In [1]:
```
import pandas as pd
```

In [2]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\ykyas\appdata\local\programs\py
thon\python312\lib\site-packages (5.21.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\ykyas\appdata\local\pr
ograms\python\python312\lib\site-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\ykyas\appdata\local\programs
\python\python312\lib\site-packages (from plotly) (24.0)
```

In [3]:
```
import warnings
```

In [4]:
```
warnings.filterwarnings('ignore')
```

In [5]:
```
df=pd.read_csv('Titanic-Dataset.csv')
```

In [6]:
```
df                                                         #we are printing the en
```

Out[6]:

	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
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450
...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376

891 rows × 12 columns

In [7]:

```
df.info() #types of data
```

```
<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
```

In [8]:
```
df.describe()  #stats of the dataset
```

Out[8]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329

In [9]:
```
df=df.drop(['PassengerId','Name','Ticket','Cabin'],axis=1) #retain whates needed
```

In [10]:
```
df
```

Out[10]:

	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
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S
887	1	1	female	19.0	0	0	30.0000	S
888	0	3	female	NaN	1	2	23.4500	S
889	1	1	male	26.0	0	0	30.0000	C
890	0	3	male	32.0	0	0	7.7500	Q

891 rows × 8 columns

In [11]:

```
df.isna()
```

Out[11]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False
887	False	False	False	False	False	False	False	False
888	False	False	False	True	False	False	False	False
889	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False

891 rows × 8 columns

In [12]:

```
df.isna().sum()    #to find the missing values
```

Out[12]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [13]:
```
df=df.dropna(subset=['Embarked'])   #all missing embarked values are dropped
```

In [14]:
```
df
```

Out[14]:

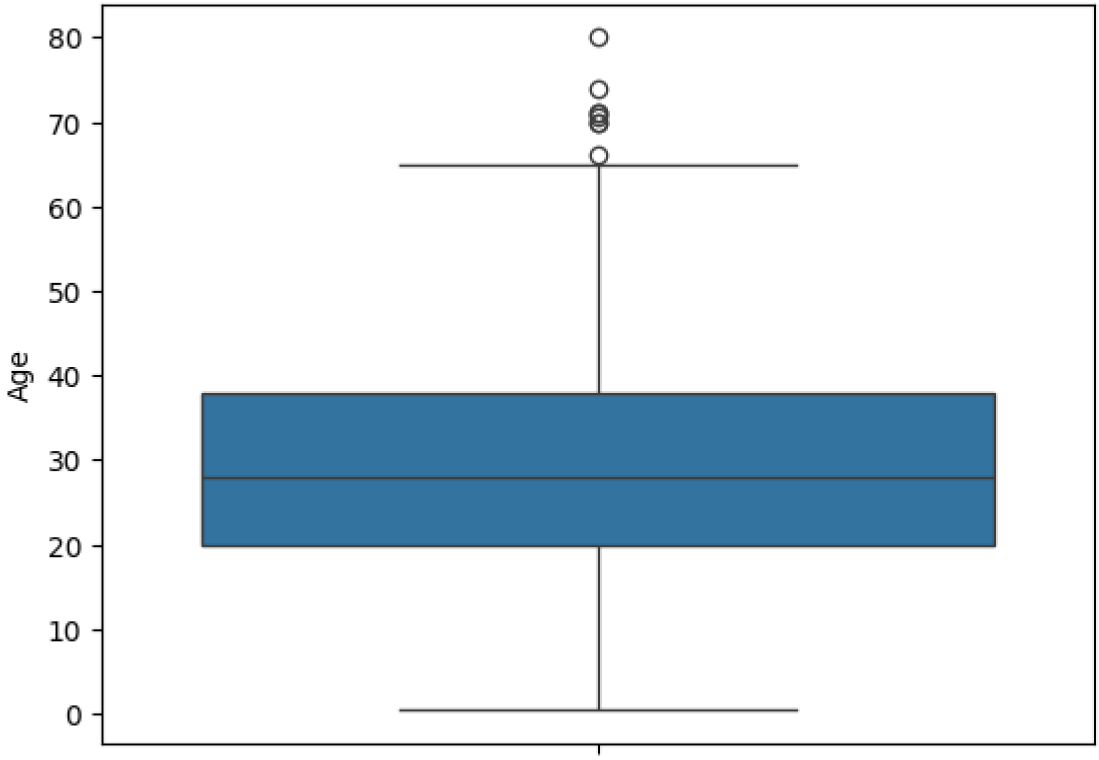
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S
887	1	1	female	19.0	0	0	30.0000	S
888	0	3	female	NaN	1	2	23.4500	S
889	1	1	male	26.0	0	0	30.0000	C
890	0	3	male	32.0	0	0	7.7500	Q

889 rows × 8 columns

In [15]:
```
import seaborn as sss  #to use graphs
```

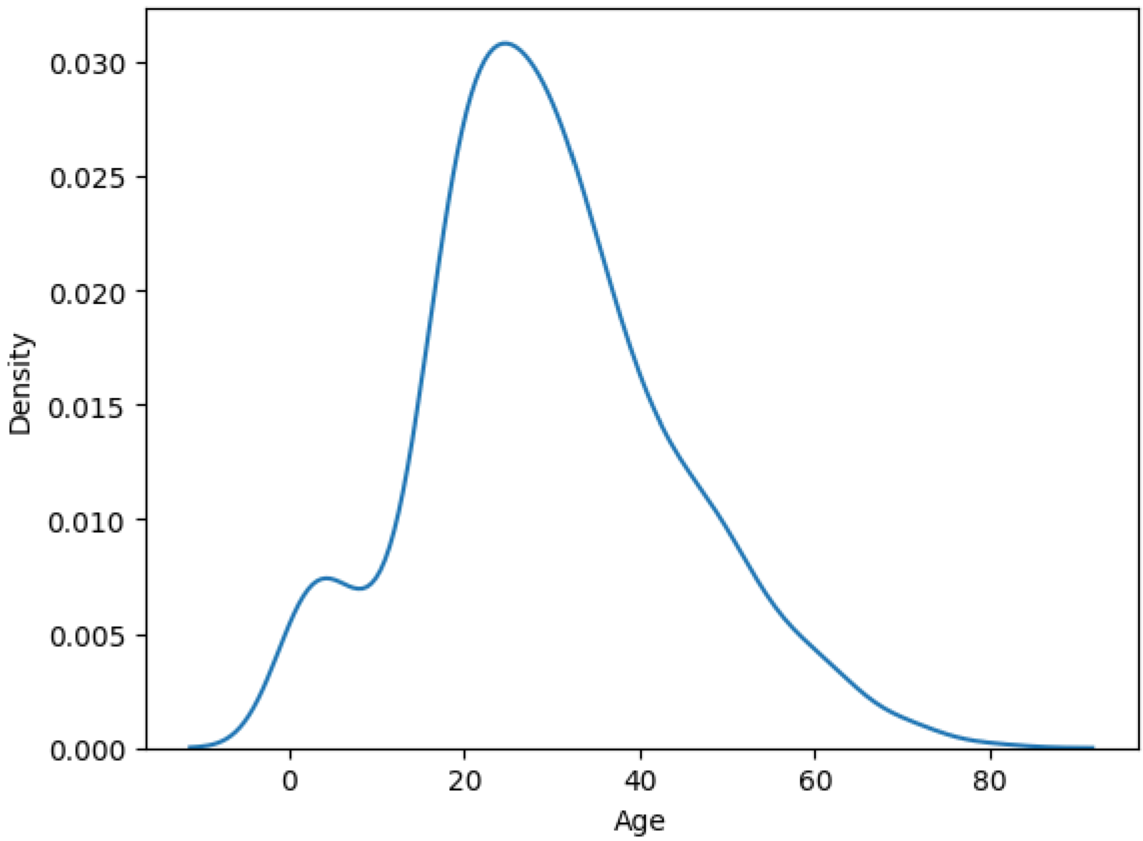
In [16]:
```
sss.boxplot(df['Age'])  #outliers
```

Out[16]:
```
<Axes: ylabel='Age'>
```



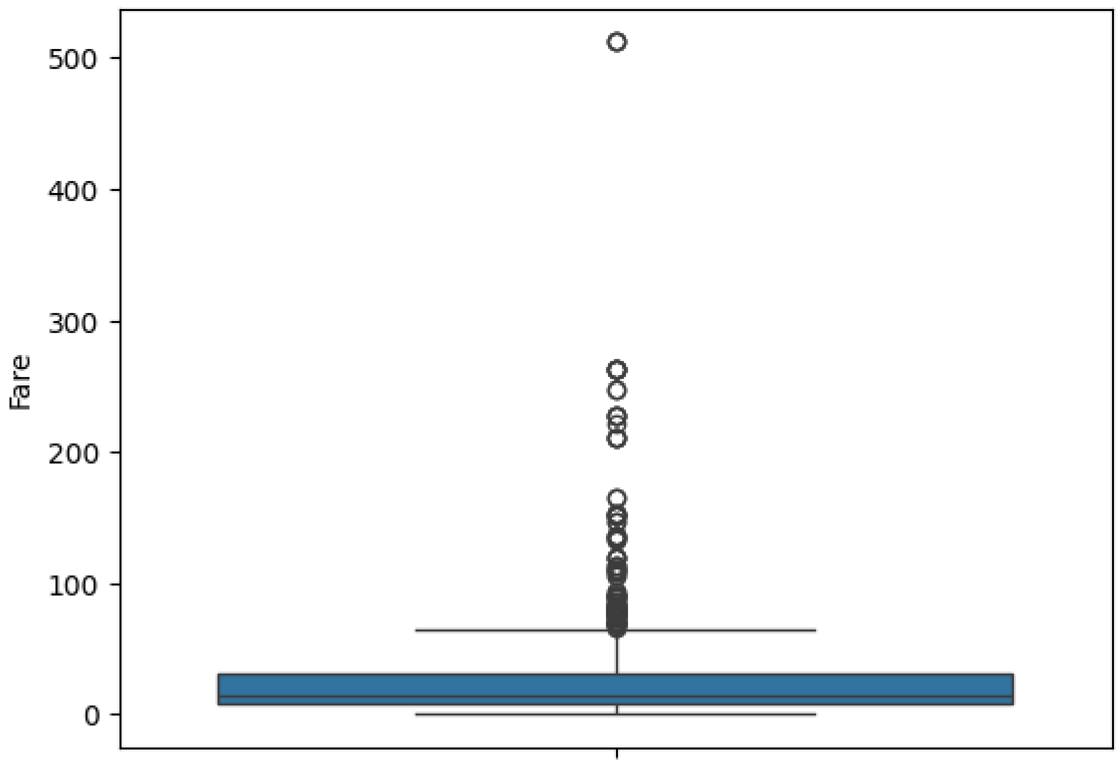
In [17]:
```
sss.kdeplot(df['Age'])      #normal
```

