titanic-dataset-assignment

September 19, 2023

0.1 Titanic Dataset

Done : Checking Null Values, Data Visualization, Outliers, Splitting into Dependent and Independent, Encoding, Splitting into Train and Test

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

Name & Reg No : CH.Ganesh & 21BDS0269

0.1.1 Import Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

0.1.2 Importing the dataset

[2]:
```
df = pd.read_csv("/content/Titanic-Dataset.csv")
```

[3]:
```
df.head(3)
```

[3]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       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

   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
```

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

[4]:
```
     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)

[5]:
```
df.shape
```

[5]:
```
(891, 12)
```

[6]:
```
df.dtypes
```

[6]:
```
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
```

[7]:
```
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
```

[8]:
```
df.describe().T
```

[8]:

	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	714.0	29.699118	14.526497	0.42	20.1250	28.0000	38.0	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	

```
Fare          512.3292
```

0.1.4 Null Values

[9]:
```
df.isnull().any()
```

[9]:
```
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
```

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

[10]:
```
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
```

- Here we can see more than 3/4th of Cabin are null , so we can remove that column or we can replace with the 0 (false) and 1(true). Lets remove directly.

[11]:
```
df.drop("Cabin",axis=1,inplace=True)
```

- Now we can see Age column it have null values 117 , so lets replace it with median.

[12]:
```
median_age = df["Age"].median()
median_age
```

[12]:
```
28.0
```

[13]:
```
df["Age"] = df["Age"].fillna(median_age)
```

- Now Embarked , as there are only 2 we can drop or replace with mode (because its object) , So lets replace with mode

[14]:
```
mode_em = df["Embarked"].mode()
mode_em
```

[14]:
```
0    S
Name: Embarked, dtype: object
```

[15]:
```
df["Embarked"] = df["Embarked"].fillna(mode_em[0])
```

[16]:
```
df.isnull().sum()
```

[16]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

- No Null Values Observed.

[17]:
```
df.describe().T
```

[17]:

	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	

```
Fare          512.3292
```

- Mainly Here only Age and Fare comes under numerical , and all othere all just numerical but comes under categorical in this dataset

[18]:
```
df.Sex.value_counts()
```

[18]:
```
male      577
female    314
Name: Sex, dtype: int64
```

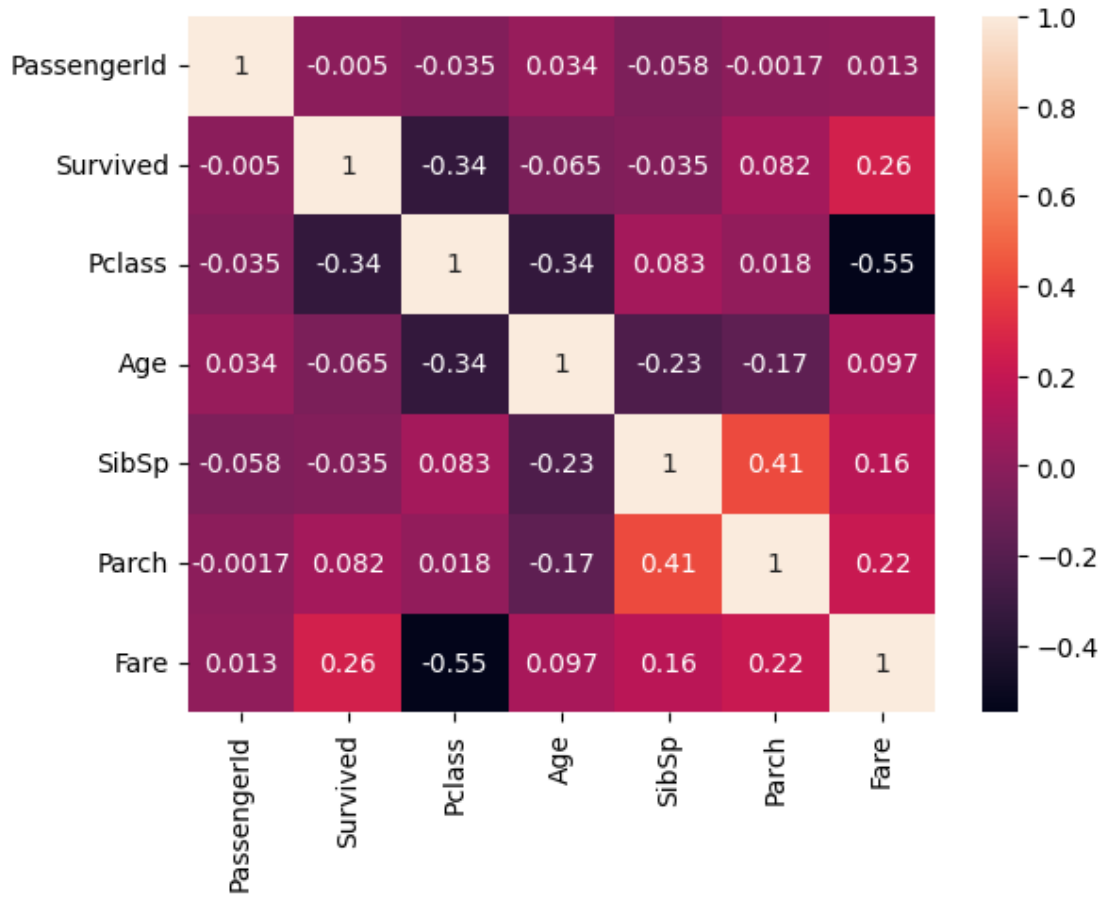
[19]:
```
df.Survived.value_counts()
```

