S AVINASH GUPTA

21BCE7754

SLOT MORNING AI ML EXTERNSHIP

Assignment : Perform Data-preprocessing for Titanitc dataset.

 # Connecting the drive Throgh the following Syntax

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly
remount, call drive.mount("/content/drive/", force_remount=True).
```

 # Importing Nesscary Libraies

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

Specifying the OS path to save the files in the current locations

```
import os
os.chdir('/content/drive/MyDrive/Smart_bridge_AI_ML')
```

 # Reading the Dataset throught the following Syntax

```
dataset=pd.read_csv("/content/drive/MyDrive/Smart_bridge_AI_ML/
Datasets/Titanic-Dataset.csv")

dataset
```

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         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
..                                                 ...     ...   ...
...
886                              Montvila, Rev. Juozas    male  27.0
0
887                       Graham, Miss. Margaret Edith  female  19.0
0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN
1
889                              Behr, Mr. Karl Howell    male  26.0
0
890                                Dooley, Mr. Patrick    male  32.0
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
..     ...               ...      ...   ...      ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q
```

```
[891 rows x 12 columns]

dataset.shape # Specify the shape to find the Number of rows and Cols

(891, 12)

dataset.info() # Here using this we can find that whether it was
belong to caterogical or numeric

<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

dataset.describe() # here we can find deep info regarding dataset mean
median and correlation values.

       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
```

 # Checking NULL VALUES HERE

```
dataset.isnull().any() # using this we can findout whehther we have null values are not

PassengerId    False
Survived       False
Pclass         False
Name           False
Sex            False
Age             True
SibSp          False
Parch          False
Ticket         False
Fare           False
Cabin           True
Embarked        True
dtype: bool

dataset.isnull().sum() # if we have null values here we use this synatx to find out how many are there.

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
```

Handling Null values

```
numerical_features = ['Age'] # making them separate arrya to specify whether it belong to categorical aor numerical if catergorical means replacing with mode if it was numerical replce with mean.
categorical_features = ['Cabin', 'Embarked']

# Handle missing values for numerical features (replace with mean)
for feature in numerical_features:
```

```
    dataset[feature].fillna(dataset[feature].mean(), inplace=True)

# Handle missing values for categorical features (replace with mode)
for feature in categorical_features:
    mode_value = dataset[feature].mode()[0]  # Get the mode (most
frequent value)
    dataset[feature].fillna(mode_value, inplace=True)

dataset.isnull().sum() # Now we check again for null values here.

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

dataset.corr() # here we find the relation between the variable using
this values if it was postive and near to 1 it means highly related to
each other and vic versa range from -1 to 1

<ipython-input-33-c187c74d1e71>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  dataset.corr()

             PassengerId  Survived    Pclass       Age     SibSp
Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.033207 -0.057527 -
0.001652
Survived       -0.005007  1.000000 -0.338481 -0.069809 -0.035322
0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.331339  0.083081
0.018443
Age             0.033207 -0.069809 -0.331339  1.000000 -0.232625 -
0.179191
SibSp          -0.057527 -0.035322  0.083081 -0.232625  1.000000
0.414838
Parch          -0.001652  0.081629  0.018443 -0.179191  0.414838
1.000000
Fare            0.012658  0.257307 -0.549500  0.091566  0.159651
0.216225
```

```
                 Fare
PassengerId  0.012658
Survived     0.257307
Pclass      -0.549500
Age          0.091566
SibSp        0.159651
Parch        0.216225
Fare         1.000000
```

```
dataset.corr().Age.sort_values(ascending=False) # making them in ascending other to understand easy
```

```
<ipython-input-35-efeb0e235e56>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  dataset.corr().Age.sort_values(ascending=False)
```

```
Age            1.000000
Fare           0.091566
PassengerId    0.033207
Survived      -0.069809
Parch         -0.179191
SibSp         -0.232625
Pclass        -0.331339
Name: Age, dtype: float64
```

