```
from google.colab import files
import pandas as pd

# Upload the file
uploaded = files.upload()
```

Choose Files Titanic-Dataset.csv

- **Titanic-Dataset.csv**(text/csv) - 61194 bytes, last modified: 12/24/2021 - 100% done

```
Saving Titanic-Dataset.csv to Titanic-Dataset.csv
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

```
titanic_df = pd.read_csv('Titanic-Dataset.csv')

# Display the first few rows of the dataset
titanic_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.
1	2	1	1	Cumings, Mrs. John Bradley (Florence	female	38.0	1	0	PC 17599	71.

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

```
titanic_df.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
```

```
# Drop columns that won't be used in the model
titanic_df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)

# Handle missing values
titanic_df['Age'].fillna(titanic_df['Age'].median(), inplace=True)
titanic_df['Embarked'].fillna(titanic_df['Embarked'].mode()[0], inplace=True)

# Convert categorical variables to numeric
titanic_df = pd.get_dummies(titanic_df, columns=['Sex', 'Embarked'], drop_first=Tru

# Display the processed dataframe
print(titanic_df.head())
```

```
   Survived  Pclass   Age  SibSp  Parch     Fare  Sex_male  Embarked_Q  \
0         0       3  22.0      1      0   7.2500      True       False
1         1       1  38.0      1      0  71.2833     False       False
2         1       3  26.0      0      0   7.9250     False       False
3         1       1  35.0      1      0  53.1000     False       False
4         0       3  35.0      0      0   8.0500      True       False

   Embarked_S
0        True
1       False
2        True
3        True
4        True
```

```
# Define features (X) and target (y)
X = titanic_df.drop('Survived', axis=1)
y = titanic_df['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_sta

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

```
# Initialize the model
model = LogisticRegression()

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

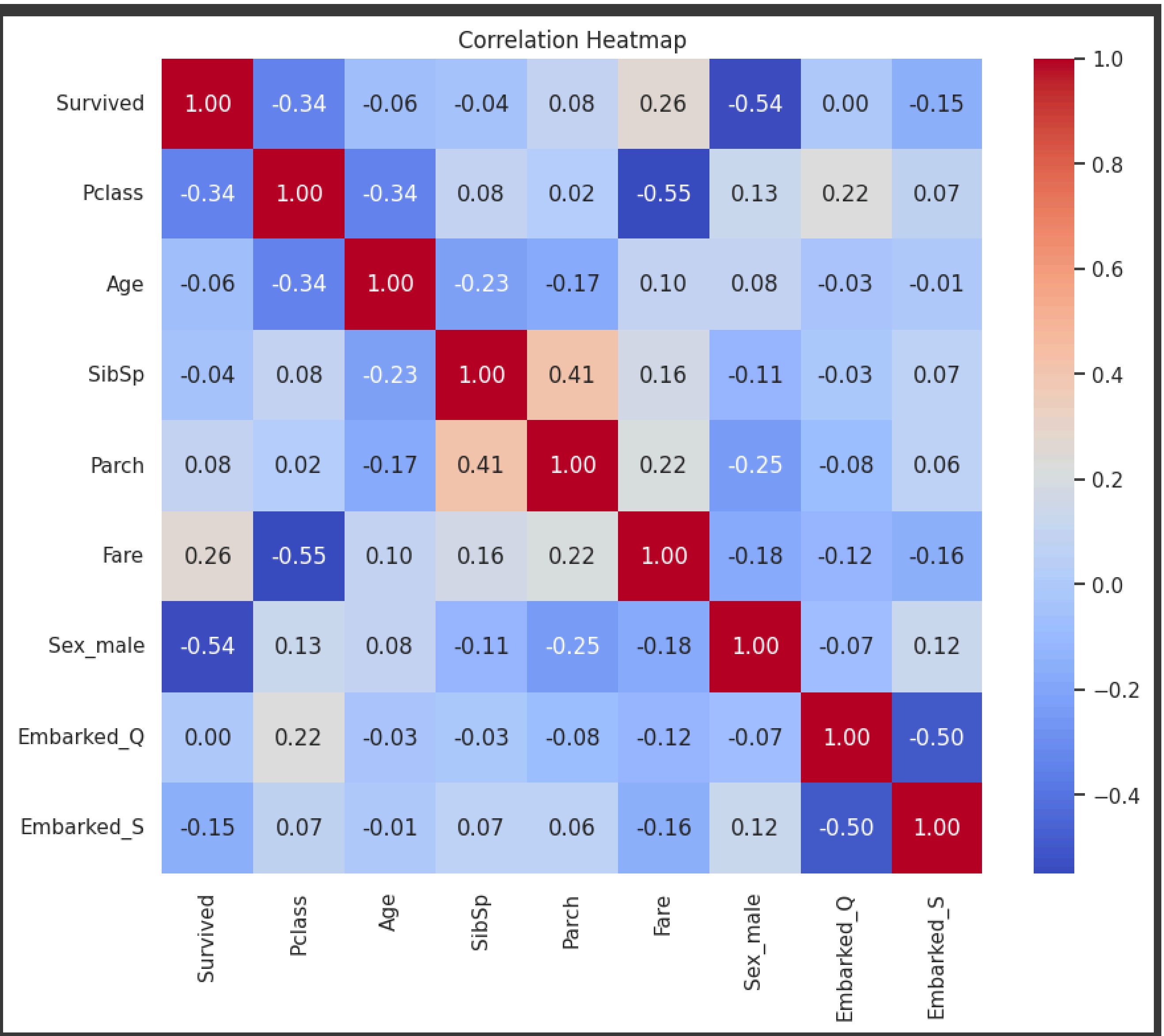
```
▾ LogisticRegression
LogisticRegression()
```

```
# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:\n", report)
```

```
Accuracy: 0.8100558659217877
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.86      0.84       105
           1       0.79      0.74      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
```