[19]:
```
0    549
1    342
Name: Survived, dtype: int64
```

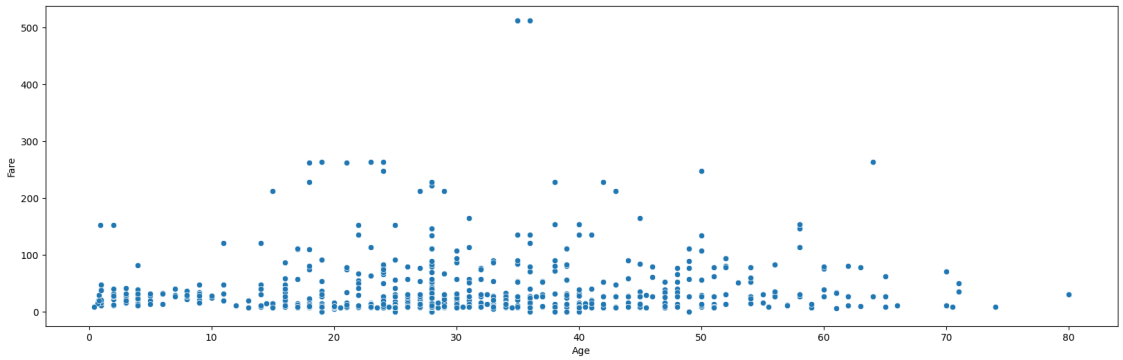
0.1.5 Data Visualization

[20]:
```
sns.heatmap(df.corr(numeric_only=True),annot=True)
```

[20]:
```
<Axes: >
```

