vit-si-assignment-3

September 20, 2023

0.1 Titanic Dataset

What has been done :

- Checking Null Values
- Data Visualization
- Outliers
- Splitting into Dependent and Independent
- Encoding
- Train and Test Splitting

Dataset : https://www.kaggle.com/datasets/yasserh/titanic-dataset

Done By : RISHIKA SAHOO 21BCB0184

0.1.1 Import Libraries

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

0.1.2 Importing the dataset

[3]:
```
df = pd.read_csv("/content/Titanic-Dataset.csv")
```

[4]:
```
df.head()
```

[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0
3	4	1	1				
4	5	0	3				

```
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
```

[5]:
```
df.tail()
```

[5]:
```
     PassengerId  Survived  Pclass                                          Name  \
886          887         0       2                         Montvila, Rev. Juozas
887          888         1       1                  Graham, Miss. Margaret Edith
888          889         0       3      Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                          Behr, Mr. Karl Howell
890          891         0       3                            Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

0.1.3 About Dataset:

Surived : Survied Or Not (0->Not survived , 1->Survived)

PassengerId: An unique identifier for each passenger.

Pclass: The ticket class (1 = 1st, 2 = 2nd, 3 = 3rd).

Name: The name of the passenger.

Sex: The sex of the passenger.

Age: The age of the passenger.

SibSp: The number of siblings/spouses aboard.

Parch: The number of parents/children aboard.

Ticket: The ticket number.

Fare: The passenger fare.

Cabin: The cabin number.

Embarked: The port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

[6]:
```
df.shape
```

[6]: (891, 12)

[7]:
```
df.dtypes
```

[7]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

[8]:
```
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
```

[12]:
```
df.describe()
```

[12]:

	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
```

[9]:
```
df.describe().T
```

[9]:
```
             count        mean         std   min       25%       50%    75%  \
PassengerId  891.0  446.000000  257.353842  1.00  223.5000  446.0000  668.5
Survived     891.0    0.383838    0.486592  0.00    0.0000    0.0000    1.0
Pclass       891.0    2.308642    0.836071  1.00    2.0000    3.0000    3.0
Age          714.0   29.699118   14.526497  0.42   20.1250   28.0000   38.0
SibSp        891.0    0.523008    1.102743  0.00    0.0000    0.0000    1.0
Parch        891.0    0.381594    0.806057  0.00    0.0000    0.0000    0.0
Fare         891.0   32.204208   49.693429  0.00    7.9104   14.4542   31.0

                max
PassengerId  891.0000
Survived       1.0000
Pclass         3.0000
Age           80.0000
SibSp          8.0000
Parch          6.0000
Fare         512.3292
```

0.1.4 Null Values

[10]:
```
df.isnull().any()
```

[10]:
```
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
```

[11]:
```
df.isnull().sum()
```

[11]:
```
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
```

[13]:
```
df.drop("Cabin",axis=1,inplace=True)
```

[16]:
```
median_age = df["Age"].median()
df["Age"] = df["Age"].fillna(median_age)
```

[17]:
```
mode_em = df["Embarked"].mode()
df["Embarked"] = df["Embarked"].fillna(mode_em[0])
```

[18]:
```
df.isnull().sum()
```

[18]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

[ ]:
```
df.describe().T
```

[ ]:

	count	mean	std	min	25%	50%	75%	max
PassengerId	891.0	446.000000	257.353842	1.00	223.5000	446.0000	668.5	891.0000
Survived	891.0	0.383838	0.486592	0.00	0.0000	0.0000	1.0	1.0000
Pclass	891.0	2.308642	0.836071	1.00	2.0000	3.0000	3.0	3.0000
Age	891.0	29.361582	13.019697	0.42	22.0000	28.0000	35.0	80.0000
SibSp	891.0	0.523008	1.102743	0.00	0.0000	0.0000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.00	0.0000	0.0000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.00	7.9104	14.4542	31.0	512.3292

