miniproject

October 2, 2024

[164]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
↪outside of the current session
```

[165]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
df = sns. load_dataset('titanic')
```

[166]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
```

```
---  ------        --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

[167]:
```python
print(df.head())
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

[168]:
```python
# Check for missing values
print(df.isnull().sum())
```

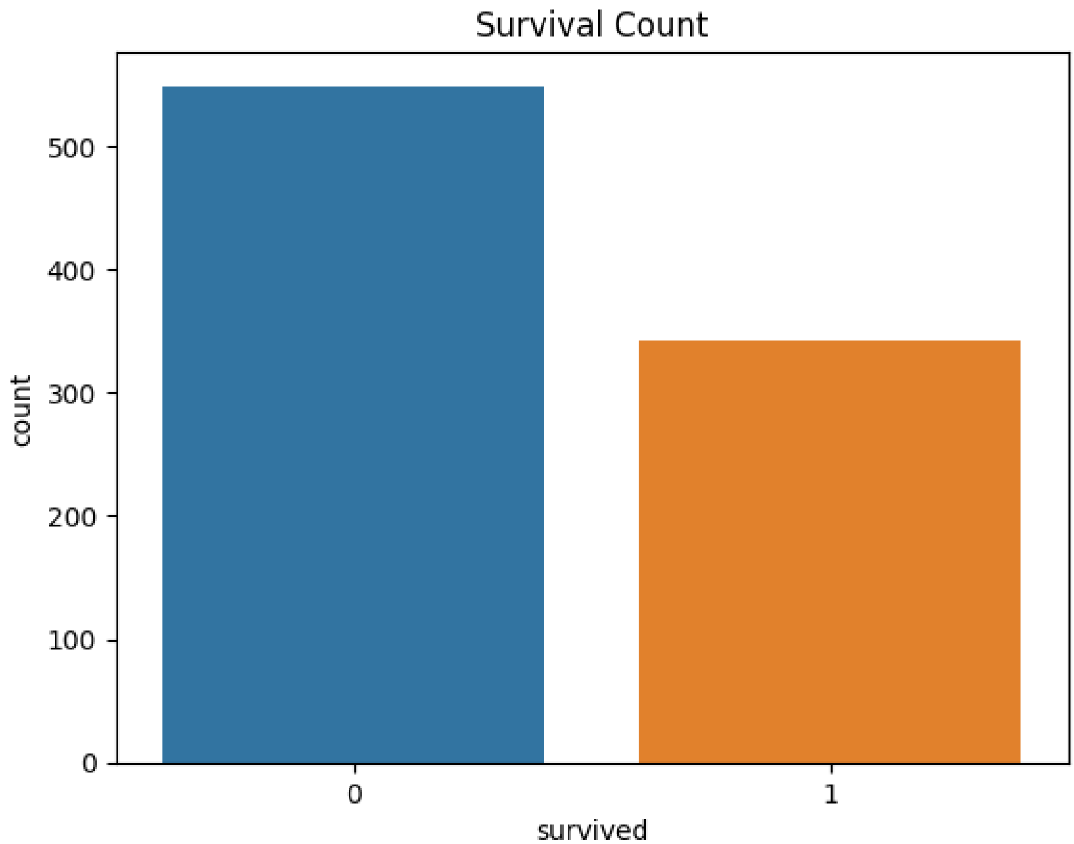
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
```

```
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

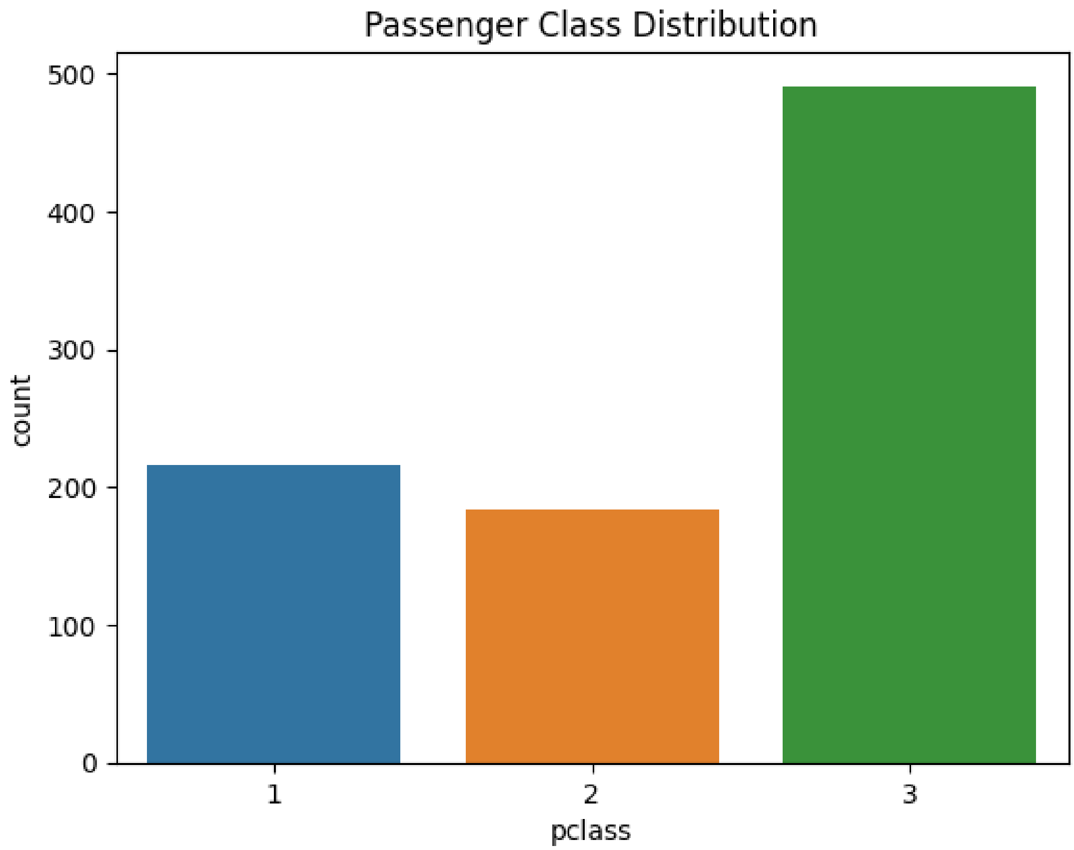
[169]:
```
# Summary statistics
print(df.describe())
```

```
         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
```

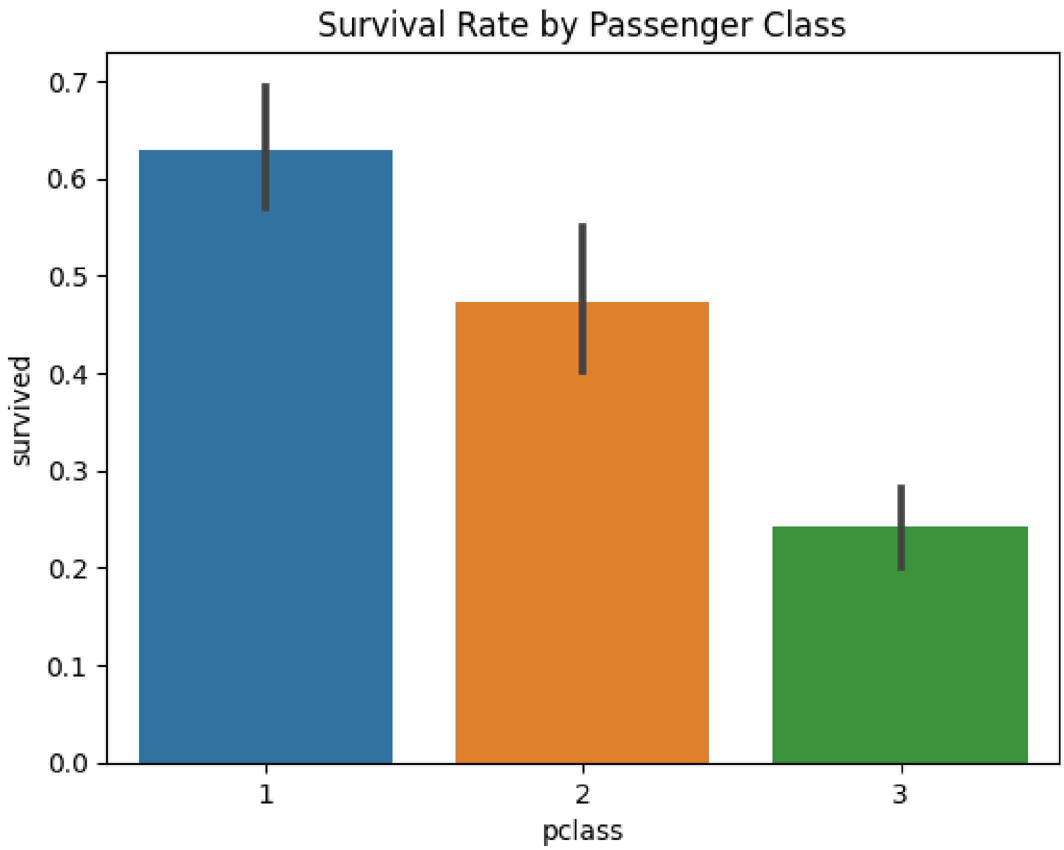
[170]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Count plot for 'Survived'
sns.countplot(x='survived', data=df)
plt.title('Survival Count')
plt.show()
```