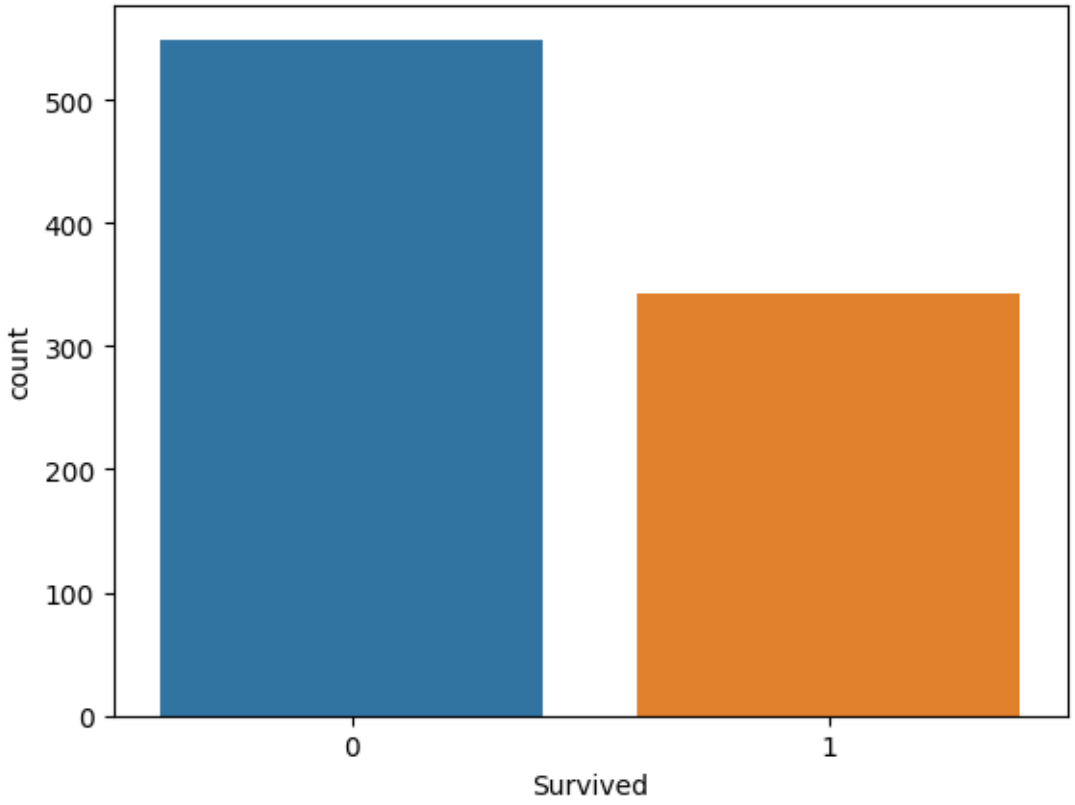


[21]:
```
plt.figure(figsize=(20, 6))
sns.scatterplot(x="Age",y="Fare",data=df)
plt.show()
```



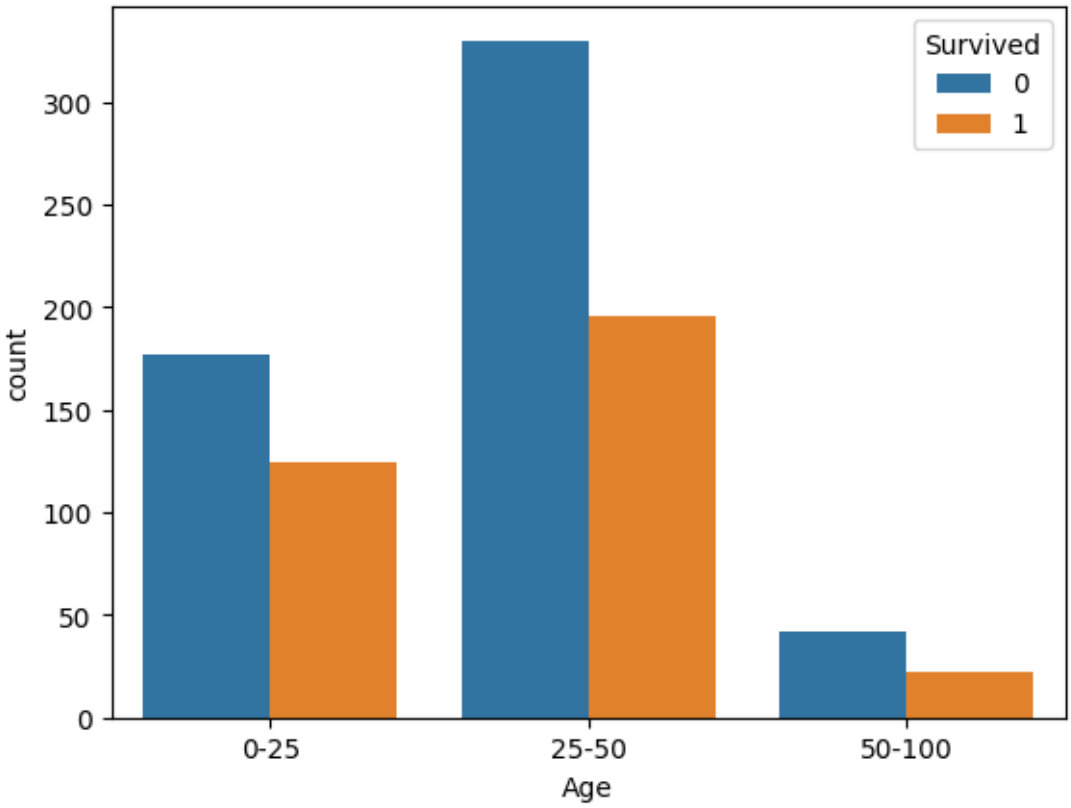
[22]:
```
sns.countplot(x="Survived",data=df)
```