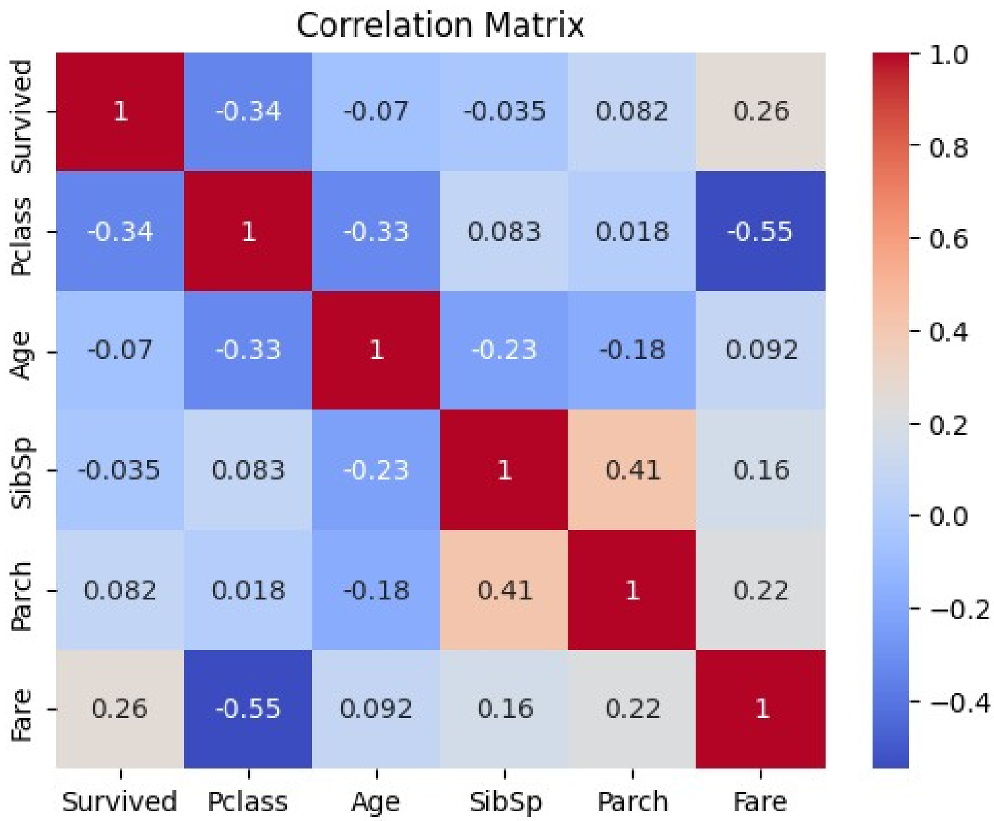
```
dataset.corr().PassengerId.sort_values(ascending=False)  # It seems like less Important and and its values also very small.
```

```
<ipython-input-37-9de62c94a563>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  dataset.corr().PassengerId.sort_values(ascending=False)
```

```
PassengerId    1.000000
Age            0.033207
Fare           0.012658
Parch         -0.001652
Survived      -0.005007
Pclass        -0.035144
SibSp         -0.057527
Name: PassengerId, dtype: float64
```

```
corr_matrix = dataset.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()  # visually representaion of correlration values
```

```
<ipython-input-45-b965ef0a5f6c>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  corr_matrix = dataset.corr()
```



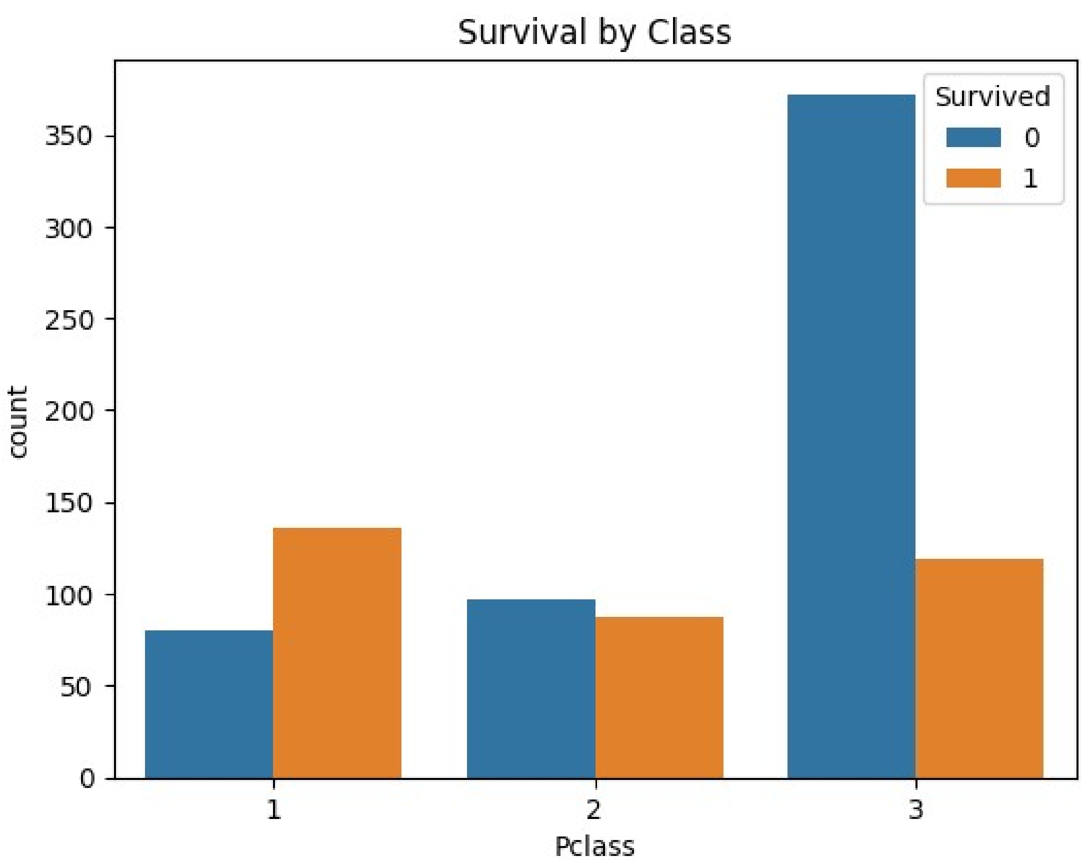

```
dataset = dataset.drop(columns=['PassengerId']) # becaauses less
correlated value to other variables

dataset.head()
```

```
   Survived  Pclass                                               Name
\
0         0       3                            Braund, Mr. Owen Harris

1         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...

2         1       3                             Heikkinen, Miss. Laina

3         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)

4         0       3                           Allen, Mr. William Henry
```

