In [1]:
```
import pandas as pd #data manipulation and analysis
import numpy as np # conjunction with pandas for data manipulation and analysis.
import warnings
warnings.filterwarnings("ignore") #ignore null values
import seaborn as sns #create a plot
import matplotlib.pyplot as plt # create a graph
```

In [2]:
```
data=pd.read_csv("Titanic Dataset.csv")
data
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [3]:
```
data.head(5) #The head() function is used to display the first few rows of a dataframe.
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:
```
data.tail(5) #The tail() function in python is used to display the last few rows of a dataframe.
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

In [5]:
```
data.describe() #The describe() method in python is used to generate summary statistics for a dataframe.
```

Out[5]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [6]:
```
data.info() #he info() function is a commonly used to know the general information about a dataframe.
```

```
<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 [7]:
```
data.shape
```

Out[7]:
```
(891, 12)
```

In [8]:
```
data.isnull().sum() #count the number of missing (null or NaN) values in each column of a DataFrame
```

Out[8]:
```
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
```

In [9]:
```
list(data)
```

Out[9]:
```
['PassengerId',
 'Survived',
 'Pclass',
 'Name',
 'Sex',
 'Age',
 'SibSp',
 'Parch',
 'Ticket',
 'Fare',
 'Cabin',
 'Embarked']
```

In [10]:
```
data['Age'].unique() #unique values present in the 'Age' column of a DataFrame
```