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



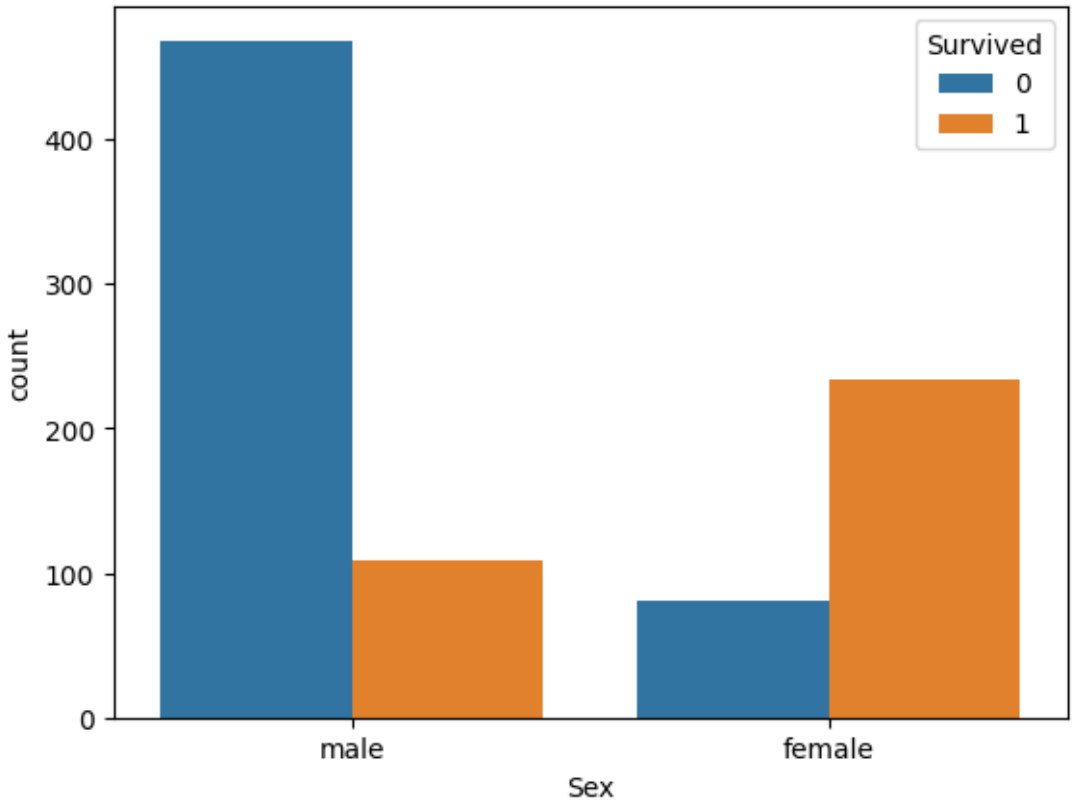
[23]:
```
AgeGroup = pd.cut(df['Age'], bins=[0, 25, 50, 100], labels=['0-25', '25-50',␣
↪'50-100'])

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

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



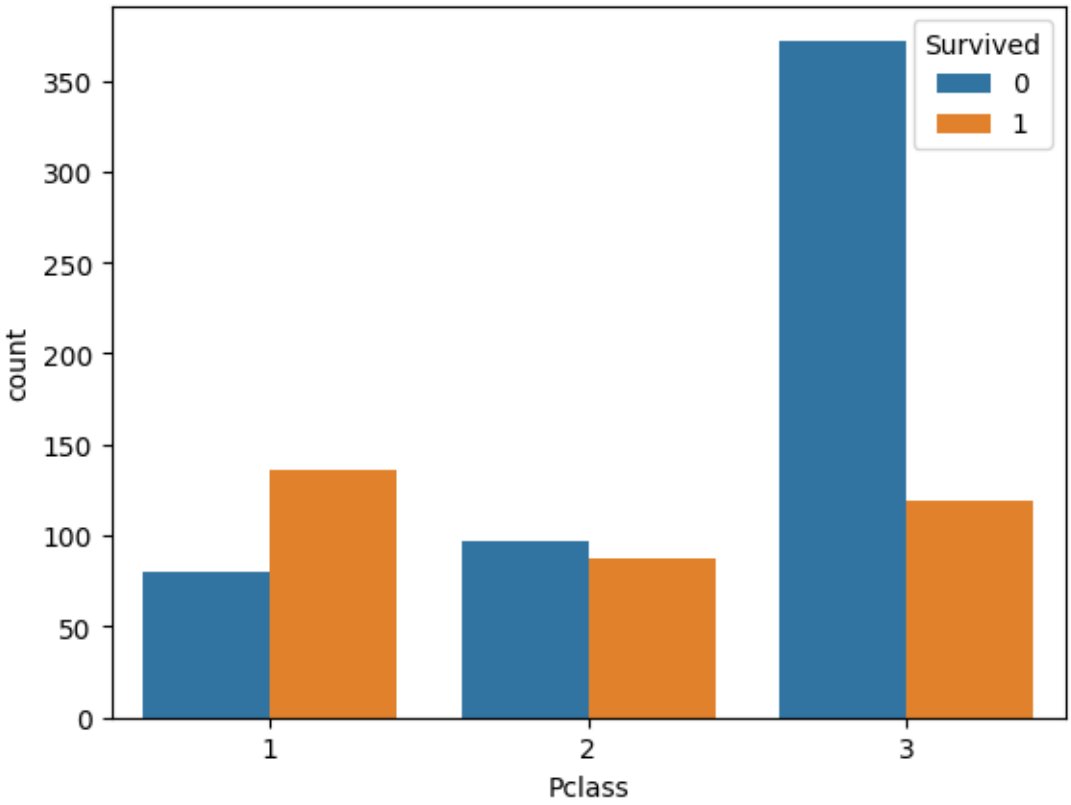
[24]:
```
sns.countplot(data=df,x="Sex",hue="Survived")
```

[24]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



[25]:
```
sns.countplot(x="Pclass",hue="Survived",data=df)
```

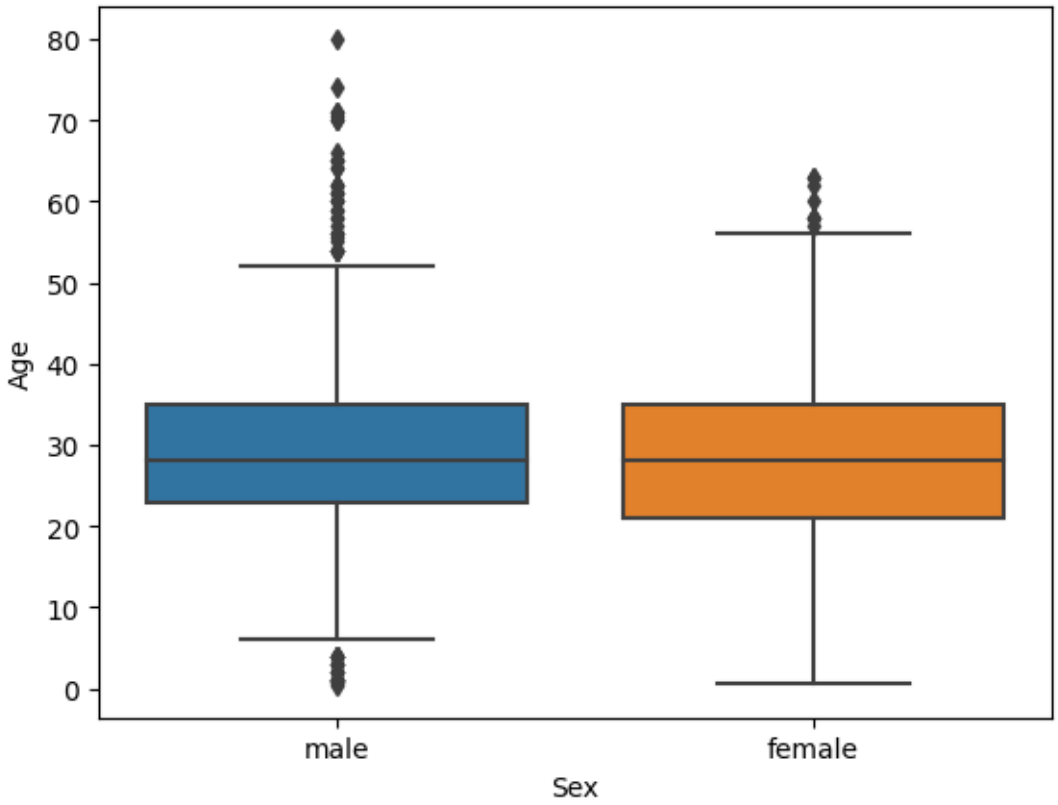
[25]:
```
<Axes: xlabel='Pclass', ylabel='count'>
```



0.1.6 Outliers

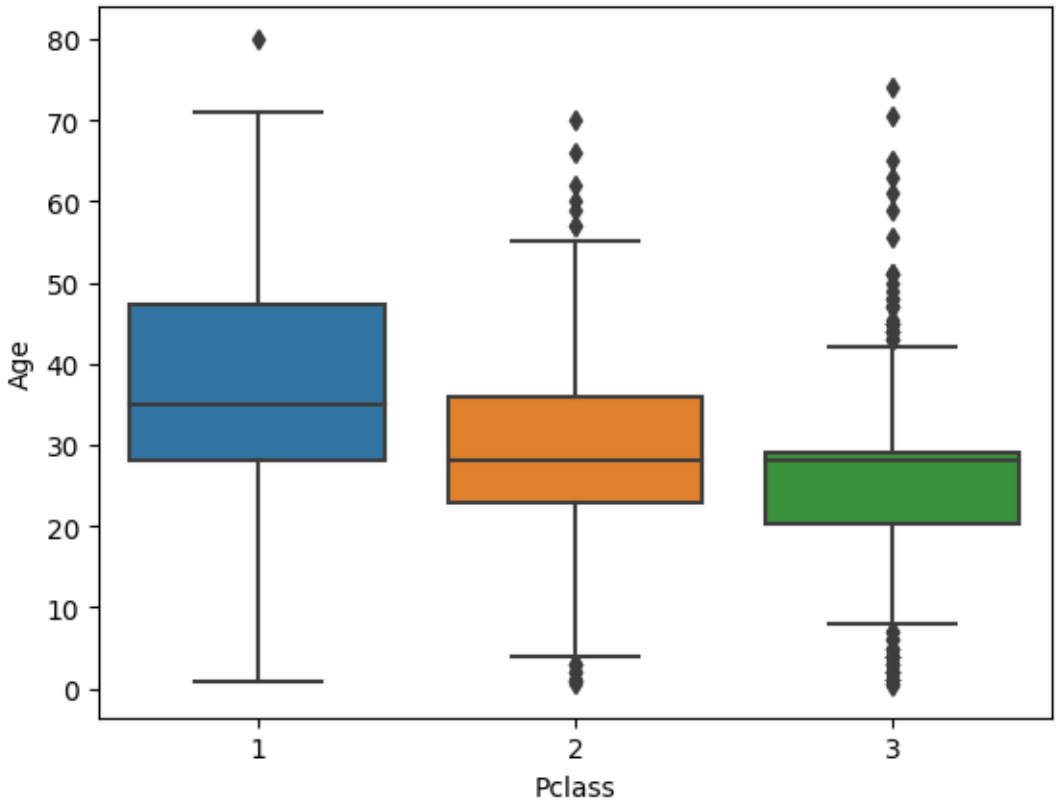
[26]:
```
sns.boxplot(x="Sex",y="Age",data=df)
```

[26]:
```
<Axes: xlabel='Sex', ylabel='Age'>
```



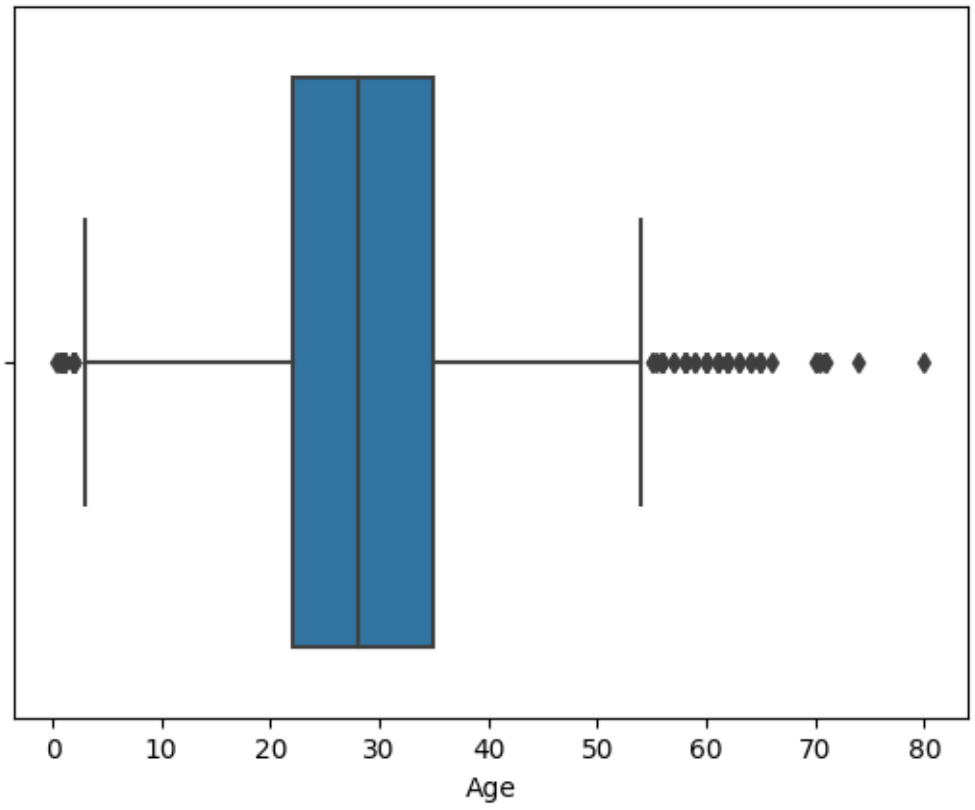
[27]:
```
sns.boxplot(x='Pclass',y="Age", data=df)
```

[27]:
```
<Axes: xlabel='Pclass', ylabel='Age'>
```



[28]:
```
sns.boxplot(x="Age",data=df)
```

[28]:
```
<Axes: xlabel='Age'>
```



- There are outliers in the graph , So than removing we can replace it with median/remove , first lets see how many are there outliers

[29]:
```
Q1 = df.Age.quantile(0.25)
Q1
```

[29]: 22.0

[30]:
```
Q3 = df.Age.quantile(0.75)
Q3
```

[30]: 35.0

[31]:
```
IQR = Q3 - Q1
IQR
```

[31]: 13.0

[32]:
```
upperLimit = Q3 + 1.5*IQR
lowerLimit = Q1 - 1.5 * IQR
```

```
print("Upper Limit : ",upperLimit)
print("Lower Limit : ",lowerLimit)
```

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

- Number of Outliers:

[33]:
```
forUpperLimit = df["Age"]>upperLimit
forLowerLimit = df["Age"]<lowerLimit

totalOutliers = forUpperLimit + forLowerLimit

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

```
Total Outliers are :  66
```

[34]:
```
df.shape
```

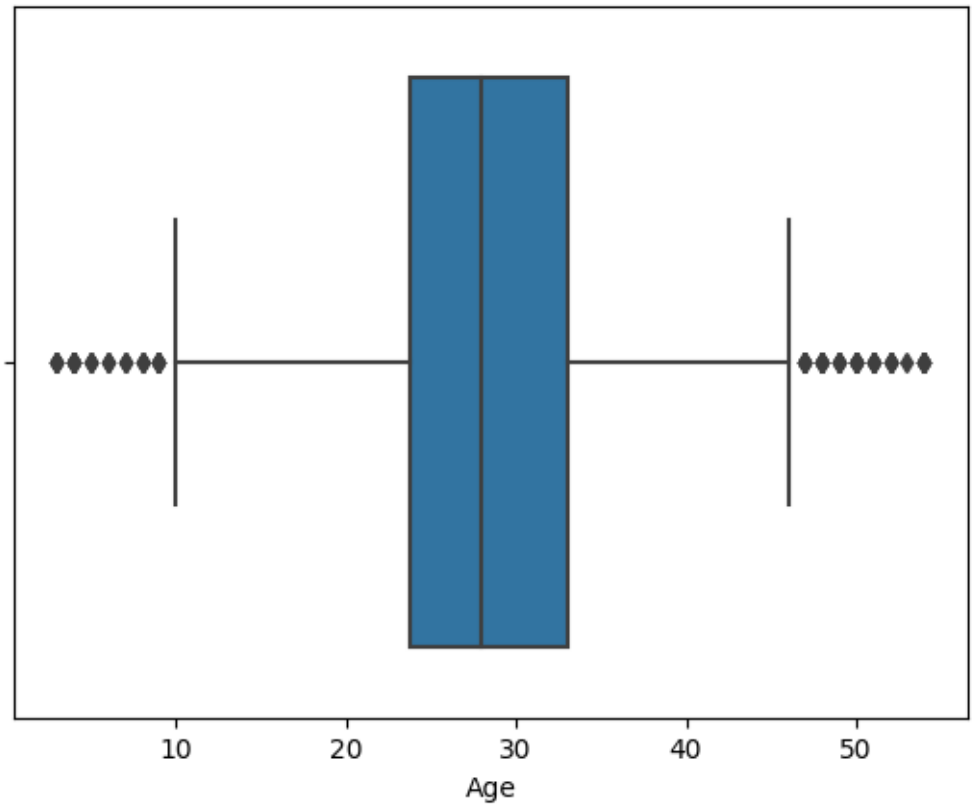
[34]:
```
(891, 11)
```

- As there are 66 , than removing lets replace with median

[35]:
```
df["Age"] = np.where((df["Age"] > upperLimit) | (df["Age"] < lowerLimit),␣
↪median_age, df["Age"])
```

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

[36]:
```
<Axes: xlabel='Age'>
```



[37]:
```
df.describe().T
```

[37]:

	count	mean	std	min	25%	50%	75%	max
PassengerId	891.0	446.000000	257.353842	1.0	223.5000	446.0000	668.5	891.0000
Survived	891.0	0.383838	0.486592	0.0	0.0000	0.0000	1.0	1.0000
Pclass	891.0	2.308642	0.836071	1.0	2.0000	3.0000	3.0	3.0000
Age	891.0	28.476992	9.793559	3.0	23.7500	28.0000	33.0	54.0000
SibSp	891.0	0.523008	1.102743	0.0	0.0000	0.0000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.0	0.0000	0.0000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.0	7.9104	14.4542	31.0	512.3292

0.1.7 Drop Unecessary Columns

- This step is not mandatory, just to reduce dataset we can do.
- Name, PassengerId, Ticket (As Pclass is there its not necessary) so these are not necessary.
- Embarked is also may be not necessary but let it be we are not 100% sure.

[38]:
```
df.drop(columns=["Name","PassengerId","Ticket"],axis=1,inplace=True)
```

[39]:
```
df.head()
```

[39]:

	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

0.1.8 Splitting Dependent and Independent variables

- Here we can say clearly the ***Survived*** is the ***Dependent*** and Remaining are independent variables
- Independent Variable should be -> 2D Array or Data Frame
- Dependent Variable Should be -> Series or 1D Array

[40]:
```
df.head()
```

[40]:

	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

[41]:
```
dependent = df["Survived"]
independent = df.drop("Survived",axis=1)
```

[42]:
```
dependent.head()
```

[42]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

[43]:
```
independent.head()
```

[43]:

	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

[44]:
```
type(dependent)
```

[44]:
```
pandas.core.series.Series
```

[45]:
```
type(independent)
```

[45]:
```
pandas.core.frame.DataFrame
```

0.1.9 Perform Encoding

[46]:
```
independent.head()
```

[46]:

	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

- We have to encode Sex,Embarked as they are in categorical

[47]:
```
from sklearn.preprocessing import LabelEncoder
```

[48]:
```
le = LabelEncoder()
```

[49]:
```
independent["Sex"] = le.fit_transform(independent["Sex"])
```

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

[50]:

	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

[51]:
```
independent.tail()
```

[51]:

	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
```

[52]:
```
embarked = pd.get_dummies(independent["Embarked"],drop_first=True)
```

[53]:
```
independent = pd.concat([independent,embarked],axis=1)
independent.drop("Embarked",axis=1,inplace=True)
```

[54]:
```
independent.head()
```

[54]:

	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

[55]:
```
independent.tail()
```

[55]:

	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

0.1.10 Splitting Data into Train and Test

[56]:
```
from sklearn.model_selection import train_test_split as tts
```

[57]:
```
independent_train,independent_test,dependent_train,dependent_test =␣
↪tts(independent,dependent,test_size=0.2,random_state=0)
```

[58]:
```
independent_train.head()
```

[58]:

	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

[59]:
```
independent_test.head()
```

[59]:

	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

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

[60]:
```
140    0
439    0
817    0
378    0
491    0
Name: Survived, dtype: int64
```

[61]:
```
dependent_test.head()
```

[61]:
```
495    0
648    0
278    0
31     1
255    1
Name: Survived, dtype: int64
```

[62]:
```
independent_train.shape,independent_test.shape,dependent_train.
↪shape,dependent_test.shape
```

[62]:
```
((712, 8), (179, 8), (712,), (179,))
```

0.1.11 Feature Scaling

- Only for independent we will perform Feature Scaling.

[63]:
```
from sklearn.preprocessing import StandardScaler
```

[64]:
```
sc = StandardScaler()
```

[65]:
```
independent_test_fs = sc.fit_transform(independent_test)
independent_test_fs
```

[65]:
```
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]])
```