[ ]:
```
df.Sex.value_counts()
```

[ ]:
```
male      577
female    314
Name: Sex, dtype: int64
```

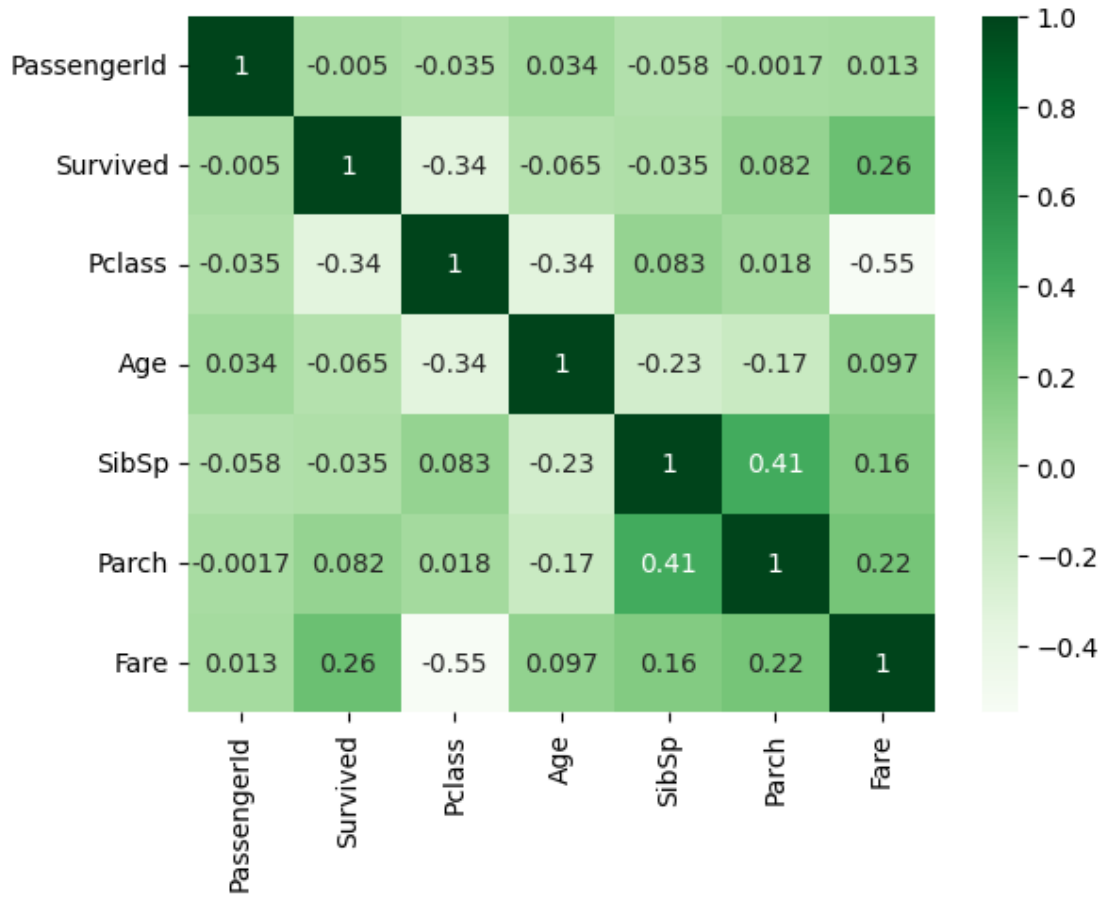
[ ]:
```
df.Survived.value_counts()
```

[ ]:
```
0    549
1    342
Name: Survived, dtype: int64
```

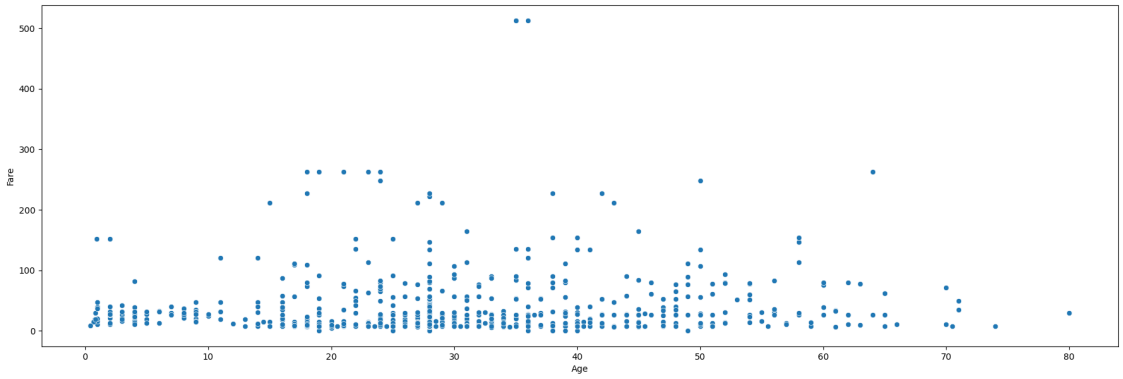
0.1.5 Data Visualization

[23]:
```