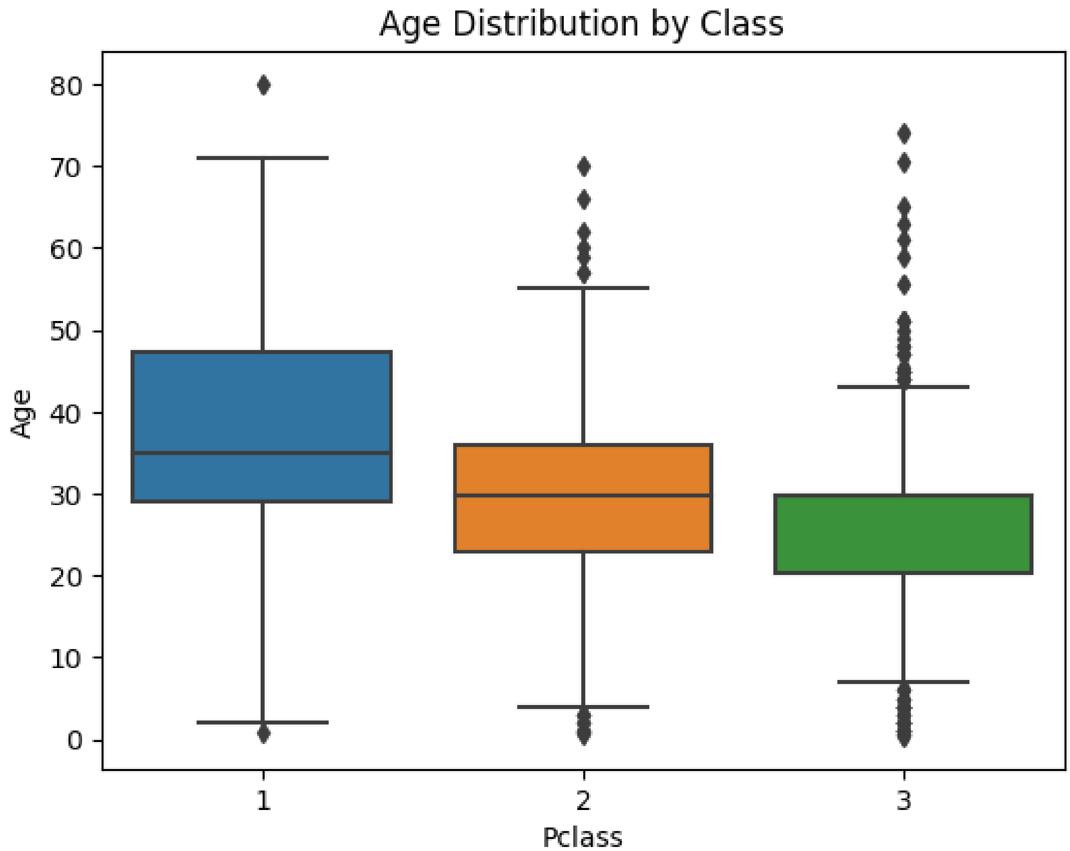
```
      Sex   Age  SibSp  Parch            Ticket     Fare    Cabin
Embarked
0    male  22.0      1      0         A/5 21171   7.2500  B96 B98
S
1  female  38.0      1      0          PC 17599  71.2833      C85
C
2  female  26.0      0      0  STON/O2. 3101282   7.9250  B96 B98
S
3  female  35.0      1      0            113803  53.1000     C123
S
4    male  35.0      0      0            373450   8.0500  B96 B98
S
```

Data Visualization

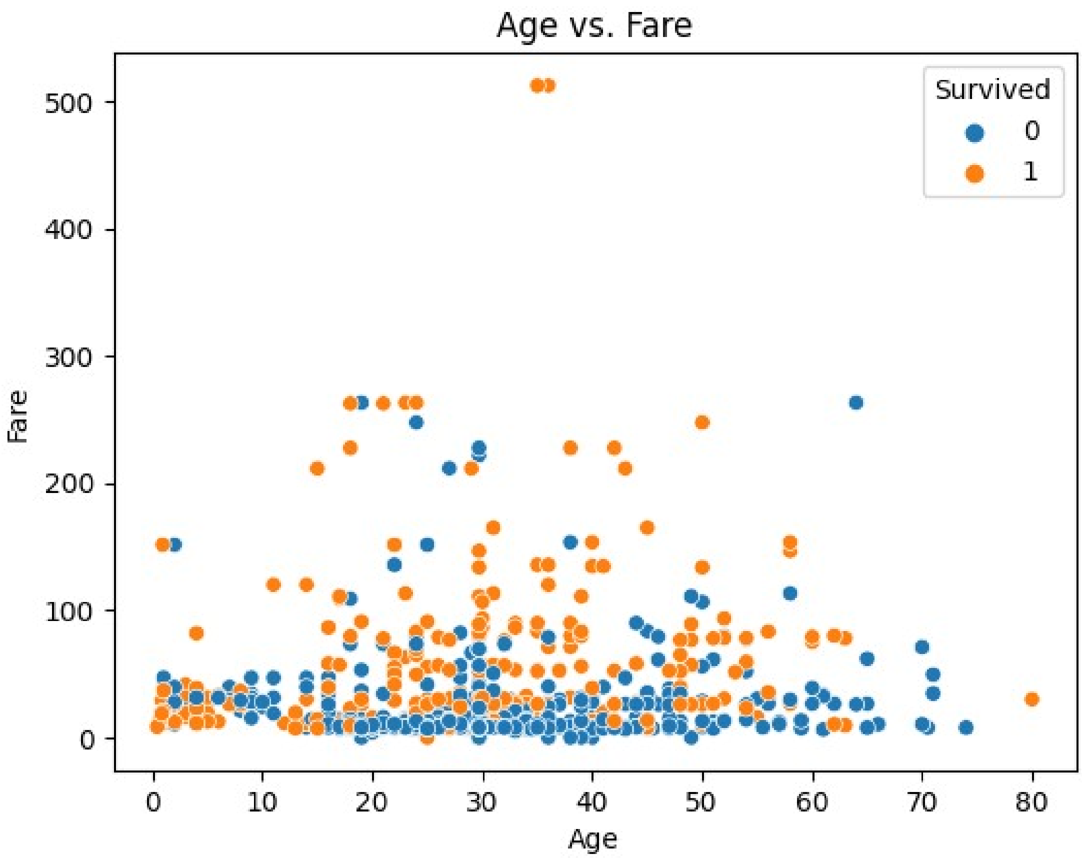
```
# here we are ploting the graph for the pclass to find out the surival
in each class it seem sthe more occurs in class 3 and less deaths
occurs in class 1
sns.countplot(data=dataset, x='Pclass', hue='Survived')
plt.title('Survival by Class')
plt.show()
```