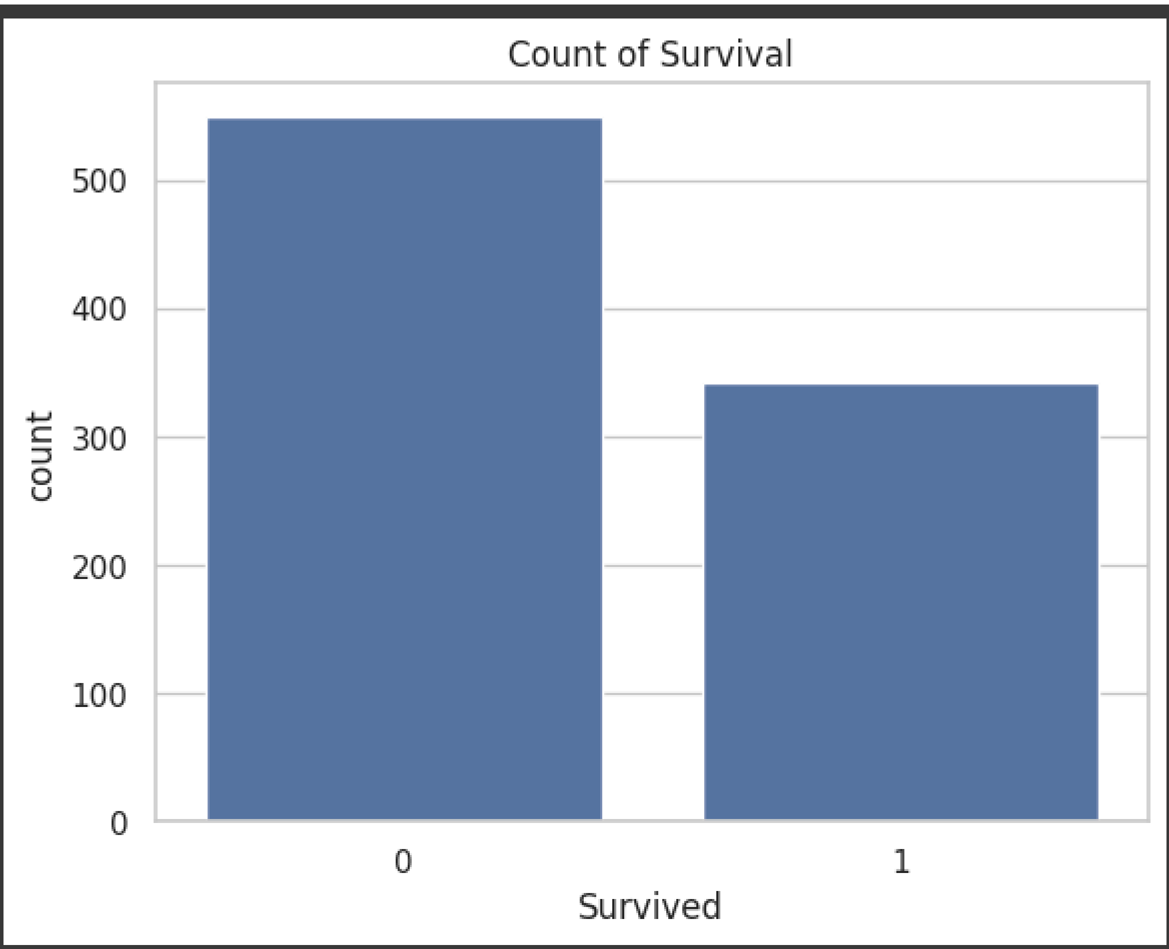
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import roc_curve, roc_auc_score, confusion_matrix