sns.heatmap(df.corr(numeric_only=True),annot=True,cmap="Greens")
```

[23]:
```
<Axes: >
```

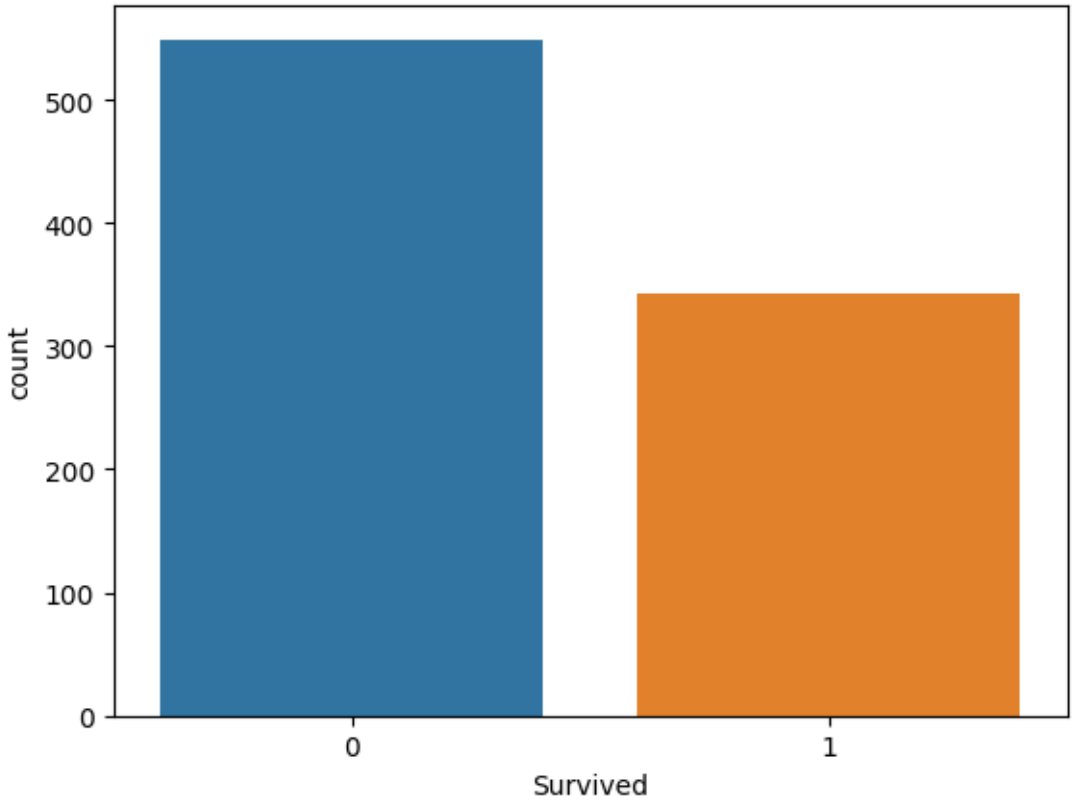


[30]:
```
plt.figure(figsize=(22, 7))
sns.scatterplot(x="Age",y="Fare",data=df)
plt.show()
```



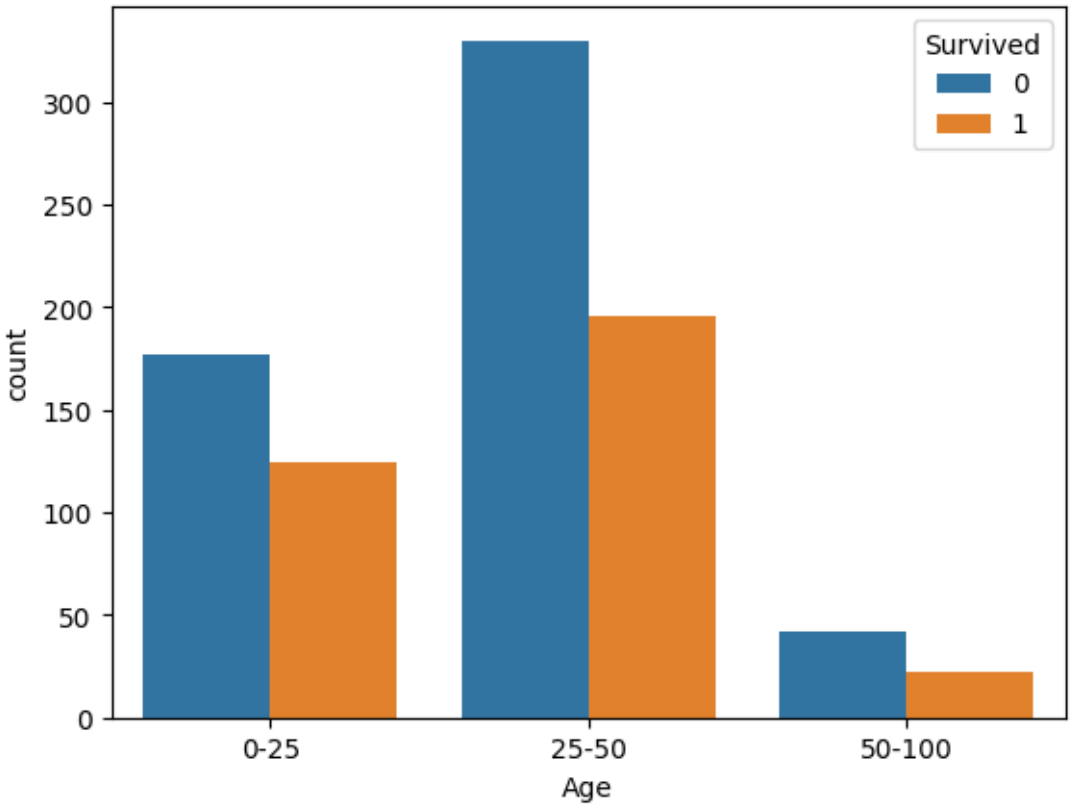
[31]:
```
sns.countplot(x="Survived",data=df)
```