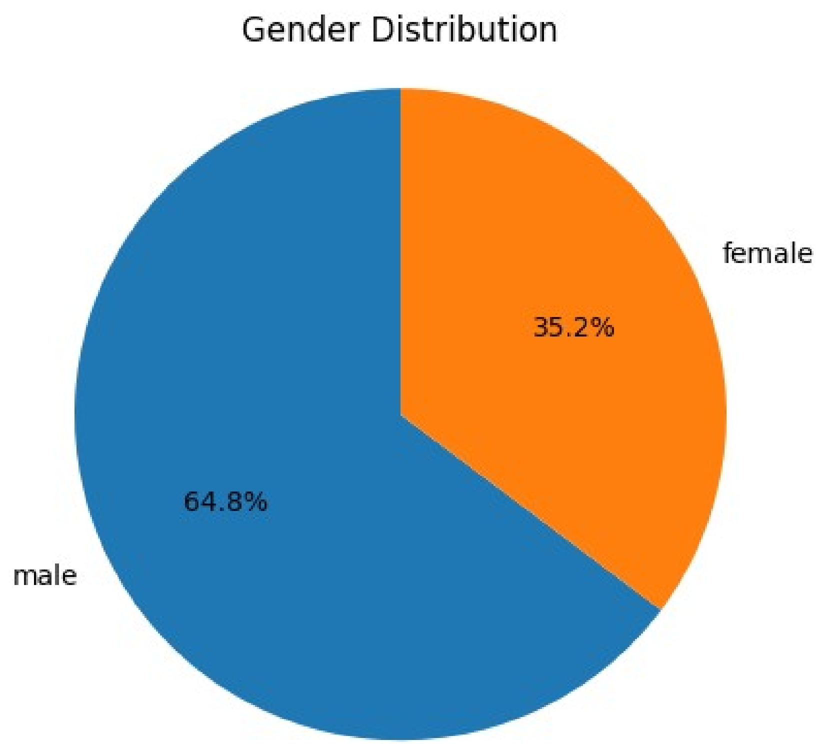
```
sns.boxplot(data=dataset, x='Pclass', y='Age')
plt.title('Age Distribution by Class')
plt.show()  # here we can see lot of outliers in this code.
```