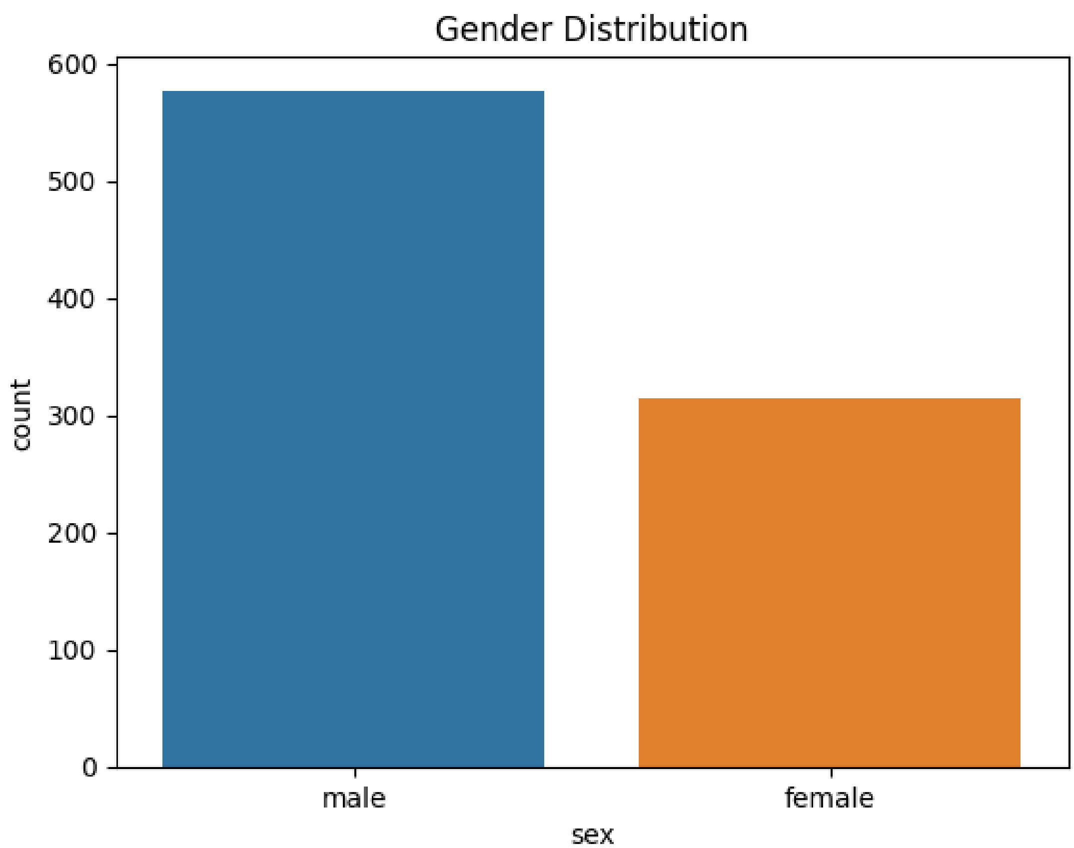
[171]:
```
# Count plot for 'Pclass'
sns.countplot(x='pclass', data=df)
plt.title('Passenger Class Distribution')
plt.show()
```



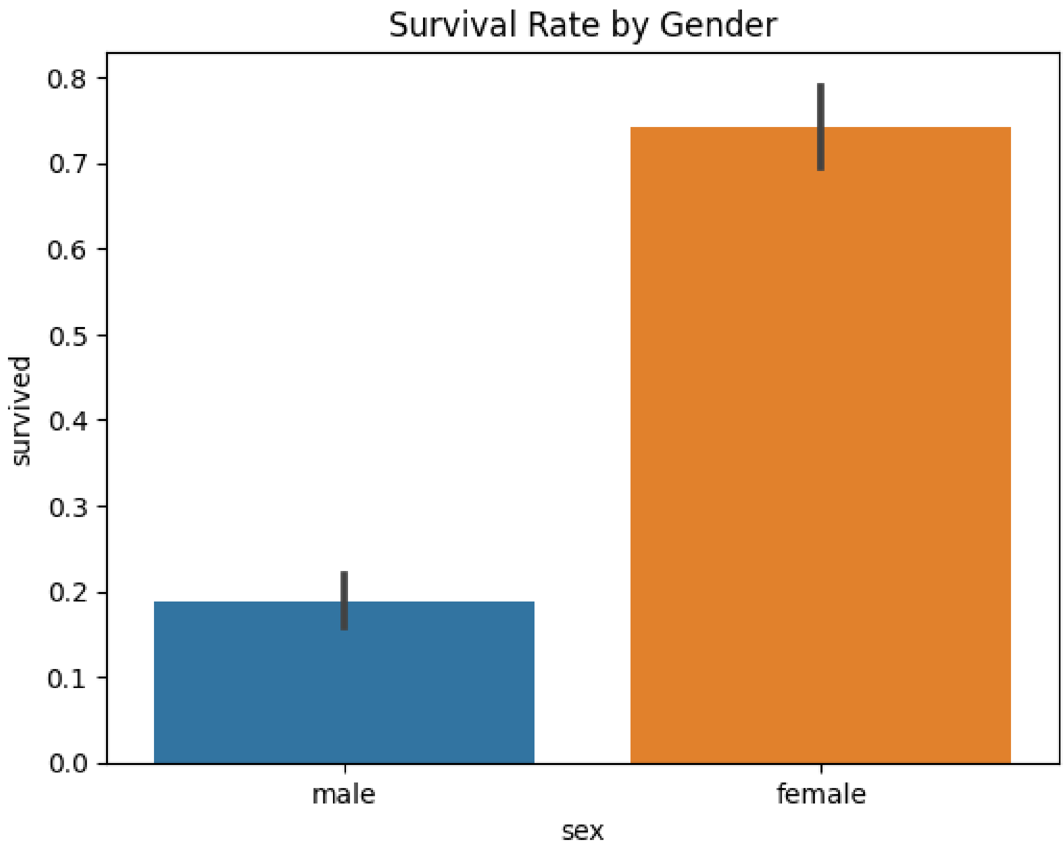
[172]:
```
# Survival rate by class
sns.barplot(x='pclass', y='survived', data=df)
plt.title('Survival Rate by Passenger Class')
plt.show()
```



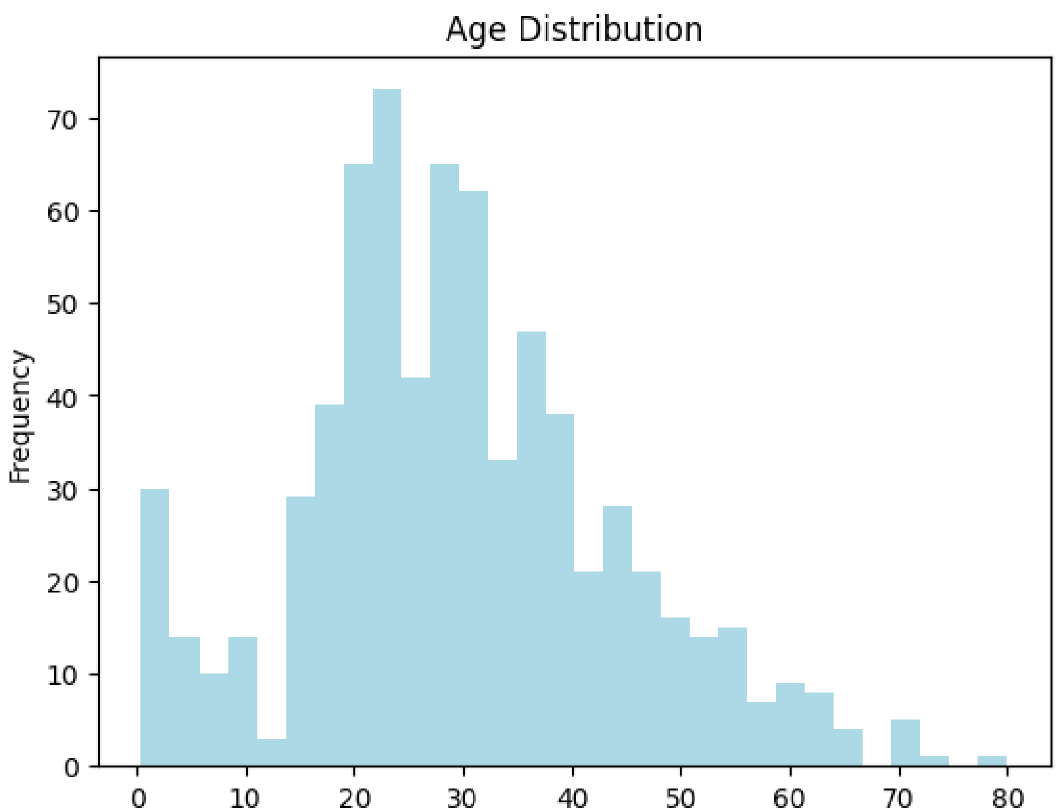
[173]:
```
# Count plot for 'Sex'
sns.countplot(x='sex', data=df)
plt.title('Gender Distribution')
plt.show()
```