[31]: <Axes: xlabel='Survived', ylabel='count'>



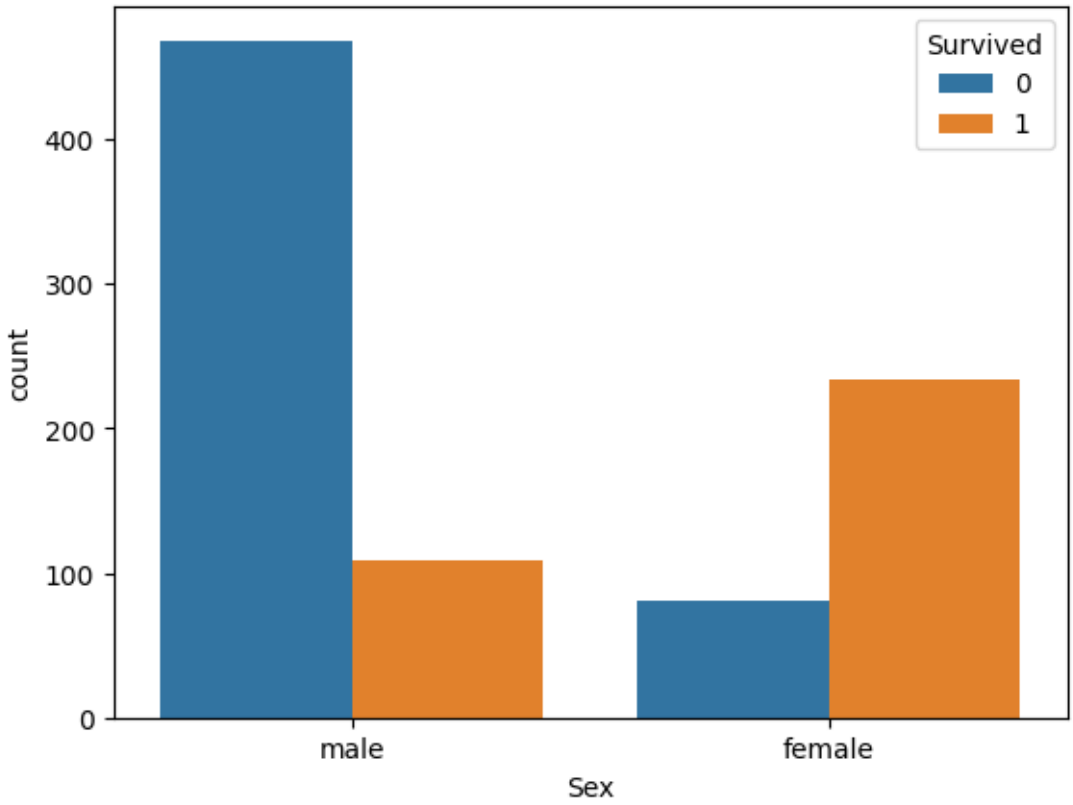
[32]:
```
AgeGroup = pd.cut(df['Age'], bins=[0, 25, 50, 100], labels=['0-25', '25-50',␣
↪'50-100'])

sns.countplot(x=AgeGroup, hue=df['Survived'])
```

[32]: <Axes: xlabel='Age', ylabel='count'>



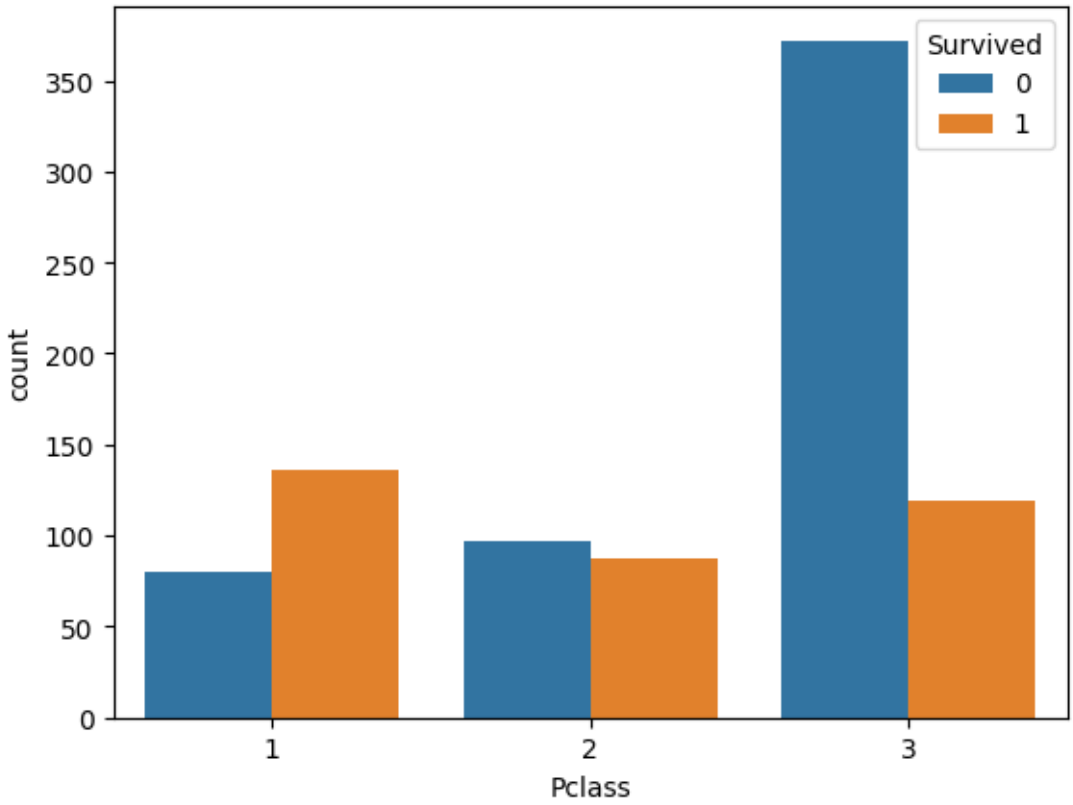
[33]:
```
sns.countplot(data=df,x="Sex",hue="Survived")
```