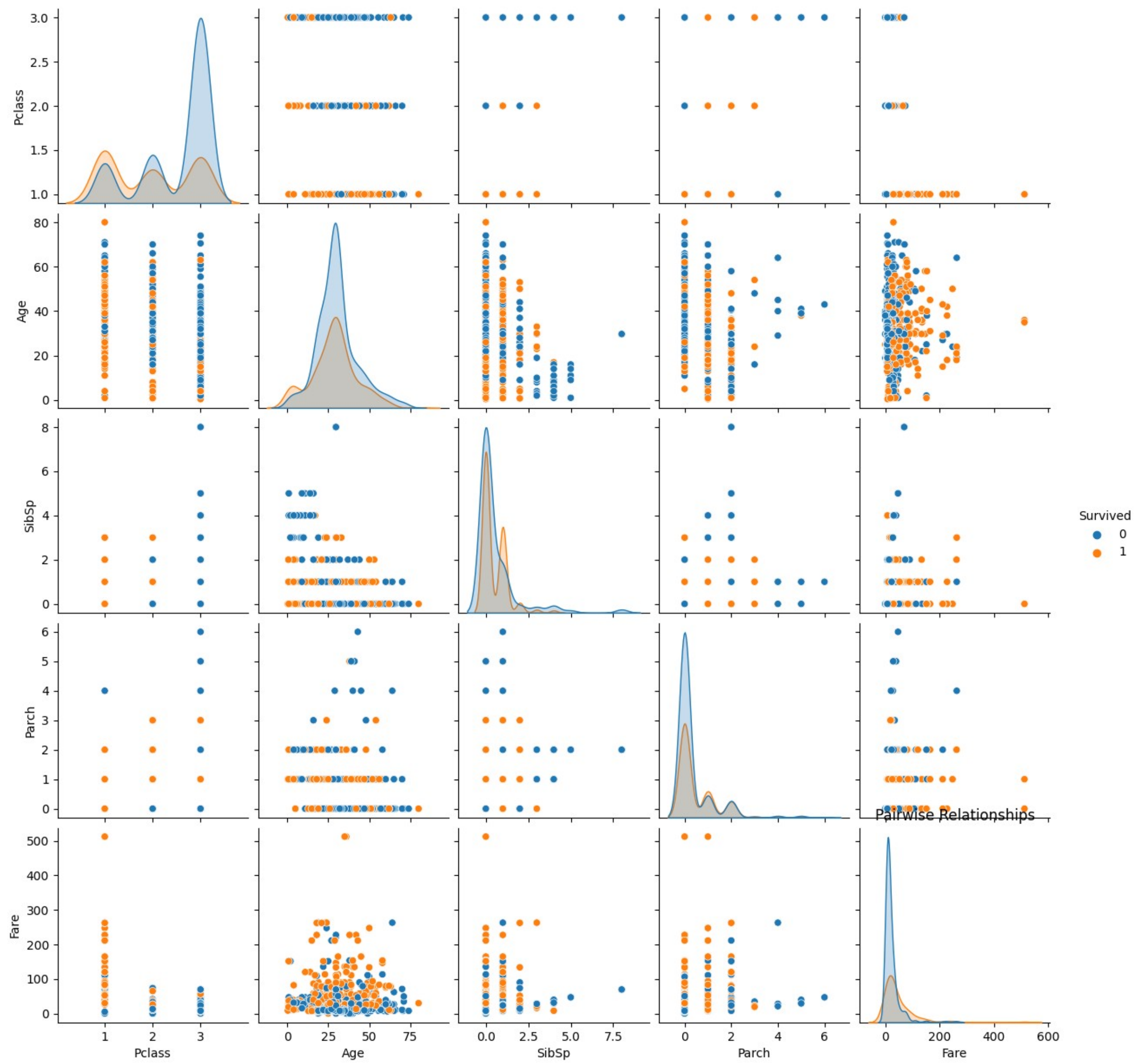

```
sns.scatterplot(data=dataset, x='Age', y='Fare', hue='Survived')
plt.title('Age vs. Fare')
plt.show() # here also we find the variation of age vs fare
```




```
gender_counts = dataset['Sex'].value_counts()
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%',
startangle=90)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a
circle.
plt.title('Gender Distribution')
plt.show()
```

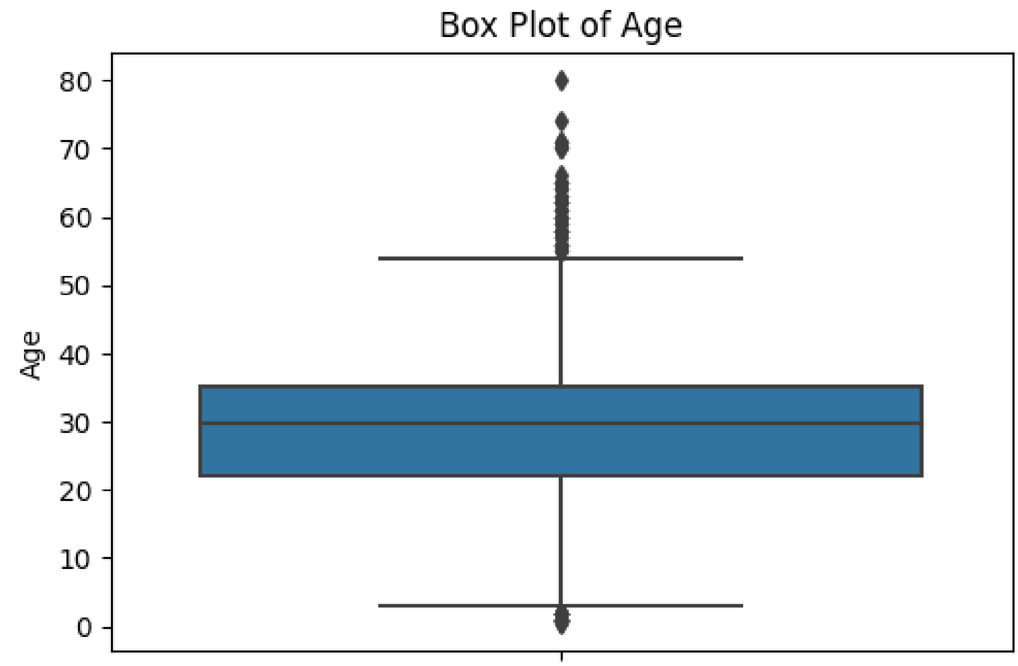



```
sns.pairplot(dataset, hue='Survived', diag_kind='kde')
plt.title('Pairwise Relationships')
plt.show()
```