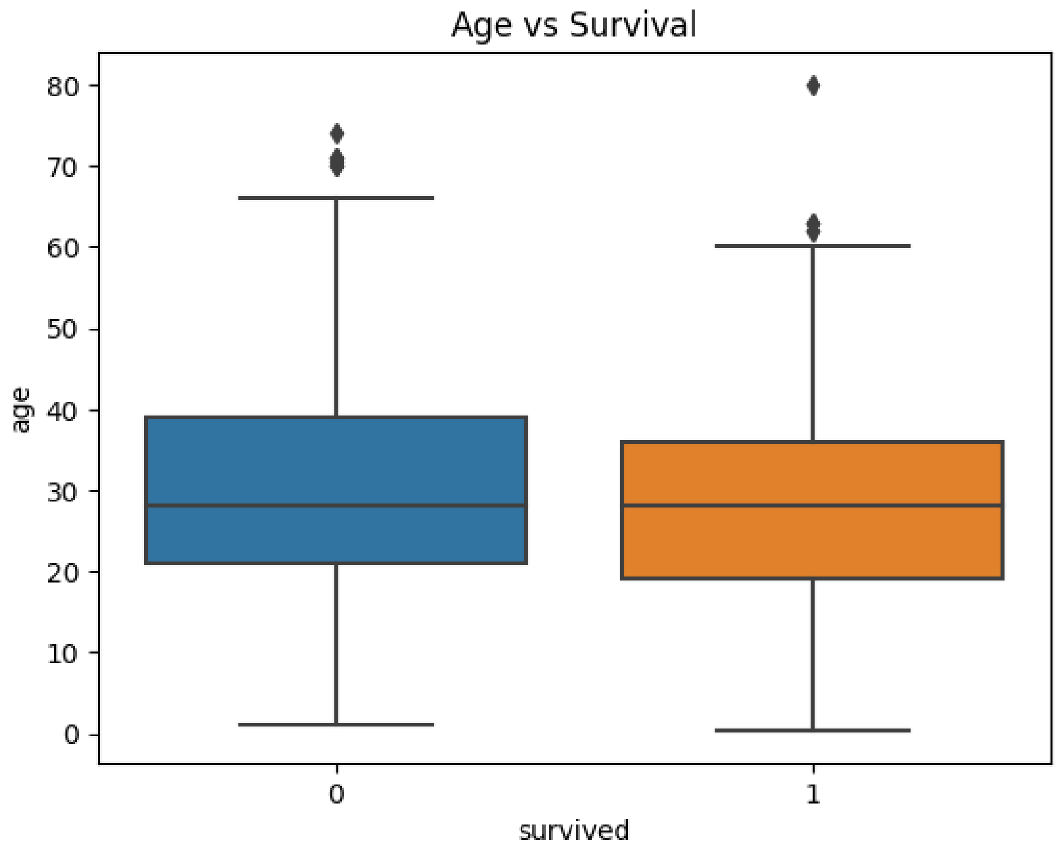
[174]:
```
# Survival rate by gender
sns.barplot(x='sex', y='survived', data=df)
plt.title('Survival Rate by Gender')
plt.show()
```



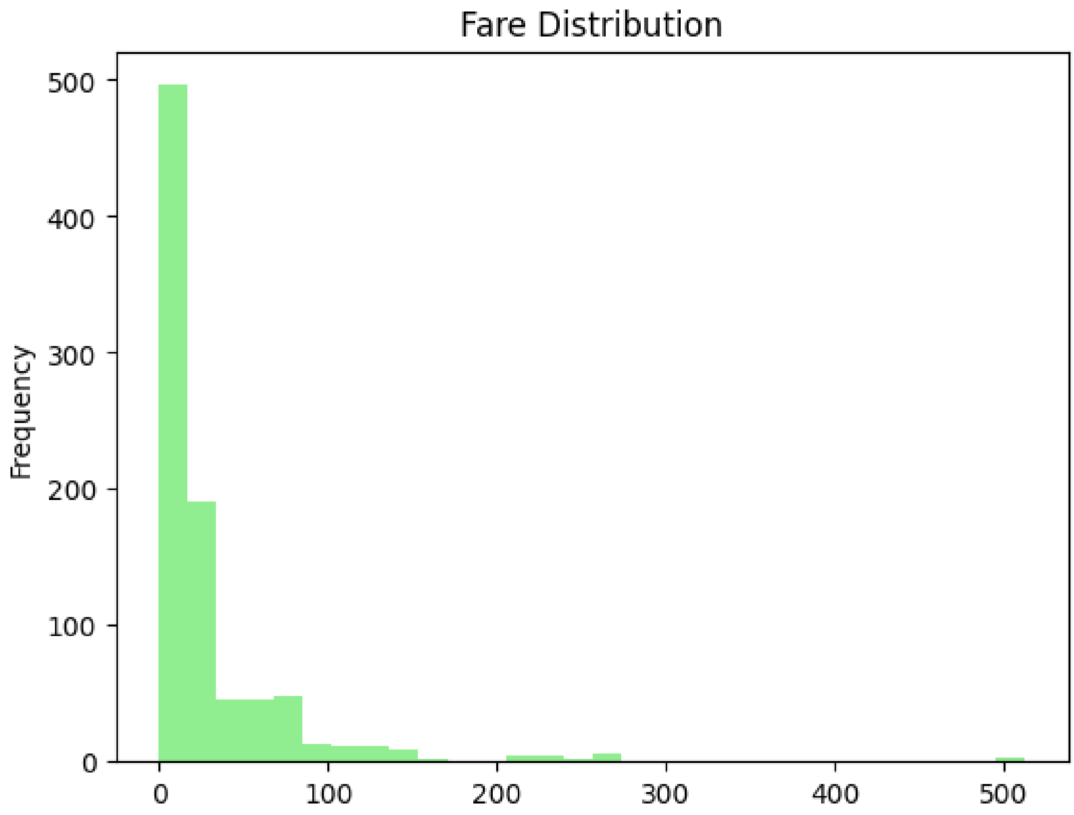
[175]:
```
# Histogram for 'Age'
df['age'].plot(kind='hist', bins=30, color='lightblue')
plt.title('Age Distribution')
plt.show()
```



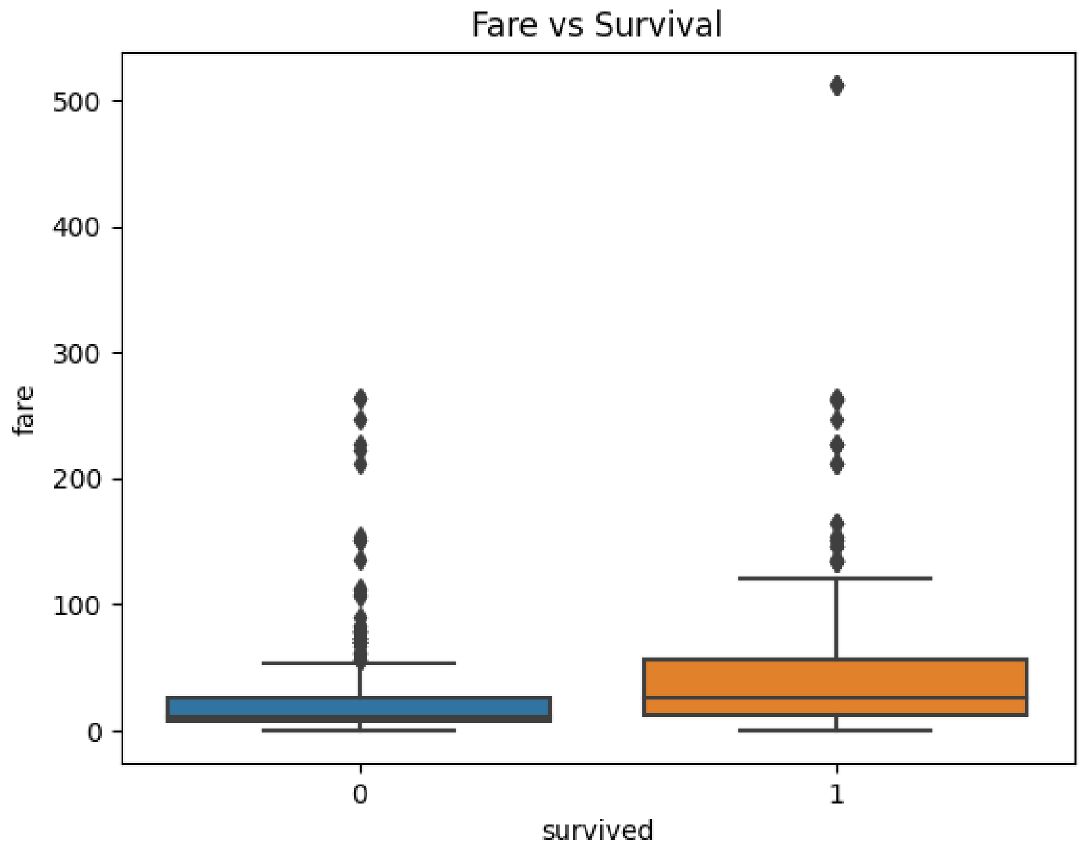
[176]:
```
# Box plot for 'Age'
sns.boxplot(x='survived', y='age', data=df)
plt.title('Age vs Survival')
plt.show()
```