[33]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



[34]:
```
sns.countplot(x="Pclass",hue="Survived",data=df)
```

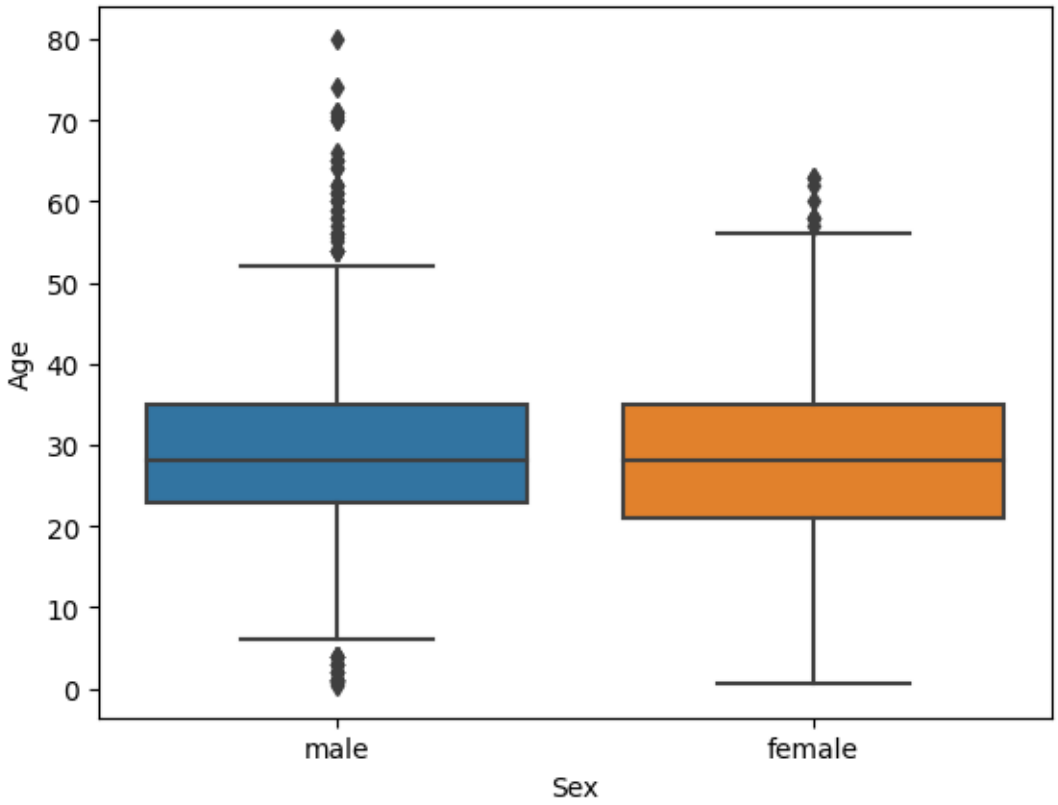
[34]:
```
<Axes: xlabel='Pclass', ylabel='count'>
```