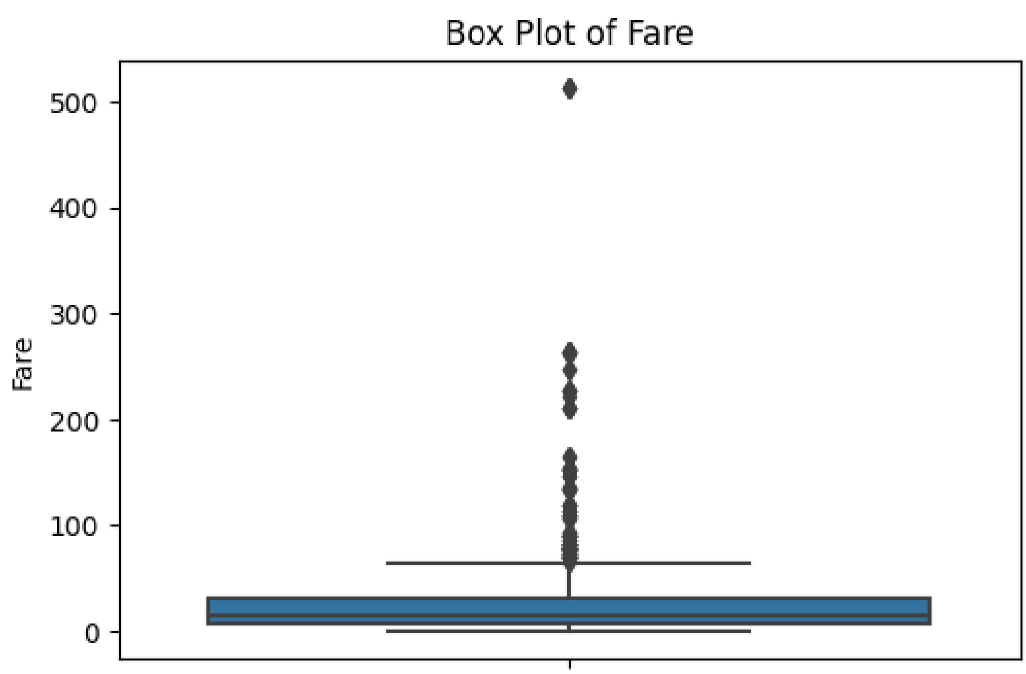


Outlier Detections

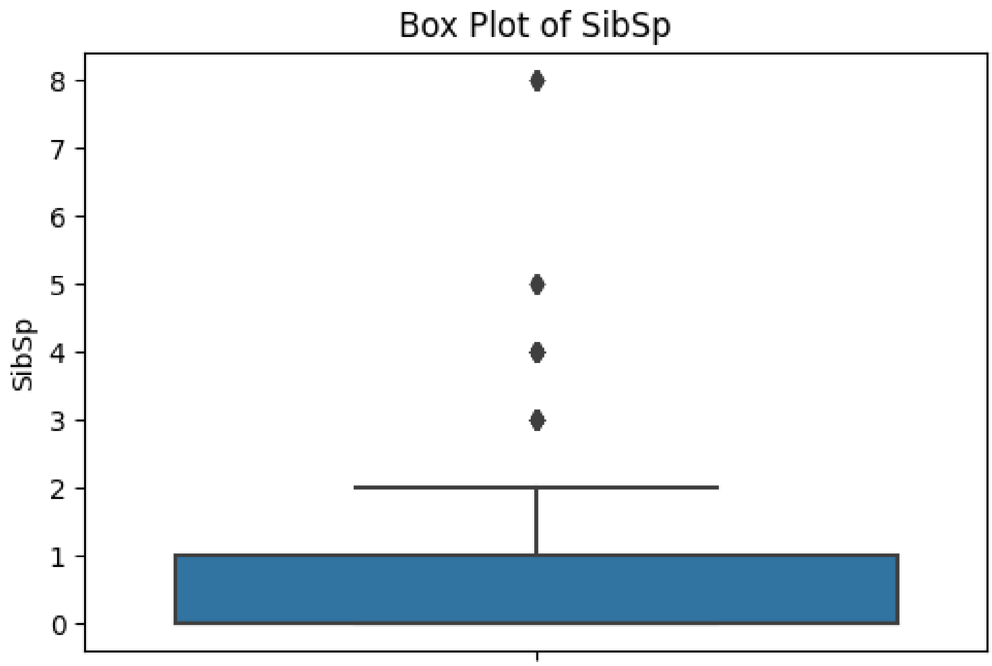
```
# Create a box plot for the 'Age' column
plt.figure(figsize=(6, 4)) # outliers are noticed here.
sns.boxplot(data=dataset, y='Age')
plt.title('Box Plot of Age')
plt.show()
```




```
# Create a box plot for the 'Fare' column
plt.figure(figsize=(6, 4))  # outliers are noticed here
sns.boxplot(data=dataset, y='Fare')
plt.title('Box Plot of Fare')
plt.show()
```

