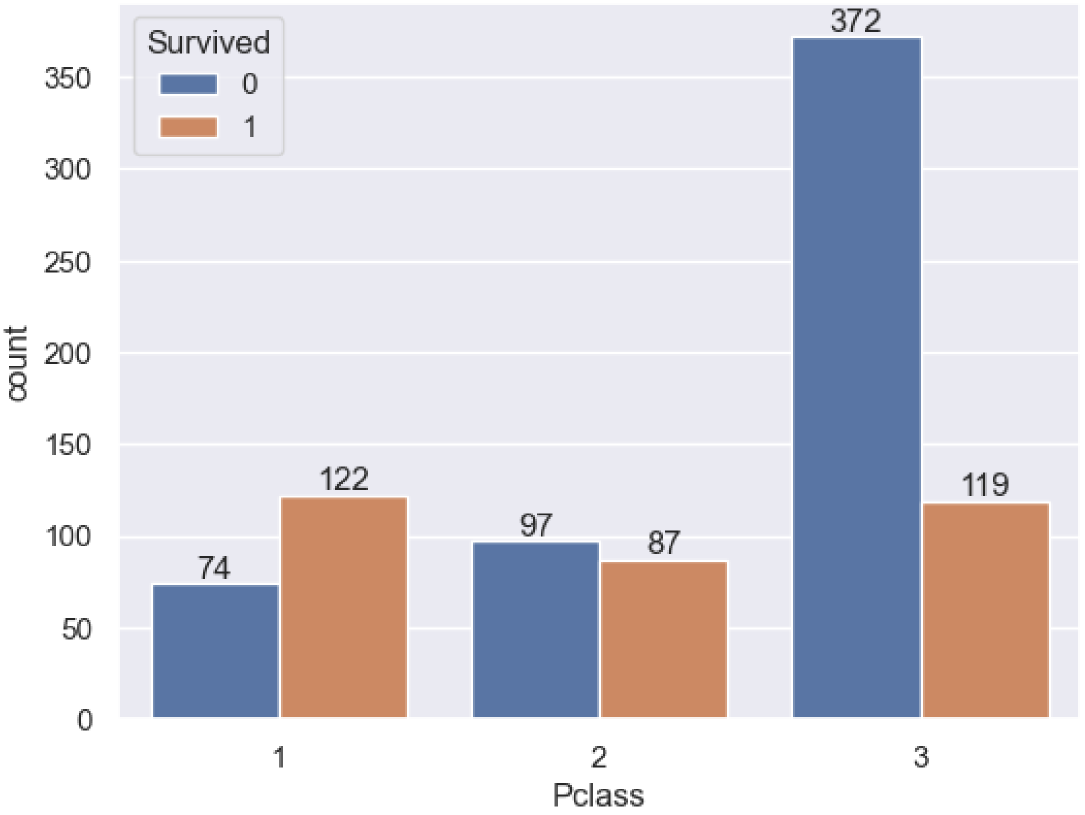
```
sns.pointplot(x = df.Survived,y = df.Sex)

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



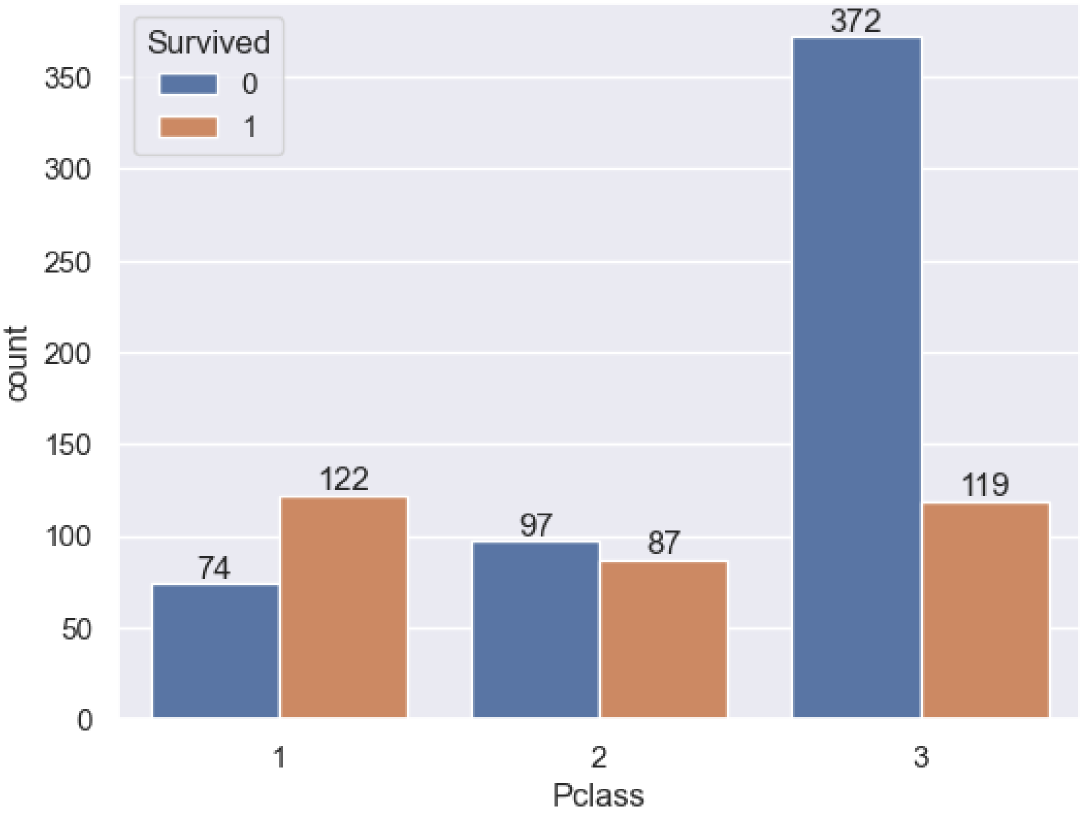
```
#use a countplot to count the number for parents/Children aboard the titanic
ax = sns.countplot(x= "Parch",data = df)
ax.bar_label(ax.containers[0])