OUTLIERS

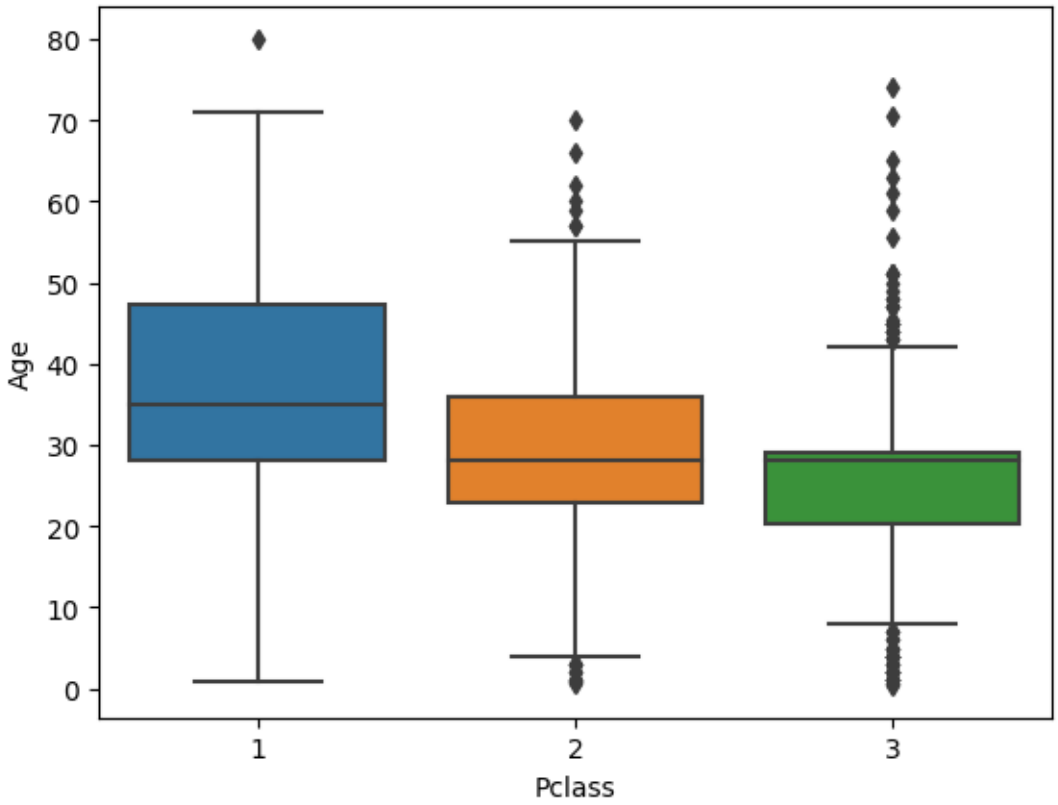
[35]:
```
sns.boxplot(x="Sex",y="Age",data=df)
```

[35]:
```
<Axes: xlabel='Sex', ylabel='Age'>
```



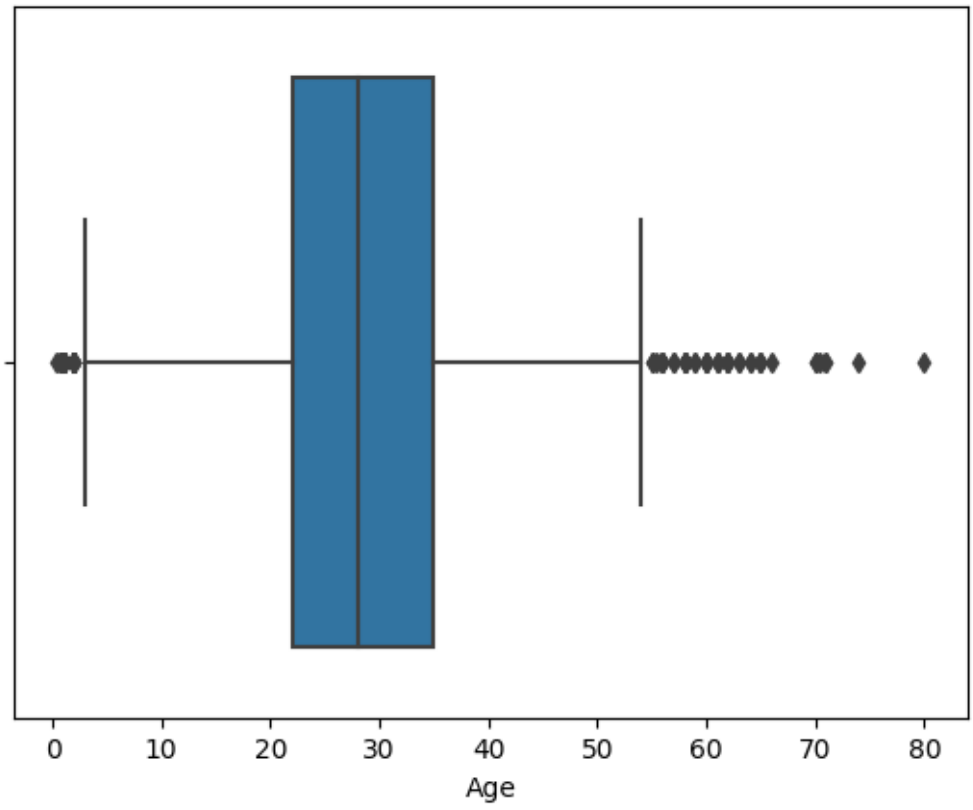
[36]:
```
sns.boxplot(x='Pclass',y="Age", data=df)
```

