Import Libraries

In [ ]:
```
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.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder, StandardScaler
import joblib
```

Load And Explore Data

In [ ]:
```
data= pd.read_csv(r'C:\Users\Administrator\Desktop\internship\codeAlpha\task 1\data/Titanic-Dataset.csv')
```

In [ ]:
```
# Display the first few rows
print(data.head())

# Basic information
print(data.info())

# Summary statistics
print(data.describe())
```

```
   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
<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
None
       PassengerId    Survived      Pclass         Age       SibSp  \
```

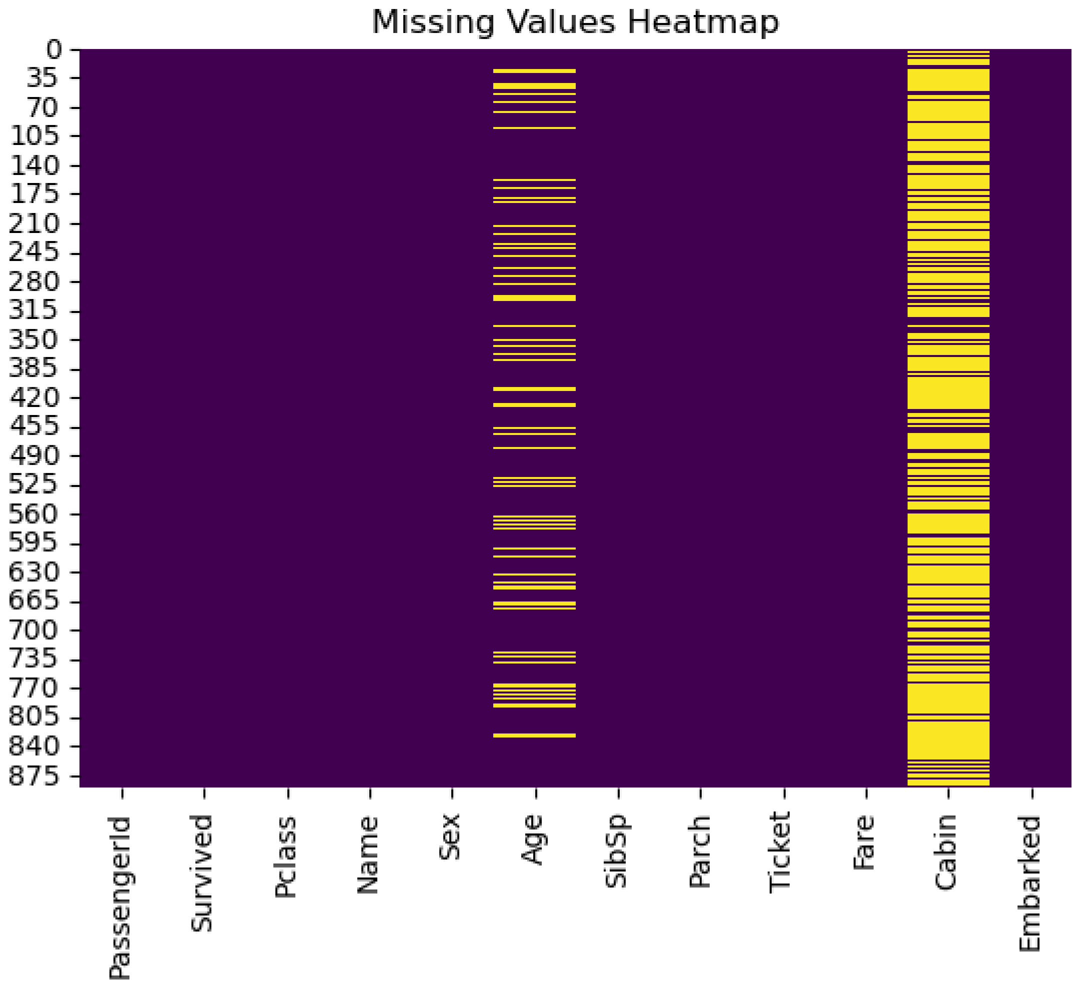
```
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    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
```