[Text(0, 0, '670'),
 Text(0, 0, '113'),
 Text(0, 0, '74'),
 Text(0, 0, '5'),
 Text(0, 0, '3'),
 Text(0, 0, '5'),
 Text(0, 0, '1')]
```



```python
ax = sns.countplot(data = df, x = 'Pclass', hue = 'Survived')

for bars in ax.containers:
    ax.bar_label(bars)
```

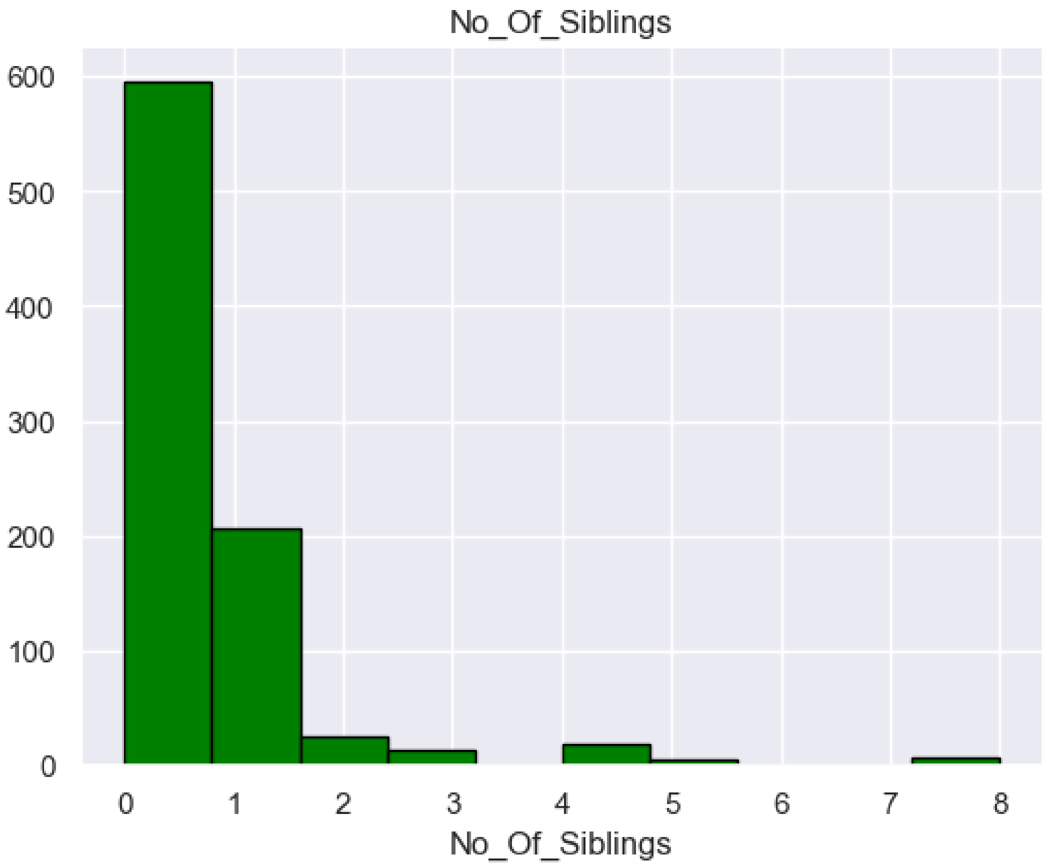


```python
ax = sns.countplot(data = df, x = 'Pclass', hue = 'Survived')

