Import libraries

```
!pip install catboost
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-pacl
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/d
```

```
import pandas as pd
from sklearn.metrics import accuracy_score
from catboost import CatBoostClassifier
import numpy as np
from catboost import Pool
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
```

Exploratory Analysis

```
df = pd.read_csv('train.csv', index_col=0)
```

The first 10 rows of the dataset to understand its structure

```
print(df.head(10))
```

```
             Survived  Pclass  \
PassengerId
1                   0       3
2                   1       1
```

```
3                   1       3
4                   1       1
5                   0       3
6                   0       3
7                   0       1
8                   0       3
9                   1       3
10                  1       2

                                                       Name     Sex   Age
PassengerId
1                                   Braund, Mr. Owen Harris    male  22.0
2            Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
3                                    Heikkinen, Miss. Laina  female  26.0
4                 Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
5                                  Allen, Mr. William Henry    male  35.0
6                                          Moran, Mr. James    male   NaN
7                                   McCarthy, Mr. Timothy J    male  54.0
8                            Palsson, Master. Gosta Leonard    male   2.0
9            Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0
10                       Nasser, Mrs. Nicholas (Adele Achem)  female  14.0

             SibSp  Parch            Ticket     Fare Cabin Embarked
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S
2                1      0          PC 17599  71.2833   C85        C
3                0      0  STON/O2. 3101282   7.9250   NaN        S
4                1      0            113803  53.1000  C123        S
5                0      0            373450   8.0500   NaN        S
6                0      0            330877   8.4583   NaN        Q
7                0      0             17463  51.8625   E46        S
8                3      1            349909  21.0750   NaN        S
9                0      2            347742  11.1333   NaN        S
10               1      0            237736  30.0708   NaN        C
```

Stats for numerical variables

```
print(df.describe())
```

```
         Survived      Pclass         Age       SibSp       Parch        Fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

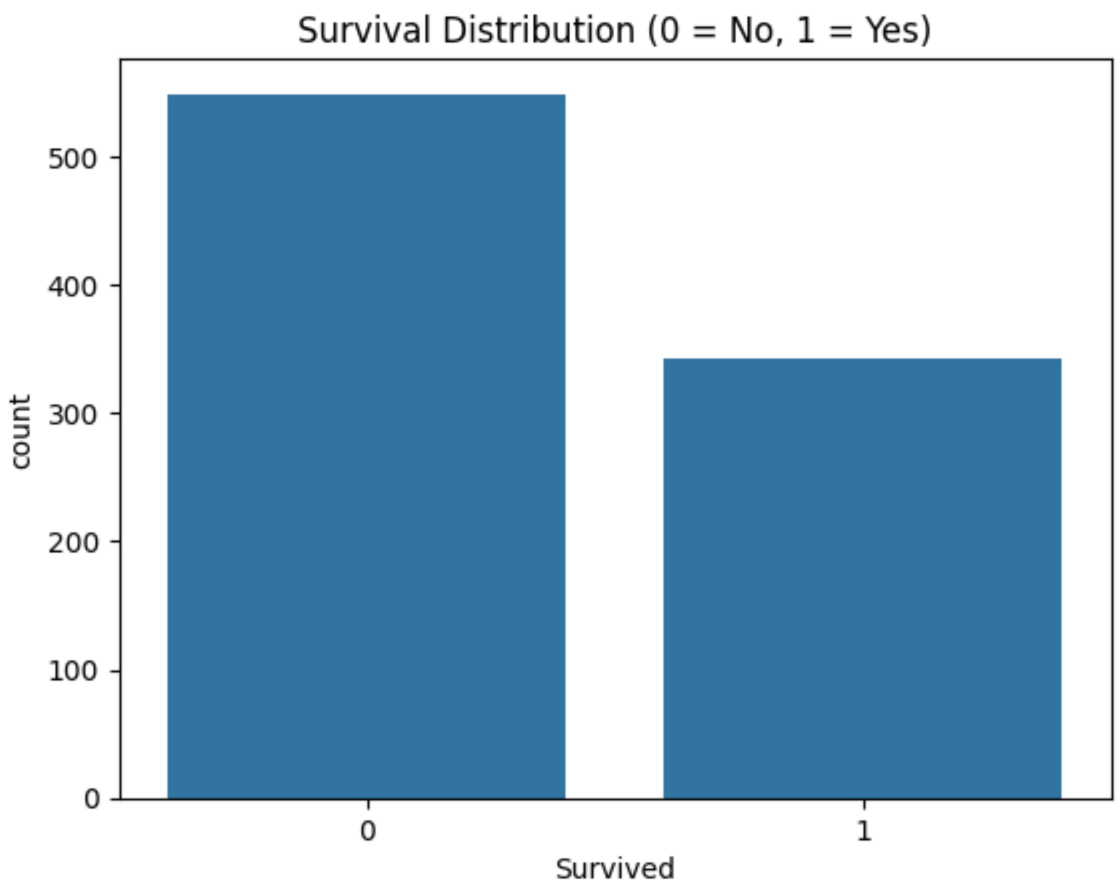
Missing values in each column