Exploratory Data Analysis

In [ ]:

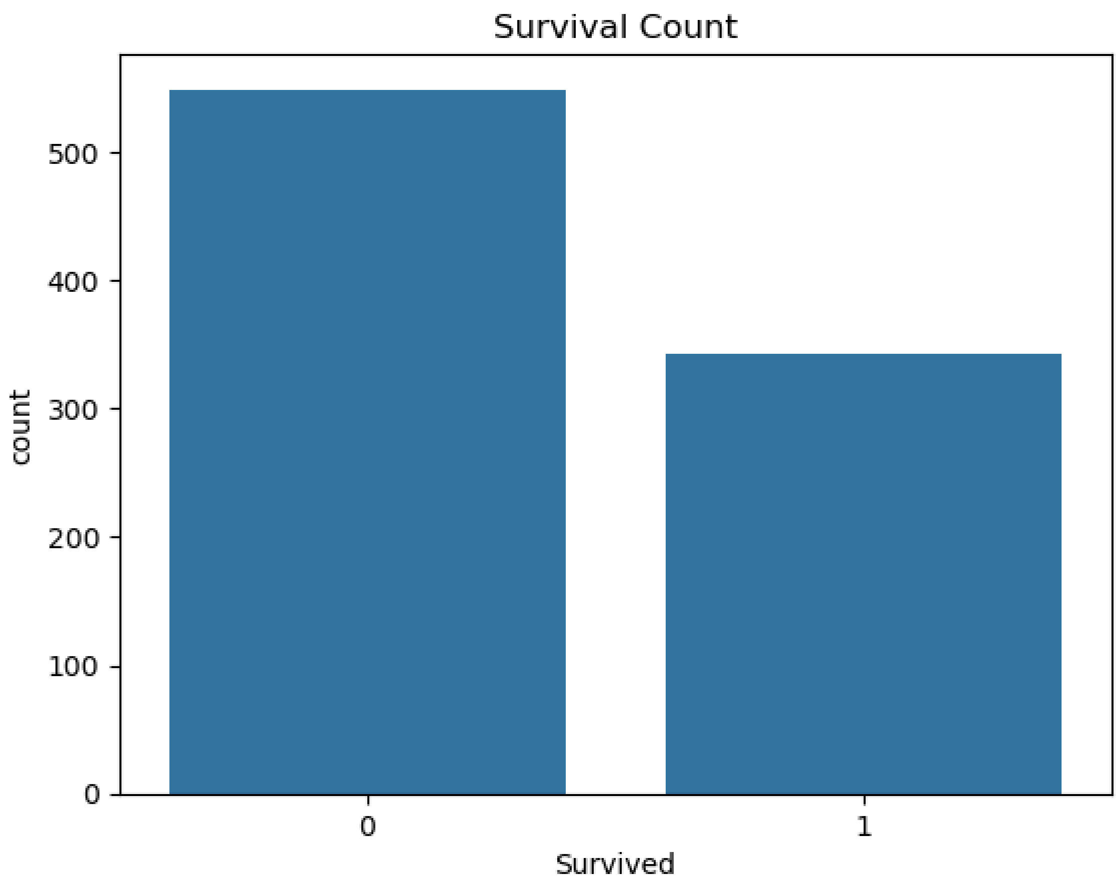
```python
# Check for missing values
print(data.isnull().sum())

# Visualize missing values
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Values Heatmap')
plt.show()
```