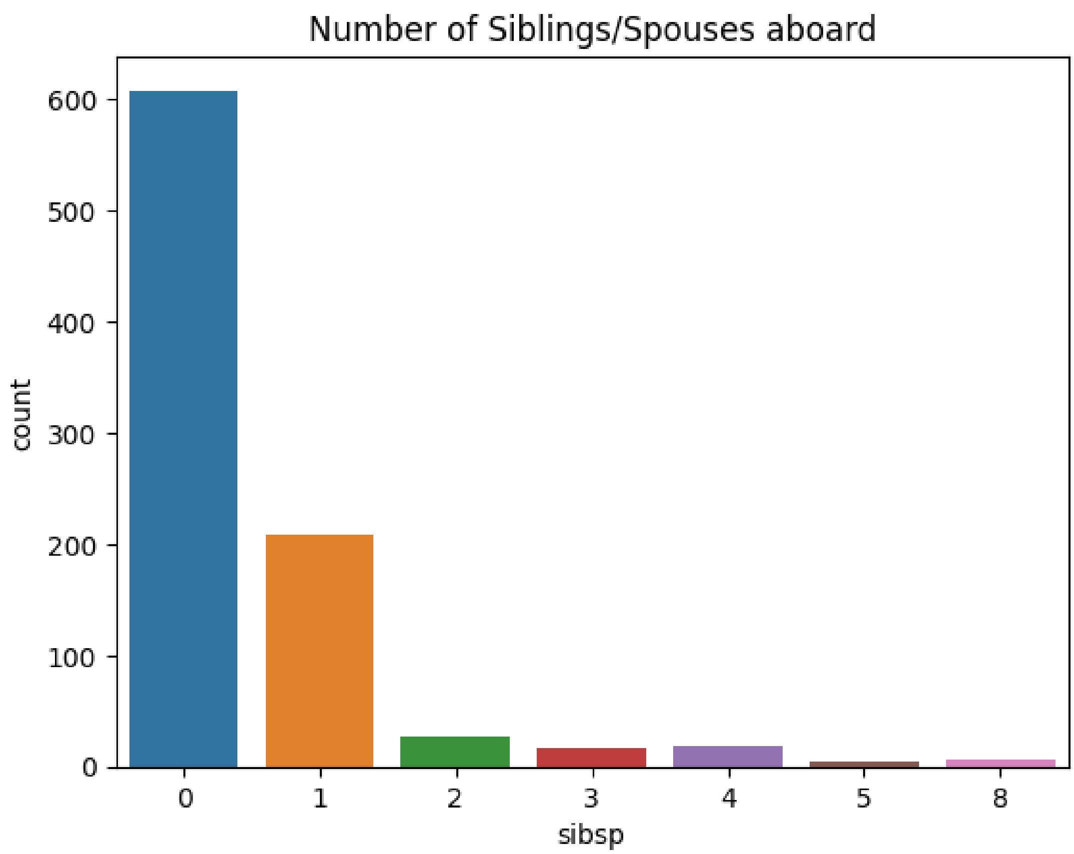
[216]:
```
# Histogram for 'Fare'
df['fare'].plot(kind='hist', bins=30, color='lightgreen')
plt.title('Fare Distribution')
plt.show()
```



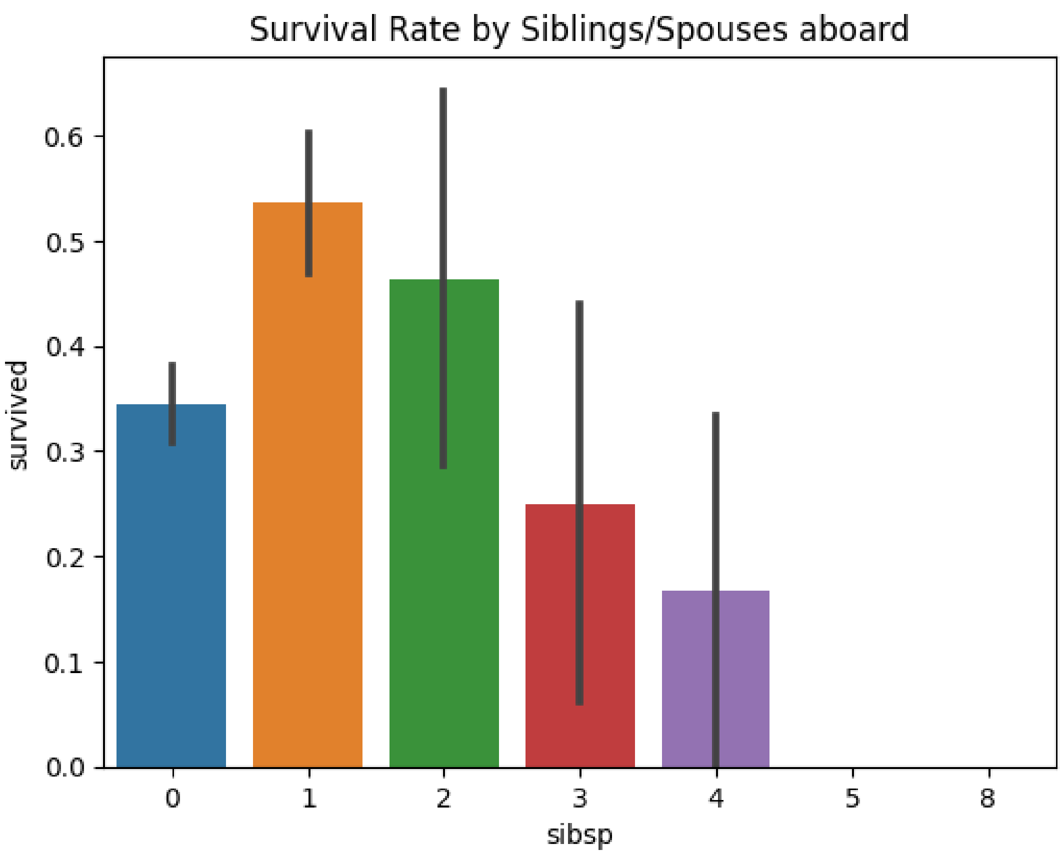
[178]:
```
# Box plot for 'Fare'
sns.boxplot(x='survived', y='fare', data=df)
plt.title('Fare vs Survival')
plt.show()
```



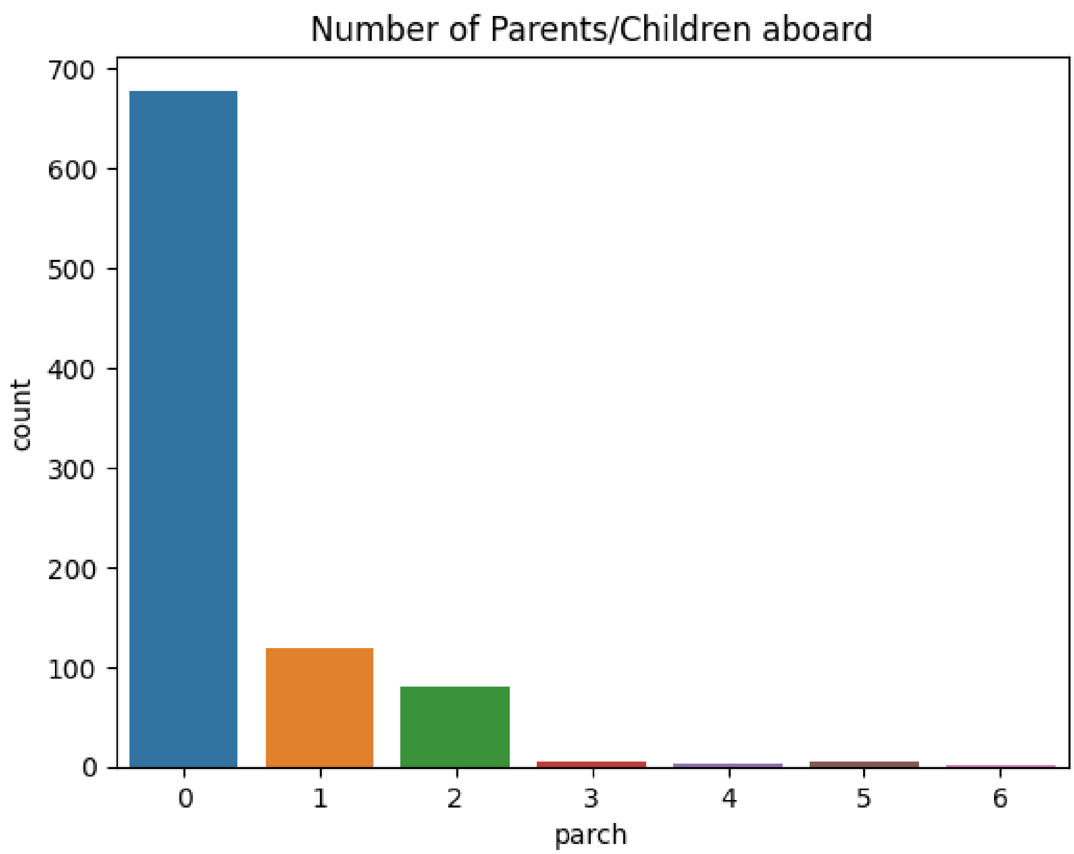
[179]:
```
# Count plot for 'SibSp'
sns.countplot(x='sibsp', data=df)
plt.title('Number of Siblings/Spouses aboard')
plt.show()
```



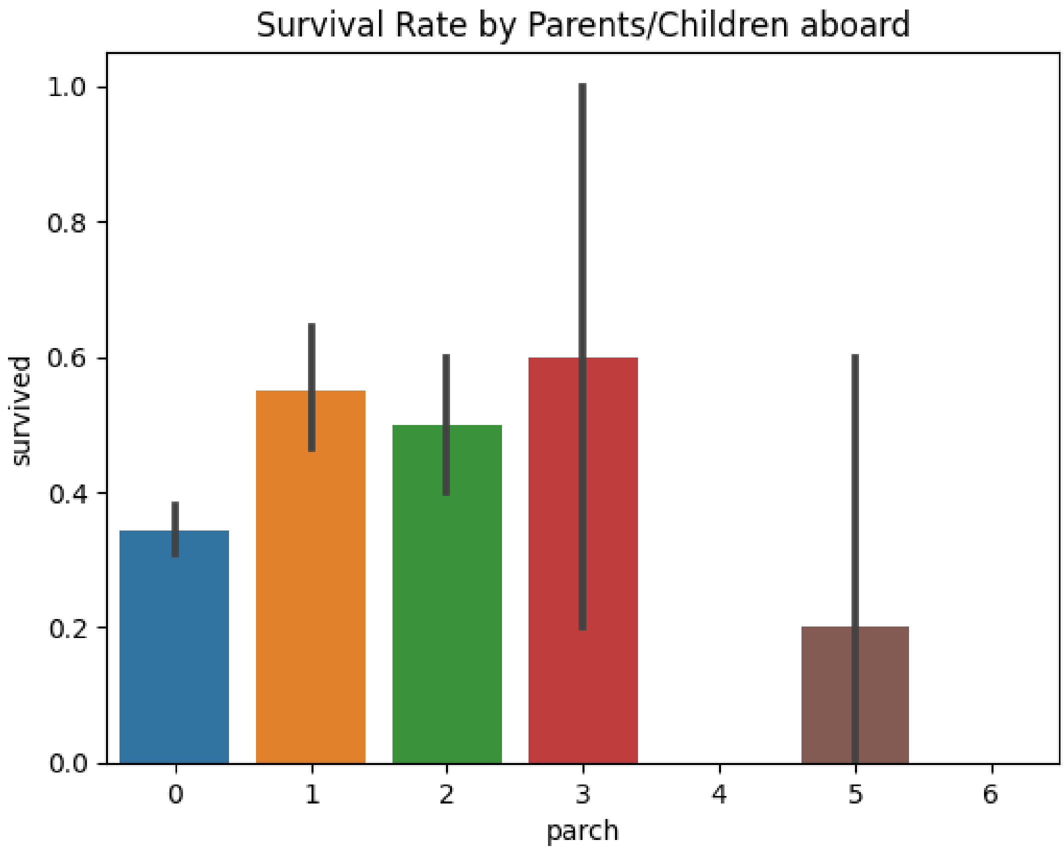
[180]:
```
# Survival rate by 'SibSp'
sns.barplot(x='sibsp', y='survived', data=df)
plt.title('Survival Rate by Siblings/Spouses aboard')
plt.show()
```



[181]:
```
# Count plot for 'Parch'
sns.countplot(x='parch', data=df)
plt.title('Number of Parents/Children aboard')
plt.show()
```

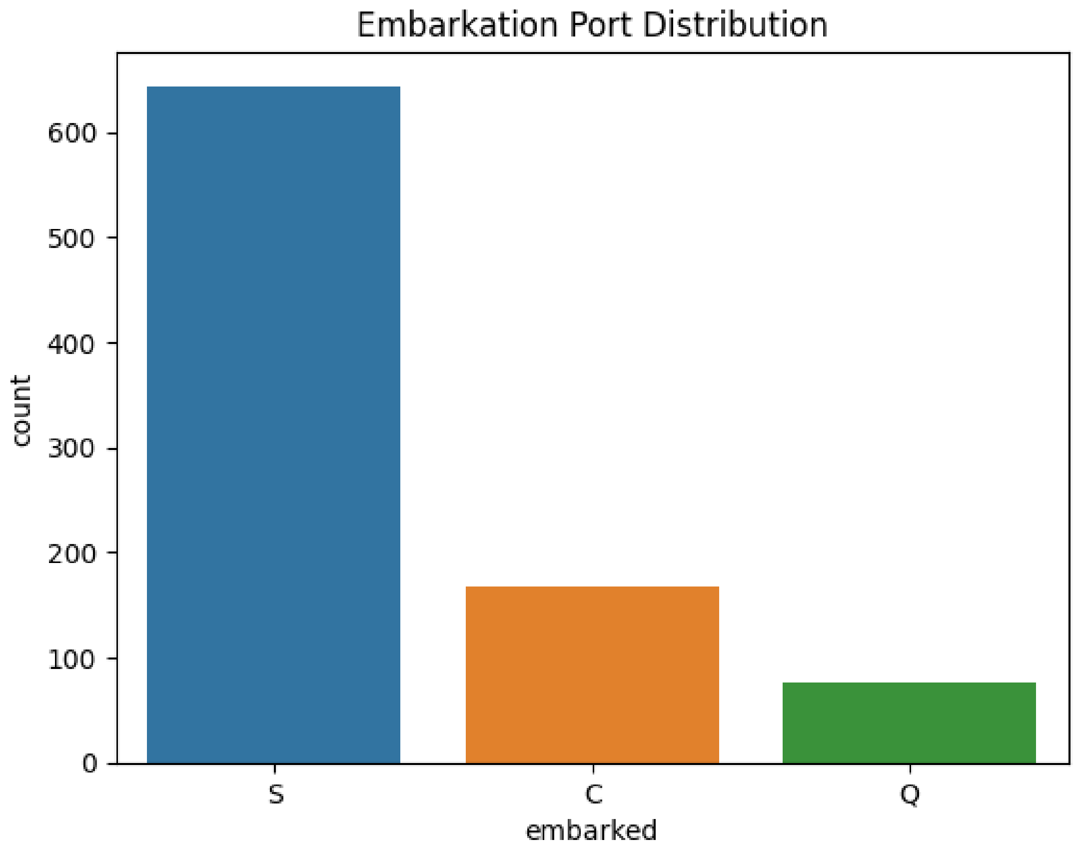


[182]:
```
# Survival rate by 'Parch'
sns.barplot(x='parch', y='survived', data=df)
plt.title('Survival Rate by Parents/Children aboard')
plt.show()
```

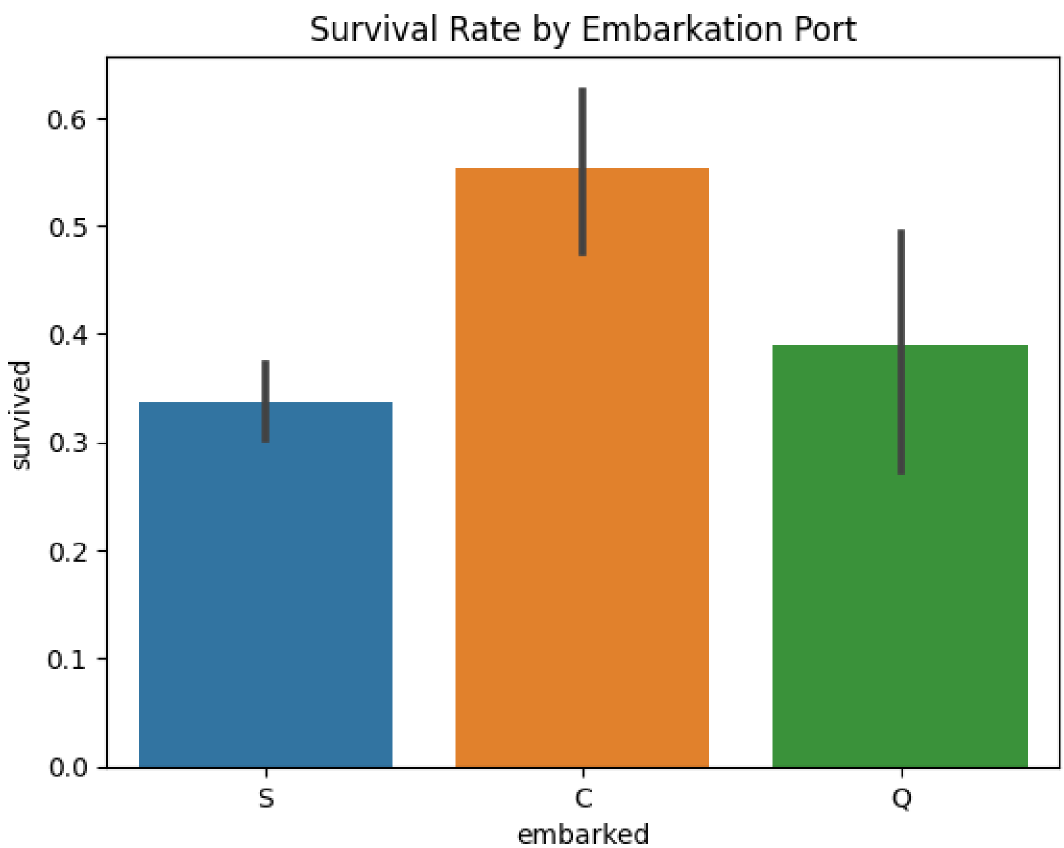


[183]:

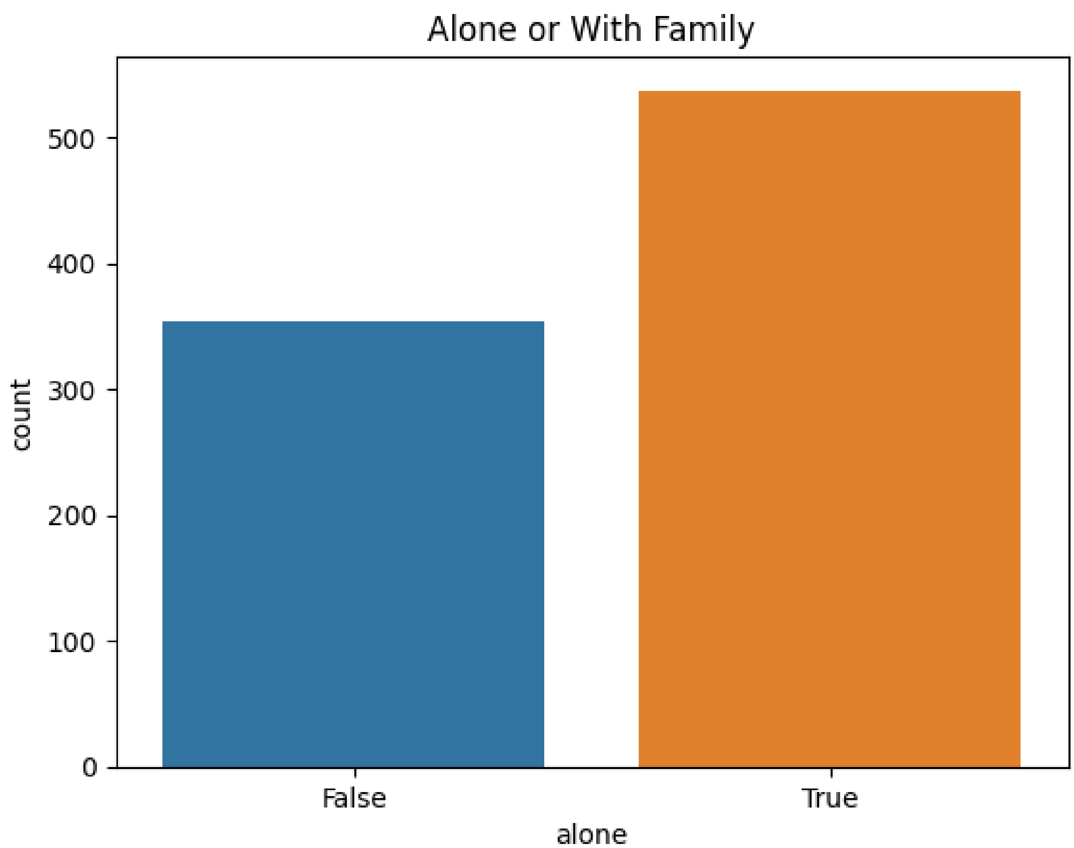
```
# Count plot for 'Embarked'
sns.countplot(x='embarked', data=df)
plt.title('Embarkation Port Distribution')
plt.show()
```



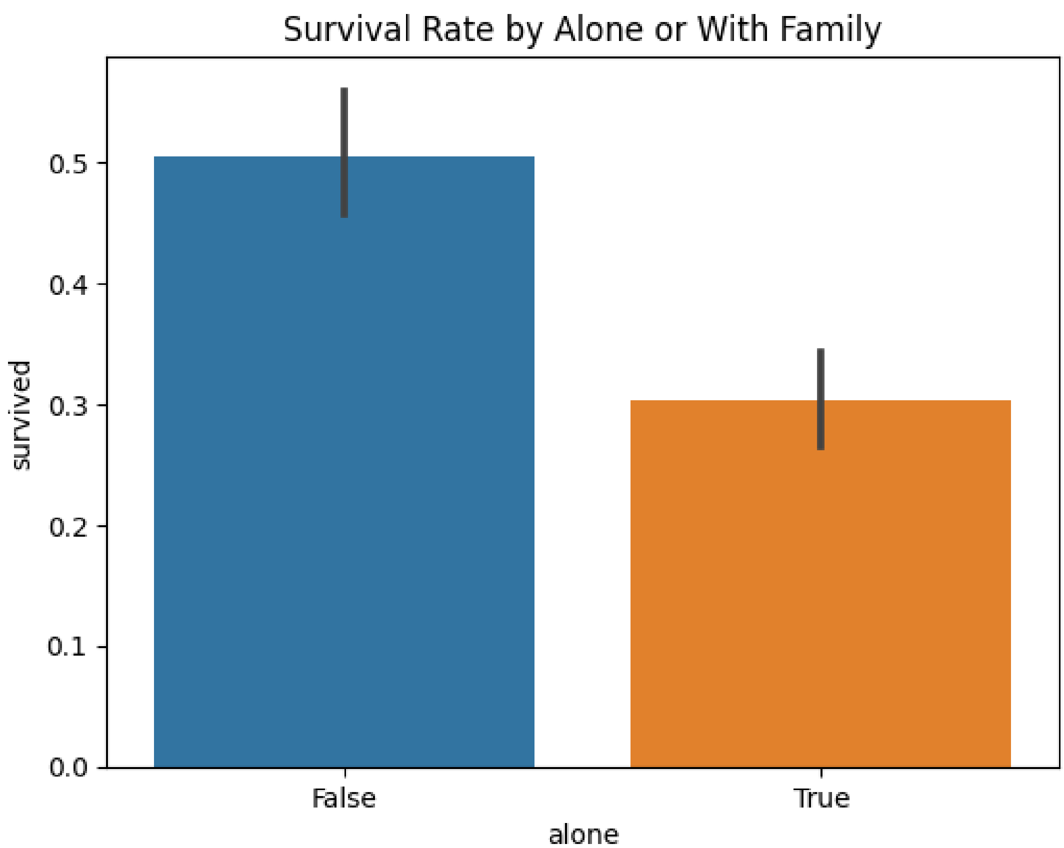
[184]:
```
# Survival rate by 'Embarked'
sns.barplot(x='embarked', y='survived', data=df)
plt.title('Survival Rate by Embarkation Port')
plt.show()
```