# Set the style for seaborn
sns.set(style="whitegrid")
```

```
# Plotting the correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(titanic_df.corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

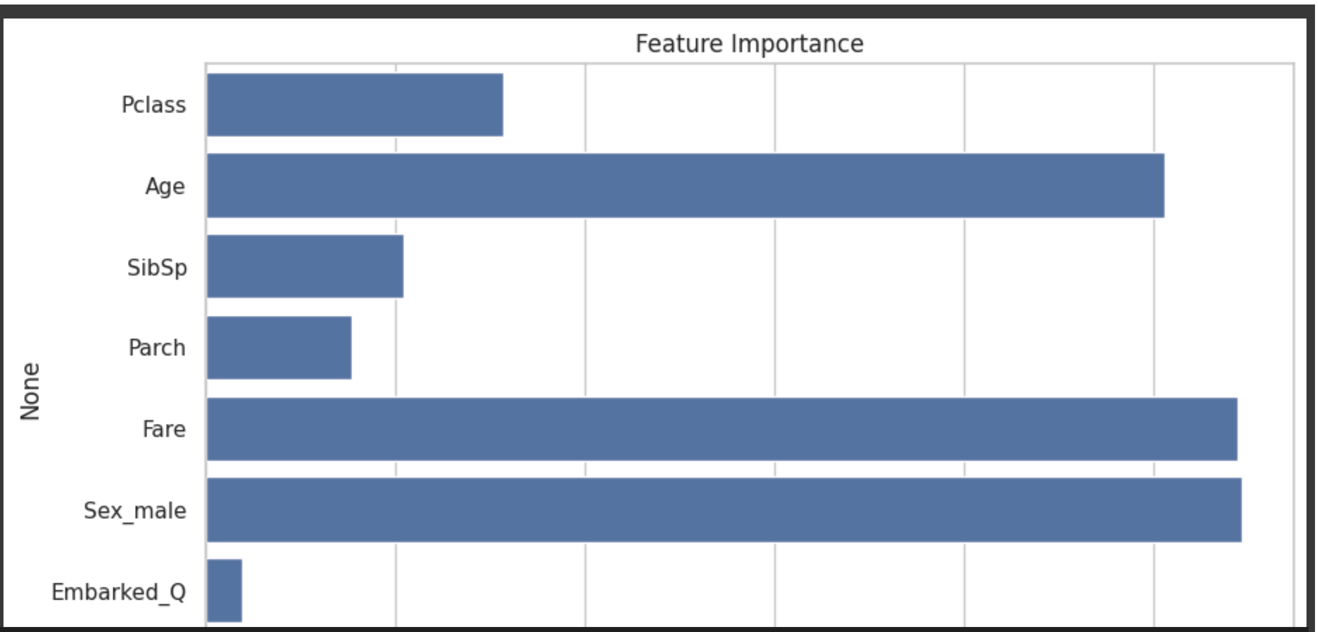


	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
Survived	1.00	-0.34	-0.06	-0.04	0.08	0.26	-0.54	0.00	-0.15
Pclass	-0.34	1.00	-0.34	0.08	0.02	-0.55	0.13	0.22	0.07
Age	-0.06	-0.34	1.00	-0.23	-0.17	0.10	0.08	-0.03	-0.01
SibSp	-0.04	0.08	-0.23	1.00	0.41	0.16	-0.11	-0.03	0.07
Parch	0.08	0.02	-0.17	0.41	1.00	0.22	-0.25	-0.08	0.06
Fare	0.26	-0.55	0.10	0.16	0.22	1.00	-0.18	-0.12	-0.16
Sex_male	-0.54	0.13	0.08	-0.11	-0.25	-0.18	1.00	-0.07	0.12
Embarked_Q	0.00	0.22	-0.03	-0.03	-0.08	-0.12	-0.07	1.00	-0.50
Embarked_S	-0.15	0.07	-0.01	0.07	0.06	-0.16	0.12	-0.50	1.00