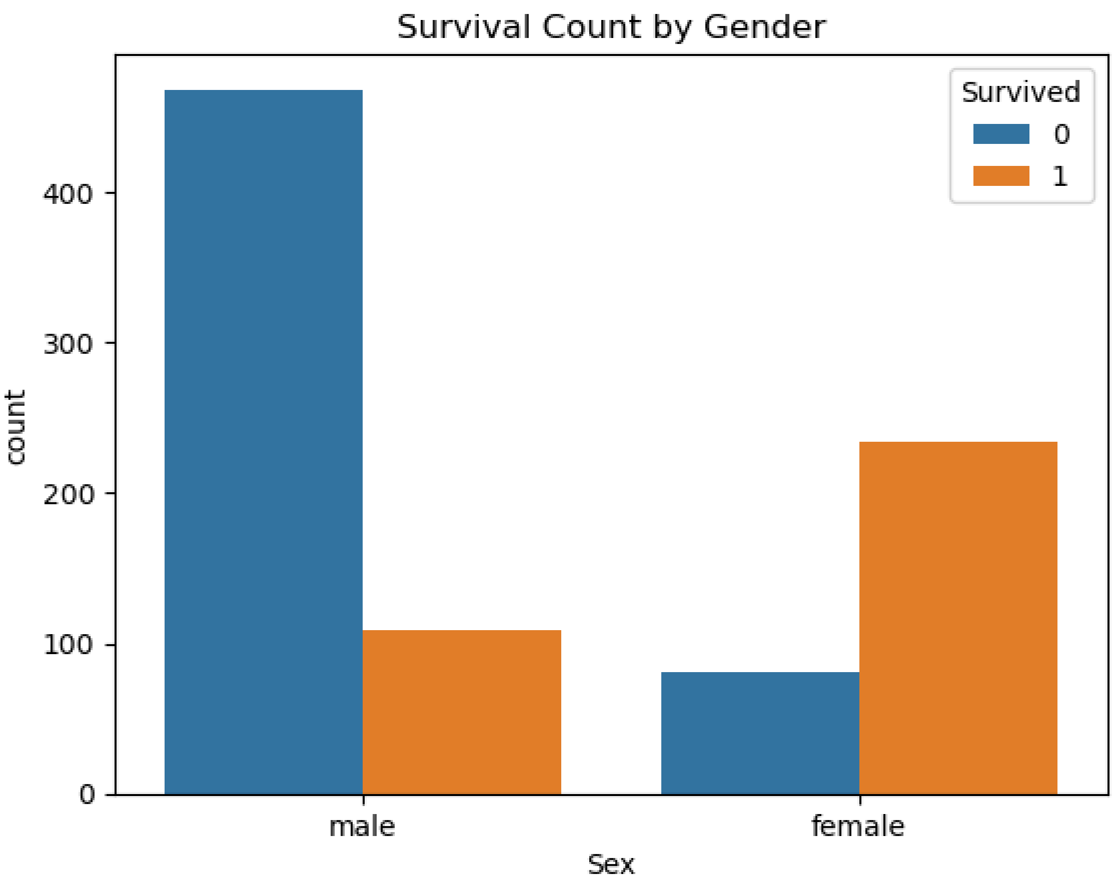
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```



In [ ]:

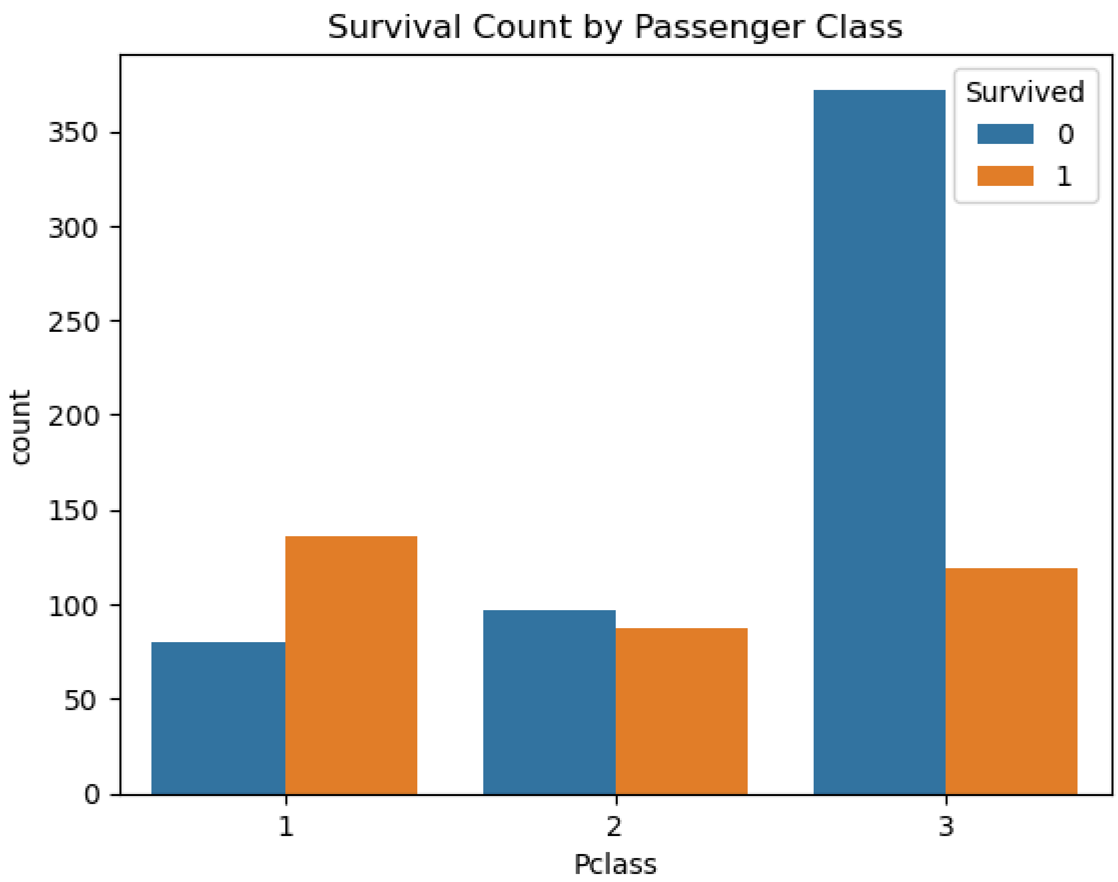
```
# Distribution of survivors
sns.countplot(x='Survived', data=data)
plt.title('Survival Count')
plt.show()

# Survival rate by gender
sns.countplot(x='Sex', hue='Survived', data=data)
plt.title('Survival Count by Gender')
plt.show()
```