Out[10]:
```
array([22.  , 38.  , 26.  , 35.  ,   nan, 54.  ,  2.  , 27.  , 14.  ,
        4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,
        8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  ,
       49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  ,
       16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  ,
       71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 ,
       51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  ,
       45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  ,
       60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  ,
       70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

In [11]:
```
data1=data.drop(columns=['PassengerId','Cabin','Ticket','Name','SibSp','Parch'])# removes the columns
data1
```

Out[11]:

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

891 rows × 6 columns

In [12]:
```
data1['Sex'] = data1['Sex'].map({'male':1,'female':0})#replace the values in the 'Sex' column of the DataFrame data1 with
data1
```

Out[12]:

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

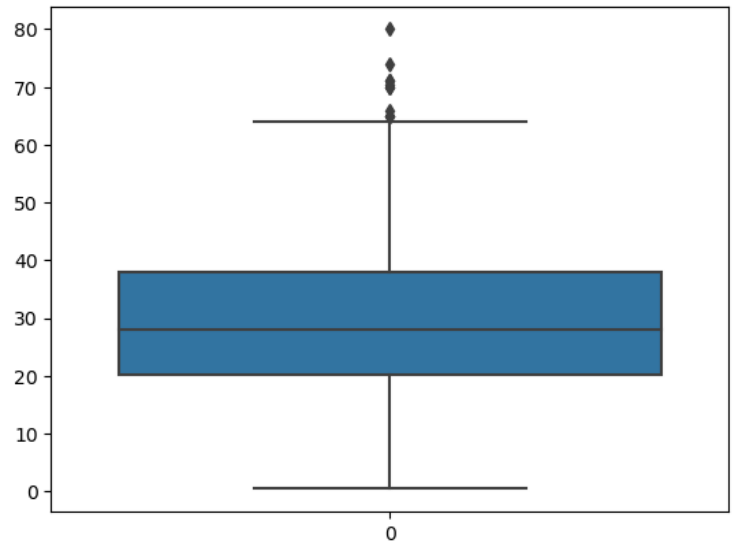
891 rows × 6 columns

In [13]:
```
data1.isnull().sum() #afer droping will use isnull:-no null values
```

Out[13]:
```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
Embarked      2
dtype: int64
```

In [14]:
```
sns.boxplot(data.Age) #to create a boxplot for the 'Age' column in the DataFrame
```

Out[14]: <Axes: >



In [15]:
```
data1['Age'] = data1['Age'].mask(data1['Age']>60,60)
```

In [16]:
```
data1['Pclass'] = data1['Pclass'].map({1:'F', 2:'S', 3:'Third'}) #mapping the numeric values
data1
```

Out[16]:

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

891 rows × 6 columns

In [17]:
```
data1=pd.get_dummies(data1) #convert categorical variable(s) into dummy/indicator variables.
data1
```

Out[17]:

	Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
0	0	1	22.0	7.2500	False	False	True	False	False	True
1	1	0	38.0	71.2833	True	False	False	True	False	False
2	1	0	26.0	7.9250	False	False	True	False	False	True
3	1	0	35.0	53.1000	True	False	False	False	False	True
4	0	1	35.0	8.0500	False	False	True	False	False	True
...	...	...	...	...	...	...	...	...	...	...
886	0	1	27.0	13.0000	False	True	False	False	False	True
887	1	0	19.0	30.0000	True	False	False	False	False	True
888	0	0	NaN	23.4500	False	False	True	False	False	True
889	1	1	26.0	30.0000	True	False	False	True	False	False
890	0	1	32.0	7.7500	False	False	True	False	True	False

891 rows × 10 columns

In [18]:
```
colnames=list(data1)
colnames
```

Out[18]:
```
['Survived',
 'Sex',
 'Age',
 'Fare',
 'Pclass_F',
 'Pclass_S',
 'Pclass_Third',
 'Embarked_C',
 'Embarked_Q',
 'Embarked_S']
```

In [19]:
```
from sklearn.impute import KNNImputer #imputing missing values in a dataset using the k-nearest neighbors approach.
```

In [20]:
```
imputer = KNNImputer(n_neighbors=3) # replaces missing values, it will consider the values of the three nearest neighbors
```

In [21]:
```
data_filled = imputer.fit_transform(data1)
```

In [22]:
```
data1 = pd.DataFrame(data=data_filled, columns = colnames)
```

In [23]:
```
data1['Age'].unique()
```

Out[23]:
```
array([22.        , 38.        , 26.        , 35.        , 53.5       ,
       54.        ,  2.        , 27.        , 14.        ,  4.        ,
       58.        , 20.        , 39.        , 55.        , 35.66666667,
       31.        , 16.66666667, 34.        , 15.        , 28.        ,
        8.        , 38.5       , 19.        , 40.        , 26.97333333,
       18.        , 60.        , 42.        , 23.66666667, 21.        ,
       32.16666667,  3.        , 25.33333333, 36.        , 18.66666667,
        7.        , 49.        , 29.        , 43.        , 28.5       ,
        5.        , 11.        , 45.        , 17.        , 32.        ,
       16.        , 25.        ,  0.83      , 30.        , 33.        ,
       23.        , 24.        , 46.        , 59.        , 37.        ,
       24.33333333, 21.33333333, 47.        , 14.5       , 32.5       ,
       12.        , 14.66666667,  9.        , 36.5       , 51.        ,
       55.5       , 40.5       , 34.33333333, 28.33333333, 44.        ,
        1.        , 55.66666667, 56.        , 50.        , 48.33333333,
       45.5       , 20.5       , 29.33333333, 25.83333333, 41.        ,
       55.33333333, 52.        , 37.16666667, 45.33333333, 31.66666667,
       23.5       , 46.33333333, 38.33333333,  0.92      , 43.66666667,
       20.33333333, 39.66666667, 35.33333333, 21.66666667, 10.        ,
       26.33333333, 13.        , 22.33333333, 48.        , 28.83333333,
        0.75      , 23.33333333, 31.83333333, 23.16666667, 42.33333333,
       24.66666667, 32.66666667, 31.16666667, 28.66666667, 34.5       ,
       53.        , 16.5       , 33.66666667, 22.66666667, 57.        ,
       24.5       , 22.16666667, 58.66666667,  6.        ,  0.67      ,
       30.5       , 50.33333333,  0.42      , 38.66666667])
```

In [24]:
```
y=data1['Survived']
x=data1.drop('Survived',axis=1)
```

In [25]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.33, random_state=42)
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
```

Out[25]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [26]:
```
y_pred= classifier.predict(x_test)
y_pred
```

Out[26]:
```
array([0., 0., 0., 0., 0., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1.,
       1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.,
       0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0.,
       0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1.,
       0., 0., 1., 1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 1., 0.,
       1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0.,
       1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 1., 1., 0., 1., 0., 1., 0., 0., 1., 1., 1., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
       0., 0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 1., 0., 1., 0.,
       1., 1., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1.,
       1., 0., 0., 1., 0., 0.])
```