[36]: <Axes: xlabel='Pclass', ylabel='Age'>



[37]:
```
sns.boxplot(x="Age",data=df)
```

[37]:
```
<Axes: xlabel='Age'>
```



[42]:
```
Q1=df.Age.quantile(0.25)
Q3=df.Age.quantile(0.75)
Q2=df.Age.quantile(0.50)
```

[40]:
```
IQR = Q3 - Q1
IQR
```

[40]: 13.0

[43]:
```
upper_Limit = Q3 + 1.5*IQR
lower_Limit = Q1 - 1.5 * IQR

print("Upper Limit : ",upper_Limit)
print("Lower Limit : ",lower_Limit)
```

```
Upper Limit :  54.5
Lower Limit :  2.5
```

[44]:
```
for_Upper_Limit = df["Age"]>upper_Limit
for_Lower_Limit = df["Age"]<lower_Limit
```

```
total_Outliers = for_Upper_Limit + for_Lower_Limit

print("Total Outliers are : ",total_Outliers.sum())
```

```
Total Outliers are :  66
```

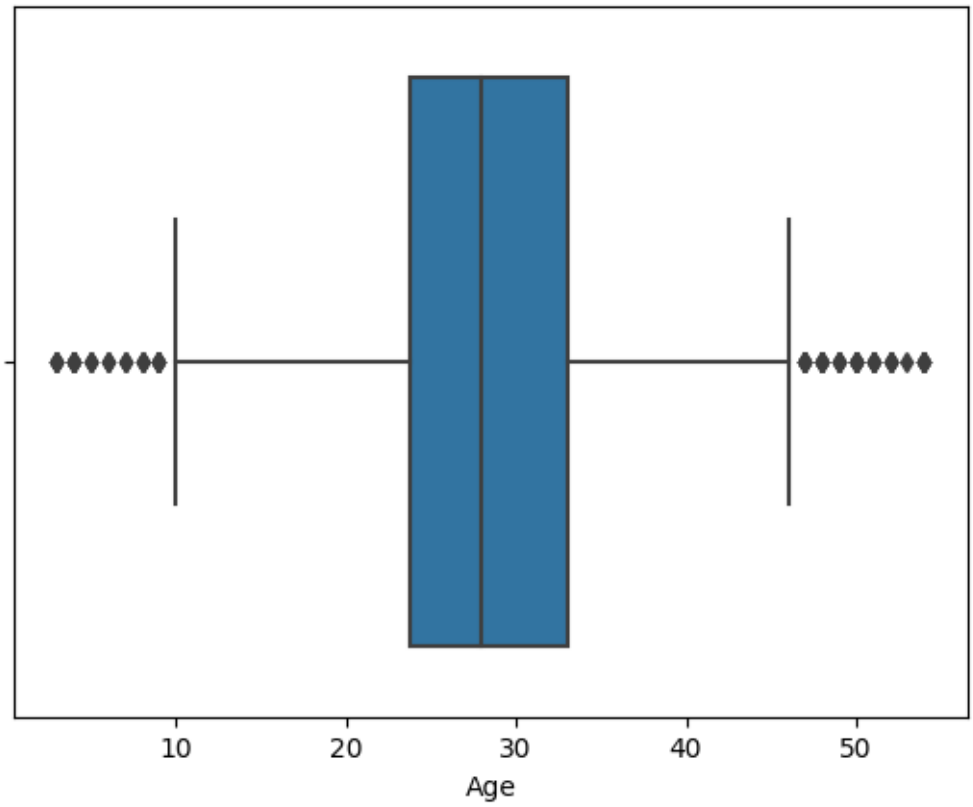
[45]:
```
df.shape
```

[45]:
```
(891, 11)
```

[46]:
```
df["Age"] = np.where((df["Age"] > upper_Limit) | (df["Age"] < lower_Limit),␣
↪median_age, df["Age"])
```

[47]:
```
sns.boxplot(x="Age",data=df)
```

[47]:
```
<Axes: xlabel='Age'>
```



[48]:
```
df.describe()
```

[48]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  891.000000  891.000000
```

```
mean    446.000000    0.383838    2.308642   28.476992    0.523008
std     257.353842    0.486592    0.836071    9.793559    1.102743
min       1.000000    0.000000    1.000000    3.000000    0.000000
25%     223.500000    0.000000    2.000000   23.750000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   33.000000    1.000000
max     891.000000    1.000000    3.000000   54.000000    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
```

[49]:
```
df.drop(columns=["Name","PassengerId","Ticket"],axis=1,inplace=True)
```

[50]:
```
df.head()
```

[50]:

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

[51]:
```
dependent = df["Survived"]
independent = df.drop("Survived",axis=1)
```

[52]:
```
dependent.head()
```

[52]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

[53]:
```
independent.head()
```

[53]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
1	1	female	38.0	1	0	71.2833	C
2	3	female	26.0	0	0	7.9250	S