Out[17]: <Axes: xlabel='Age', ylabel='Density'>



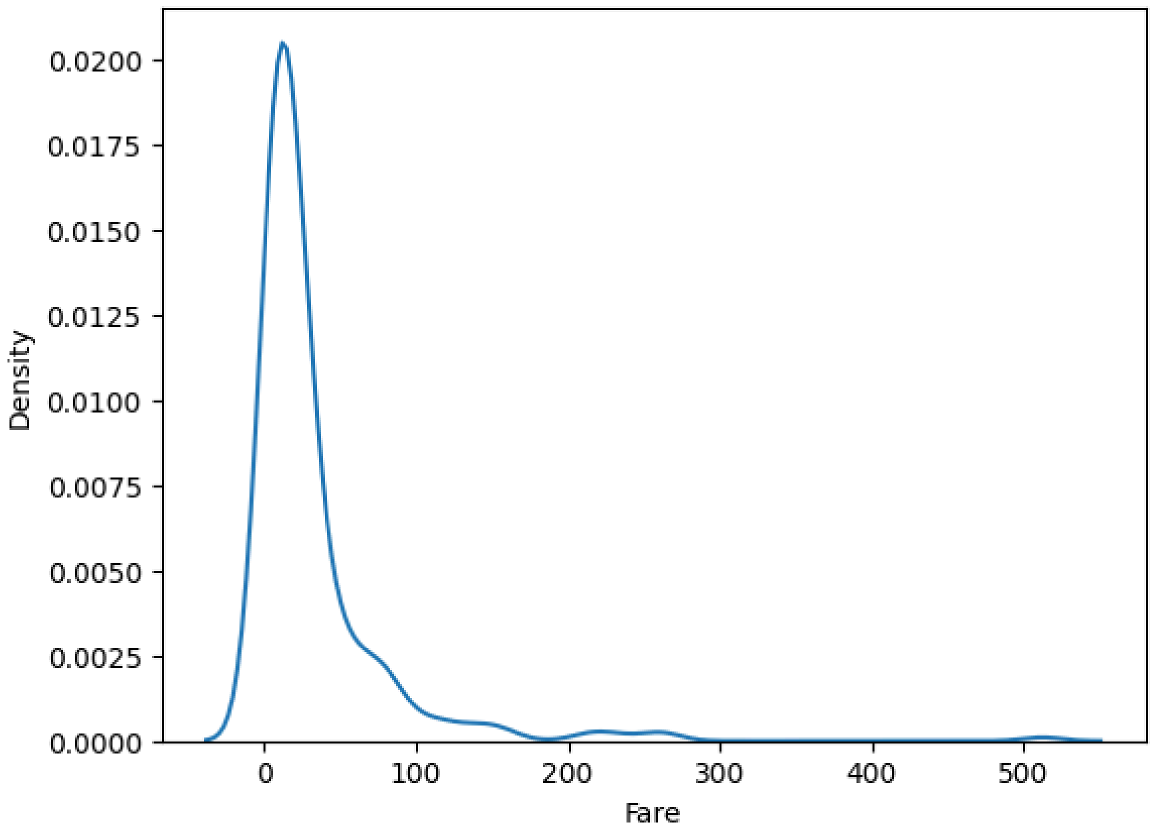
In [18]:
```
sss.boxplot(df['Fare'])  #outliers
```

Out[18]: <Axes: ylabel='Fare'>



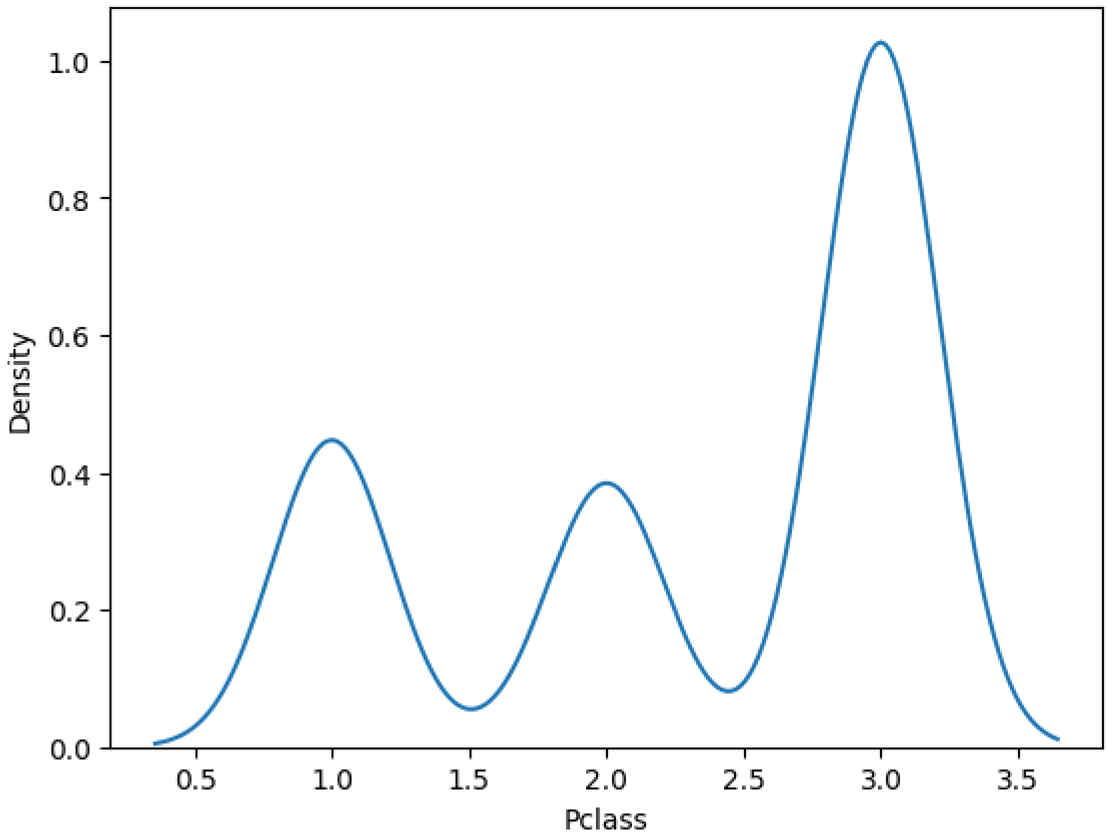
In [19]:
```
sss.kdeplot(df['Fare'])    #right sqewed
```

Out[19]: <Axes: xlabel='Fare', ylabel='Density'>



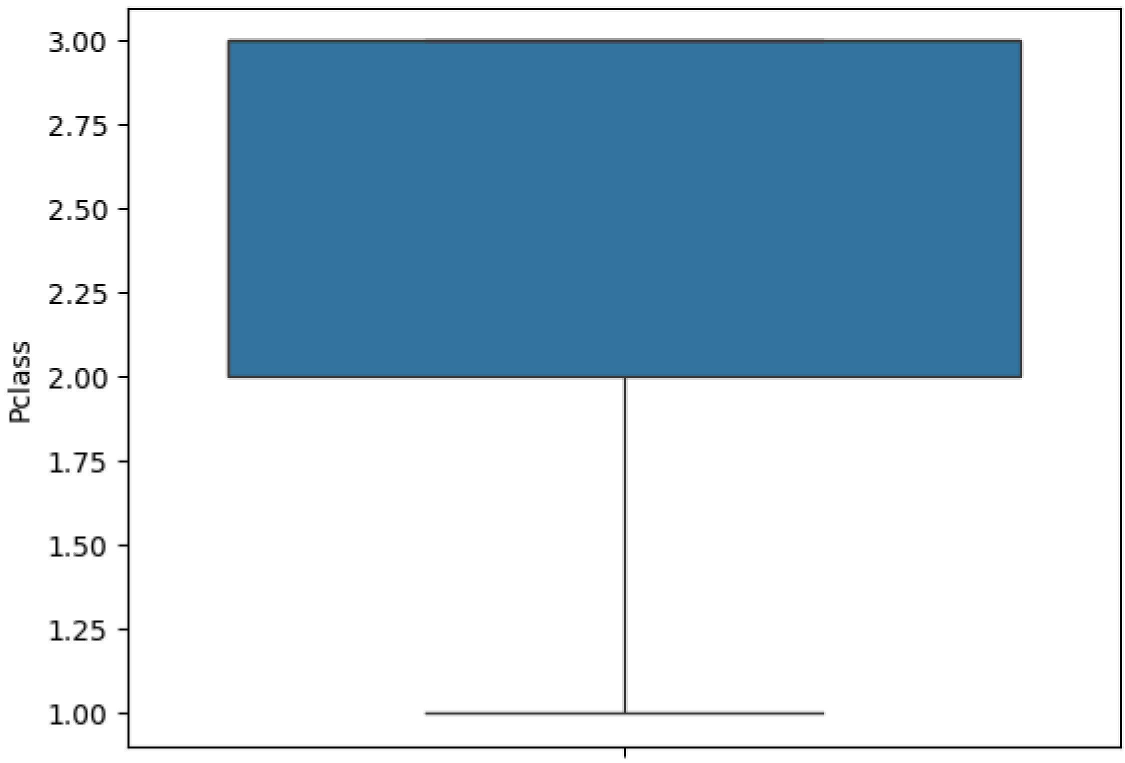
In [20]:
```
sss.kdeplot(df['Pclass'])    #normal graph
```

Out[20]: <Axes: xlabel='Pclass', ylabel='Density'>



In [21]:
```
sss.boxplot(df['Pclass'])  #no outliers
```

Out[21]: <Axes: ylabel='Pclass'>



In [22]:
```
dfage=df['Age']
dffare=df['Fare']
#we need to fix age and fare
```

In [23]:
```python
import numpy as np
dfage_abs=dfage.abs()  #avoid non negative numbers
dffare_abs=dffare.abs()