In [27]:
```
from sklearn.metrics import confusion_matrix #
con= confusion_matrix(y_test, y_pred)
con
```

Out[27]:
```
array([[147,  28],
       [ 58,  62]], dtype=int64)
```

In [28]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[28]:
```
0.7084745762711865
```

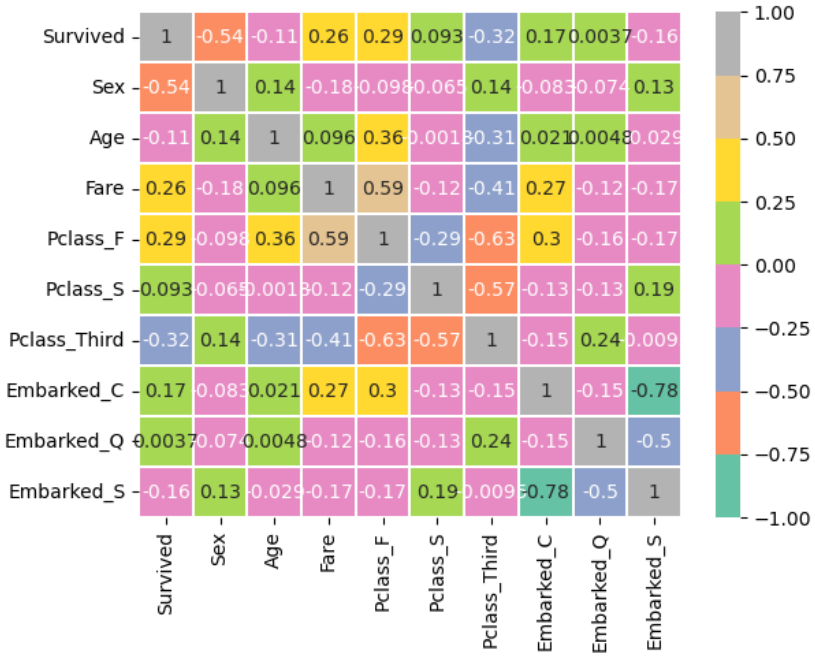
In [29]:
```
cor_mat = data1.corr()
cor_mat
```

Out[29]:

	Survived	Sex	Age	Fare	Pclass_F	Pclass_S	Pclass_Third	Embarked_C	Embarked_Q	Embarked_S
Survived	1.000000	-0.543351	-0.107151	0.257307	0.285904	0.093349	-0.322308	0.168240	0.003650	-0.155660
Sex	-0.543351	1.000000	0.136861	-0.182333	-0.098013	-0.064746	0.137143	-0.082853	-0.074115	0.125722
Age	-0.107151	0.136861	1.000000	0.095942	0.356562	-0.001802	-0.305749	0.021250	0.004760	-0.028685
Fare	0.257307	-0.182333	0.095942	1.000000	0.591711	-0.118557	-0.413333	0.269335	-0.117216	-0.166603
Pclass_F	0.285904	-0.098013	0.356562	0.591711	1.000000	-0.288585	-0.626738	0.296423	-0.155342	-0.170379
Pclass_S	0.093349	-0.064746	-0.001802	-0.118557	-0.288585	1.000000	-0.565210	-0.125416	-0.127301	0.192061
Pclass_Third	-0.322308	0.137143	-0.305749	-0.413333	-0.626738	-0.565210	1.000000	-0.153329	0.237449	-0.009511
Embarked_C	0.168240	-0.082853	0.021250	0.269335	0.296423	-0.125416	-0.153329	1.000000	-0.148258	-0.778359
Embarked_Q	0.003650	-0.074115	0.004760	-0.117216	-0.155342	-0.127301	0.237449	-0.148258	1.000000	-0.496624
Embarked_S	-0.155660	0.125722	-0.028685	-0.166603	-0.170379	0.192061	-0.009511	-0.778359	-0.496624	1.000000

In [30]:
```
sns.heatmap(cor_mat,vmax=1,vmin=-1,annot=True,linewidth=0.01,cmap='Set2')
```

Out[30]: <Axes: >



In [ ]:
```

```