```
3       1  female  35.0      1      0  53.1000        S
4       3    male  35.0      0      0   8.0500        S
```

[54]:
```
type(dependent)
```

[54]:
```
pandas.core.series.Series
```

[55]:
```
type(independent)
```

[55]:
```
pandas.core.frame.DataFrame
```

0.1.6 Perform Encoding

[56]:
```
independent.head()
```

[56]:
```
   Pclass     Sex   Age  SibSp  Parch     Fare Embarked
0       3    male  22.0      1      0   7.2500        S
1       1  female  38.0      1      0  71.2833        C
2       3  female  26.0      0      0   7.9250        S
3       1  female  35.0      1      0  53.1000        S
4       3    male  35.0      0      0   8.0500        S
```

[57]:
```
from sklearn.preprocessing import LabelEncoder
```

[58]:
```
le = LabelEncoder()
```

[59]:
```
independent["Sex"] = le.fit_transform(independent["Sex"])
```

[60]:
```
independent.head()
```

[60]:
```
   Pclass  Sex   Age  SibSp  Parch     Fare Embarked
0       3    1  22.0      1      0   7.2500        S
1       1    0  38.0      1      0  71.2833        C
2       3    0  26.0      0      0   7.9250        S
3       1    0  35.0      1      0  53.1000        S
4       3    1  35.0      0      0   8.0500        S
```

[61]:
```
independent.tail()
```

[61]:
```
     Pclass  Sex   Age  SibSp  Parch   Fare Embarked
886       2    1  27.0      0      0  13.00        S
887       1    0  19.0      0      0  30.00        S
888       3    0  28.0      1      2  23.45        S
889       1    1  26.0      0      0  30.00        C
890       3    1  32.0      0      0   7.75        Q
```

[62]:
```
embarked = pd.get_dummies(independent["Embarked"],drop_first=True)
```

[63]:
```
independent = pd.concat([independent,embarked],axis=1)
independent.drop("Embarked",axis=1,inplace=True)
```

[64]:
```
independent.head()
```

[64]:

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

[65]:
```
independent.tail()
```

[65]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Q	S
886	2	1	27.0	0	0	13.00	0	1
887	1	0	19.0	0	0	30.00	0	1
888	3	0	28.0	1	2	23.45	0	1
889	1	1	26.0	0	0	30.00	0	0
890	3	1	32.0	0	0	7.75	1	0

0.1.7 Train and Test split

[66]:
```
from sklearn.model_selection import train_test_split as tts
```

[67]:
```
independent_train,independent_test,dependent_train,dependent_test =␣
↪tts(independent,dependent,test_size=0.2,random_state=0)
```

[68]:
```
independent_train.head()
```

[68]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Q	S
140	3	0	28.0	0	2	15.2458	0	0
439	2	1	31.0	0	0	10.5000	0	1
817	2	1	31.0	1	1	37.0042	0	0
378	3	1	20.0	0	0	4.0125	0	0
491	3	1	21.0	0	0	7.2500	0	1

[69]:
```
independent_test.head()
```

[69]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Q	S
495	3	1	28.0	0	0	14.4583	0	0
648	3	1	28.0	0	0	7.5500	0	1
278	3	1	7.0	4	1	29.1250	1	0
31	1	0	28.0	1	0	146.5208	0	0
255	3	0	29.0	0	2	15.2458	0	0

[70]:
```
dependent_train.head()
```

[70]:
```
140    0
439    0
817    0
378    0
491    0
Name: Survived, dtype: int64
```

[71]:
```
dependent_test.head()
```

[71]:
```
495    0
648    0
278    0
31     1
255    1
Name: Survived, dtype: int64
```

[72]:
```
independent_train.shape,independent_test.shape,dependent_train.
↪shape,dependent_test.shape
```

[72]:
```
((712, 8), (179, 8), (712,), (179,))
```

0.1.8 Feature Scaling

[73]:
```
from sklearn.preprocessing import StandardScaler
```

[74]:
```
sc = StandardScaler()
```

[75]:
```
independent_test_fs = sc.fit_transform(independent_test)
independent_test_fs
```

[75]:
```
array([[ 0.86022947,  0.77344314, -0.05003246, …, -0.39903373,
        -0.27984505, -1.56278843],
       [ 0.86022947,  0.77344314, -0.05003246, …, -0.54333564,
        -0.27984505,  0.63988188],
       [ 0.86022947,  0.77344314, -2.12817628, …, -0.09267286,
         3.57340605, -1.56278843],
       …,
       [-1.50871015, -1.29291987,  0.24684523, …,  1.66506862,
        -0.27984505, -1.56278843],
       [ 0.86022947,  0.77344314, -0.54482861, …, -0.53698145,
        -0.27984505,  0.63988188],
       [ 0.86022947,  0.77344314, -0.94066553, …, -0.53289154,
        -0.27984505,  0.63988188]])
```