Survival Count
count
500
400
300
200
100
0
0
1
Survived

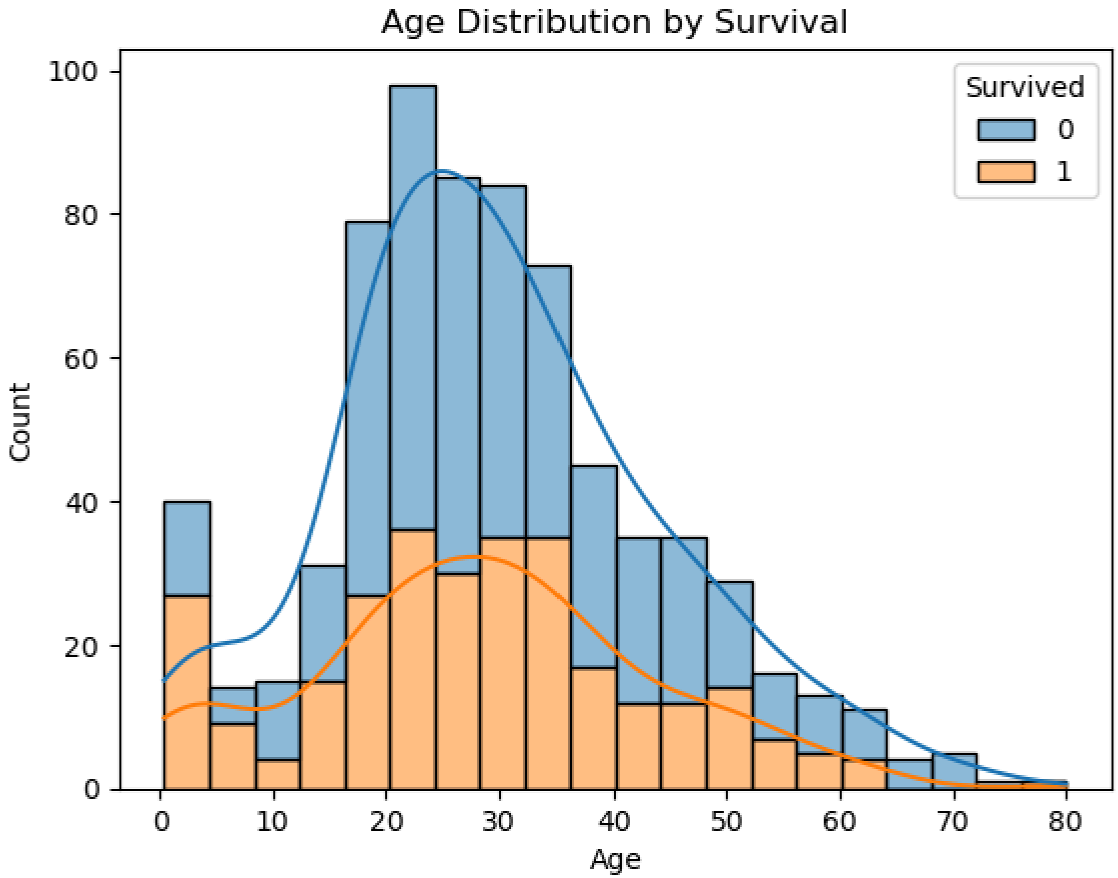


In [ ]:

```
# Survival rate by passenger class
sns.countplot(x='Pclass', hue='Survived', data=data)
plt.title('Survival Count by Passenger Class')
plt.show()

# Age distribution of survivors vs non-survivors
sns.histplot(data=data, x='Age', hue='Survived', kde=True, multiple='stack')
plt.title('Age Distribution by Survival')
plt.show()
```