changedage=dfage_abs.apply(lambda x: x**(1/2)) #reducing the values using square
changedfare=dffare_abs.apply(lambda x: np.log1p(x))

df['Age']=changedage
df['Fare']=changedfare
```

In [24]:
```python
df['Age']
```

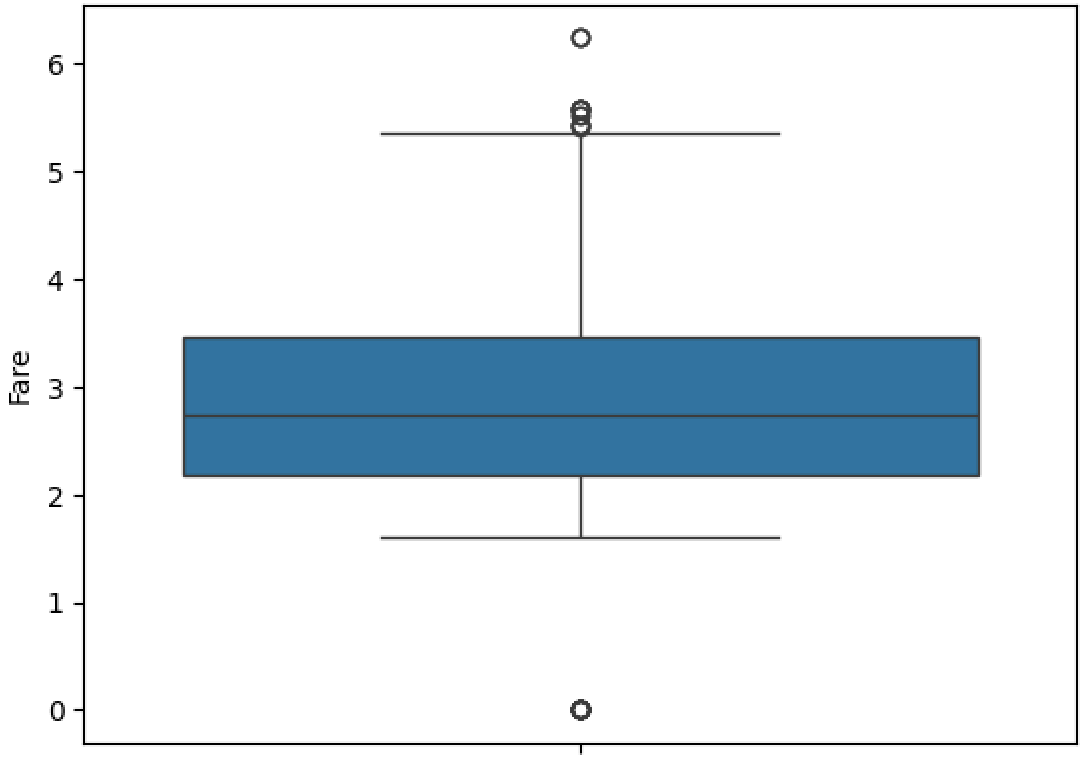
Out[24]:
```
0      4.690416
1      6.164414
2      5.099020
3      5.916080
4      5.916080
         ...
886    5.196152
887    4.358899
888         NaN
889    5.099020
890    5.656854
Name: Age, Length: 889, dtype: float64
```

In [25]:
```python
df['Fare']
```

Out[25]:
```
0      2.110213
1      4.280593
2      2.188856
3      3.990834
4      2.202765
         ...
886    2.639057
887    3.433987
888    3.196630
889    3.433987
890    2.169054
Name: Fare, Length: 889, dtype: float64
```

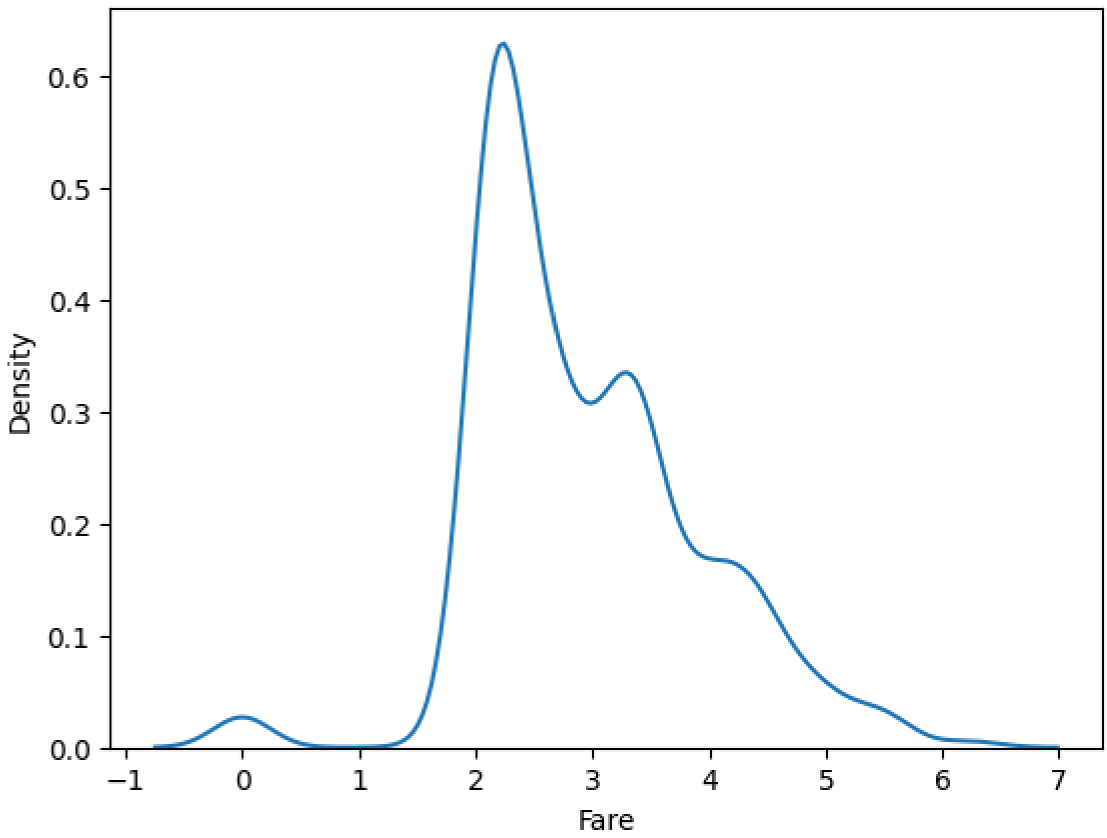
In [26]:
```python
sss.boxplot(df['Fare'])
```

Out[26]:
```
<Axes: ylabel='Fare'>
```



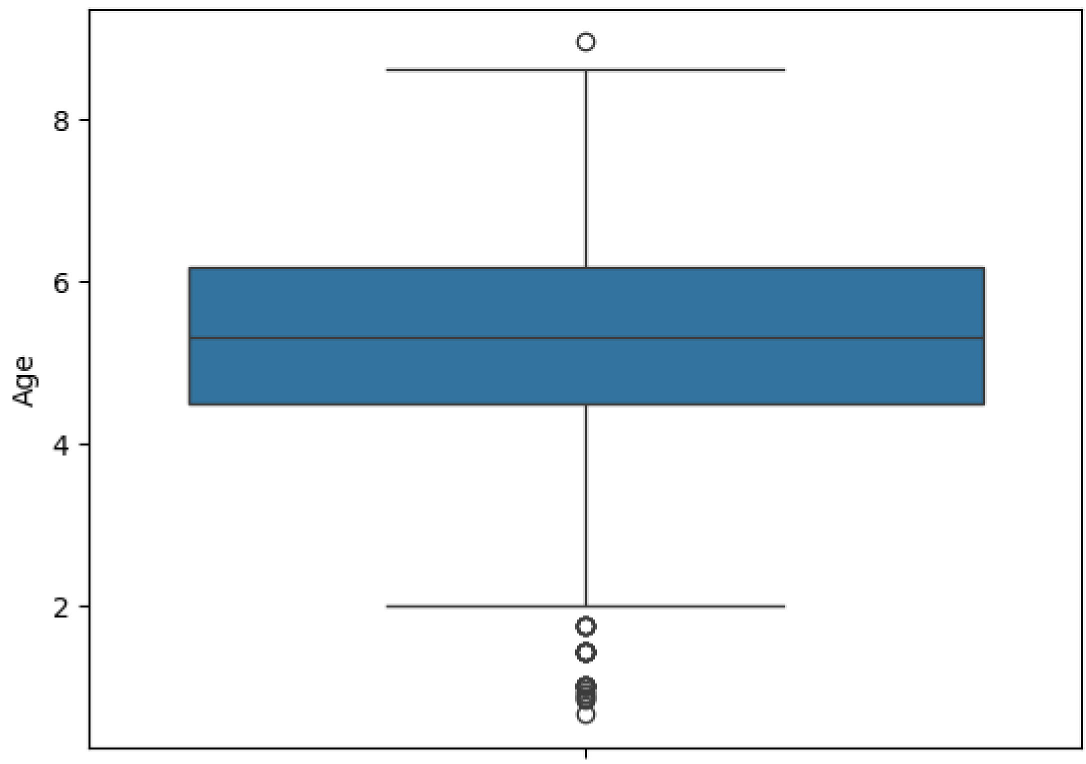
In [27]:
```
sss.kdeplot(df['Fare'])
```

Out[27]: <Axes: xlabel='Fare', ylabel='Density'>



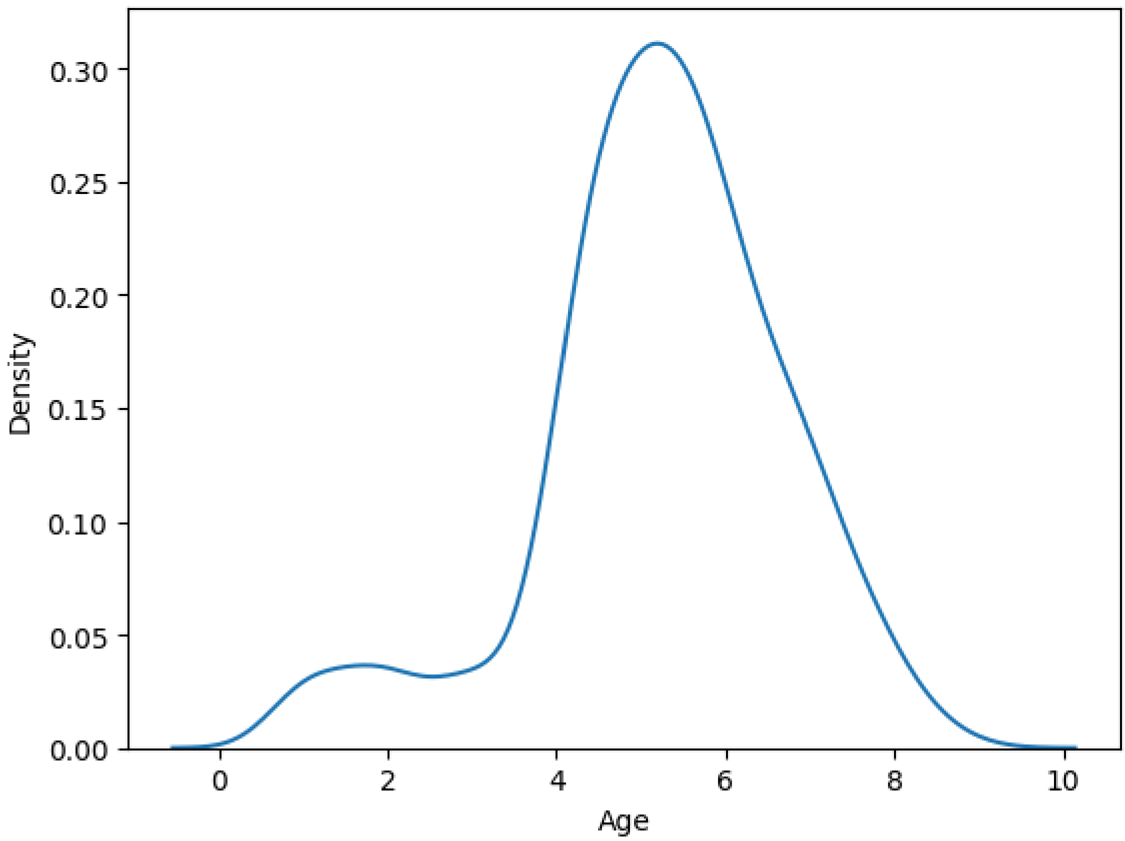
In [28]:
```
sss.boxplot(df['Age'])
```

Out[28]: <Axes: ylabel='Age'>



In [29]:
```
sss.kdeplot(df['Age'])
```

Out[29]: <Axes: xlabel='Age', ylabel='Density'>



In [30]:
```
df
```

Out[30]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	4.690416	1	0	2.110213	S
1	1	1	female	6.164414	1	0	4.280593	C
2	1	3	female	5.099020	0	0	2.188856	S
3	1	1	female	5.916080	1	0	3.990834	S
4	0	3	male	5.916080	0	0	2.202765	S
...	...	...	...	...	...	...	...	...
886	0	2	male	5.196152	0	0	2.639057	S
887	1	1	female	4.358899	0	0	3.433987	S
888	0	3	female	NaN	1	2	3.196630	S
889	1	1	male	5.099020	0	0	3.433987	C
890	0	3	male	5.656854	0	0	2.169054	Q

889 rows × 8 columns

In [31]:

```
from sklearn.preprocessing import OneHotEncoder   #we are converting string valu
```

In [32]:

```
attr_val=['Sex','Embarked']
data_to_encode=df[attr_val]

data_to_enocde=data_to_encode.dropna() #remove missing values as encoder will no

encoder=OneHotEncoder(drop='first') #we remove sex_female,embarked_c because the

encoded_data=encoder.fit_transform(data_to_encode) #transform data into encoded

encoded_df=pd.DataFrame(encoded_data.toarray(),columns=encoder.get_feature_names

df.reset_index(drop=True,inplace=True)#reset indices
encoded_df.reset_index(drop=True,inplace=True)

df_encoded=pd.concat([df,encoded_df],axis=1)#joining old dataframe with new data

df=df_encoded.drop(columns=attr_val)  #remove original columns that got encoded
```

In [33]:

```
df
```

Out[33]:

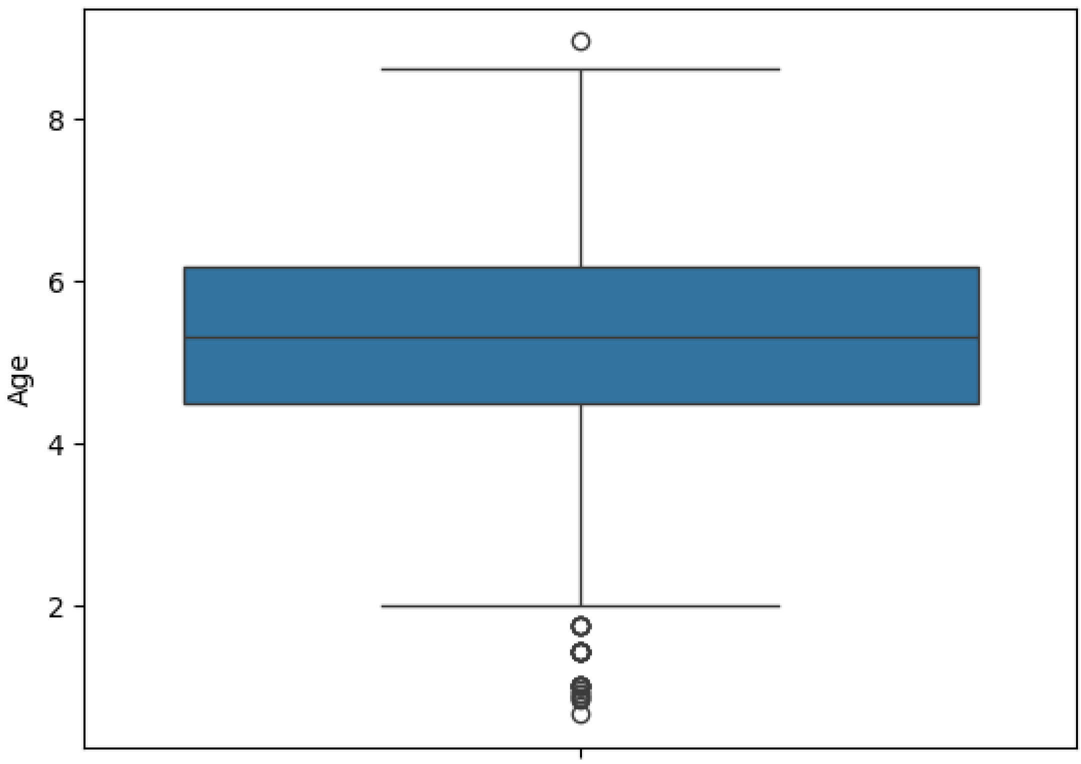
	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embai
0	0	3	4.690416	1	0	2.110213	1.0	0.0	
1	1	1	6.164414	1	0	4.280593	0.0	0.0	
2	1	3	5.099020	0	0	2.188856	0.0	0.0	
3	1	1	5.916080	1	0	3.990834	0.0	0.0	
4	0	3	5.916080	0	0	2.202765	1.0	0.0	
...	...	...	...	...	...	...	...	...	
884	0	2	5.196152	0	0	2.639057	1.0	0.0	
885	1	1	4.358899	0	0	3.433987	0.0	0.0	
886	0	3	NaN	1	2	3.196630	0.0	0.0	
887	1	1	5.099020	0	0	3.433987	1.0	0.0	
888	0	3	5.656854	0	0	2.169054	1.0	1.0	

889 rows × 9 columns

In [34]:

```
sss.boxplot(df['Age'])
```

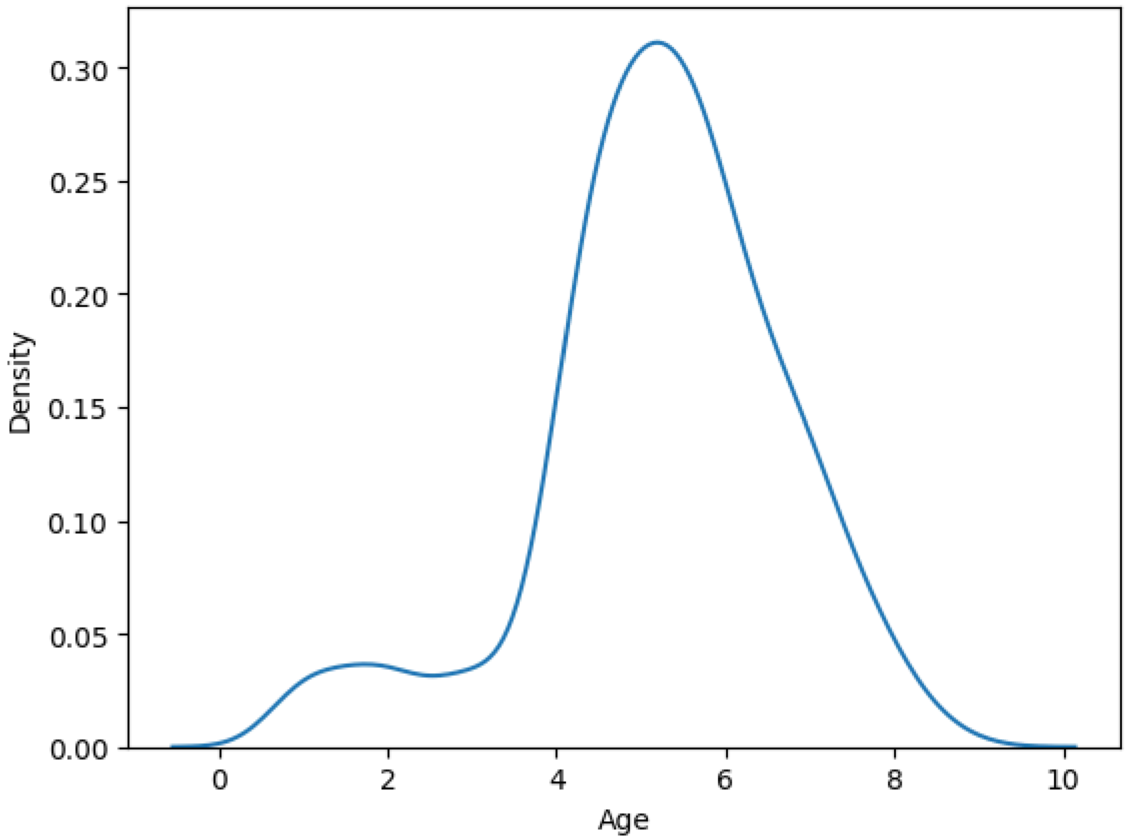
Out[34]: <Axes: ylabel='Age'>



In [35]:

```
sss.kdeplot(df['Age'])
```

Out[35]: <Axes: xlabel='Age', ylabel='Density'>



In [36]:
```
df.isna().sum()  #we still have outliers in age
```

Out[36]:
```
Survived        0
Pclass          0
Age           177
SibSp           0
Parch           0
Fare            0
Sex_male        0
Embarked_Q      0
Embarked_S      0
dtype: int64
```

In [37]:
```
#we could not eliminate missing values,so we use linear regression to predict mi
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
```

In [38]:
```
#seperating rows with values and without values
without_age=df[df['Age'].isnull()]
with_age=df.dropna(subset=['Age'])

#training using with_age
x_train=with_age.drop(columns=['Age'])
y_train=with_age['Age']
regressor_obj=LinearRegression()
regressor_obj.fit(x_train,y_train)

#predicting missing data
missing_x_values=without_age.drop(columns=['Age'])
predicted_age=regressor_obj.predict(missing_x_values)