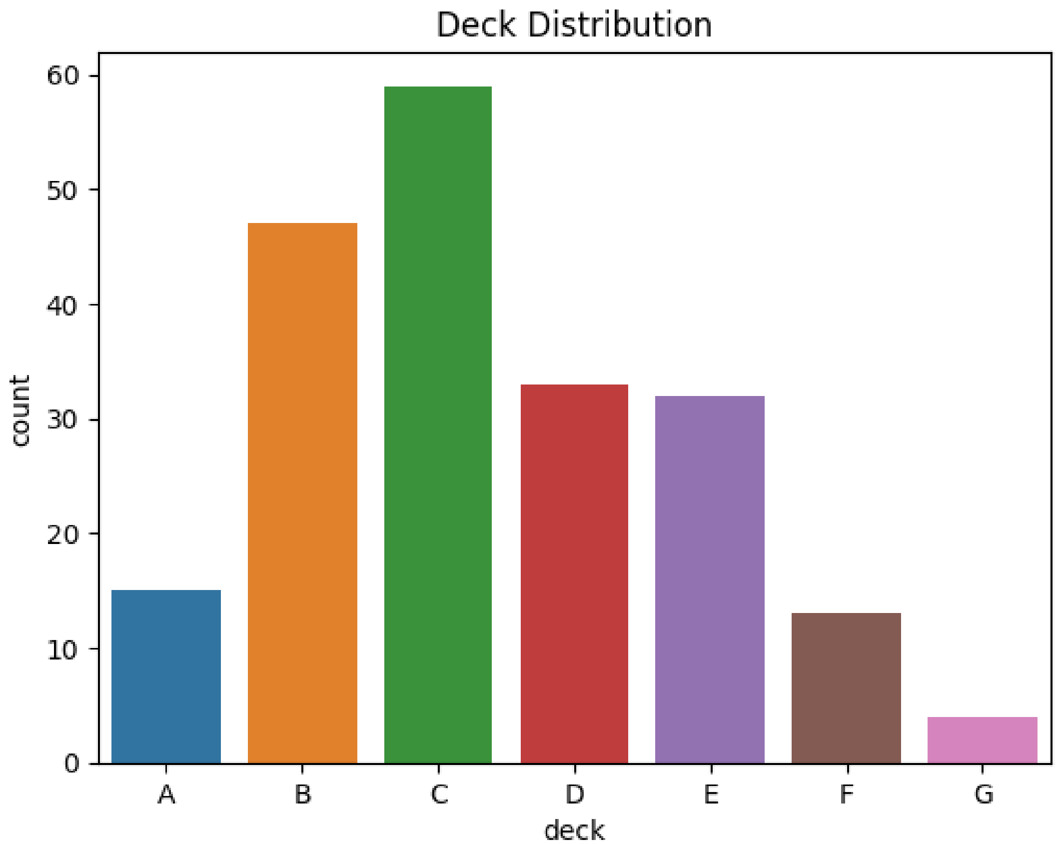
[185]:
```
# Count plot for 'Alone'
sns.countplot(x='alone', data=df)
plt.title('Alone or With Family')
plt.show()
```



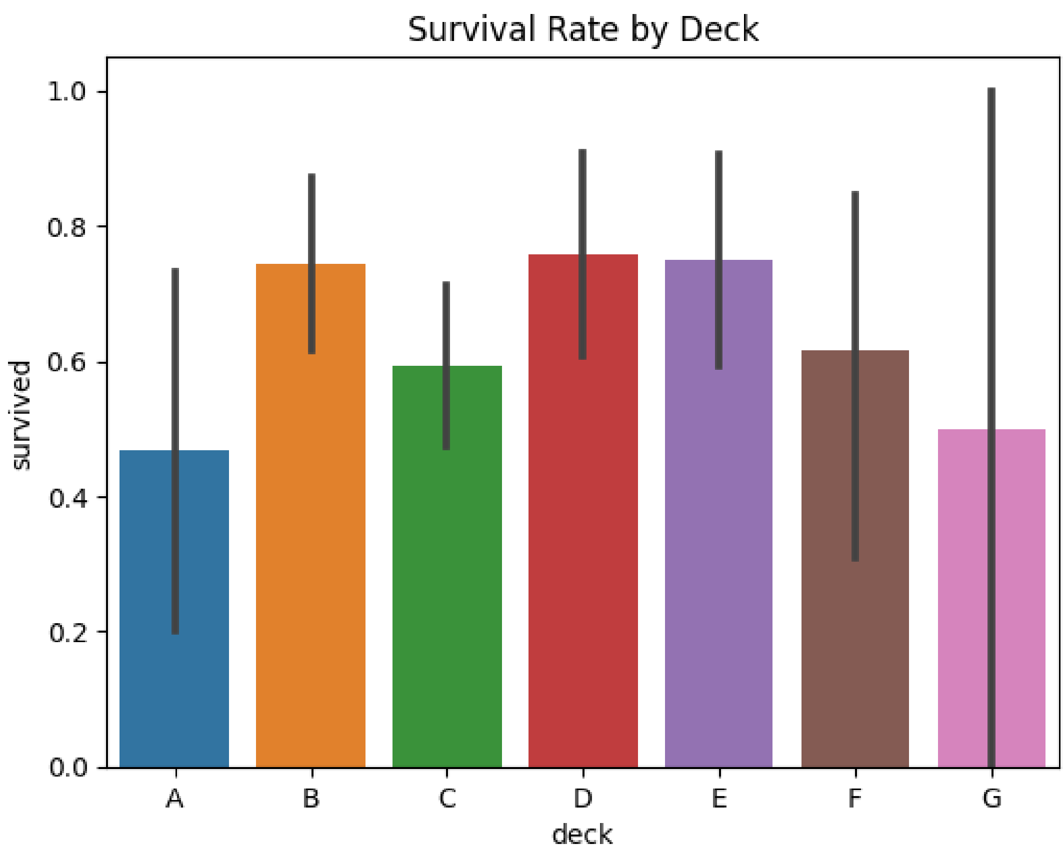
[186]:
```
# Survival rate by 'Alone'
sns.barplot(x='alone', y='survived', data=df)
plt.title('Survival Rate by Alone or With Family')
plt.show()
```



[187]:
```
# Count plot for 'Deck'
sns.countplot(x='deck', data=df)
plt.title('Deck Distribution')
plt.show()
```



[188]:
```
# Survival rate by 'Deck'
sns.barplot(x='deck', y='survived', data=df)
plt.title('Survival Rate by Deck')
plt.show()
```



[189]:
```
# Fill missing 'age' based on median within 'pclass' and 'sex' groups
df['age'] = df.groupby(['pclass', 'sex'])['age'].transform(lambda x: x.fillna(x.
    ↪median()))
```

[190]:
```
# Add 'Unknown' to the list of categories in 'deck' column
df['deck'] = df['deck'].cat.add_categories('Unknown')

# Fill missing 'deck' values with 'Unknown'
df['deck'].fillna('Unknown', inplace=True)
```

[191]:
```
# Fill missing 'embarked' with mode
df['embarked'].fillna(df['embarked'].mode()[0], inplace=True)
```

[192]:
```
# Fill missing 'embark_town' with mode
df['embark_town'].fillna(df['embark_town'].mode()[0], inplace=True)
```

[193]:
```
df.isnull().sum()
```

[193]:
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```

[194]:
```
# Encoding categorical variables
df['sex'] = df['sex'].map({'male': 0, 'female': 1})
df = pd.get_dummies(df, columns=['embarked', 'class', 'who', 'deck',␣
↪'embark_town'], drop_first=True)
```

[195]:
```
# Create a 'family_size' feature
df['family_size'] = df['sibsp'] + df['parch']
```

[196]:
```
# Drop irrelevant columns
df.drop(['alive', 'adult_male', 'alone'], axis=1, inplace=True)
```

[197]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 23 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   survived                 891 non-null    int64
 1   pclass                   891 non-null    int64
 2   sex                      891 non-null    int64
 3   age                      891 non-null    float64
 4   sibsp                    891 non-null    int64
 5   parch                    891 non-null    int64
 6   fare                     891 non-null    float64
 7   embarked_Q               891 non-null    bool
 8   embarked_S               891 non-null    bool
 9   class_Second             891 non-null    bool
 10  class_Third              891 non-null    bool
 11  who_man                  891 non-null    bool
```

 12 who_woman   891 non-null   bool
 13 deck_B   891 non-null   bool
 14 deck_C   891 non-null   bool
 15 deck_D   891 non-null   bool
 16 deck_E   891 non-null   bool
 17 deck_F   891 non-null   bool
 18 deck_G   891 non-null   bool
 19 deck_Unknown   891 non-null   bool
 20 embark_town_Queenstown   891 non-null   bool
 21 embark_town_Southampton   891 non-null   bool
 22 family_size   891 non-null   int64
dtypes: bool(15), float64(2), int64(6)
memory usage: 68.9 KB

[198]:
```
from sklearn.model_selection import train_test_split

X = df.drop('survived', axis=1)
y = df['survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[199]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# Logistic Regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Random Forest model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

[199]: RandomForestClassifier()

