ML Assignment 1

Perform data preprocessing and exploratory data analysis on a given dataset to prepare it for machine learning modeling.

In [21]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

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

Load CSV

In [22]:
```
df = pd.read_csv("D:/Machine Learning/titanic - titanic.csv")
df.head()
```

Out[22]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500

Data Understanding

In [23]:
```
df. shape
```

Out[23]: (887, 8)

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 887 entries, 0 to 886
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Survived                  887 non-null    int64  
 1   Pclass                    887 non-null    int64  
 2   Name                      887 non-null    object 
 3   Sex                       887 non-null    object 
 4   Age                       887 non-null    float64
 5   Siblings/Spouses Aboard   887 non-null    int64  
 6   Parents/Children Aboard   887 non-null    int64  
 7   Fare                      887 non-null    float64
dtypes: float64(2), int64(4), object(2)
memory usage: 55.6+ KB
```

In [25]:
```
df.describe()
```

Out[25]:

	Survived	Pclass	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
count	887.000000	887.000000	887.000000	887.000000	887.000000	887.00000
mean	0.385569	2.305524	29.471443	0.525366	0.383315	32.30542
std	0.487004	0.836662	14.121908	1.104669	0.807466	49.78204
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000
25%	0.000000	2.000000	20.250000	0.000000	0.000000	7.92500
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.13750
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

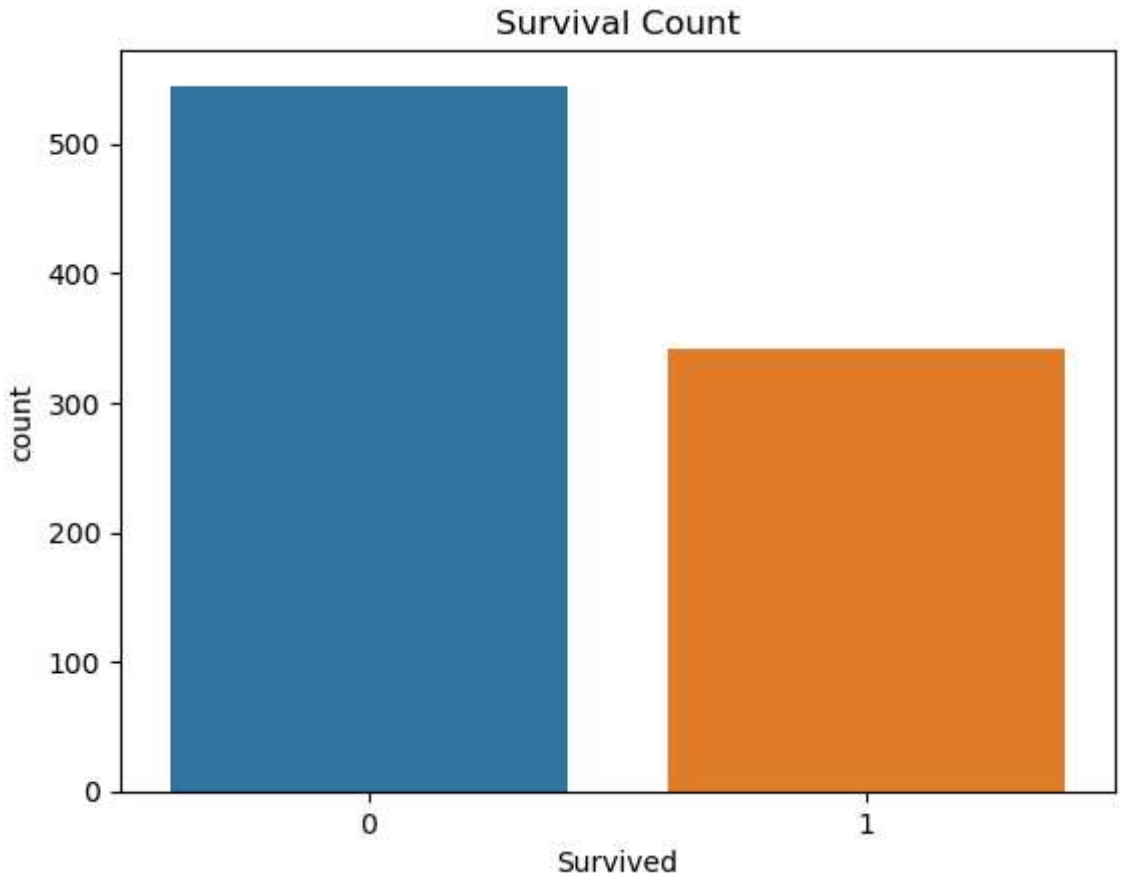
In [26]:
```
# Check null values
df.isnull().sum()
```

Out[26]:
```
Survived                  0
Pclass                    0
Name                      0
Sex                       0
Age                       0
Siblings/Spouses Aboard   0
Parents/Children Aboard   0
Fare                      0
dtype: int64
```

Exploratory Data Analysis

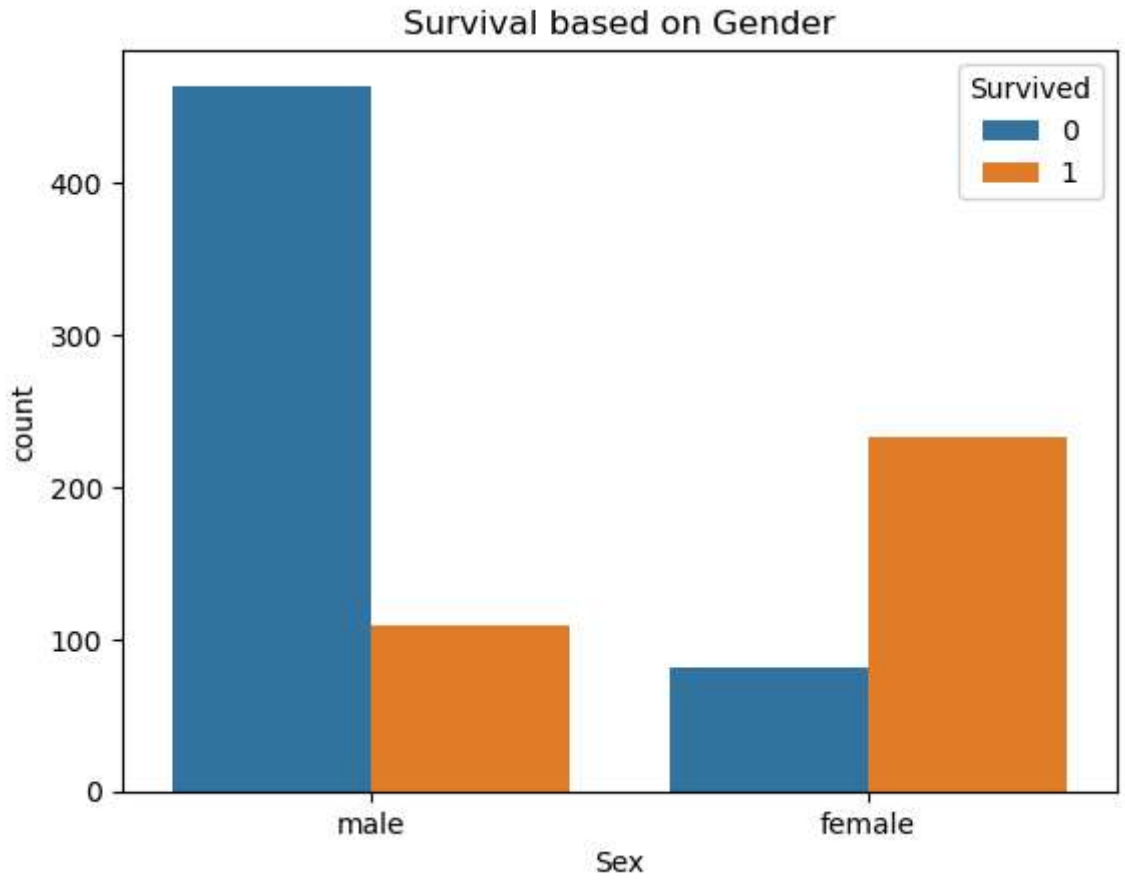
1. Target value distribution

In [27]:
```
sns.countplot(x='Survived', data=df)
plt.title("Survival Count")
plt.show()
```



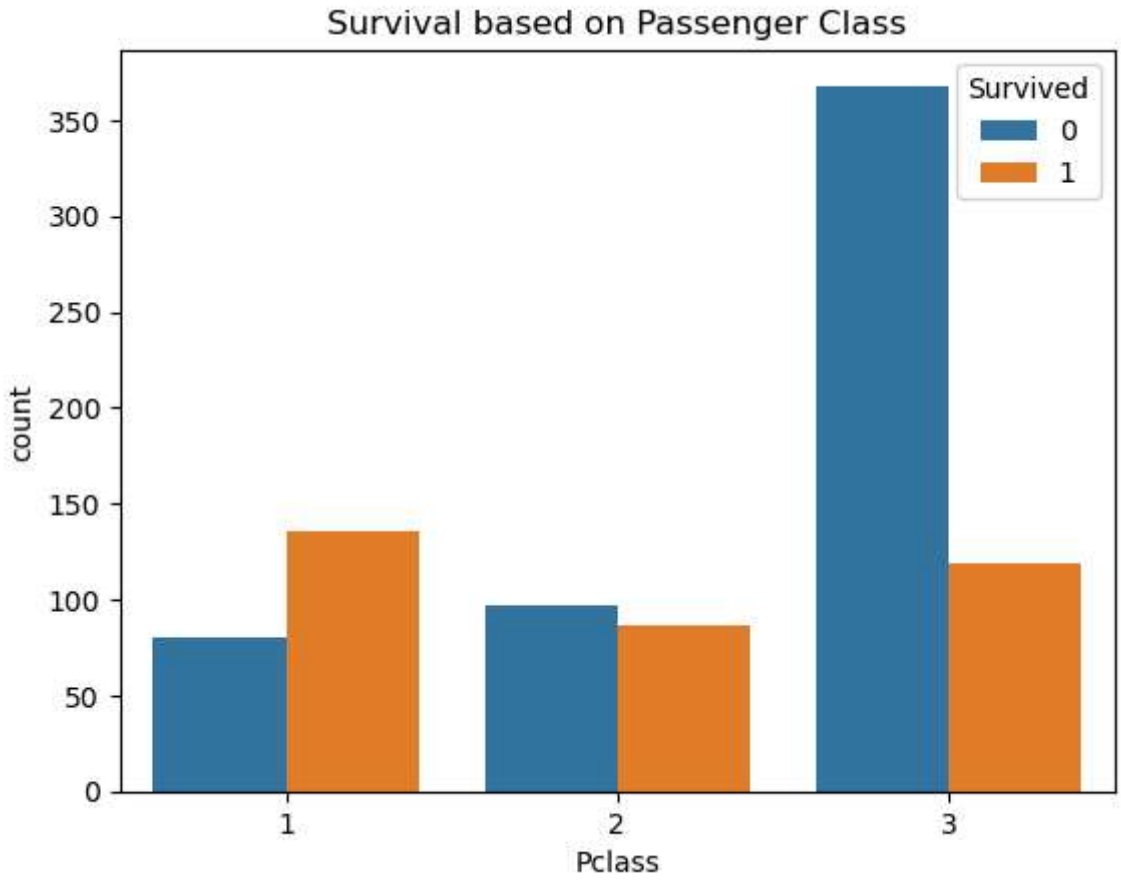
2. Survival VS Gender

In [28]:
```
sns.countplot(x='Sex', hue='Survived', data=df)
plt.title("Survival based on Gender")
plt.show()
```



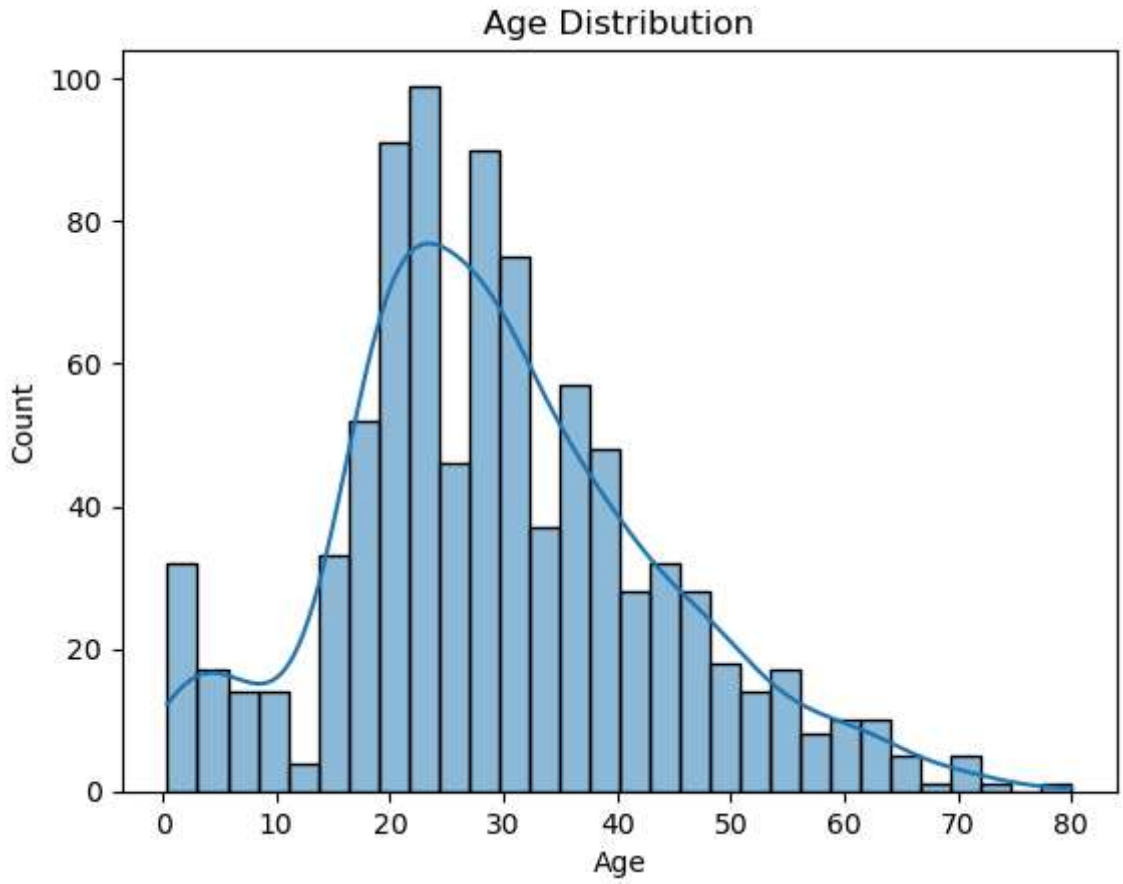
3. Survival VS Passenger Class

In [29]:
```
sns.countplot(x='Pclass', hue='Survived', data=df)
plt.title("Survival based on Passenger Class")
plt.show()
```