#replacing missing values with predicted values in the dataset
df.loc[df['Age'].isnull(),'Age']=predicted_age
```

In [39]:
```
df
```

Out[39]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Emba
0	0	3	4.690416	1	0	2.110213	1.0	0.0	
1	1	1	6.164414	1	0	4.280593	0.0	0.0	
2	1	3	5.099020	0	0	2.188856	0.0	0.0	
3	1	1	5.916080	1	0	3.990834	0.0	0.0	
4	0	3	5.916080	0	0	2.202765	1.0	0.0	
...	...	...	...	...	...	...	...	...	
884	0	2	5.196152	0	0	2.639057	1.0	0.0	
885	1	1	4.358899	0	0	3.433987	0.0	0.0	
886	0	3	4.583992	1	2	3.196630	0.0	0.0	
887	1	1	5.099020	0	0	3.433987	1.0	0.0	
888	0	3	5.656854	0	0	2.169054	1.0	1.0	

889 rows × 9 columns

In [40]:
```
df.isna().sum()   #no outliers left in age
```

Out[40]:
```
Survived      0
Pclass        0
Age           0
SibSp         0
Parch         0
Fare          0
Sex_male      0
Embarked_Q    0
Embarked_S    0
dtype: int64
```

In [41]:
```
#remove any duplicates if present
df=df.drop_duplicates()
```

In [42]:
```
df  #around 100 rows were removed
```

Out[42]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embar
0	0	3	4.690416	1	0	2.110213	1.0	0.0	
1	1	1	6.164414	1	0	4.280593	0.0	0.0	
2	1	3	5.099020	0	0	2.188856	0.0	0.0	
3	1	1	5.916080	1	0	3.990834	0.0	0.0	
4	0	3	5.916080	0	0	2.202765	1.0	0.0	
...	...	...	...	...	...	...	...	...	
883	0	3	6.244998	0	5	3.405355	0.0	1.0	
885	1	1	4.358899	0	0	3.433987	0.0	0.0	
886	0	3	4.583992	1	2	3.196630	0.0	0.0	
887	1	1	5.099020	0	0	3.433987	1.0	0.0	
888	0	3	5.656854	0	0	2.169054	1.0	1.0	

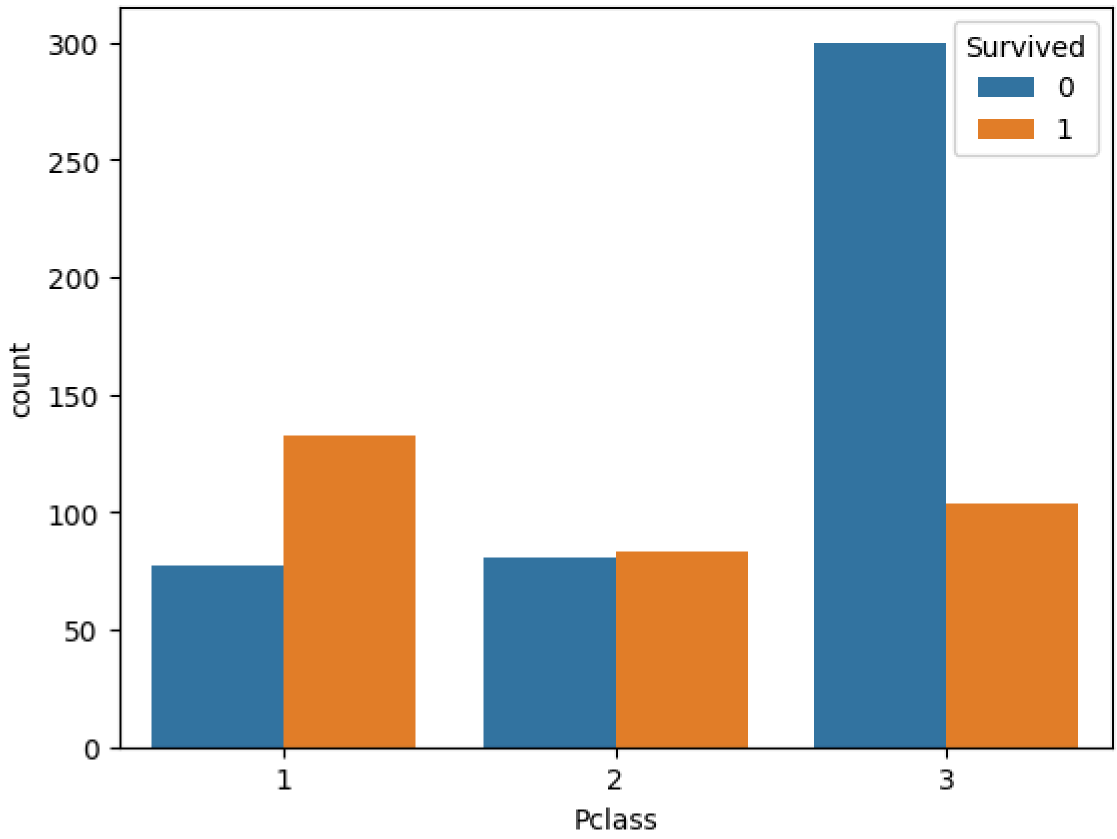
778 rows × 9 columns

In [43]:
```
#we have finished the preprocessing of data ,now we need to do visualization
```

In [44]:
```
import matplotlib.pyplot as plt
import seaborn as sss
```

In [45]:
```
sss.countplot(df,x='Pclass',hue='Survived')
```

Out[45]: <Axes: xlabel='Pclass', ylabel='count'>



In [46]:
```
#we can see 3rd class people have not survived ,so 3rd class is in lower berth o
#1st and second class have survived

df
```

Out[46]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embar
0	0	3	4.690416	1	0	2.110213	1.0	0.0	
1	1	1	6.164414	1	0	4.280593	0.0	0.0	
2	1	3	5.099020	0	0	2.188856	0.0	0.0	
3	1	1	5.916080	1	0	3.990834	0.0	0.0	
4	0	3	5.916080	0	0	2.202765	1.0	0.0	
...	...	...	...	...	...	...	...	...	
883	0	3	6.244998	0	5	3.405355	0.0	1.0	
885	1	1	4.358899	0	0	3.433987	0.0	0.0	
886	0	3	4.583992	1	2	3.196630	0.0	0.0	
887	1	1	5.099020	0	0	3.433987	1.0	0.0	
888	0	3	5.656854	0	0	2.169054	1.0	1.0	

778 rows × 9 columns

In [49]:
```
df['Family']=df['SibSp']+df['Parch']
df['Family']   #we created dataset of people containing family
```

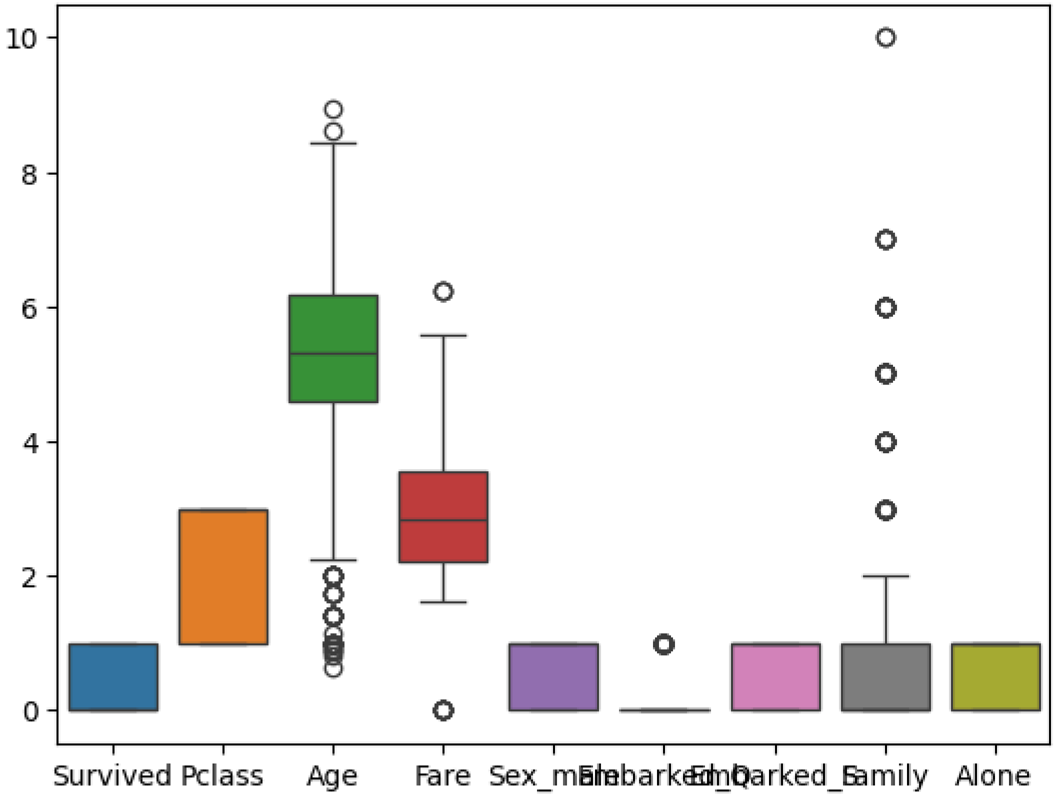
Out[49]:
```
0      1
1      1
2      0
3      1
4      0
      ..
883    5
885    0
886    3
887    0
888    0
Name: Family, Length: 778, dtype: int64
```

In [50]:
```
df['Alone']=df['Family']==0  #create dataset of people who dont have family
```

In [51]:
```
df=df.drop(['SibSp','Parch'],axis=1)
```

In [52]:
```
sss.boxplot(df)
```

Out[52]: `<Axes: >`



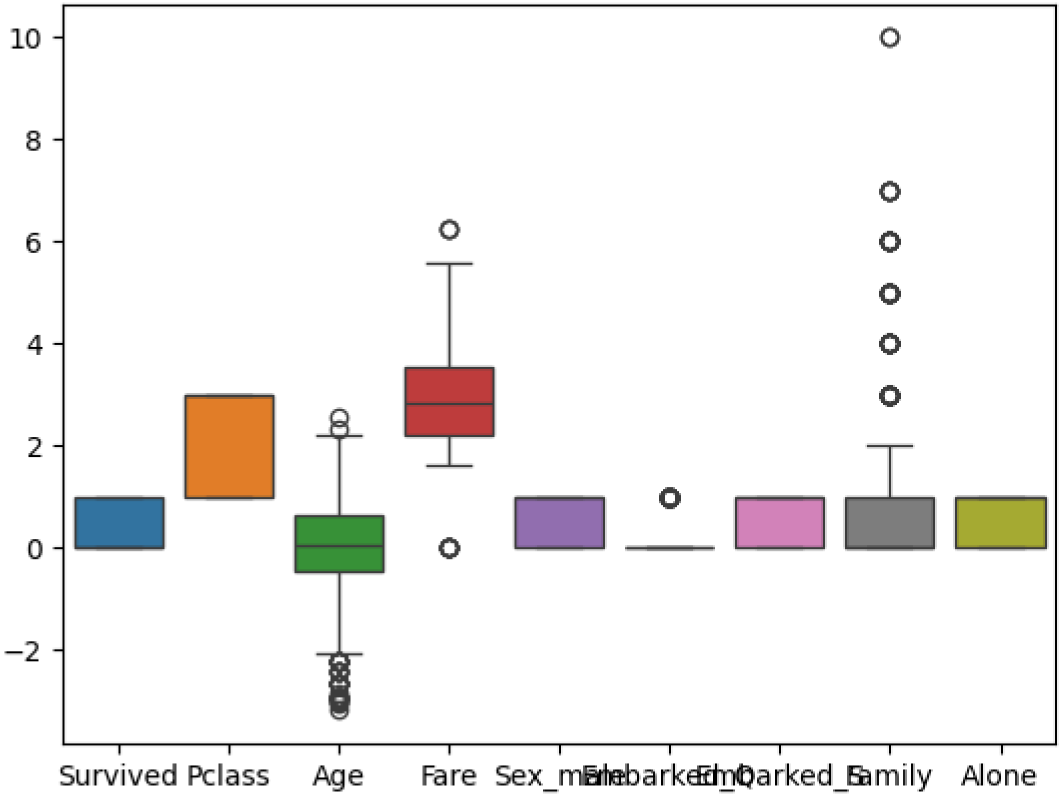
In [53]:
```
#outliers in family ,age,embarked,fare
#standardize the age first