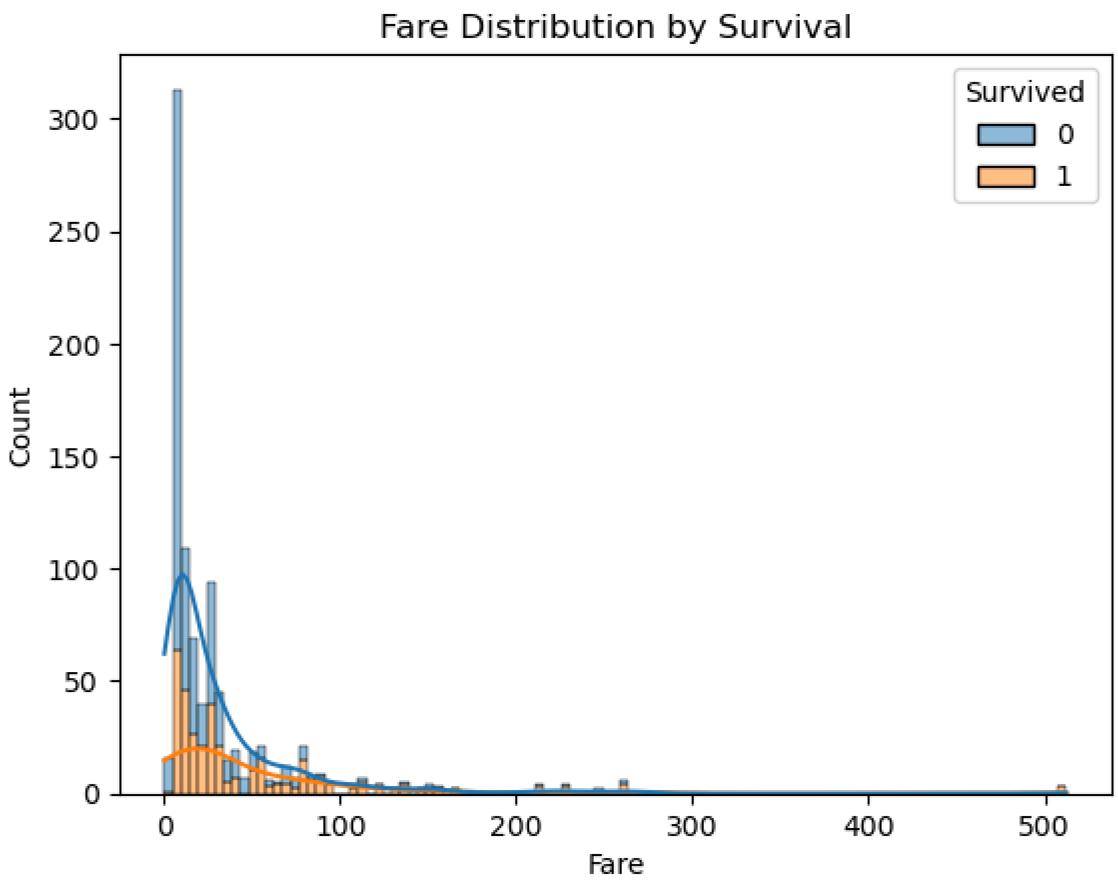


Survival Count by Passenger Class
Survived
0
1
count
350
300
250
200
150
100
50
0
1
2
3
Pclass



In [ ]:

```
# Fare distribution of survivors vs non-survivors
sns.histplot(data=data, x='Fare', hue='Survived', kde=True, multiple='stack')
plt.title('Fare Distribution by Survival')
plt.show()
```



Data Preprocessing

In [ ]:
```
# Fill missing Age with median
data['Age'] = data['Age'].fillna(data['Age'].median())

# Fill missing Embarked with mode
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])

# Check the columns in the DataFrame
print(data.columns)

# Drop Cabin column if it exists
```

```
data = data.drop('Cabin', axis=1, errors='ignore')

# Drop rows with missing Fare values
data = data.dropna(subset=['Fare'])
```

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

```
In [ ]: # Convert categorical variables to numerical
label_encoder = LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex'])
data['Embarked'] = label_encoder.fit_transform(data['Embarked'])

# Drop unnecessary columns
data.drop(['PassengerId', 'Name', 'Ticket'], axis=1, inplace=True)

# Check the processed data
print(data.head())
```

```
   Survived  Pclass  Sex   Age  SibSp  Parch     Fare  Embarked
0         0       3    1  22.0      1      0   7.2500         2
1         1       1    0  38.0      1      0  71.2833         0
2         1       3    0  26.0      0      0   7.9250         2
3         1       1    0  35.0      1      0  53.1000         2
4         0       3    1  35.0      0      0   8.0500         2
```

Feature Engineering

```
In [ ]: # Create a new feature 'FamilySize'
data['FamilySize'] = data['SibSp'] + data['Parch'] + 1

# Create a new feature 'IsAlone'
data['IsAlone'] = 1
data.loc[data['FamilySize'] > 1, 'IsAlone'] = 0

# Drop SibSp and Parch columns
data.drop(['SibSp', 'Parch'], axis=1, inplace=True)

# Check the final dataset
print(data.head())
```

	Survived	Pclass	Sex	Age	Fare	Embarked	FamilySize	IsAlone
0	0	3	1	22.0	7.2500	2	2	0
1	1	1	0	38.0	71.2833	0	2	0
2	1	3	0	26.0	7.9250	2	1	1
3	1	1	0	35.0	53.1000	2	2	0
4	0	3	1	35.0	8.0500	2	1	1

Machine Learning Models

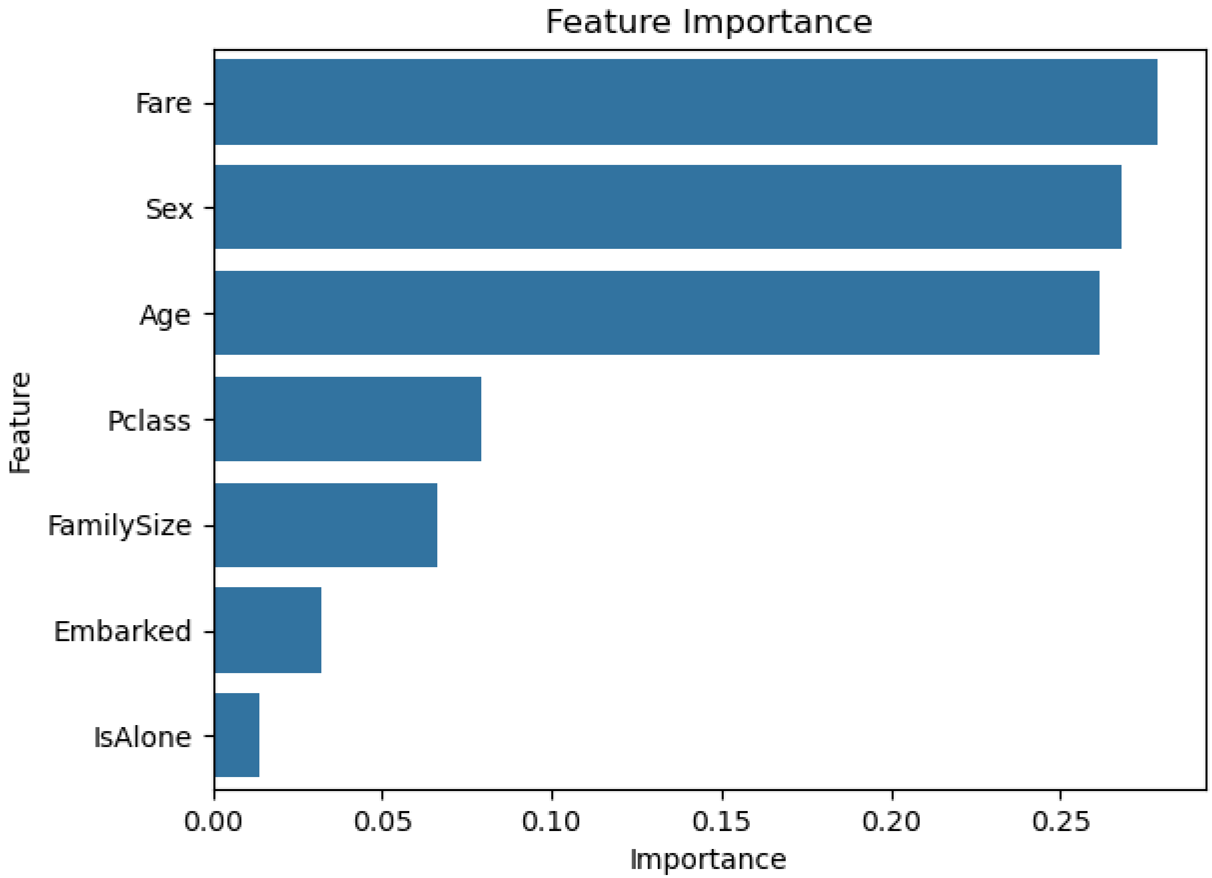
In [ ]:
```
#Split Data into Training and Testing Sets

# Define features and target
X = data.drop('Survived', axis=1)
y = data['Survived']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [ ]:
```
# Initialize the model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8324022346368715
[[91 14]
 [16 58]]
              precision    recall  f1-score   support

           0       0.85      0.87      0.86       105
           1       0.81      0.78      0.79        74

    accuracy                           0.83       179
   macro avg       0.83      0.83      0.83       179
weighted avg       0.83      0.83      0.83       179
```

In [ ]:

```python
# Get feature importances
importances = model.feature_importances_
feature_names = X.columns

# Create a DataFrame for visualization
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Plot feature importances
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance')
plt.show()
```



In [ ]:
```
# Save the model and scaler
joblib.dump(model, 'titanic_model.pkl')
joblib.dump(scaler, 'titanic_scaler.pkl')
print("Model and scaler saved to disk.")
```

```
Model and scaler saved to disk.
```