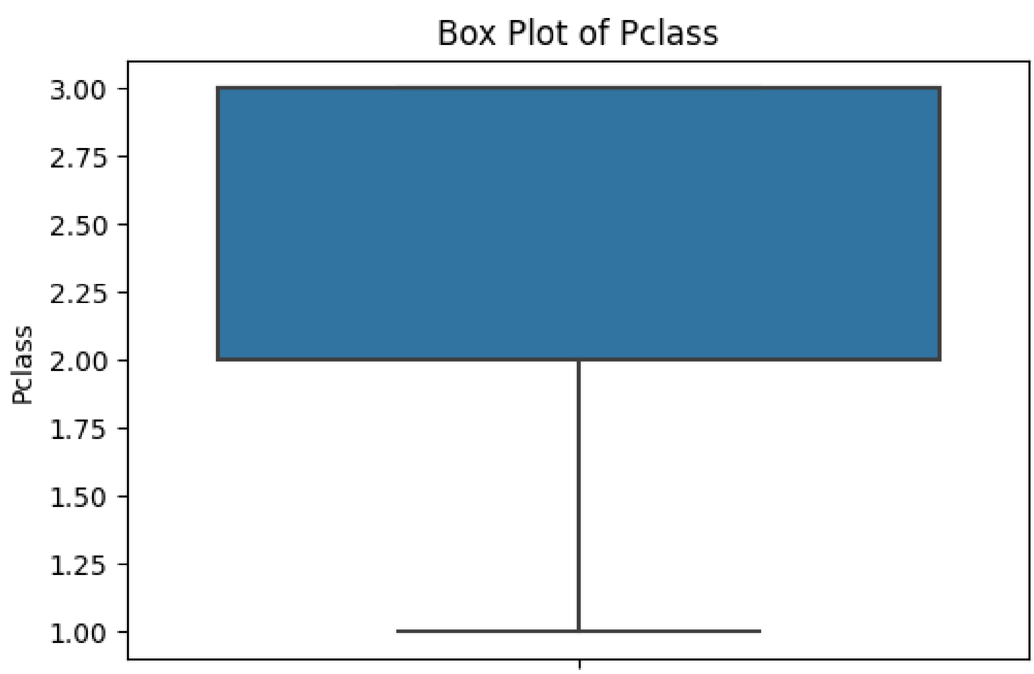


```
# Create a box plot for the 'SibSp' column
plt.figure(figsize=(6, 4))  # oulier are noticed here
sns.boxplot(data=dataset, y='SibSp')
plt.title('Box Plot of SibSp')
plt.show()
```




```
# Create a box plot for the 'Parch' column
plt.figure(figsize=(6, 4))  # ouliers are noticed here
sns.boxplot(data=dataset, y='Parch')
plt.title('Box Plot of Parch')
plt.show()
```


```
# Create a box plot for the 'Pclass' column
plt.figure(figsize=(6, 4))  # no outliers here
sns.boxplot(data=dataset, y='Pclass')
plt.title('Box Plot of Pclass')
plt.show()
```



HANDLING OUTLIERS

```
# Calculate the IQR for 'Age'
Q1_age = dataset['Age'].quantile(0.25)
Q3_age = dataset['Age'].quantile(0.75)
IQR_age = Q3_age - Q1_age

# Define a multiplier (e.g., 1.5 times the IQR)
multiplier = 1.5

# Define a condition to identify outliers
age_outlier_condition = (dataset['Age'] < (Q1_age - multiplier *
IQR_age)) | (dataset['Age'] > (Q3_age + multiplier * IQR_age))

# Remove outliers from the 'Age' column
df_cleaned = dataset[~age_outlier_condition]

# Verify the removal of outliers
print("Number of rows before removing outliers:", dataset.shape[0])
print("Number of rows after removing outliers:", df_cleaned.shape[0])
```

```
Number of rows before removing outliers: 891
Number of rows after removing outliers: 815
```

```
# Calculate the IQR for 'Fare'
Q1_Fare = dataset['Fare'].quantile(0.25)
Q3_Fare = dataset['Fare'].quantile(0.75)
IQR_Fare = Q3_Fare - Q1_Fare

# Define a multiplier (e.g., 1.5 times the IQR)
multiplier = 1.5

# Define a condition to identify outliers
Fare_outlier_condition = (dataset['Fare'] < (Q1_Fare - multiplier *
IQR_Fare)) | (dataset['Fare'] > (Q3_Fare + multiplier * IQR_Fare))

# Remove outliers from the 'Age' column
dataset = dataset[~Fare_outlier_condition]

# Verify the removal of outliers
print("Number of rows before removing outliers:", dataset.shape[0])
```

```
Number of rows before removing outliers: 775
```

```
# Calculate the IQR for 'SibSp'
Q1_SibSp = dataset['SibSp'].quantile(0.25)
Q3_SibSp = dataset['SibSp'].quantile(0.75)
IQR_SibSp = Q3_SibSp - Q1_SibSp

# Define a multiplier (e.g., 1.5 times the IQR)
multiplier = 1.5
```

```
# Define a condition to identify outliers
SibSp_outlier_condition = (dataset['SibSp'] < (Q1_SibSp - multiplier *
IQR_SibSp)) | (dataset['SibSp'] > (Q3_SibSp + multiplier * IQR_SibSp))

# Remove outliers from the 'Age' column
dataset = dataset[~SibSp_outlier_condition]

# Verify the removal of outliers
print("Number of rows before removing outliers:", dataset.shape[0])
```

```
Number of rows before removing outliers: 739
```

```
# Calculate the IQR for 'Parch'
Q1_Parch = dataset['Parch'].quantile(0.25)
Q3_Parch= dataset['Parch'].quantile(0.75)
IQR_Parch = Q3_Parch - Q1_Parch

# Define a multiplier (e.g., 1.5 times the IQR)
multiplier = 1.5

# Define a condition to identify outliers
Parch_outlier_condition = (dataset['Parch'] < (Q1_Parch - multiplier *
IQR_Parch)) | (dataset['Parch'] > (Q3_Parch + multiplier * IQR_Parch))

# Remove outliers from the 'Age' column
dataset = dataset[~Parch_outlier_condition]

# Verify the removal of outliers
print("Number of rows before removing outliers:", dataset.shape[0])
```

```
Number of rows before removing outliers: 607
```