for bars in ax.containers:
    ax.bar_label(bars)
```



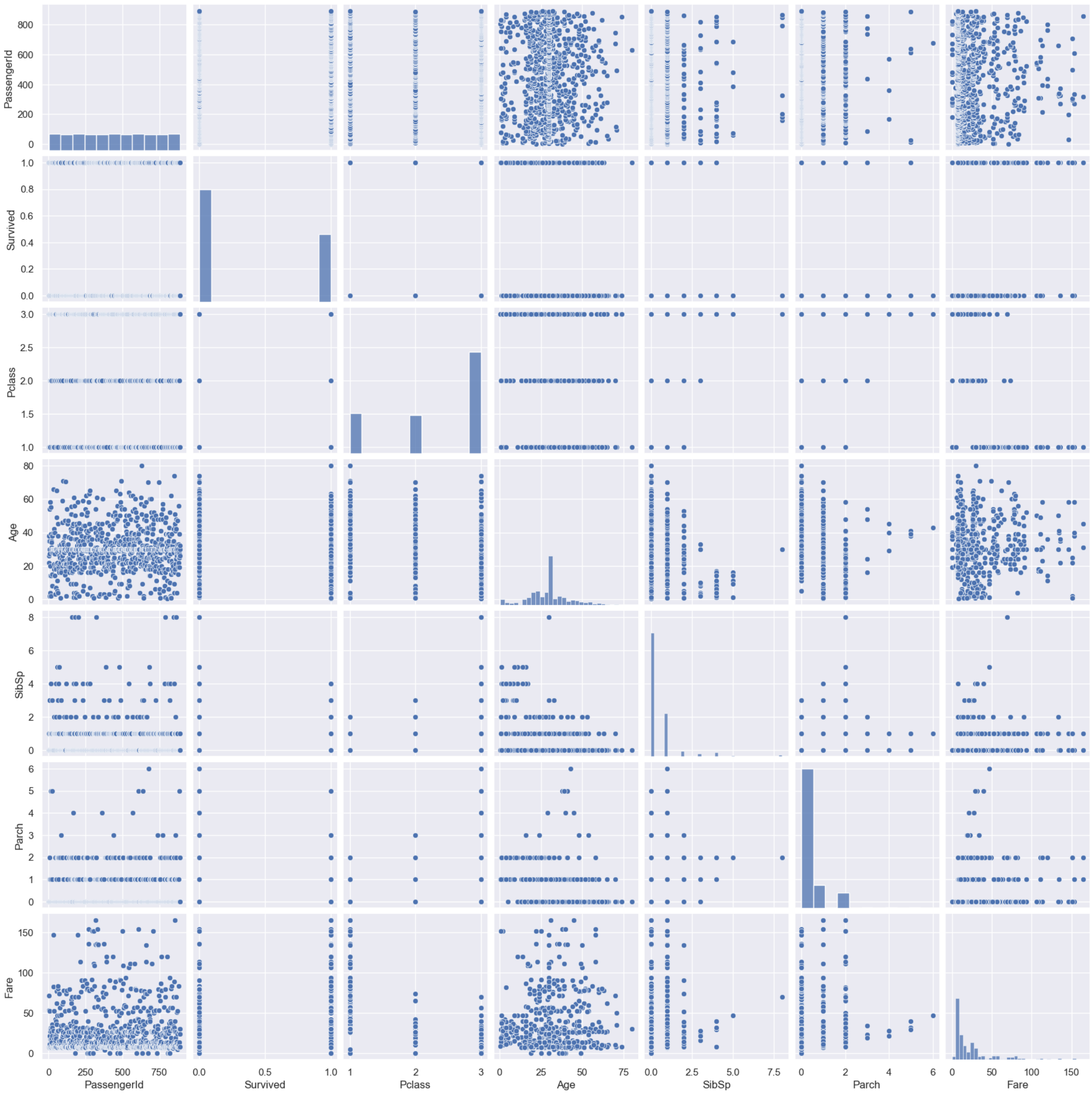
```
No_Of_Siblings = df['SibSp']
plt.hist(No_Of_Siblings, color='green', edgecolor='black')
plt.title('No_Of_Siblings')
plt.xlabel('No_Of_Siblings')
```

```
Text(0.5, 0, 'No_Of_Siblings')
```



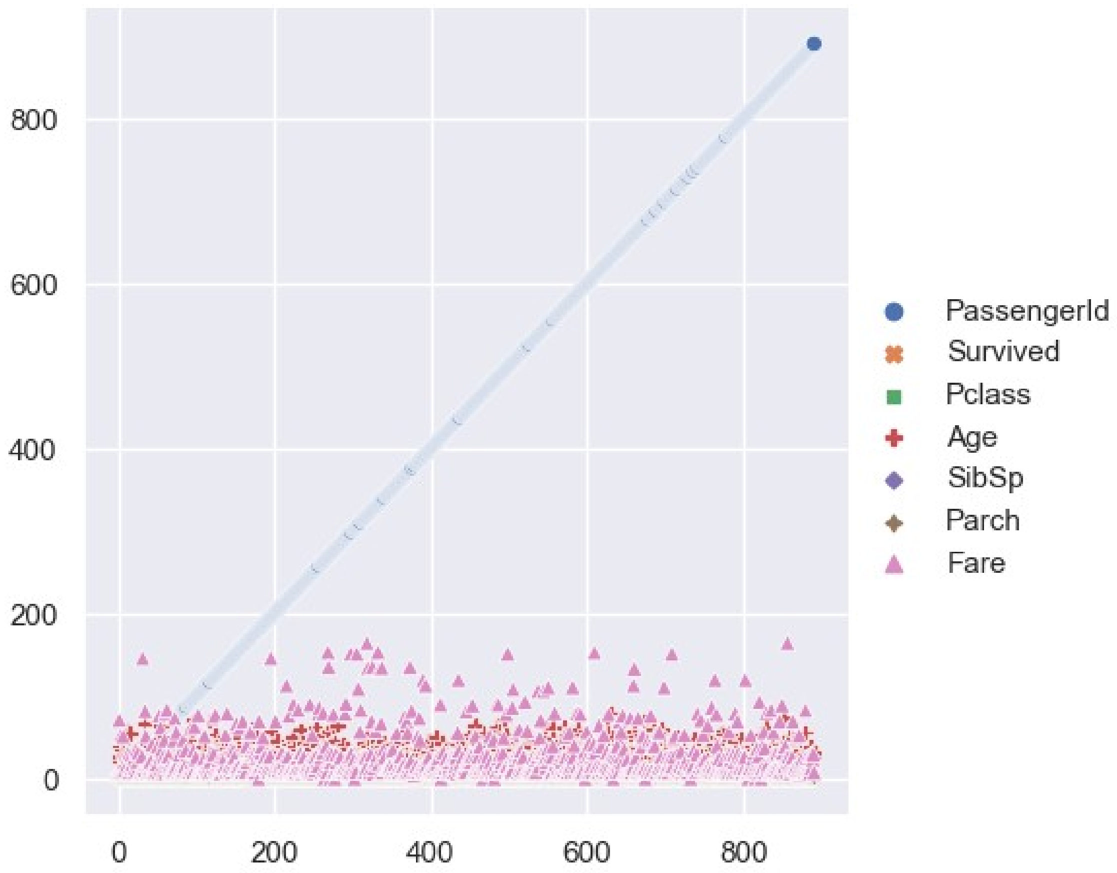
```
sns.pairplot(df)

C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
```

```
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):

<seaborn.axisgrid.PairGrid at 0x1fc65a3b940>
```



```
plt.figure(figsize =(15,5))
sns.relplot(df)
```

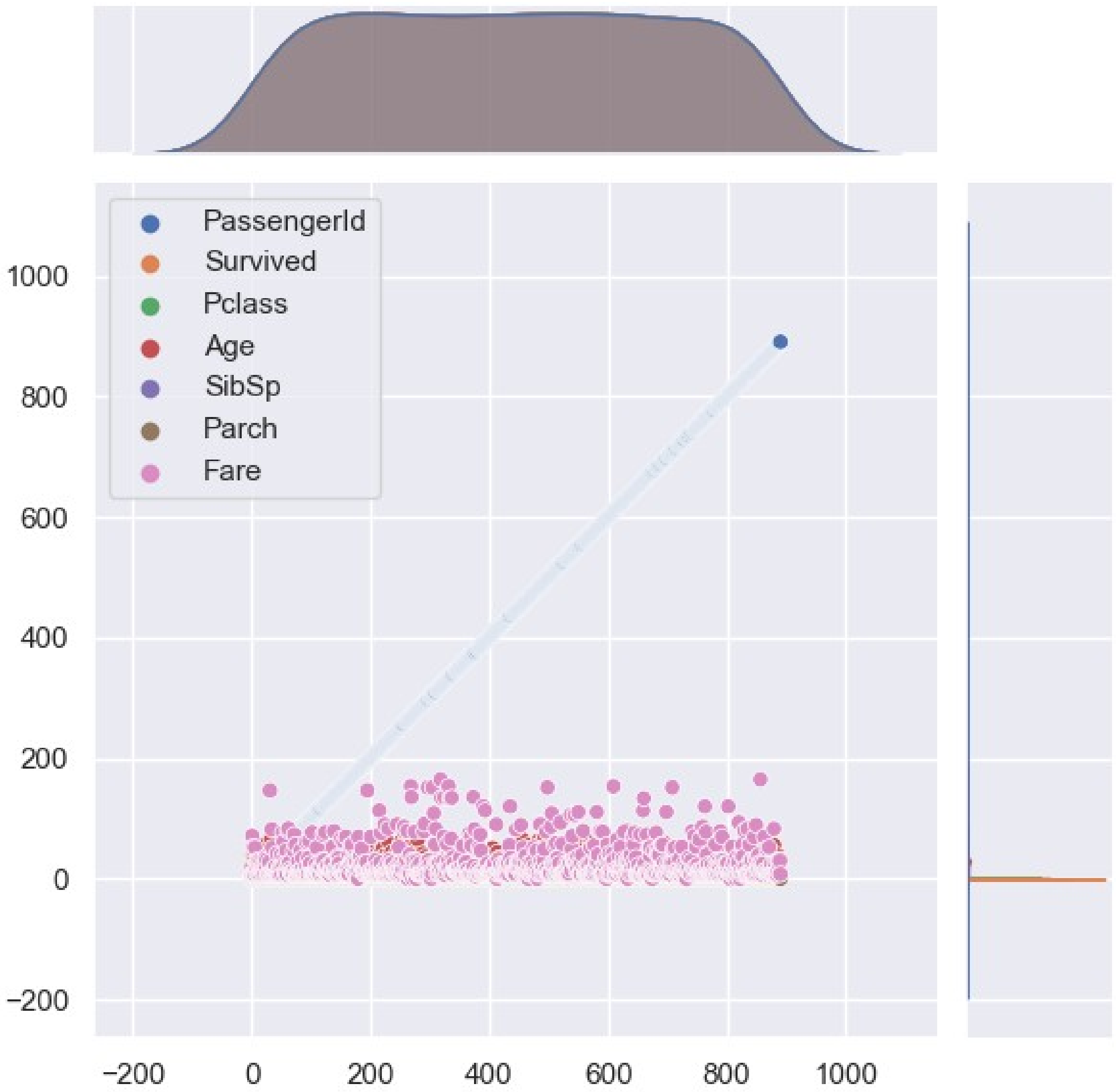
```
<seaborn.axisgrid.FacetGrid at 0x1fc69ba4ca0>

<Figure size 1500x500 with 0 Axes>
```



```
sns.jointplot(df)
```

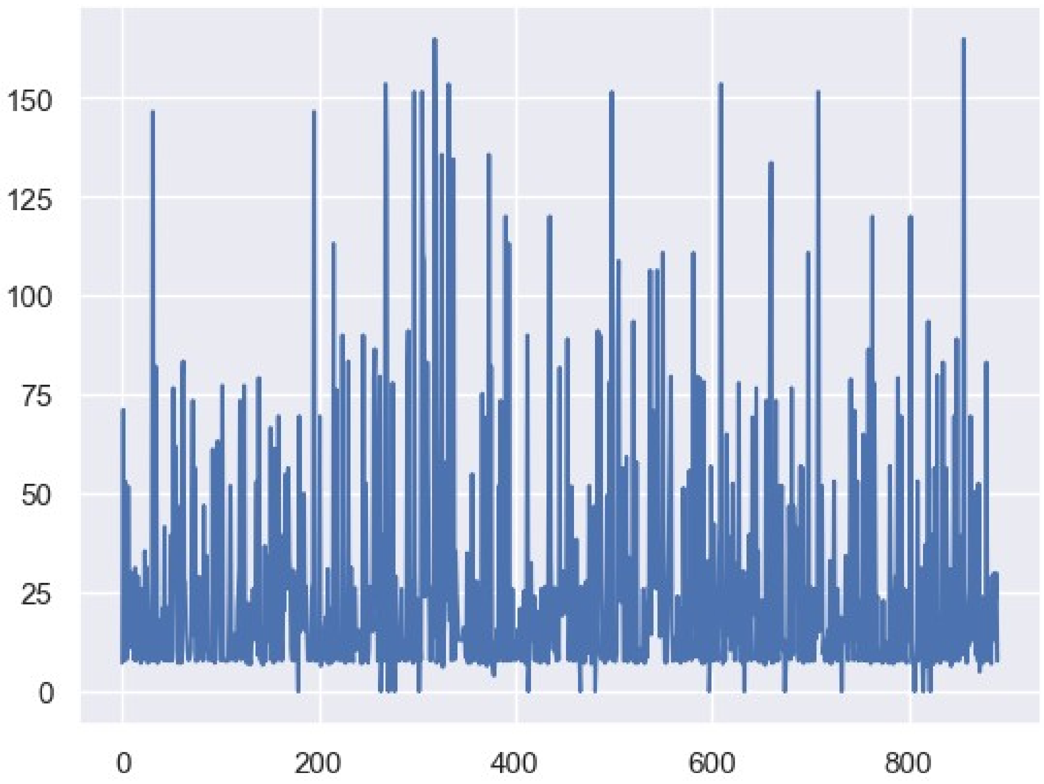
```
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1119:
```

```
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
C:\TURBOC3\python39\lib\site-packages\seaborn\_oldcore.py:1075:
FutureWarning: When grouping with a length-1 list-like, you will need
to pass a length-1 tuple to get_group in a future version of pandas.
Pass `(name,)` instead of `name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)

<seaborn.axisgrid.JointGrid at 0x1fc66b64580>
```


PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare
1000
800
600
400
200
0
−200
−200
0
200
400
600
800
1000

```
df['Fare'].plot()
```

```
<Axes: >
```



Encoding the Categorical Columns

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

```
Sex
male      569
female    302
Name: count, dtype: int64
```

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

```
Embarked
S    638
C    156
Q     77
Name: count, dtype: int64
```

```
# converting categorical Columns

df.replace({'Sex':{'male':0,'female':1}, 'Embarked':
{'S':0,'C':1,'Q':2}}, inplace=True)
```

```
C:\Users\nidhi kushwaha\AppData\Local\Temp\
ipykernel_2296\604960066.py:3: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To
```

```
retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior,
set `pd.set_option('future.no_silent_downcasting', True)`
  df.replace({'Sex':{'male':0,'female':1}, 'Embarked':
{'S':0,'C':1,'Q':2}}, inplace=True)

df.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  Embarked
0         A/5 21171   7.2500         0
1          PC 17599  71.2833         1
2  STON/O2. 3101282   7.9250         0
3            113803  53.1000         0
4            373450   8.0500         0
```

Separating features & Target

```
X = df.drop(columns =
['PassengerId','Name','Ticket','Survived'],axis=1)
Y = df['Survived']

print(X)

     Pclass  Sex        Age  SibSp  Parch     Fare  Embarked
0         3    0  22.000000      1      0   7.2500         0
1         1    1  38.000000      1      0  71.2833         1
2         3    1  26.000000      0      0   7.9250         0
3         1    1  35.000000      1      0  53.1000         0
4         3    0  35.000000      0      0   8.0500         0
..      ...  ...        ...    ...    ...      ...       ...
```

```
886       2    0  27.000000      0      0  13.0000         0
887       1    1  19.000000      0      0  30.0000         0
888       3    1  29.699118      1      2  23.4500         0
889       1    0  26.000000      0      0  30.0000         1
890       3    0  32.000000      0      0   7.7500         2

[871 rows x 7 columns]
```

```
print(Y)
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 871, dtype: int64
```

Splitting the data into training data & Test data

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,
test_size=0.2, random_state=2)

print(X.shape, X_train.shape, X_test.shape)
```

```
(871, 7) (696, 7) (175, 7)
```

Model Training

```
model = LogisticRegression()

# training the Logistic Regression model with training data
model.fit(X_train, Y_train)
```

```
LogisticRegression()
```

Model Evaluation

```
# accuracy on training data
X_train_prediction = model.predict(X_train)

print(X_train_prediction)
```

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

```
from sklearn.metrics import accuracy_score

training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print('Accuracy score of training data : ', training_data_accuracy)
```

```
Accuracy score of training data :  0.8117816091954023

# accuracy on test data
X_test_prediction = model.predict(X_test)

print(X_test_prediction)

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

test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print('Accuracy score of test data : ', test_data_accuracy)

Accuracy score of test data :  0.7942857142857143
```

prediction by RandomForest

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, Y_train)

RandomForestClassifier(random_state=42)
```

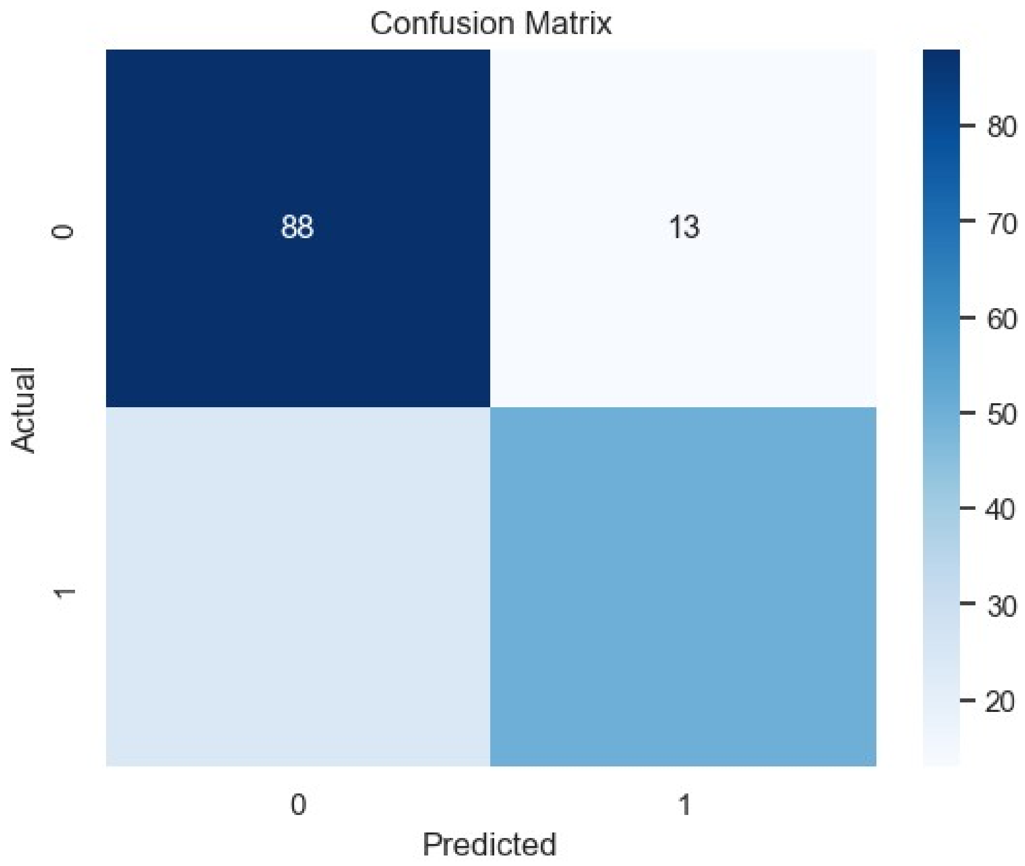
Make Predictions

```
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)
```

Evaluate the Model

```
# Accuracy
train_accuracy = accuracy_score(Y_train, y_train_pred)
test_accuracy = accuracy_score(Y_test, y_test_pred)

print(f"Training Accuracy: {train_accuracy:.2f}")
print(f"Testing Accuracy: {test_accuracy:.2f}")

# Confusion Matrix
conf_matrix = confusion_matrix(Y_test, y_test_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Classification Report
print(classification_report(Y_test, y_test_pred))
```

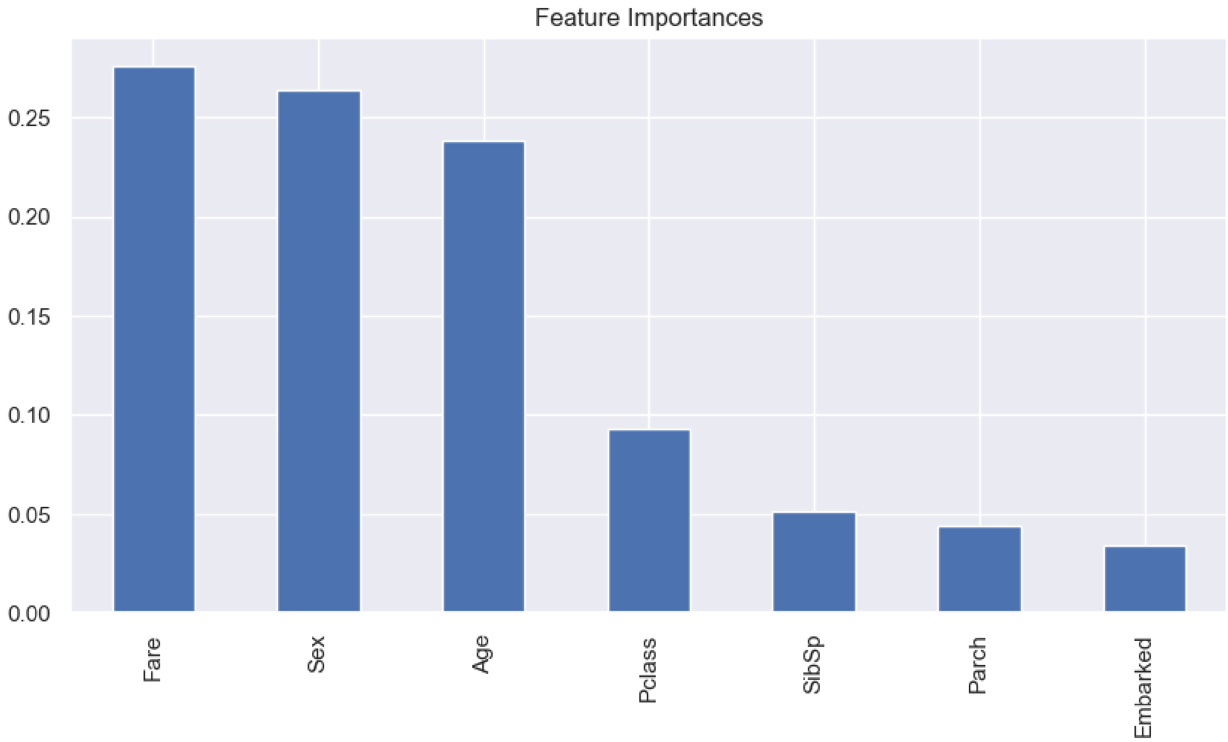
```
Training Accuracy: 0.98
Testing Accuracy: 0.79
```



```
              precision    recall  f1-score   support

           0       0.79      0.87      0.83       101
           1       0.79      0.68      0.73        74

    accuracy                           0.79       175
   macro avg       0.79      0.77      0.78       175
weighted avg       0.79      0.79      0.79       175
```

Visualize Feature Importance

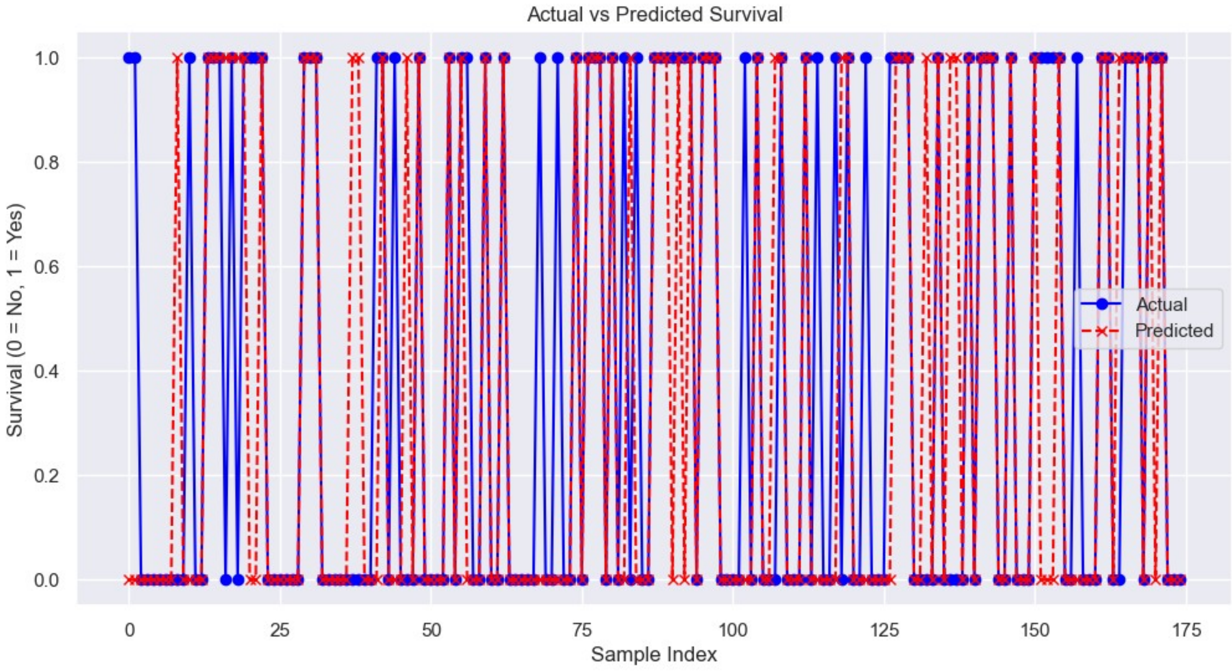
```
feature_importances = pd.Series(model.feature_importances_,
index=X.columns)
feature_importances.sort_values(ascending=False).plot(kind='bar',
figsize=(10, 5))
plt.title("Feature Importances")
plt.show()
```



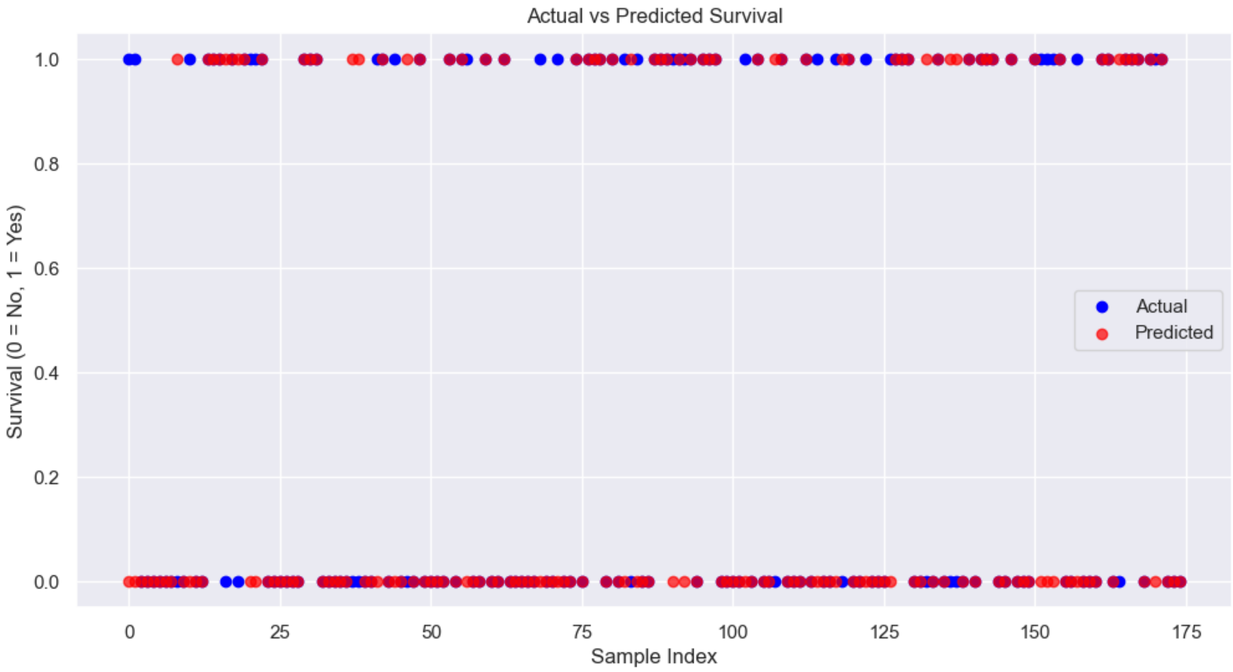
```
y_test_pred = model.predict(X_test)
```

Prepare the Data for Visualization

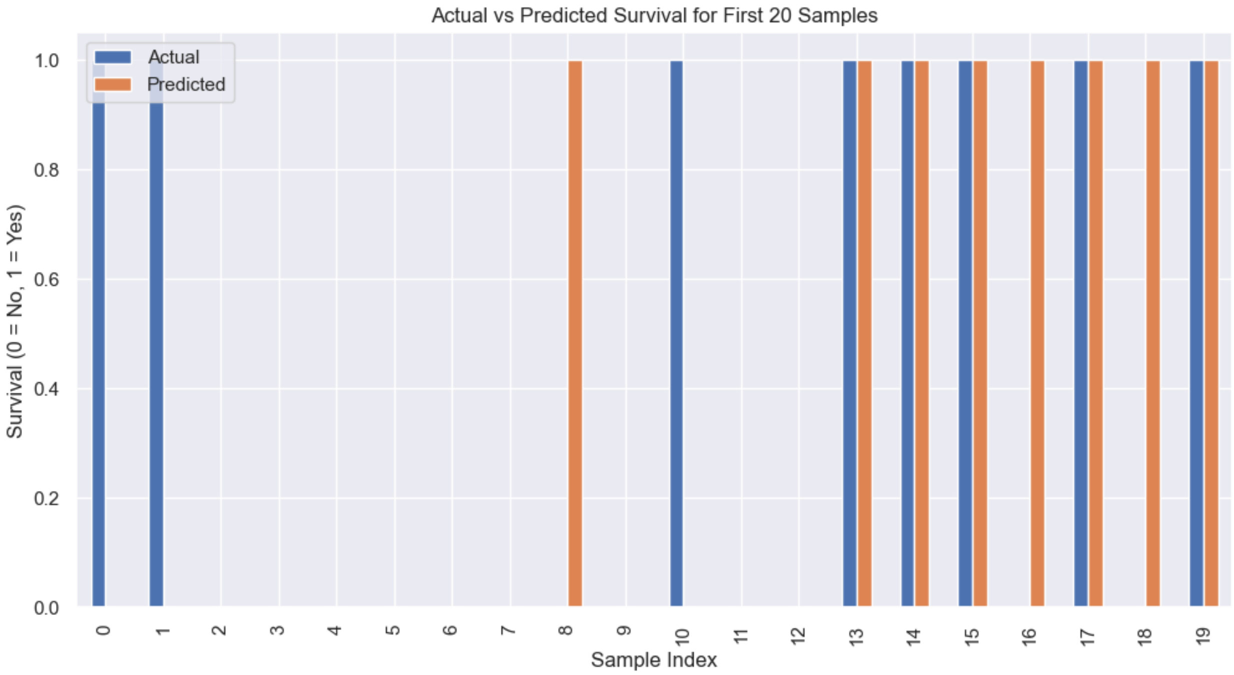
```
import pandas as pd

# Create a DataFrame
results = pd.DataFrame({
    'Actual': Y_test.values,
    'Predicted': y_test_pred
})

# Reset the index for plotting
results.reset_index(drop=True, inplace=True)

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(results['Actual'], label='Actual', color='blue', marker='o')
plt.plot(results['Predicted'], label='Predicted', color='red',
linestyle='--', marker='x')
plt.title('Actual vs Predicted Survival')
plt.xlabel('Sample Index')
plt.ylabel('Survival (0 = No, 1 = Yes)')
plt.legend()
plt.grid(True)
plt.show()
```



```
plt.figure(figsize=(12, 6))
plt.scatter(range(len(results)), results['Actual'], color='blue',
label='Actual')
plt.scatter(range(len(results)), results['Predicted'], color='red',
label='Predicted', alpha=0.7)
plt.title('Actual vs Predicted Survival')
plt.xlabel('Sample Index')
plt.ylabel('Survival (0 = No, 1 = Yes)')
plt.legend()
plt.show()
```



```
results.head(20).plot(kind='bar', figsize=(12, 6))
plt.title('Actual vs Predicted Survival for First 20 Samples')
plt.xlabel('Sample Index')
plt.ylabel('Survival (0 = No, 1 = Yes)')
plt.legend(['Actual', 'Predicted'])
plt.show()
```


Actual vs Predicted Survival for First 20 Samples
Actual
Predicted
Survival (0 = No, 1 = Yes)
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Sample Index