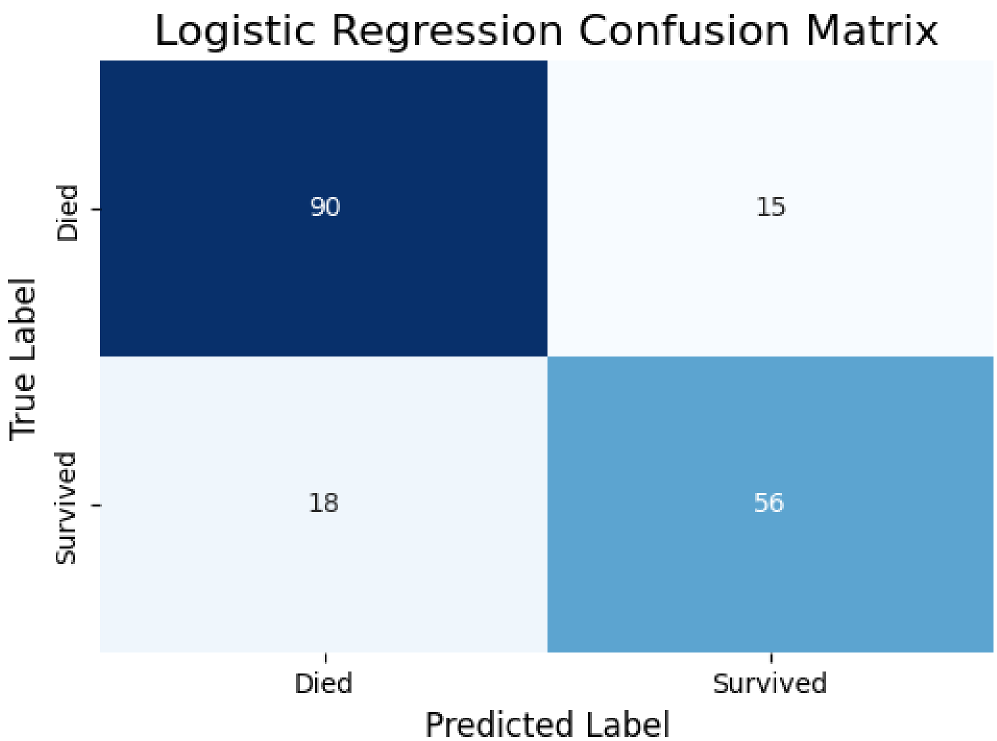
[200]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Function to plot the confusion matrix for Titanic dataset
def plot_confusion_matrix(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)

    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
```

```
                xticklabels=['Died', 'Survived'], yticklabels=['Died',␣
↪'Survived'])
    plt.title(f'{model_name} Confusion Matrix', fontsize=16)
    plt.xlabel('Predicted Label', fontsize=12)
    plt.ylabel('True Label', fontsize=12)
    plt.show()
```

[201]:
```
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix

# Logistic Regression Evaluation
y_pred_logreg = logreg.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_logreg))
print(classification_report(y_test, y_pred_logreg))
```

```
Logistic Regression Accuracy: 0.8156424581005587
              precision    recall  f1-score   support

           0       0.83      0.86      0.85       105
           1       0.79      0.76      0.77        74

    accuracy                           0.82       179
   macro avg       0.81      0.81      0.81       179
weighted avg       0.81      0.82      0.82       179
```

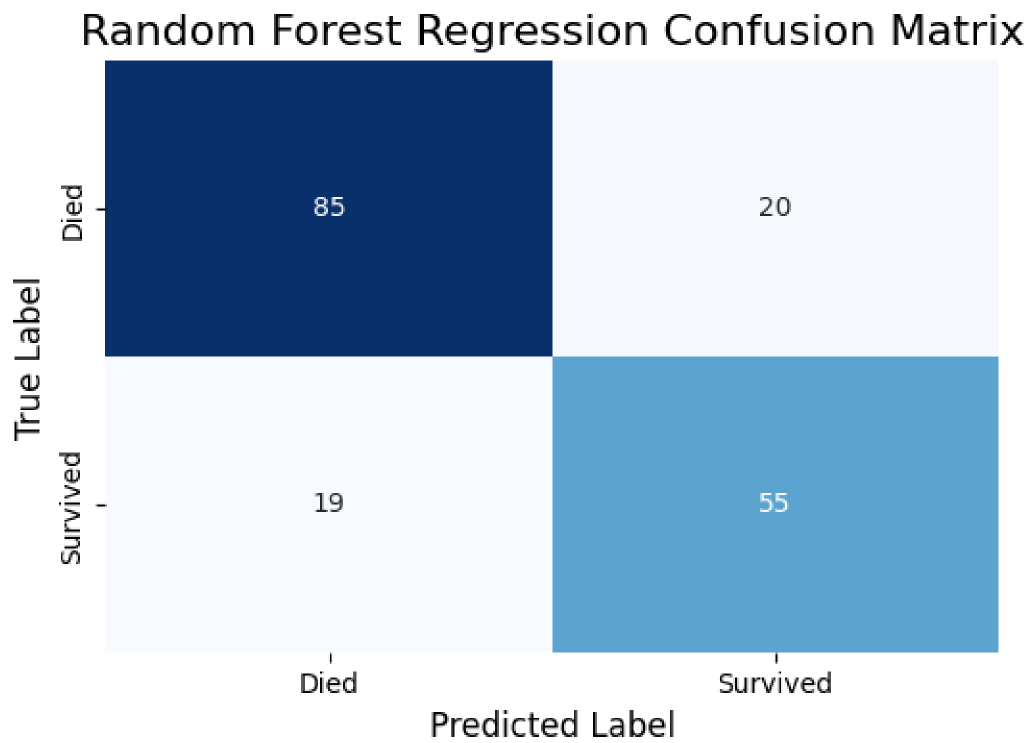
[202]:
```
plot_confusion_matrix(y_test, y_pred_logreg, "Logistic Regression")
```



[203]:
```
# Random Forest Evaluation
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.7821229050279329
              precision    recall  f1-score   support

           0       0.82      0.81      0.81       105
           1       0.73      0.74      0.74        74

    accuracy                           0.78       179
   macro avg       0.78      0.78      0.78       179
weighted avg       0.78      0.78      0.78       179
```

[204]:
```
plot_confusion_matrix(y_test, y_pred_rf, "Random Forest Regression")
```



[206]:
```
from sklearn.model_selection import GridSearchCV

# Example for Random Forest
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

# Perform GridSearchCV with Random Forest
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3)
grid_search_rf.fit(X_train, y_train)

# Print the best parameters
print("Best parameters for Random Forest:", grid_search_rf.best_params_)
```

```
Best parameters for Random Forest: {'max_depth': 20, 'min_samples_split': 5,
'n_estimators': 200}
```

[211]:
```
# Use the best parameters to create a new Random Forest model
best_rf = RandomForestClassifier(
    max_depth=grid_search_rf.best_params_['max_depth'],
    min_samples_split=grid_search_rf.best_params_['min_samples_split'],
```

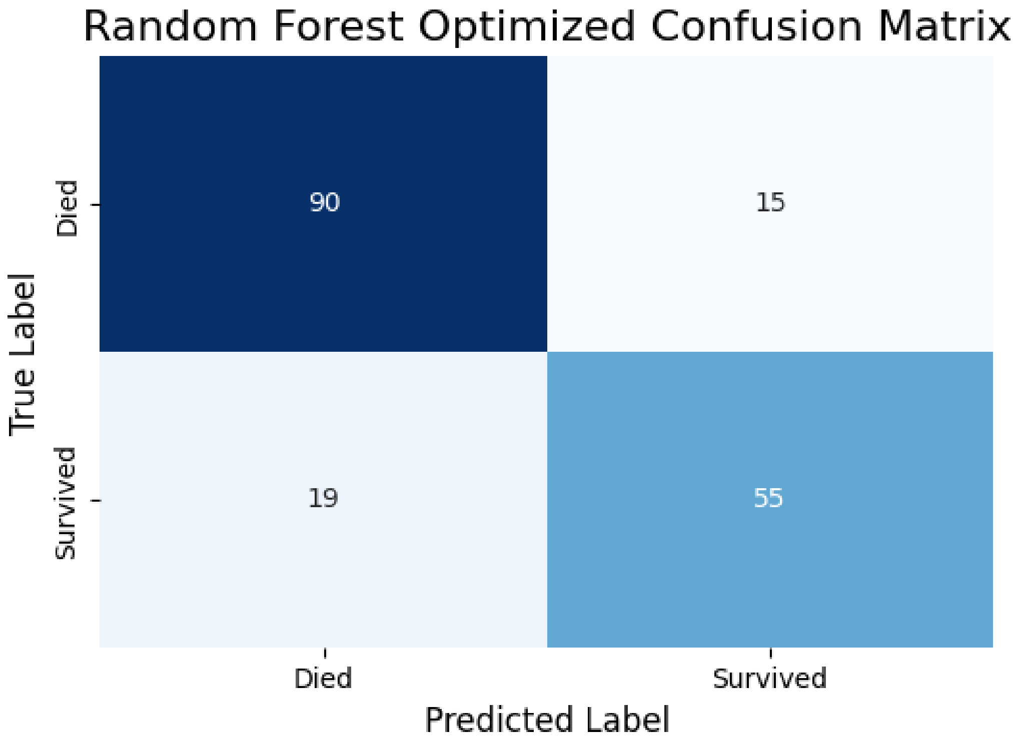
```python
    n_estimators=grid_search_rf.best_params_['n_estimators']
)

# Fit the model on the training data
best_rf.fit(X_train, y_train)

# Evaluate on the test data
y_pred = best_rf.predict(X_test)

# Accuracy
accuracy = best_rf.score(X_test, y_test)
print(f'Accuracy of the optimized Random Forest: {accuracy:.4f}')
```

```
Accuracy of the optimized Random Forest: 0.8101
```

[212]:
```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.88      0.84       105
           1       0.80      0.72      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179
```

[209]:
```python
plot_confusion_matrix(y_test, y_pred, "Random Forest Optimized")
```


Random Forest Optimized Confusion Matrix
True Label
Died
Survived
90
15
19
55
Died
Survived
Predicted Label