```
print(df.isnull().sum())
```

```
Survived      0
Pclass        0
Name          0
Sex           0
Age         177
SibSp         0
Parch         0
Ticket        0
Fare          0
Cabin       687
Embarked      2
dtype: int64
```

Distribution of the variable: 'Survived'

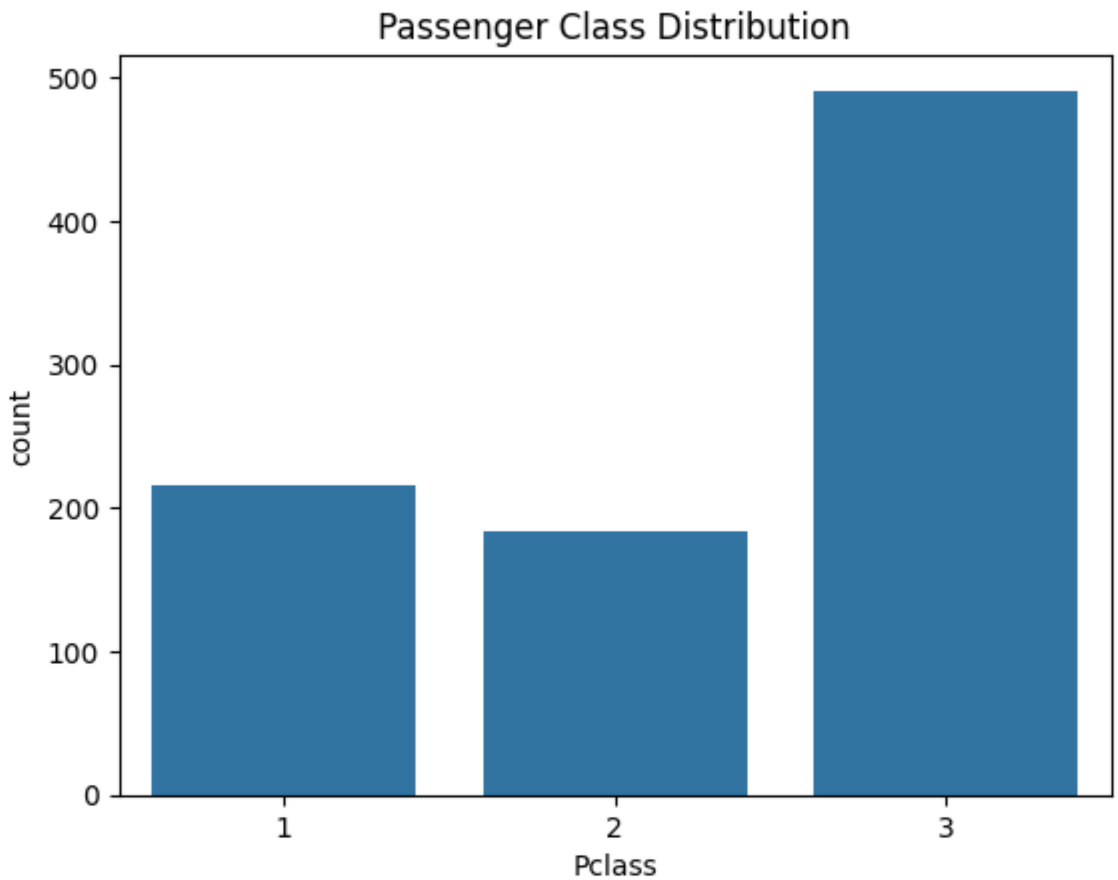
```
sns.countplot(x='Survived', data=df)
plt.title('Survival Distribution (0 = No, 1 = Yes)')
plt.show()
```



Distribution of varieble: 'Pclass'

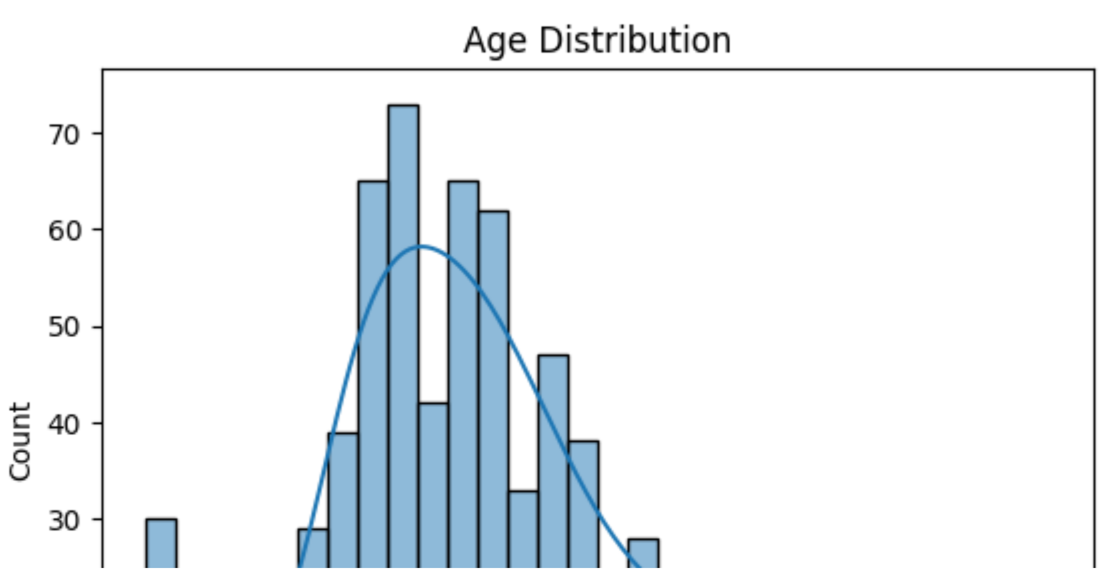