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scale_age=['Age']

df[scale_age]=scaler.fit_transform(df[scale_age])
```

In [54]:
```
sss.boxplot(df)
```

Out[54]: `<Axes: >`



In [55]:
```
rel_mat=df.corr()
rel_with_surv=rel_mat['Survived'].drop('Survived')
```

In [56]:
```
rel_with_surv
```

Out[56]:
```
Pclass        -0.333291
Age           -0.115660
Fare           0.309705
Sex_male      -0.511686
Embarked_Q    -0.038347
Embarked_S    -0.131168
Family         0.015938
Alone         -0.179048
Name: Survived, dtype: float64
```

In [57]:
```
#lets assign a threshold and find values nearest to target variable
threshold=0.2
close_rel_with_surv=rel_with_surv[abs(rel_with_surv)>threshold]
```

In [59]:
```
close_rel_with_surv
```

Out[59]:
```
Pclass      -0.333291
Fare         0.309705
Sex_male    -0.511686
Name: Survived, dtype: float64
```

In [60]:
```
ID=df[['Pclass','Age','Fare','Sex_male']]
D=df['Survived']
```

In [61]:
```
ID
```

Out[61]:

	Pclass	Age	Fare	Sex_male
0	3	-0.383212	2.110213	1.0
1	1	0.638601	4.280593	0.0
2	3	-0.099957	2.188856	0.0
3	1	0.466450	3.990834	0.0
4	3	0.466450	2.202765	1.0
...	...	...	...	...
883	3	0.694464	3.405355	0.0
885	1	-0.613028	3.433987	0.0
886	3	-0.456988	3.196630	0.0
887	1	-0.099957	3.433987	1.0
888	3	0.286748	2.169054	1.0

778 rows × 4 columns

In [62]:
```
D
```

Out[62]:
```
0      0
1      1
2      1
3      1
4      0
      ..
883    0
885    1
886    0
887    1
888    0
Name: Survived, Length: 778, dtype: int64
```

In [63]:
```
from sklearn.model_selection import train_test_split
ID_train,ID_test,D_train,D_test=train_test_split(ID,D,test_size=0.2,random_state
```

In [64]:
```
ID_train
```

Out[64]:

	Pclass	Age	Fare	Sex_male
788	1	1.066956	4.384524	1.0
722	2	1.267117	2.639057	1.0
141	3	-0.238630	2.824351	0.0
388	2	-0.776486	2.564949	0.0
56	2	-0.457969	2.442347	0.0
...	...	...	...	...
72	3	-0.099957	2.737881	1.0
112	3	-0.534529	2.381858	0.0
287	2	0.857887	2.639057	1.0
483	1	-0.168600	4.522649	1.0
108	3	-0.637370	3.224858	0.0

622 rows × 4 columns

In [65]:
```
ID_test
```

Out[65]:

	Pclass	Age	Fare	Sex_male
676	3	-0.693621	2.383400	0.0
667	3	0.911056	2.202765	1.0
614	2	-0.238630	4.189655	0.0
728	3	-0.168600	2.188856	0.0
545	2	-0.613028	3.295837	0.0
...	...	...	...	...
382	1	0.466450	3.970292	0.0
82	1	0.033477	3.873282	1.0
156	3	0.162225	2.202765	1.0
362	3	0.466450	2.085672	1.0
177	2	0.162225	2.639057	1.0

156 rows × 4 columns

In [66]:
```
D_train
```

Out[66]:
```
788    0
722    0
141    1
388    1
56     1
      ..
72     0
112    0
287    1
483    1
108    1
Name: Survived, Length: 622, dtype: int64
```

In [67]:
```
D_test
```

Out[67]:
```
676    1
667    0
614    1
728    0
545    1
      ..
382    1
82     0
156    0
362    0
177    0
Name: Survived, Length: 156, dtype: int64
```

In [80]:
```
#using logistic regression model for training
#first technique
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report

#training the model
model1=LogisticRegression()
model1.fit(ID_train,D_train)
D_pred=model1.predict(ID_test)

#lets determine the model accuracy
accuracy=accuracy_score(D_test,D_pred)
print(accuracy)
```

```
0.782051282051282
```

In [82]:
```
#using decsion tree second method
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matri
from sklearn.model_selection import cross_val_score

model2=DecisionTreeClassifier()
model2.fit(ID_train,D_train)
D_pred=model2.predict(ID_test)

accuracy=accuracy_score(D_test,D_pred)
print(accuracy)
```

```
0.7243589743589743
```

In [85]:

```
!pip install xgboost
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matri
from sklearn.model_selection import cross_val_score

model3=XGBClassifier()
model3.fit(ID_train,D_train)
D_pred=model3.predict(ID_test)