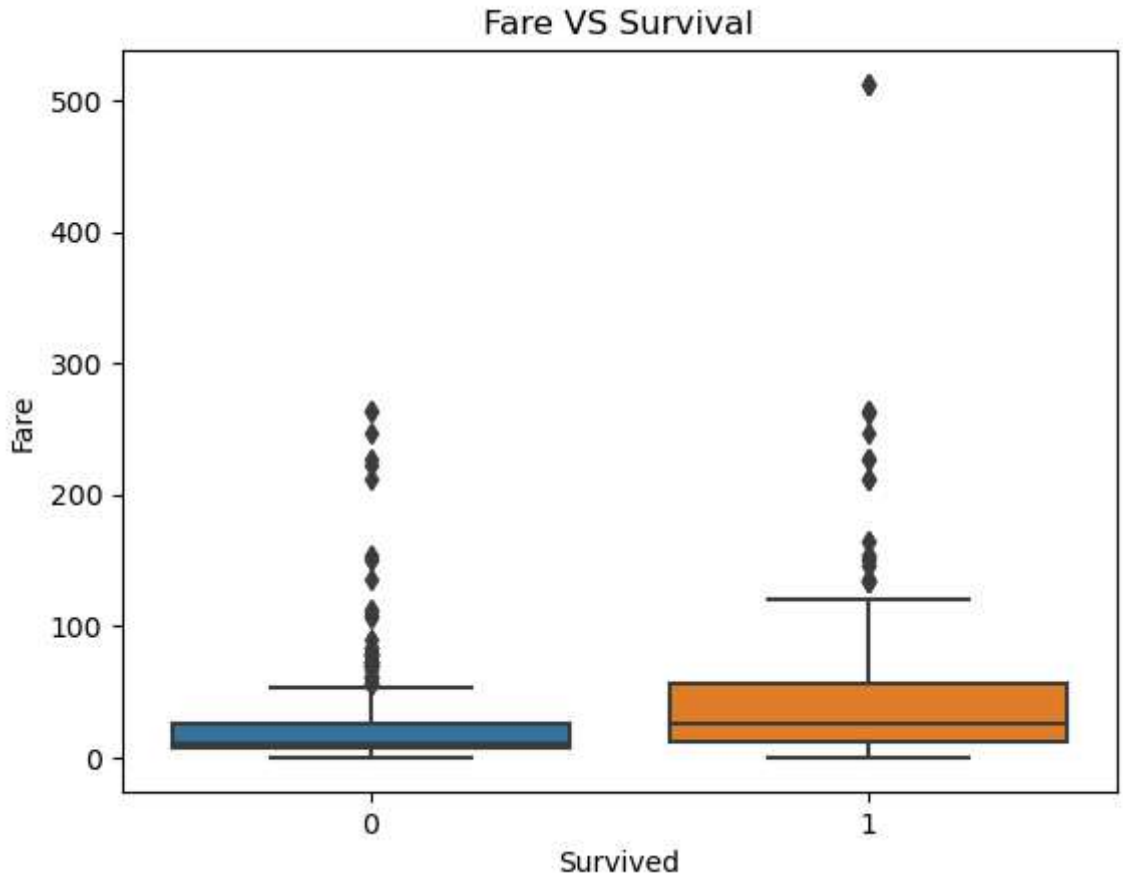
4. Age Distribution

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



5. Fare VS Survival

In [33]:
```
sns.boxplot(x='Survived', y='Fare', data=df)
plt.title("Fare VS Survival")
plt.show()
```



Data Preprocessing

1. Drop unnecessary Columns

In [36]:
```
# Remove name as it won't help directly for prediction
df.drop(columns=['Name'], inplace=True)
```

2. Drop Missing Values

There are no missing values

3. Encode Categorical Variables

Sex (Male / Female)

In [38]:
```
le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])

# Male = 1, Female = 0
```

4. Feature and target split

In [40]:
```
X = df.drop('Survived', axis=1)
y = df['Survived']
```

5. Train-Test Split

In [41]:
```
X_train,  X_test,  y_train,  y_test = train_test_split(
    X,  y, test_size=0.2, random_state=42
)
```

6. Feature Scaling

In [42]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Model Training

Logistic Regression

In [43]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[43]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Model Evaluation

Predictions

In [45]:
```
y_pred = model.predict(X_test)
```

Accuracy

In [46]:
```
accuracy_score(y_test, y_pred)
```

Out[46]: 0.7471910112359551

Confusion Matrix

In [47]:
```
confusion_matrix(y_test, y_pred)
```

Out[47]:
```
array([[96, 15],
       [30, 37]], dtype=int64)
```

Classification Report

In [48]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.86      0.81       111
           1       0.71      0.55      0.62        67

    accuracy                           0.75       178
   macro avg       0.74      0.71      0.72       178
weighted avg       0.74      0.75      0.74       178
```

In [ ]:
```

```