```
sns.countplot(x='Pclass', data=df)
```

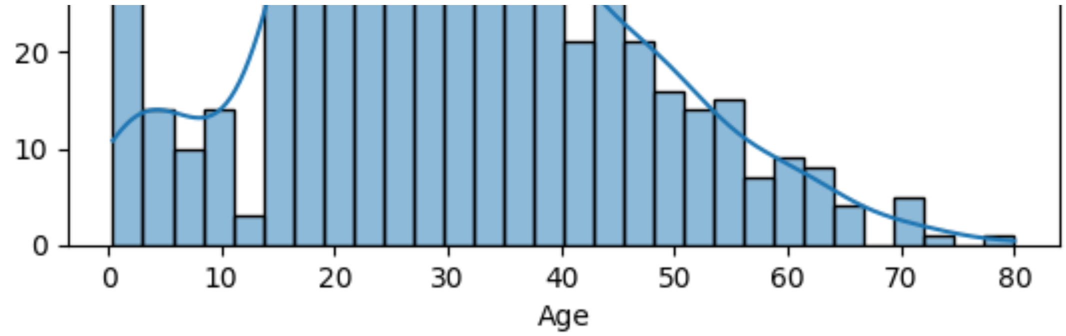
```
plt.title('Passenger Class Distribution')
plt.show()
```



Age distribution

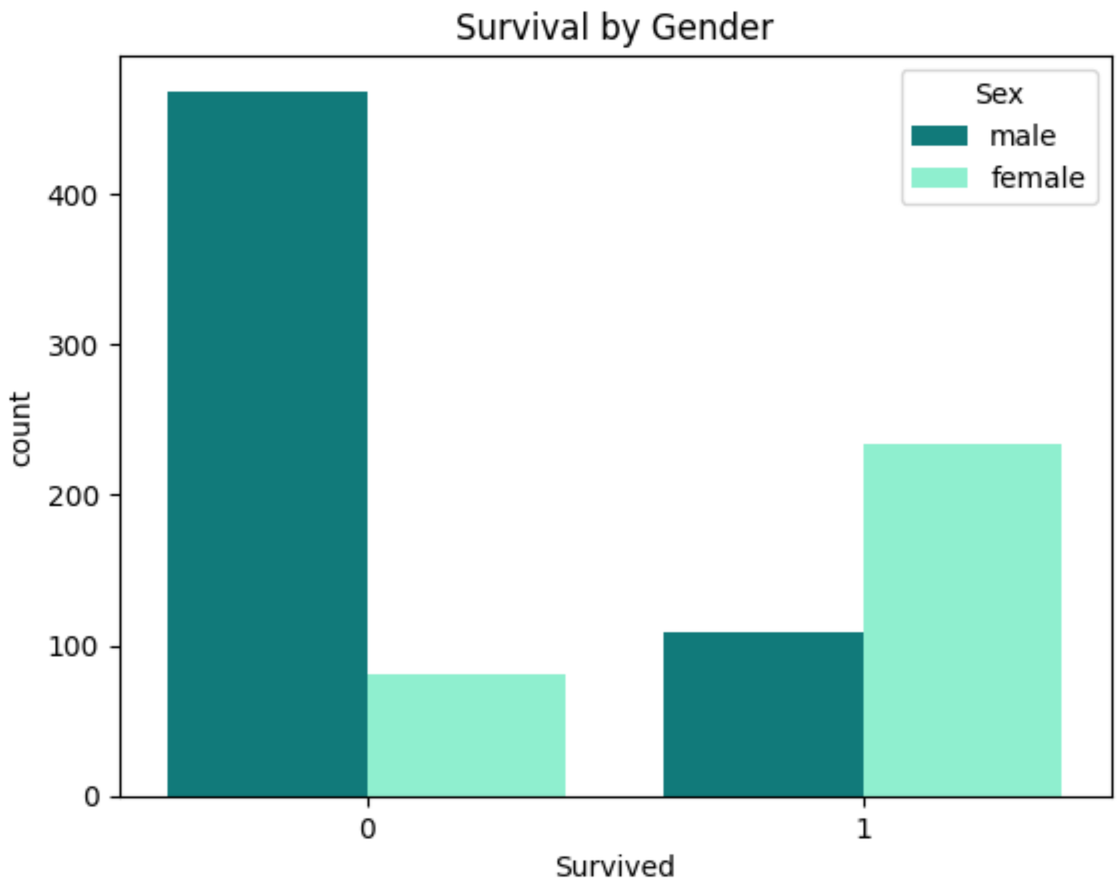
```
sns.histplot(df['Age'].dropna(), kde=True, bins=30)
plt.title('Age Distribution')
plt.show()
```





Survival comparison by gender

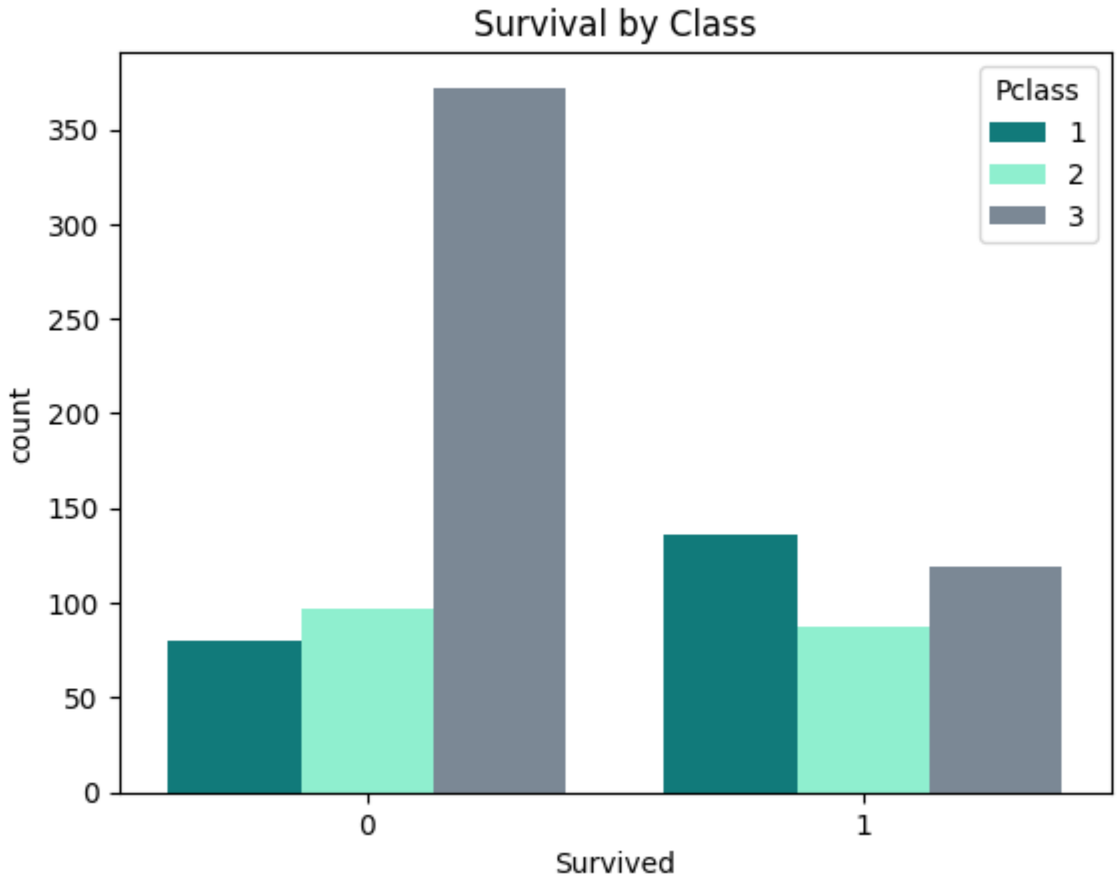
```
custom_palette = ['darkcyan', 'aquamarine']
sns.countplot(x='Survived', hue='Sex', data=df, palette=custom_palette)
plt.title('Survival by Gender')
plt.show()
```



Survival comparison by class

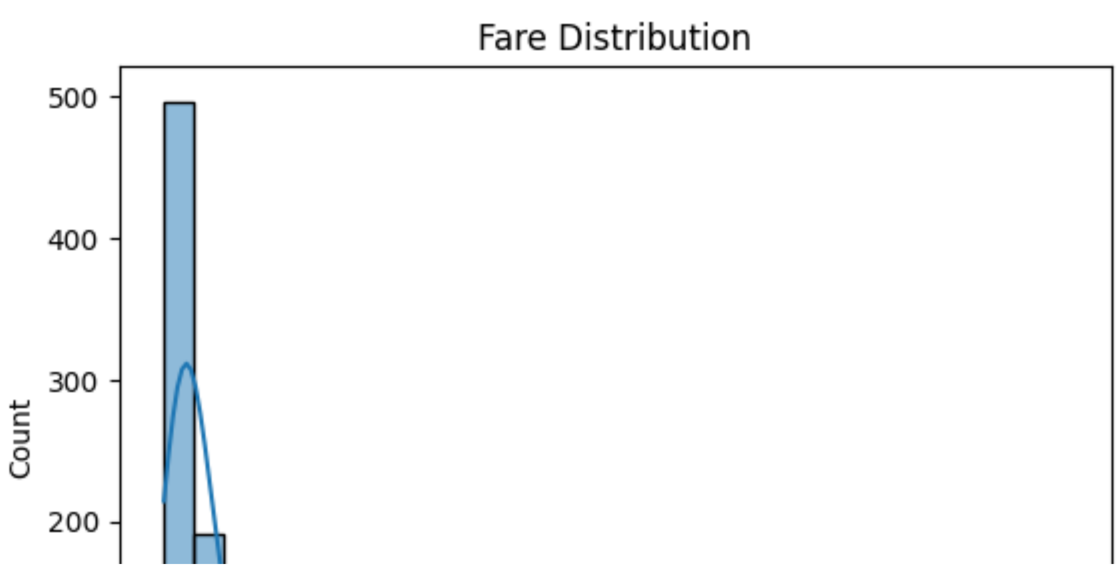
```
custom_palette = ['darkcyan', 'aquamarine', 'lightslategrey']
sns.countplot(x='Survived', hue='Pclass', data=df, palette=custom_palette)
```

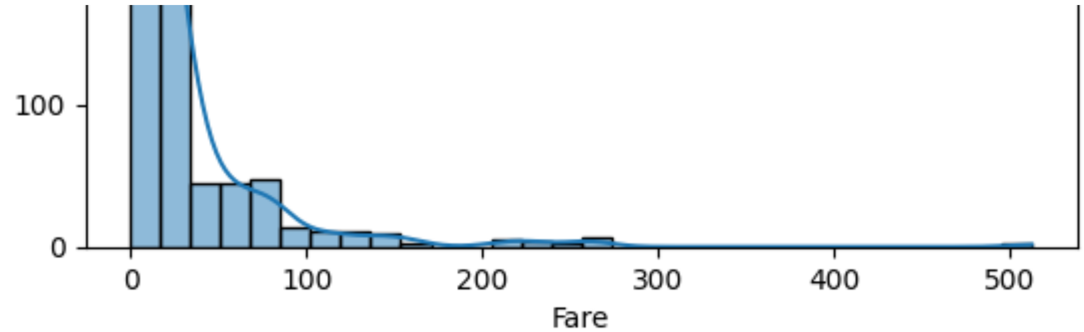
```
sns.countplot(x='Survived', hue='Pclass', data=df, palette=custom_palette)
plt.title('Survival by Class')
plt.show()
```



Analysis of the fare paid

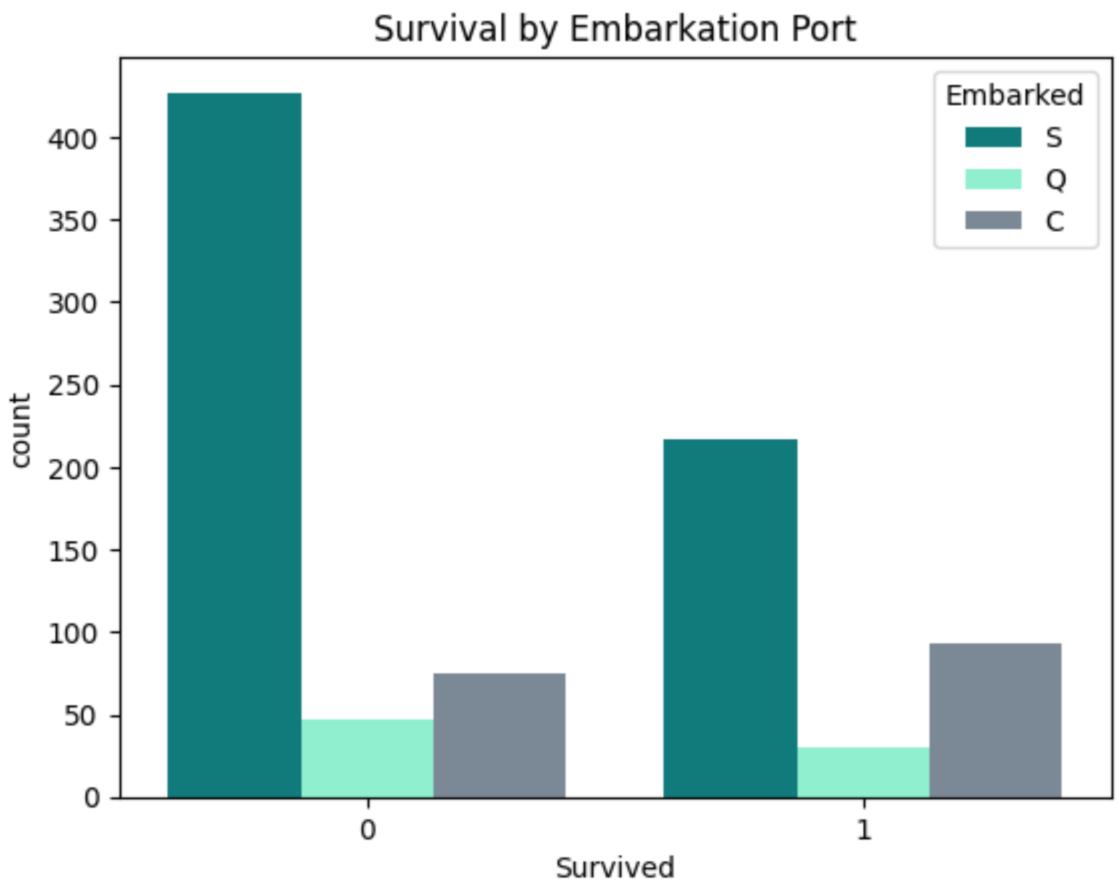
```
sns.histplot(df['Fare'], kde=True, bins=30)
plt.title('Fare Distribution')
plt.show()
```





Survival comparison by embarkation port

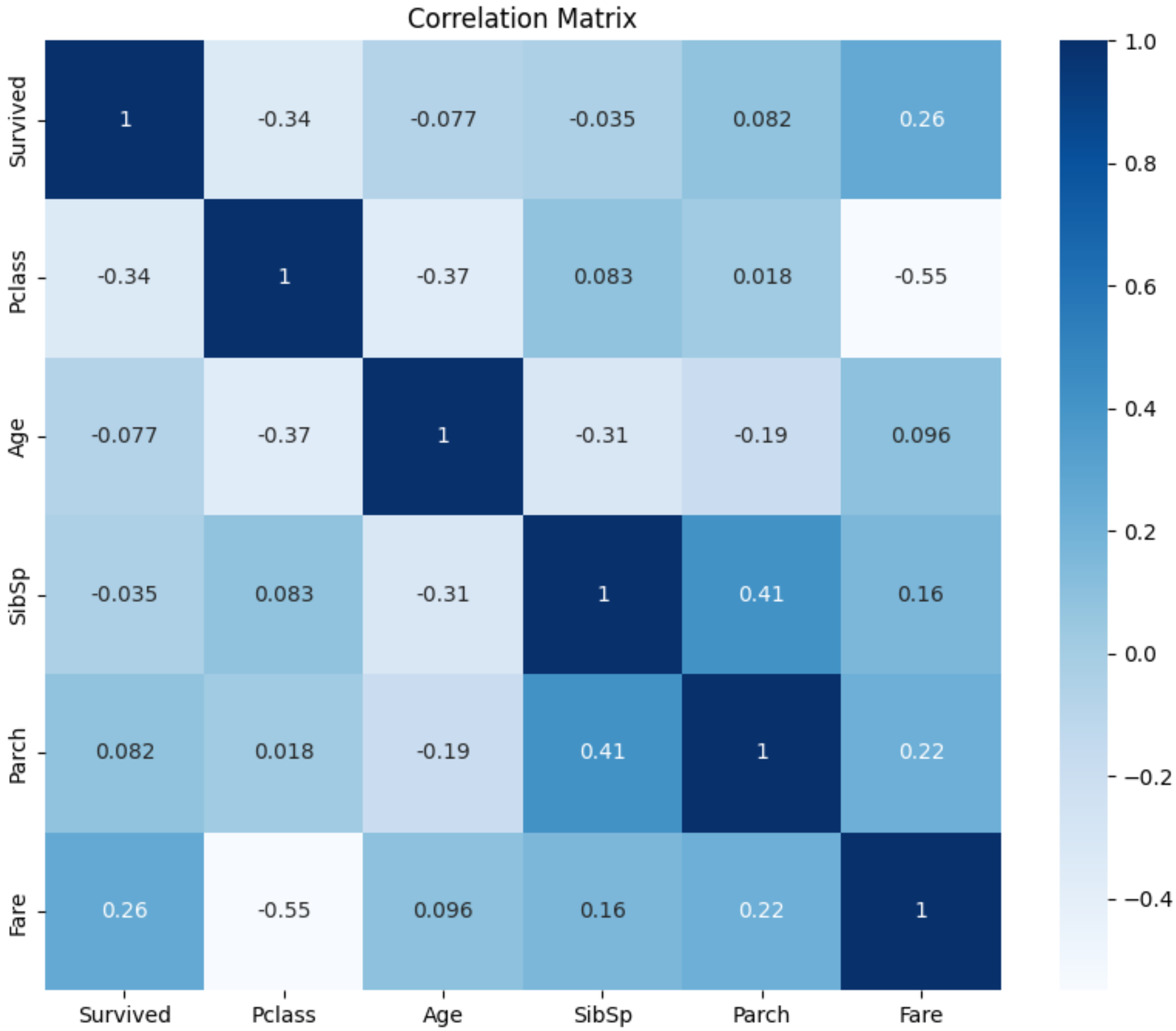
```
custom_palette = ['darkcyan', 'aquamarine', 'lightslategrey']
sns.countplot(x='Survived', hue='Embarked', data=df, palette=custom_palette)
plt.title('Survival by Embarkation Port')
plt.show()
```



Correlation matrix between numerical variables

```
plt.figure(figsize=(10, 8))
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
```

```
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap='Blues')
plt.title('Correlation Matrix')
plt.show()
```



Cleaning the data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 891 entries, 1 to 891
Data columns (total 11 columns):
```

```
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Name      891 non-null    object 
 3   Sex       891 non-null    object 
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64  
 6   Parch     891 non-null    int64  
 7   Ticket    891 non-null    object 
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object 
 10  Embarked  889 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

Get the amount of null values in percentage

```
df.isnull().sum() / len(df) * 100
```

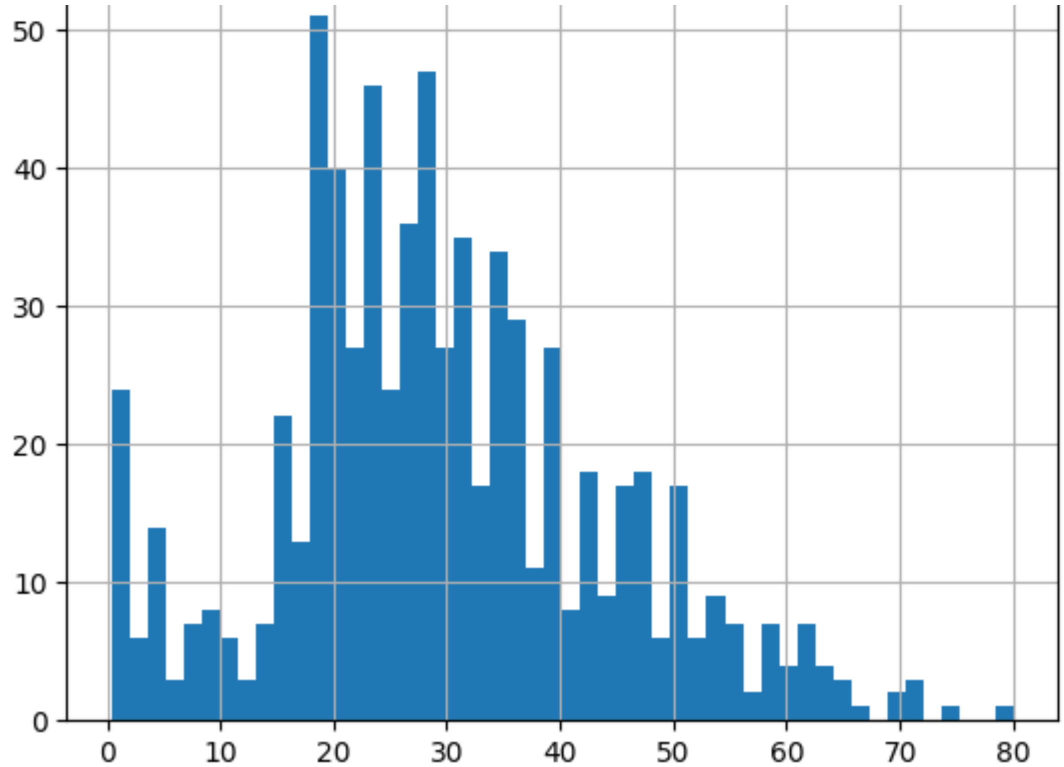
	0
Survived	0.000000
Pclass	0.000000
Name	0.000000
Sex	0.000000
Age	19.865320
SibSp	0.000000
Parch	0.000000
Ticket	0.000000
Fare	0.000000
Cabin	77.104377
Embarked	0.224467

dtype: float64

Look at the distributions of the features to decide with wich method we will fill the null values

```
df["Age"].hist(bins=50)
```

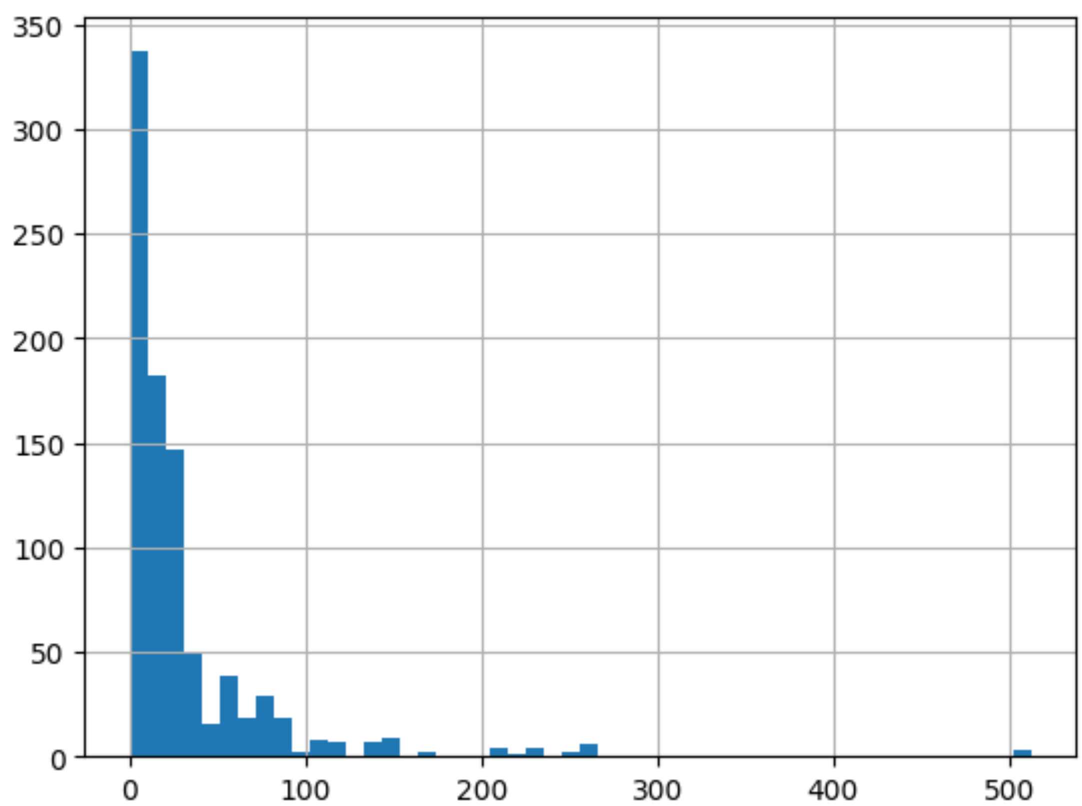
<Axes: >



We will use the median to fill the null values in Age

```
df["Fare"].hist(bins=50)
```

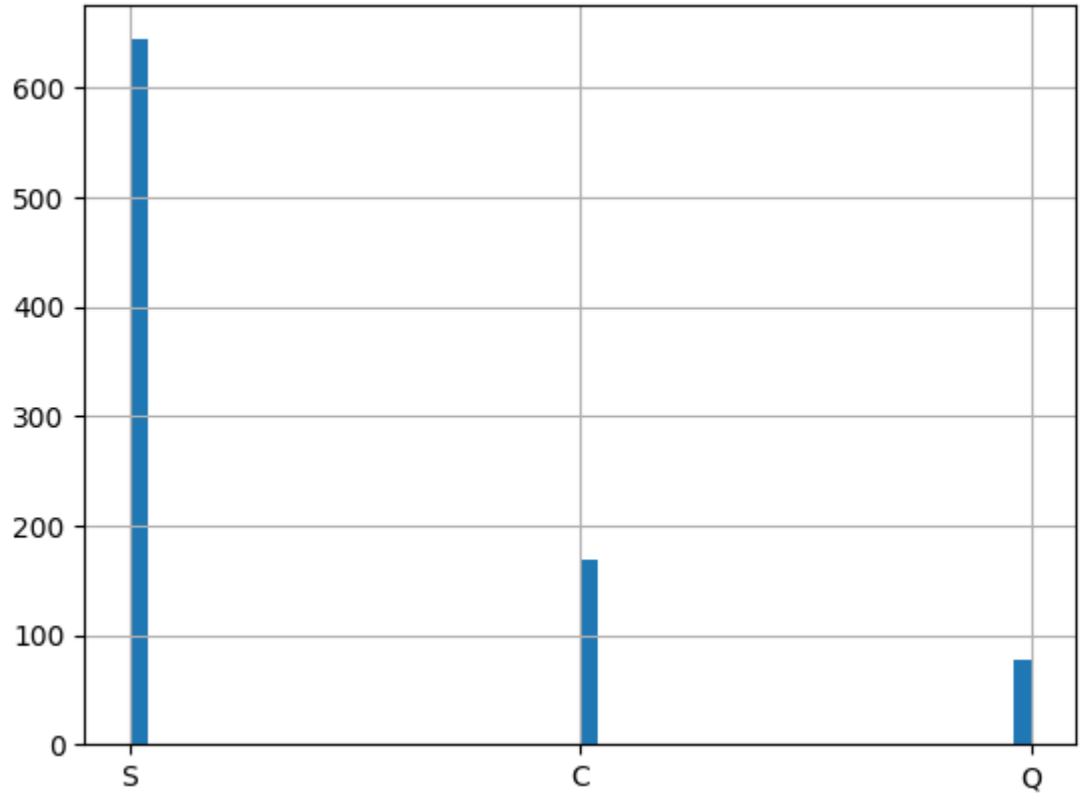
```
<Axes: >
```



We will use the median to fill the null values in Fare

```
df["Embarked"].hist(bins=50)
```

```
<Axes: >
```



We will use mode to fill the null values in Embarked

Sabemos que hay un 77.1% de datos faltantes en la columna de 'cabin', intentamos rellenarlo usando un modelo de knn pero no tuvimos un accuracy bueno, despues decidimos probar con random forest, y despues de varios intentos más decidimos eliminar la columna de 'cabin' ya que no obtuvimos los resultados esperados para rellenar esta columna y no aportaba algo bueno a nuestros resultados por lo que lo mejor es mejor no usar la columna.

Feature engineering

We are going to do the analysis based on the idea that women and children were given priority to board the lifeboats.

We will deduce the passengers gender based on the title in their names.

```
# Extract titles from the 'Name' column
df['Title'] = df['Name'].apply(lambda x: x.split(",")[1].split(".")[0].strip())
df.value_counts("Title")
```

	count
Title	
Mr	517
Miss	182
Mrs	125
Master	40
Dr	7
Rev	6
Major	2
Col	2
Mlle	2
Sir	1
Ms	1
Capt	1
Mme	1
Lady	1
Jonkheer	1
Don	1
the Countess	1

dtype: int64

Replace titles with general categories

```
df['Title'] = df['Title'].replace({
    "Capt": "man", "Don": "man", "Major": "man", "Col": "man",
    "Rev": "man", "Dr": "man", "Sir": "man", "Mr": "man", "Jonkheer": "man",
    "Dona": "woman", "the Countess": "woman", "Mme": "woman",
    "Mlle": "woman", "Ms": "woman", "Miss": "woman", "Lady": "woman", "Mrs": "wom
```

```
    "Title": "woman", "Ms": "woman", "Miss": "woman", "Lady": "woman", "Mrs": "wom
    "Master": "boy"
})
df.value_counts("Title")
```

	count
Title	
man	538
woman	313
boy	40

dtype: int64

We will group people by families

```
# Extract surnames from the 'Name' column
df['Surname'] = df['Name'].apply(lambda x: x.split(",")[0])

# Group 'man' titles under 'noGroup'
df.loc[df['Title'] == 'man', 'Surname'] = 'noGroup'

# Calculate the frequency of surnames
df['SurnameFreq'] = df.groupby('Surname')['Surname'].transform('count')

# Group surnames that appear only once under 'noGroup'
df.loc[df['SurnameFreq'] <= 1, 'Surname'] = 'noGroup'

df['SurnameSurvival'] = df.groupby('Surname')['Survived'].transform('mean')
```

The SurnameSurvival feature is calculated by grouping the data by Surname and taking the mean of the Survived column for each group. This gives the overall survival rate for each surname.

```
df['AdjustedSurvival'] = (df['SurnameSurvival'] * df['SurnameFreq'] - df['Survive
```

AdjustedSurvival adjusts the SurnameSurvival rate to provide a better estimate of the survival rate for each passenger, taking into account the number of people with the same surname (SurnameFreq) and the actual survival of the individual passenger (Survived).

This feature is intended to provide a more reliable estimate of the survival probability for each passenger based on their surname.

Data Preprocessing Function

```
def preprocess_data(data):

    # Extract titles from the 'Name' column
    data['Title'] = data['Name'].apply(lambda x: x.split(",")[1].split(".")[0].st

    # Replace titles with general categories
    data['Title'] = data['Title'].replace({
        "Capt": "man", "Don": "man", "Major": "man", "Col": "man",
        "Rev": "man", "Dr": "man", "Sir": "man", "Mr": "man", "Jonkheer": "man",
        "Dona": "woman", "the Countess": "woman", "Mme": "woman",
        "Mlle": "woman", "Ms": "woman", "Miss": "woman", "Lady": "woman", "Mrs":
        "Master": "boy"
    })

    # Extract surnames from the 'Name' column
    data['Surname'] = data['Name'].apply(lambda x: x.split(",")[0])

    # Group 'man' titles under 'noGroup'
    data.loc[data['Title'] == 'man', 'Surname'] = 'noGroup'

    # Calculate the frequency of surnames
    data['SurnameFreq'] = data.groupby('Surname')['Surname'].transform('count')

    # Group surnames that appear only once under 'noGroup'
    data.loc[data['SurnameFreq'] <= 1, 'Surname'] = 'noGroup'

    # Calculate the survival rates for 'woman-child-groups'
    data['SurnameSurvival'] = data.groupby('Surname')['Survived'].transform('mean

    # Adjust survival rates for use on the training set
    data['AdjustedSurvival'] = (data['SurnameSurvival'] * data['SurnameFreq'] - d

    # if the adjust survival rate is -inf or inf
    data['AdjustedSurvival'] = data['AdjustedSurvival'].replace(np.inf, 1) # this
    data['AdjustedSurvival'] = data['AdjustedSurvival'].replace(-np.inf, 0) # thi

    data.drop(['Name', 'Ticket', 'Cabin', 'Surname', 'SurnameFreq', 'Title'], axi

    data['Age'].fillna(data['Age'].median(), inplace=True)
    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
    data['Fare'].fillna(data['Fare'].median(), inplace=True)
    data['Sex'] = data['Sex'].map({'female': 1, 'male': 0})
    data = pd.get_dummies(data, columns=['Embarked'])

    return data
```

Prepare the Data

```
train = pd.read_csv("train.csv", index_col=0)
train = preprocess_data(train)

y_train = train['Survived']
X_train = train.drop('Survived', axis=1)

len(X_train), len(y_train)
```

```
(891, 891)
```

```
test = pd.read_csv('test_with_survived.csv', index_col=0)
test = preprocess_data(test)

y_test = test['Survived']
X_test = test.drop(["Survived"], axis=1)

len(X_test), len(y_test)
```

```
(418, 418)
```

Model

We will use catboost, therefore we need to indicate the categorical features.

```
cat_features = np.where(X_train.dtypes != float)[0]
cat_features
```

```
array([ 0,  1,  3,  4,  8,  9, 10])
```

Create the Model

```
# fit the model
model = CatBoostClassifier(loss_function='Logloss', eval_metric='Accuracy', verbo
model.fit(X_train, y_train, eval_set=(X_test, y_test))

# make predictions
y_pred = model.predict(X_test)

# evaluate predictions
accuracy = accuracy_score(y_test, y_pred)
accuracy # 0.8947368421052632
```

```
0.8947368421052632
```

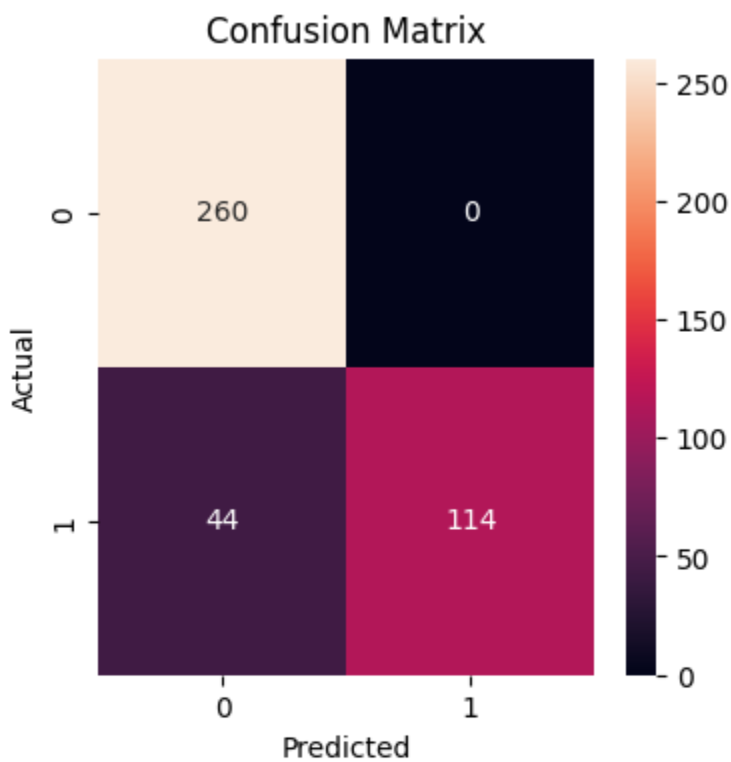
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      1.00      0.92       260
           1       1.00      0.72      0.84       158

    accuracy                           0.89       418
   macro avg       0.93      0.86      0.88       418
weighted avg       0.91      0.89      0.89       418
```

```
plt.figure(figsize = (4,4))
cm = sns.heatmap(confusion_matrix(y_test, y_pred), fmt='g', annot=True)
cm.set(title='Confusion Matrix')
cm.set(xlabel='Predicted', ylabel='Actual')
plt.show()
```




```
# get the feature importance
feature_importance = model.get_feature_importance(prettified=True)
feature_importance
```

	Feature Id	Importances
0	AdjustedSurvival	65.674739
1	SurnameSurvival	33.395220
2	Sex	0.930041

3	Pclass	0.000000
4	Age	0.000000
5	SibSp	0.000000
6	Parch	0.000000
7	Fare	0.000000
8	Embarked_C	0.000000
9	Embarked_Q	0.000000
10	Embarked_S	0.000000

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

Hyperparameter Tuning

We are going to find the best parameters for the model using GridSearchCV. Also, we are going to use Cross Validation to avoid overfitting.

```
'''
cat_for_search = CatBoostClassifier(loss_function='Logloss',
                                    eval_metric='Accuracy',
                                    verbose=False,
                                    random_state=42)

params = {
    'depth': [4, 6, 8, 10],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'iterations': [100, 200, 300, 500],
    'l2_leaf_reg': [1, 3, 5, 7],
}

grid = cat_for_search.grid_search(params, Pool(X_train, y_train, cat_features=cat_

best_model = CatBoostClassifier(depth=grid['params']['depth'],
                                loss_function='Logloss',
                                eval_metric='Accuracy',
                                use_best_model=True,
                                random_seed=42,
                                verbose=False)

best_model.fit(X_train, y_train, cat_features=cat_features, eval_set = (X_test, y_
```

```
accuracy_score(y_test, best_model.predict(X_test))
'''
```

```
'\ncat_for_search = CatBoostClassifier(loss_function='Logloss',\n
eval_metric='Accuracy',\n                                                  verbose=False,\n
random_state=42)\n\nparams = {\n    'depth': [4, 6, 8, 10],\n    'learning_ra
te': [0.01, 0.05, 0.1, 0.2],\n    'iterations': [100, 200, 300, 500],\n    'l
2_leaf_reg': [1, 3, 5, 7],\n}\n\ngrid = cat_for_search.grid_search(params, Po
ol(X_train, y_train, cat_features=cat_features), shuffle=True, cv=5, verbose=
False,  plot=False)\n\nbest_model = CatBoostClassifier(depth=grid['params']
['depth'],\n                              loss_function='Logloss',\n
```