Splitting Dataset Like Dependent and Independent variables

```
# Define the dependent variable (target)
y = dataset['Survived'] # IT IS ALWAYS IN SERAIL FORM

# Define the independent variables (features)
X = dataset.drop(columns=['Survived'],axis=1)  # 2-D ARRAY

y.head()
```

```
0    0
2    1
```

```
3    1
4    0
5    0
Name: Survived, dtype: int64
```

```
X.head()
```

```
   Pclass                                          Name     Sex
Age  \
0       3                            Braund, Mr. Owen Harris    male
22.000000
2       3                             Heikkinen, Miss. Laina  female
26.000000
3       1  Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
35.000000
4       3                           Allen, Mr. William Henry    male
35.000000
5       3                                   Moran, Mr. James    male
29.699118

   SibSp  Parch            Ticket     Fare    Cabin Embarked
0      1      0         A/5 21171   7.2500  B96 B98        S
2      0      0  STON/O2. 3101282   7.9250  B96 B98        S
3      1      0            113803  53.1000     C123        S
4      0      0            373450   8.0500  B96 B98        S
5      0      0            330877   8.4583  B96 B98        Q
```

```
# here For Name,Ticket,Cabin Have the uniuq values we know and we also
know that this cols arent that much corrletated that much
# So we are Removing those cols . because of contains more null
values.

X = dataset.drop(columns=['Ticket', 'Cabin', 'Name'])

X
```

```
     Survived  Pclass     Sex        Age  SibSp  Parch     Fare
Embarked
0           0       3    male  22.000000      1      0   7.2500
S
2           1       3  female  26.000000      0      0   7.9250
S
3           1       1  female  35.000000      1      0  53.1000
S
4           0       3    male  35.000000      0      0   8.0500
S
5           0       3    male  29.699118      0      0   8.4583
Q
..        ...     ...     ...        ...    ...    ...      ...       .
..
884         0       3    male  25.000000      0      0   7.0500
```

```
S
886          0         2     male  27.000000       0       0  13.0000
S
887          1         1   female  19.000000       0       0  30.0000
S
889          1         1     male  26.000000       0       0  30.0000
C
890          0         3     male  32.000000       0       0   7.7500
Q

[607 rows x 8 columns]
```

#Perform Encoding to change the catergorical values to Numerical vaules

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

X["Sex"]=le.fit_transform(X["Sex"])

X

     Survived  Pclass  Sex        Age  SibSp  Parch     Fare Embarked
0           0       3    1  22.000000      1      0   7.2500        S
2           1       3    0  26.000000      0      0   7.9250        S
3           1       1    0  35.000000      1      0  53.1000        S
4           0       3    1  35.000000      0      0   8.0500        S
5           0       3    1  29.699118      0      0   8.4583        Q
..        ...     ...  ...        ...    ...    ...      ...      ...
884         0       3    1  25.000000      0      0   7.0500        S
886         0       2    1  27.000000      0      0  13.0000        S
887         1       1    0  19.000000      0      0  30.0000        S
889         1       1    1  26.000000      0      0  30.0000        C
890         0       3    1  32.000000      0      0   7.7500        Q

[607 rows x 8 columns]

X["Embarked"]=le.fit_transform(X["Embarked"])

X

     Survived  Pclass  Sex        Age  SibSp  Parch     Fare  Embarked
0           0       3    1  22.000000      1      0   7.2500         2
2           1       3    0  26.000000      0      0   7.9250         2
3           1       1    0  35.000000      1      0  53.1000         2
4           0       3    1  35.000000      0      0   8.0500         2
5           0       3    1  29.699118      0      0   8.4583         1
..        ...     ...  ...        ...    ...    ...      ...       ...
884         0       3    1  25.000000      0      0   7.0500         2
886         0       2    1  27.000000      0      0  13.0000         2
887         1       1    0  19.000000      0      0  30.0000         2
```

```
889          1        1    1  26.000000      0      0  30.0000          0
890          0        3    1  32.000000      0      0   7.7500          1

[607 rows x 8 columns]
```

Performing the Feature Scaling here where to make them equal measure while calcuting

```python
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()

X_Scaled=pd.DataFrame(ms.fit_transform(X),columns=X.columns)

X_Scaled
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0.0	1.0	1.0	0.346939	0.5	0.0	0.118512	1.0
1	1.0	1.0	0.0	0.428571	0.0	0.0	0.129546	1.0
2	1.0	0.0	0.0	0.612245	0.5	0.0	0.868002	1.0
3	0.0	1.0	1.0	0.612245	0.0	0.0	0.131590	1.0
4	0.0	1.0	1.0	0.504064	0.0	0.0	0.138264	0.5
..	...	...	...	...	...	...	...	...
602	0.0	1.0	1.0	0.408163	0.0	0.0	0.115243	1.0
603	0.0	0.5	1.0	0.448980	0.0	0.0	0.212505	1.0
604	1.0	0.0	0.0	0.285714	0.0	0.0	0.490396	1.0
605	1.0	0.0	1.0	0.428571	0.0	0.0	0.490396	0.0
606	0.0	1.0	1.0	0.551020	0.0	0.0	0.126686	0.5

[607 rows x 8 columns]

Spliting Dataset into Train and Test for futher evalution

```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X_Scaled,y,test_size=0.
2,random_state=0)

print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(485, 8) (122, 8) (485,) (122,)
```