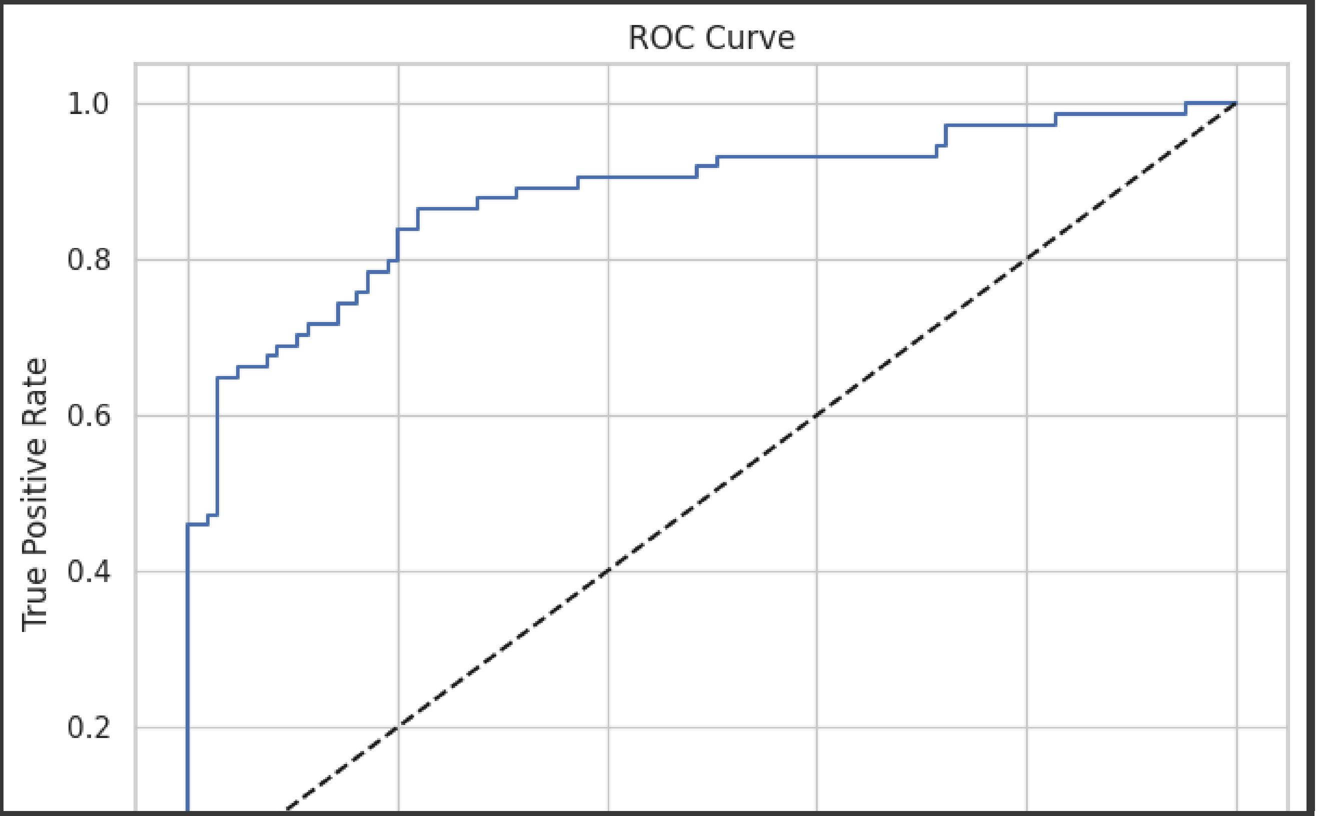
```python
# Plotting the survival count
sns.countplot(x='Survived', data=titanic_df)
plt.title("Count of Survival")
plt.show()
```



```python
from sklearn.ensemble import RandomForestClassifier

# Train a Random Forest model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Get feature importances
importances = rf_model.feature_importances_
feature_names = X.columns

# Plot feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x=importances, y=feature_names)
plt.title("Feature Importance")
plt.show()
```



```
# Calculate ROC curve and AUC
y_pred_prob = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = roc_auc_score(y_test, y_pred_prob)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"ROC Curve (area = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend(loc="lower right")
plt.show()
```


ROC Curve
True Positive Rate