accuracy=accuracy_score(D_test,D_pred)
print(accuracy)
```

```
Collecting xgboost
  Downloading xgboost-2.0.3-py3-none-win_amd64.whl.metadata (2.0 kB)
Requirement already satisfied: numpy in c:\users\ykyas\appdata\local\programs\pyt
hon\python312\lib\site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\users\ykyas\appdata\local\programs\pyt
hon\python312\lib\site-packages (from xgboost) (1.13.0)
Downloading xgboost-2.0.3-py3-none-win_amd64.whl (99.8 MB)
   ---------------------------------------- 0.0/99.8 MB ? eta -:--:--
   ---------------------------------------- 0.0/99.8 MB 660.6 kB/s eta 0:02:31
   ---------------------------------------- 0.4/99.8 MB 5.5 MB/s eta 0:00:19
    --------------------------------------- 1.4/99.8 MB 11.1 MB/s eta 0:00:09
    --------------------------------------- 2.3/99.8 MB 13.2 MB/s eta 0:00:08
   - -------------------------------------- 3.5/99.8 MB 15.8 MB/s eta 0:00:07
   - -------------------------------------- 4.6/99.8 MB 17.5 MB/s eta 0:00:06
   -- ------------------------------------- 5.8/99.8 MB 18.6 MB/s eta 0:00:06
   -- ------------------------------------- 6.8/99.8 MB 19.7 MB/s eta 0:00:05
   --- ------------------------------------ 8.1/99.8 MB 19.8 MB/s eta 0:00:05
   --- ------------------------------------ 9.2/99.8 MB 20.2 MB/s eta 0:00:05
   ---- ----------------------------------- 10.4/99.8 MB 24.2 MB/s eta 0:00:04
   ---- ----------------------------------- 10.6/99.8 MB 21.8 MB/s eta 0:00:05
   ---- ----------------------------------- 11.8/99.8 MB 23.4 MB/s eta 0:00:04
   ----- ---------------------------------- 13.3/99.8 MB 23.4 MB/s eta 0:00:04
   ----- ---------------------------------- 14.4/99.8 MB 24.2 MB/s eta 0:00:04
   ------ --------------------------------- 15.7/99.8 MB 24.3 MB/s eta 0:00:04
   ------ --------------------------------- 17.0/99.8 MB 25.2 MB/s eta 0:00:04
   ------- -------------------------------- 18.4/99.8 MB 24.2 MB/s eta 0:00:04
   ------- -------------------------------- 19.1/99.8 MB 25.2 MB/s eta 0:00:04
   -------- ------------------------------- 20.3/99.8 MB 24.2 MB/s eta 0:00:04
   -------- ------------------------------- 21.3/99.8 MB 26.2 MB/s eta 0:00:03
   -------- ------------------------------- 22.3/99.8 MB 25.1 MB/s eta 0:00:04
   --------- ------------------------------ 23.4/99.8 MB 24.2 MB/s eta 0:00:04
   --------- ------------------------------ 24.8/99.8 MB 25.2 MB/s eta 0:00:03
   ---------- ----------------------------- 26.1/99.8 MB 25.2 MB/s eta 0:00:03
   ---------- ----------------------------- 27.1/99.8 MB 24.2 MB/s eta 0:00:03
   ----------- ---------------------------- 28.1/99.8 MB 24.2 MB/s eta 0:00:03
   ----------- ---------------------------- 29.1/99.8 MB 23.4 MB/s eta 0:00:04
   ------------ --------------------------- 30.3/99.8 MB 23.4 MB/s eta 0:00:03
   ------------ --------------------------- 31.5/99.8 MB 24.2 MB/s eta 0:00:03
   ------------- -------------------------- 32.9/99.8 MB 25.1 MB/s eta 0:00:03
   ------------- -------------------------- 33.8/99.8 MB 25.2 MB/s eta 0:00:03
   -------------- ------------------------- 35.0/99.8 MB 24.2 MB/s eta 0:00:03
   -------------- ------------------------- 36.1/99.8 MB 25.1 MB/s eta 0:00:03
   --------------- ------------------------ 37.5/99.8 MB 25.2 MB/s eta 0:00:03
   --------------- ------------------------ 38.9/99.8 MB 25.2 MB/s eta 0:00:03
   ---------------- ----------------------- 40.0/99.8 MB 26.2 MB/s eta 0:00:03
   ---------------- ----------------------- 41.5/99.8 MB 26.2 MB/s eta 0:00:03
   ----------------- ---------------------- 42.8/99.8 MB 27.3 MB/s eta 0:00:03
   ----------------- ---------------------- 43.8/99.8 MB 27.3 MB/s eta 0:00:03
   ------------------ --------------------- 45.0/99.8 MB 28.4 MB/s eta 0:00:02
   ------------------ --------------------- 46.0/99.8 MB 26.2 MB/s eta 0:00:03
   ------------------ --------------------- 47.0/99.8 MB 27.3 MB/s eta 0:00:02
   ------------------- -------------------- 48.0/99.8 MB 26.2 MB/s eta 0:00:02
   ------------------- -------------------- 49.0/99.8 MB 25.2 MB/s eta 0:00:03
   -------------------- ------------------- 50.4/99.8 MB 25.2 MB/s eta 0:00:02
   -------------------- ------------------- 51.7/99.8 MB 25.2 MB/s eta 0:00:02
   --------------------- ------------------ 52.9/99.8 MB 24.2 MB/s eta 0:00:02
   --------------------- ------------------ 54.4/99.8 MB 26.2 MB/s eta 0:00:02
   ---------------------- ----------------- 55.7/99.8 MB 27.3 MB/s eta 0:00:02
   ---------------------- ----------------- 57.2/99.8 MB 27.3 MB/s eta 0:00:02
   ----------------------- ---------------- 58.5/99.8 MB 28.5 MB/s eta 0:00:02
```

```
   ------------------------ --------------- 59.9/99.8 MB 29.7 MB/s eta 0:00:02
   ------------------------ --------------- 61.1/99.8 MB 29.7 MB/s eta 0:00:02
   ------------------------- -------------- 62.6/99.8 MB 29.7 MB/s eta 0:00:02
   ------------------------- -------------- 63.9/99.8 MB 29.7 MB/s eta 0:00:02
   -------------------------- ------------- 65.1/99.8 MB 28.4 MB/s eta 0:00:02
   -------------------------- ------------- 66.5/99.8 MB 28.4 MB/s eta 0:00:02
   --------------------------- ------------ 67.5/99.8 MB 27.3 MB/s eta 0:00:02
   --------------------------- ------------ 68.4/99.8 MB 26.2 MB/s eta 0:00:02
   ---------------------------- ----------- 69.9/99.8 MB 27.3 MB/s eta 0:00:02
   ---------------------------- ----------- 70.8/99.8 MB 26.2 MB/s eta 0:00:02
   ---------------------------- ----------- 71.9/99.8 MB 26.2 MB/s eta 0:00:02
   ----------------------------- ---------- 73.3/99.8 MB 26.2 MB/s eta 0:00:02
   ----------------------------- ---------- 74.6/99.8 MB 26.2 MB/s eta 0:00:01
   ------------------------------ --------- 75.6/99.8 MB 26.2 MB/s eta 0:00:01
   ------------------------------ --------- 76.8/99.8 MB 24.3 MB/s eta 0:00:01
   ------------------------------- -------- 77.7/99.8 MB 25.1 MB/s eta 0:00:01
   ------------------------------- -------- 78.9/99.8 MB 24.2 MB/s eta 0:00:01
   -------------------------------- ------- 80.3/99.8 MB 26.2 MB/s eta 0:00:01
   -------------------------------- ------- 81.6/99.8 MB 26.2 MB/s eta 0:00:01
   --------------------------------- ------ 82.6/99.8 MB 26.2 MB/s eta 0:00:01
   --------------------------------- ------ 83.8/99.8 MB 26.2 MB/s eta 0:00:01
   --------------------------------- ------ 84.4/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------- ----- 85.8/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------- ----- 86.9/99.8 MB 24.2 MB/s eta 0:00:01
   ----------------------------------- ---- 88.0/99.8 MB 25.1 MB/s eta 0:00:01
   ----------------------------------- ---- 89.2/99.8 MB 25.2 MB/s eta 0:00:01
   ------------------------------------ --- 90.2/99.8 MB 24.2 MB/s eta 0:00:01
   ------------------------------------ --- 91.8/99.8 MB 24.2 MB/s eta 0:00:01
   ------------------------------------- -- 92.9/99.8 MB 25.2 MB/s eta 0:00:01
   ------------------------------------- -- 94.0/99.8 MB 24.2 MB/s eta 0:00:01
   -------------------------------------- - 95.3/99.8 MB 25.2 MB/s eta 0:00:01
   -------------------------------------- - 96.6/99.8 MB 26.2 MB/s eta 0:00:01
   ---------------------------------------  97.3/99.8 MB 26.2 MB/s eta 0:00:01
   ---------------------------------------  97.8/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------  99.3/99.8 MB 25.1 MB/s eta 0:00:01
   ---------------------------------------  99.7/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------  99.7/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------  99.7/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------  99.7/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------  99.7/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------  99.7/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------  99.7/99.8 MB 24.2 MB/s eta 0:00:01
   ---------------------------------------- 99.8/99.8 MB 13.3 MB/s eta 0:00:00
Installing collected packages: xgboost
Successfully installed xgboost-2.0.3
0.7692307692307693
```

In [93]:

```python
#using radnom forest method
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matri
from sklearn.model_selection import cross_val_score

model4=RandomForestClassifier()
model4.fit(ID_train,D_train)

D_pred=model4.predict(ID_test)

accuracy=accuracy_score(D_test,D_pred)
print(accuracy)
```

```python
print(classification_report(D_test,D_pred))
```

```
0.782051282051282
              precision    recall  f1-score   support

           0       0.78      0.87      0.82        91
           1       0.78      0.66      0.72        65

    accuracy                           0.78       156
   macro avg       0.78      0.76      0.77       156
weighted avg       0.78      0.78      0.78       156
```

In [94]:

```python
#at the moment we have random forest giving  best accuracy
```

In [ ]: