



DATA SCIENCE AND VISUALIZATION LABORATORY

Lab Journal



NAME :- OJUS P. JAISWAL

YEAR & DIV :- TE A

SR. NO. :- 98

ROLL NO. :- TACO19108

SEAT NO. :- S191094290

In []:

```
# Practical No. 1
# Data Science and Visualization

'''Access an open source dataset "Titanic".
Apply pre-processing techniques on the raw dataset.'''
```

In [2]:

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

In [3]:

```
pd.__version__
```

Out[3]:

```
'1.2.4'
```

In [4]:

```
np.__version__
```

Out[4]:

```
'1.20.1'
```

In [5]:

```
sns.__version__
```

Out[5]:

```
'0.11.1'
```

In [6]:

```
sns.get_dataset_names()
```

Out[6]:

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'taxis',
 'tips',
 'titanic']
```

In [7]:

```
dataset = sns.load_dataset('titanic')
```

```
In [0]:
```

```
dataset
```

```
Out[8]:
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no

891 rows x 15 columns



```
In [9]:
```

```
df = pd.read_csv('https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv')
```

```
In [10]:
```

```
df
```

```
Out[10]:
```

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500
...
882	0	2	Rev. Juozas Montvila	male	27.0	0	0	13.0000
883	1	1	Miss. Margaret Edith Graham	female	19.0	0	0	30.0000
884	0	3	Miss. Catherine Helen Johnston	female	7.0	1	2	23.4500
885	1	1	Mr. Karl Howell Behr	male	26.0	0	0	30.0000
886	0	3	Mr. Patrick Dooley	male	32.0	0	0	7.7500

887 rows x 8 columns

```
In [11]:
```

```
import os
os.getcwd()
os.chdir('C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DS\\Data Set')
os.getcwd()
```

```
Out[11]:
'C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DS\\Data Set'
```

```
In [12]:
df = pd.read_csv('titanic.csv')
```

```
In [13]:
df
```

Out[13]:

Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare	
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500
...
882	0	2	Rev. Juozas Montvila	male	27.0	0	0	13.0000
883	1	1	Miss. Margaret Edith Graham	female	19.0	0	0	30.0000
884	0	3	Miss. Catherine Helen Johnston	female	7.0	1	2	23.4500
885	1	1	Mr. Karl Howell Behr	male	26.0	0	0	30.0000
886	0	3	Mr. Patrick Dooley	male	32.0	0	0	7.7500

887 rows x 8 columns

```
In [14]:
df.columns
```

```
Out[14]:
Index(['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'Siblings/Spouses Aboard',
      'Parents/Children Aboard', 'Fare'],
      dtype='object')
```

```
In [15]:
df.shape
```

```
Out[15]:
(887, 8)
```

```
In [16]:
dataset.shape
```

```
Out[16]:
(891, 15)
```

```
In [17]:
dataset.columns
```

```
Out[17]:
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
      'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
```

```
'alive', 'alone'],
dtype='object')
```

In [18]:

```
df.head()
```

Out[18]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500

In [19]:

```
dataset.head()
```

Out[19]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [20]:

```
df.tail()
```

Out[20]:

	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
882	0	2	Rev. Juozas Montvila	male	27.0	0	0	13.00
883	1	1	Miss. Margaret Edith Graham	female	19.0	0	0	30.00
884	0	3	Miss. Catherine Helen Johnston	female	7.0	1	2	23.45
885	1	1	Mr. Karl Howell Behr	male	26.0	0	0	30.00
886	0	3	Mr. Patrick Dooley	male	32.0	0	0	7.75

In [21]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 887 entries, 0 to 886
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Survived    887 non-null   int64
1   Pclass      887 non-null   int64
2   Name        887 non-null   object
3   Sex         887 non-null   object
```

```
4   Age      887 non-null    float64
5   Siblings/Spouses Aboard 887 non-null    int64
6   Parents/Children Aboard 887 non-null    int64
7   Fare     887 non-null    float64
```

```
dtypes: float64(2), int64(4), object(2)
memory usage: 55.6+ KB
```

In [22]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  -
0   survived    891 non-null    int64
1   pclass      891 non-null    int64
2   sex         891 non-null    object
3   age         714 non-null    float64
4   sibsp       891 non-null    int64
5   parch       891 non-null    int64
6   fare        891 non-null    float64
7   embarked    889 non-null    object
8   class       891 non-null    category
9   who         891 non-null    object
10  adult_male  891 non-null    bool
11  deck        203 non-null    category
12  embark_town 889 non-null    object
13  alive       891 non-null    object
14  alone       891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [23]:

```
dataset.describe()
```

Out[23]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [24]:

```
df.describe()
```

Out[24]:

	Survived	Pclass	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
count	887.000000	887.000000	887.000000	887.000000	887.000000	887.000000
mean	0.385569	2.305524	29.471443	0.525366	0.383315	32.30542
std	0.487004	0.836662	14.121908	1.104669	0.807466	49.78204
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.250000	0.000000	0.000000	7.92500

50%	Survived	Pclass	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.13750
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

In [25]:

```
df.count()
```

Out[25]:

```
Survived      887
Pclass        887
Name          887
Sex           887
Age           887
Siblings/Spouses Aboard  887
Parents/Children Aboard  887
Fare          887
dtype: int64
```

In [26]:

```
dataset.count()
```

Out[26]:

```
survived      891
pclass        891
sex           891
age           714
sibsp         891
parch         891
fare          891
embarked      889
class         891
who           891
adult_male    891
deck         203
embark_town   889
alive         891
alone         891
dtype: int64
```

In [27]:

```
dataset.isnull().sum()
```

Out[27]:

```
survived      0
pclass        0
sex           0
age          177
sibsp         0
parch         0
fare          0
embarked      2
class         0
who           0
adult_male    0
deck         688
embark_town   2
alive         0
alone         0
dtype: int64
```

In [28]:

```
dataset = dataset.drop('deck', axis = 1)
```

In [29]:

```
dataset.isnull().sum()
```

Out[29]:

```
survived      0
pclass        0
sex           0
age          177
sibsp         0
parch         0
fare          0
embarked      2
class         0
who           0
adult_male    0
embark_town   2
alive         0
alone         0
dtype: int64
```

In [30]:

```
dataset['age'] = dataset['age'].fillna(dataset['age'].median())
```

In [31]:

```
dataset.isnull().sum()
```

Out[31]:

```
survived      0
pclass        0
sex           0
age           0
sibsp         0
parch         0
fare          0
embarked      2
class         0
who           0
adult_male    0
embark_town   2
alive         0
alone         0
dtype: int64
```

In [32]:

```
dataset['embarked'].mode()[0]
```

Out[32]:

```
'S'
```

In [33]:

```
dataset['embark_town'].mode()[0]
```

Out[33]:

```
'Southampton'
```

In [34]:

```
dataset['embarked'] = dataset['embarked'].fillna(
    dataset['embarked'].mode()[0])
```

In [35]:

```
dataset['embark_town'] = dataset['embark_town'].fillna(
    dataset['embark_town'].mode()[0])
```


In [36]:

```
dataset.isnull().sum()
```

Out[36]:

```
survived      0
pclass        0
sex           0
age           0
sibsp         0
parch         0
fare          0
embarked      0
class         0
who           0
adult_male    0
embark_town   0
alive         0
alone         0
dtype: int64
```

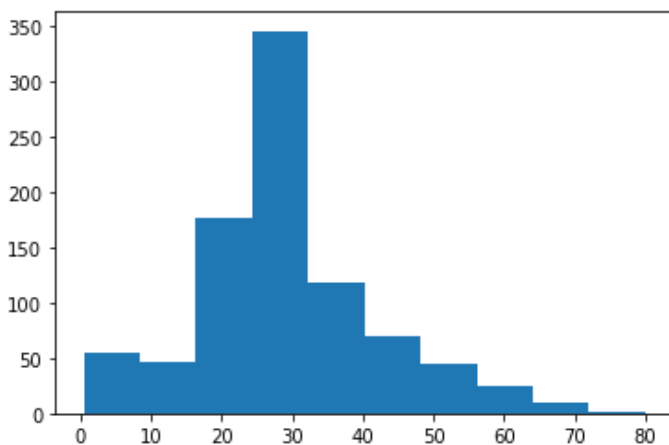
In [37]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  -
 0   survived      891 non-null   int64  
 1   pclass        891 non-null   int64  
 2   sex           891 non-null   object  
 3   age           891 non-null   float64 
 4   sibsp         891 non-null   int64  
 5   parch         891 non-null   int64  
 6   fare          891 non-null   float64 
 7   embarked      891 non-null   object  
 8   class         891 non-null   category
 9   who           891 non-null   object  
10  adult_male    891 non-null   bool    
11  embark_town   891 non-null   object  
12  alive         891 non-null   object  
13  alone         891 non-null   bool    
dtypes: bool(2), category(1), float64(2), int64(4), object(5)
memory usage: 79.4+ KB
```

In [38]:

```
plt.hist(dataset['age']);
```

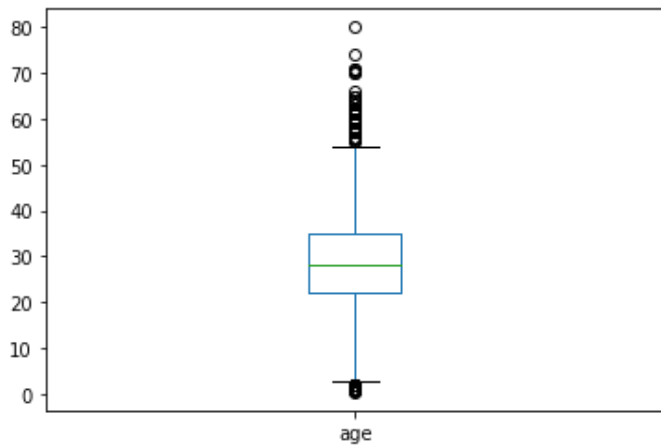


In [39]:

```
dataset['age'].plot(kind='box')
```

Out[39]:

<AxesSubplot:>

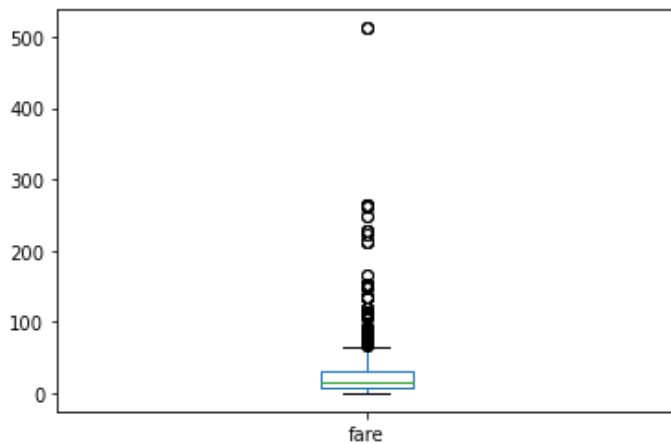


In [40]:

```
dataset['fare'].plot(kind='box')
```

Out[40]:

<AxesSubplot:>



In [41]:

```
dataset.info()
```

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 14 columns):

#	Column	Non-Null Count	Dtype
0	survived	891 non-null	int64
1	pclass	891 non-null	int64
2	sex	891 non-null	object
3	age	891 non-null	float64
4	sibsp	891 non-null	int64
5	parch	891 non-null	int64
6	fare	891 non-null	float64
7	embarked	891 non-null	object
8	class	891 non-null	category
9	who	891 non-null	object
10	adult_male	891 non-null	bool
11	embark_town	891 non-null	object
12	alive	891 non-null	object
13	alone	891 non-null	bool

dtypes: bool(2), category(1), float64(2), int64(4), object(5)

memory usage: 79.4+ KB

In [42]:

```
pd.get_dummies(dataset).head()
```

Out[42]:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone	sex_female	sex_male	...	class_Second	class_Third	who
0	0	3	22.0	1	0	7.2500	True	False	0	1	...	0	1	
1	1	1	38.0	1	0	71.2833	False	False	1	0	...	0	0	
2	1	3	26.0	0	0	7.9250	False	True	1	0	...	0	1	
3	1	1	35.0	1	0	53.1000	False	False	1	0	...	0	0	
4	0	3	35.0	0	0	8.0500	True	True	0	1	...	0	1	

5 rows x 24 columns



In [43]:

```
from sklearn.model_selection import train_test_split
```

In [44]:

```
train, test = train_test_split(dataset, test_size=0.20)
```

In [45]:

```
len(dataset)
```

Out[45]:

891

In [46]:

```
len(train)
```

Out[46]:

712

In [47]:

```
len(test)
```

Out[47]:

179

In []:

In []:

```
# Practical No. 2
# Data Science and Visualization

'''Build training and testing dataset of assignment 1 to predict the probability of a survival
of a person based on gender, age and passenger-class.'''
```

In [2]:

```
import numpy as np
import seaborn as sns
import pandas as pd
```

In [3]:

```
ds = sns.load_dataset('titanic')
```

In [4]:

```
ds.head()
```

Out[4]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [5]:

```
len(ds)
```

Out[5]:

891

In [6]:

```
ds['age'] = ds['age'].fillna(ds['age'].median())
# Data Cleaning

x = ds['age'].values #input
y = ds['survived'] #output
```

In [7]:

```
x = x.reshape(-1,1)
x.shape
```

Out[7]:

(891, 1)

In [8]:

```
from sklearn.model_selection import train_test_split
```

In [9]:

```
x_train, x_test, y_train, y_test = train_test_split(
```

```
x, y, random_state = 0, test_size = 0.25)  
# build train and test
```

In [10]:

```
len(x_train)
```

Out[10]:

668

In [11]:

```
len(x_test)
```

Out[11]:

223

In [12]:

```
from collections import Counter
```

In [13]:

```
Counter(y)
```

Out[13]:

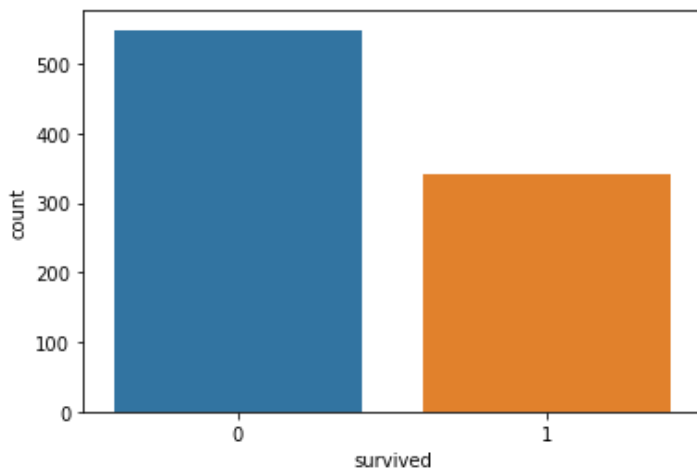
Counter({0: 549, 1: 342})

In [14]:

```
sns.countplot(x=y)
```

Out[14]:

<AxesSubplot:xlabel='survived', ylabel='count'>



In [15]:

```
from sklearn.naive_bayes import GaussianNB
```

In [16]:

```
model = GaussianNB()
```

In [17]:

```
# train the algorithm with given training data  
model.fit(x_train, y_train)
```

Out[17]:

GaussianNB()

In [18]:

```
y_pred = model.predict_proba(x_test)
```

In [19]:

```
y_pred
```

Out[19]:

```
array([[0.62876821, 0.37123179],
       [0.62876821, 0.37123179],
       [0.51657653, 0.48342347],
       [0.62876821, 0.37123179],
       [0.63148867, 0.36851133],
       [0.62876821, 0.37123179],
       [0.64689984, 0.35310016],
       [0.63625703, 0.36374297],
       [0.6192395 , 0.3807605 ],
       [0.62876821, 0.37123179],
       [0.62264556, 0.37735444],
       [0.64689984, 0.35310016],
       [0.62876821, 0.37123179],
       [0.51657653, 0.48342347],
       [0.61560095, 0.38439905],
       [0.56600603, 0.43399397],
       [0.61172787, 0.38827213],
       [0.59385049, 0.40614951],
       [0.64314813, 0.35685187],
       [0.45943048, 0.54056952],
       [0.58877545, 0.41122455],
       [0.60761825, 0.39238175],
       [0.62876821, 0.37123179],
       [0.62876821, 0.37123179],
       [0.60761825, 0.39238175],
       [0.64689984, 0.35310016],
       [0.63830829, 0.36169171],
       [0.60761825, 0.39238175],
       [0.6192395 , 0.3807605 ],
       [0.47403058, 0.52596942],
       [0.64013966, 0.35986034],
       [0.63835894, 0.36164106],
       [0.62876821, 0.37123179],
       [0.62876821, 0.37123179],
       [0.63148867, 0.36851133],
       [0.63830829, 0.36169171],
       [0.64658851, 0.35341149],
       [0.62876821, 0.37123179],
       [0.6192395 , 0.3807605 ],
       [0.62884219, 0.37115781],
       [0.60772869, 0.39227131],
       [0.6192395 , 0.3807605 ],
       [0.62876821, 0.37123179],
       [0.56600603, 0.43399397],
       [0.64314813, 0.35685187],
       [0.62876821, 0.37123179],
       [0.62876821, 0.37123179],
       [0.58877545, 0.41122455],
       [0.64657737, 0.35342263],
       [0.6340466 , 0.3659534 ],
       [0.62876821, 0.37123179],
       [0.61172787, 0.38827213],
       [0.49173773, 0.50826227],
       [0.59385049, 0.40614951],
       [0.62876821, 0.37123179],
       [0.61560095, 0.38439905],
       [0.59880446, 0.40119554],
       [0.5463121 , 0.4536879 ],
       [0.53924847, 0.46075153],
       [0.62876821, 0.37123179],
       [0.60761825, 0.39238175],
       [0.63148867, 0.36851133],
       [0.62884219, 0.37115781]
```

[0.62876821, 0.37123179],
[0.62876821, 0.37123179],
[0.62582114, 0.37417886],
[0.63830829, 0.36169171],
[0.64018459, 0.35981541],
[0.62876821, 0.37123179],
[0.47403058, 0.52596942],
[0.59385049, 0.40614951],
[0.61560095, 0.38439905],
[0.64179176, 0.35820824],
[0.62876821, 0.37123179],
[0.62876821, 0.37123179],
[0.61744941, 0.38255059],
[0.63398437, 0.36601563],
[0.59385049, 0.40614951],
[0.64689984, 0.35310016],
[0.64175253, 0.35824747],
[0.62876821, 0.37123179],
[0.62876821, 0.37123179],
[0.53193679, 0.46806321],
[0.62264556, 0.37735444],
[0.57788703, 0.42211297],
[0.64435549, 0.35564451],
[0.6032701 , 0.3967299],
[0.64690541, 0.35309459],
[0.59397985, 0.40602015],
[0.64175253, 0.35824747],
[0.58877545, 0.41122455],
[0.62876821, 0.37123179],
[0.59385049, 0.40614951],
[0.48299524, 0.51700476],
[0.64605821, 0.35394179],
[0.53193679, 0.46806321],
[0.62876821, 0.37123179],
[0.61560095, 0.38439905],
[0.57788703, 0.42211297],
[0.60772869, 0.39227131],
[0.62582114, 0.37417886],
[0.56600603, 0.43399397],
[0.62884219, 0.37115781],
[0.59385049, 0.40614951],
[0.58877545, 0.41122455],
[0.63148867, 0.36851133],
[0.64314813, 0.35685187],
[0.62876821, 0.37123179],
[0.64700915, 0.35299085],
[0.6192395 , 0.3807605],
[0.62876821, 0.37123179],
[0.64179176, 0.35820824],
[0.60761825, 0.39238175],
[0.63830829, 0.36169171],
[0.62876821, 0.37123179],
[0.64432756, 0.35567244],
[0.62876821, 0.37123179],
[0.61560095, 0.38439905],
[0.58877545, 0.41122455],
[0.64689984, 0.35310016],
[0.64531408, 0.35468592],
[0.62876821, 0.37123179],
[0.62876821, 0.37123179],
[0.59868146, 0.40131854],
[0.64604149, 0.35395851],
[0.64013966, 0.35986034],
[0.61569902, 0.38430098],
[0.62876821, 0.37123179],
[0.6192395 , 0.3807605],
[0.62876821, 0.37123179],
[0.63631345, 0.36368655],
[0.6032701 , 0.3967299],
[0.6032701 , 0.3967299],
[0.6192395 , 0.3807605],
[0.59868146, 0.40131854],
[0.58891121 0.41108879]

[0.6032701 , 0.3967299],
[0.62876821, 0.37123179],
[0.64689984, 0.35310016],
[0.64605821, 0.35394179],
[0.61172787, 0.38827213],
[0.62876821, 0.37123179],
[0.64013966, 0.35986034],
[0.62876821, 0.37123179],
[0.58877545, 0.41122455],
[0.62876821, 0.37123179],
[0.61560095, 0.38439905],
[0.63398437, 0.36601563],
[0.64690541, 0.35309459],
[0.63835894, 0.36164106],
[0.62876821, 0.37123179],
[0.64531408, 0.35468592],
[0.63398437, 0.36601563],
[0.64529175, 0.35470825],
[0.63148867, 0.36851133],
[0.58345477, 0.41654523],
[0.62264556, 0.37735444],
[0.64531408, 0.35468592],
[0.59880446, 0.40119554],
[0.62876821, 0.37123179],
[0.46252279, 0.53747721],
[0.62876821, 0.37123179],
[0.59868146, 0.40131854],
[0.62884219, 0.37115781],
[0.64013966, 0.35986034],
[0.57207109, 0.42792891],
[0.6192395 , 0.3807605],
[0.63631345, 0.36368655],
[0.60761825, 0.39238175],
[0.61560095, 0.38439905],
[0.58345477, 0.41654523],
[0.63631345, 0.36368655],
[0.64604149, 0.35395851],
[0.62876821, 0.37123179],
[0.62876821, 0.37123179],
[0.55969127, 0.44030873],
[0.62876821, 0.37123179],
[0.63625703, 0.36374297],
[0.61172787, 0.38827213],
[0.59385049, 0.40614951],
[0.62876821, 0.37123179],
[0.63625703, 0.36374297],
[0.63625703, 0.36374297],
[0.59868146, 0.40131854],
[0.6032701 , 0.3967299],
[0.64486164, 0.35513836],
[0.60761825, 0.39238175],
[0.62876821, 0.37123179],
[0.62876821, 0.37123179],
[0.62264556, 0.37735444],
[0.6192395 , 0.3807605],
[0.6032701 , 0.3967299],
[0.63625703, 0.36374297],
[0.57207109, 0.42792891],
[0.62876821, 0.37123179],
[0.62876821, 0.37123179],
[0.58359697, 0.41640303],
[0.62876821, 0.37123179],
[0.46485056, 0.53514944],
[0.64175253, 0.35824747],
[0.64018459, 0.35981541],
[0.58877545, 0.41122455],
[0.62876821, 0.37123179],
[0.55330133, 0.44669867],
[0.56600603, 0.43399397],
[0.59385049, 0.40614951],
[0.63398437, 0.36601563],
[0.63625703, 0.36374297]


```
[0.63830829, 0.36169171],  
[0.57788703, 0.42211297],  
[0.63835894, 0.36164106],  
[0.61560095, 0.38439905],  
[0.62273148, 0.37726852],  
[0.51657653, 0.48342347],  
[0.53193679, 0.46806321],  
[0.64013966, 0.35986034],  
[0.59385049, 0.40614951],  
[0.63925137, 0.36074863],  
[0.46485056, 0.53514944],  
[0.64531408, 0.35468592],  
[0.62876821, 0.37123179],  
[0.59385049, 0.40614951],  
[0.6032701 , 0.3967299 ],  
[0.49173773, 0.50826227]])
```

In [20]:

```
y_pred = model.predict(x_test)  
from sklearn.metrics import accuracy_score
```

In [21]:

```
accuracy_score(y_test, y_pred) * 100
```

Out[21]:

65.47085201793722

In [22]:

```
x = ds['pclass'].values  
y = ds['survived']  
x = x.reshape(-1,1)
```

In [23]:

```
x_train,x_test,y_train,y_test=train_test_split(  
    x, y, random_state = 0, test_size = 0.25)  
# build train and test
```

In [24]:

```
model = GaussianNB()  
model.fit(x_train, y_train)
```

Out[24]:

GaussianNB()

In [25]:

```
model.predict_proba(x_test);
```

In [26]:

```
y_pred = model.predict(x_test)
```

In [27]:

```
accuracy_score(y_test, y_pred) * 100
```

Out[27]:

70.85201793721974

In [28]:

```
x = ds[['sex','pclass']]  
y = ds['survived']
```

In [29]:

```
from sklearn.preprocessing import LabelEncoder
enc = LabelEncoder()
x['sex'] = enc.fit_transform(x['sex'])
```

<ipython-input-29-61f95b308646>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
x['sex'] = enc.fit_transform(x['sex'])

In [30]:

```
#x = x.reshape(-1,1)
```

In [31]:

```
x_train,x_test,y_train,y_test=train_test_split(
    x, y, random_state = 0, test_size = 0.25)
# build train and test
model = GaussianNB()
model.fit(x_train, y_train)
```

Out[31]:

GaussianNB()

In [32]:

```
model.predict_proba(x_test);
```

In [33]:

```
y_pred = model.predict(x_test)
accuracy_score(y_test, y_pred) * 100
```

Out[33]:

78.02690582959642

In []:

In []:

```
# Practical No. 3
# Data Science and Visualization

'''Download Abalone dataset. (URL: http://archive.ics.uci.edu/ml/datasets/Abalone)
Data set has total 8 Number of Attributes.
Load the data from data file and split it into training and test datasets. Summarize the
properties in the training dataset.
The number of rings is the value to predict: either as a continuous value or as a classification
problem.
Predict the age of abalone from physical measurements using linear regression or predict ring
class as classification problem.'''
```

In [2]:

```
import pandas as pd
```

In [3]:

```
col = ['sex', 'length', 'diameter', 'height', 'weight', 'sweight',
       'vweight', 'shweight', 'rings']
df = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data',
names=col)
```

In [4]:

```
df.head()
```

Out[4]:

	sex	length	diameter	height	weight	sweight	vweight	shweight	rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [5]:

```
df.describe()
```

Out[5]:

	length	diameter	height	weight	sweight	vweight	shweight	rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   sex         4177 non-null   object
 1   length      4177 non-null   float64
 2   diameter    4177 non-null   float64
 3   height      4177 non-null   float64
 4   weight      4177 non-null   float64
 5   sweight     4177 non-null   float64
 6   vweight     4177 non-null   float64
 7   shweight    4177 non-null   float64
 8   rings       4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

In [7]:

```
X = df.drop('rings', axis=1) #input
y = df['rings'] #output
```

In [8]:

```
X.head()
```

Out[8]:

	sex	length	diameter	height	weight	sweight	vweight	shweight
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055

In [9]:

```
from collections import Counter
Counter(y)
```

Out[9]:

```
Counter({15: 103,
          7: 391,
          9: 689,
          10: 634,
          8: 568,
          20: 26,
          16: 67,
          19: 32,
          14: 126,
          11: 487,
          12: 267,
          18: 42,
          13: 203,
          5: 115,
          4: 57,
          6: 259,
          21: 14,
          17: 58,
          22: 6,
          1: 1,
          3: 15,
          26: 1,
          23: 9,
          29: 1,
          2: 1,
          27: 2,
          25: 1,
          28: 1,
          24: 1,
          30: 1,
          16: 1,
          17: 1,
          18: 1,
          19: 1,
          20: 1,
          21: 1,
          22: 1,
          23: 1,
          24: 1,
          25: 1,
          26: 1,
          27: 1,
          28: 1,
          29: 1,
          30: 1,
          31: 1,
          32: 1,
          33: 1,
          34: 1,
          35: 1,
          36: 1,
          37: 1,
          38: 1,
          39: 1,
          40: 1,
          41: 1,
          42: 1,
          43: 1,
          44: 1,
          45: 1,
          46: 1,
          47: 1,
          48: 1,
          49: 1,
          50: 1,
          51: 1,
          52: 1,
          53: 1,
          54: 1,
          55: 1,
          56: 1,
          57: 1,
          58: 1,
          59: 1,
          60: 1,
          61: 1,
          62: 1,
          63: 1,
          64: 1,
          65: 1,
          66: 1,
          67: 1,
          68: 1,
          69: 1,
          70: 1,
          71: 1,
          72: 1,
          73: 1,
          74: 1,
          75: 1,
          76: 1,
          77: 1,
          78: 1,
          79: 1,
          80: 1,
          81: 1,
          82: 1,
          83: 1,
          84: 1,
          85: 1,
          86: 1,
          87: 1,
          88: 1,
          89: 1,
          90: 1,
          91: 1,
          92: 1,
          93: 1,
          94: 1,
          95: 1,
          96: 1,
          97: 1,
          98: 1,
          99: 1,
          100: 1,
          101: 1,
          102: 1,
          103: 1,
          104: 1,
          105: 1,
          106: 1,
          107: 1,
          108: 1,
          109: 1,
          110: 1,
          111: 1,
          112: 1,
          113: 1,
          114: 1,
          115: 1,
          116: 1,
          117: 1,
          118: 1,
          119: 1,
          120: 1,
          121: 1,
          122: 1,
          123: 1,
          124: 1,
          125: 1,
          126: 1,
          127: 1,
          128: 1,
          129: 1,
          130: 1,
          131: 1,
          132: 1,
          133: 1,
          134: 1,
          135: 1,
          136: 1,
          137: 1,
          138: 1,
          139: 1,
          140: 1,
          141: 1,
          142: 1,
          143: 1,
          144: 1,
          145: 1,
          146: 1,
          147: 1,
          148: 1,
          149: 1,
          150: 1,
          151: 1,
          152: 1,
          153: 1,
          154: 1,
          155: 1,
          156: 1,
          157: 1,
          158: 1,
          159: 1,
          160: 1,
          161: 1,
          162: 1,
          163: 1,
          164: 1,
          165: 1,
          166: 1,
          167: 1,
          168: 1,
          169: 1,
          170: 1,
          171: 1,
          172: 1,
          173: 1,
          174: 1,
          175: 1,
          176: 1,
          177: 1,
          178: 1,
          179: 1,
          180: 1,
          181: 1,
          182: 1,
          183: 1,
          184: 1,
          185: 1,
          186: 1,
          187: 1,
          188: 1,
          189: 1,
          190: 1,
          191: 1,
          192: 1,
          193: 1,
          194: 1,
          195: 1,
          196: 1,
          197: 1,
          198: 1,
          199: 1,
          200: 1,
          201: 1,
          202: 1,
          203: 1,
          204: 1,
          205: 1,
          206: 1,
          207: 1,
          208: 1,
          209: 1,
          210: 1,
          211: 1,
          212: 1,
          213: 1,
          214: 1,
          215: 1,
          216: 1,
          217: 1,
          218: 1,
          219: 1,
          220: 1,
          221: 1,
          222: 1,
          223: 1,
          224: 1,
          225: 1,
          226: 1,
          227: 1,
          228: 1,
          229: 1,
          230: 1,
          231: 1,
          232: 1,
          233: 1,
          234: 1,
          235: 1,
          236: 1,
          237: 1,
          238: 1,
          239: 1,
          240: 1,
          241: 1,
          242: 1,
          243: 1,
          244: 1,
          245: 1,
          246: 1,
          247: 1,
          248: 1,
          249: 1,
          250: 1,
          251: 1,
          252: 1,
          253: 1,
          254: 1,
          255: 1,
          256: 1,
          257: 1,
          258: 1,
          259: 1,
          260: 1,
          261: 1,
          262: 1,
          263: 1,
          264: 1,
          265: 1,
          266: 1,
          267: 1,
          268: 1,
          269: 1,
          270: 1,
          271: 1,
          272: 1,
          273: 1,
          274: 1,
          275: 1,
          276: 1,
          277: 1,
          278: 1,
          279: 1,
          280: 1,
          281: 1,
          282: 1,
          283: 1,
          284: 1,
          285: 1,
          286: 1,
          287: 1,
          288: 1,
          289: 1,
          290: 1,
          291: 1,
          292: 1,
          293: 1,
          294: 1,
          295: 1,
          296: 1,
          297: 1,
          298: 1,
          299: 1,
          300: 1,
          301: 1,
          302: 1,
          303: 1,
          304: 1,
          305: 1,
          306: 1,
          307: 1,
          308: 1,
          309: 1,
          310: 1,
          311: 1,
          312: 1,
          313: 1,
          314: 1,
          315: 1,
          316: 1,
          317: 1,
          318: 1,
          319: 1,
          320: 1,
          321: 1,
          322: 1,
          323: 1,
          324: 1,
          325: 1,
          326: 1,
          327: 1,
          328: 1,
          329: 1,
          330: 1,
          331: 1,
          332: 1,
          333: 1,
          334: 1,
          335: 1,
          336: 1,
          337: 1,
          338: 1,
          339: 1,
          340: 1,
          341: 1,
          342: 1,
          343: 1,
          344: 1,
          345: 1,
          346: 1,
          347: 1,
          348: 1,
          349: 1,
          350: 1,
          351: 1,
          352: 1,
          353: 1,
          354: 1,
          355: 1,
          356: 1,
          357: 1,
          358: 1,
          359: 1,
          360: 1,
          361: 1,
          362: 1,
          363: 1,
          364: 1,
          365: 1,
          366: 1,
          367: 1,
          368: 1,
          369: 1,
          370: 1,
          371: 1,
          372: 1,
          373: 1,
          374: 1,
          375: 1,
          376: 1,
          377: 1,
          378: 1,
          379: 1,
          380: 1,
          381: 1,
          382: 1,
          383: 1,
          384: 1,
          385: 1,
          386: 1,
          387: 1,
          388: 1,
          389: 1,
          390: 1,
          391: 1,
          392: 1,
          393: 1,
          394: 1,
          395: 1,
          396: 1,
          397: 1,
          398: 1,
          399: 1,
          400: 1,
          401: 1,
          402: 1,
          403: 1,
          404: 1,
          405: 1,
          406: 1,
          407: 1,
          408: 1,
          409: 1,
          410: 1,
          411: 1,
          412: 1,
          413: 1,
          414: 1,
          415: 1,
          416: 1,
          417: 1,
          418: 1,
          419: 1,
          420: 1,
          421: 1,
          422: 1,
          423: 1,
          424: 1,
          425: 1,
          426: 1,
          427: 1,
          428: 1,
          429: 1,
          430: 1,
          431: 1,
          432: 1,
          433: 1,
          434: 1,
          435: 1,
          436: 1,
          437: 1,
          438: 1,
          439: 1,
          440: 1,
          441: 1,
          442: 1,
          443: 1,
          444: 1,
          445: 1,
          446: 1,
          447: 1,
          448: 1,
          449: 1,
          450: 1,
          451: 1,
          452: 1,
          453: 1,
          454: 1,
          455: 1,
          456: 1,
          457: 1,
          458: 1,
          459: 1,
          460: 1,
          461: 1,
          462: 1,
          463: 1,
          464: 1,
          465: 1,
          466: 1,
          467: 1,
          468: 1,
          469: 1,
          470: 1,
          471: 1,
          472: 1,
          473: 1,
          474: 1,
          475: 1,
          476: 1,
          477: 1,
          478: 1,
          479: 1,
          480: 1,
          481: 1,
          482: 1,
          483: 1,
          484: 1,
          485: 1,
          486: 1,
          487: 1,
          488: 1,
          489: 1,
          490: 1,
          491: 1,
          492: 1,
          493: 1,
          494: 1,
          495: 1,
          496: 1,
          497: 1,
          498: 1,
          499: 1,
          500: 1,
          501: 1,
          502: 1,
          503: 1,
          504: 1,
          505: 1,
          506: 1,
          507: 1,
          508: 1,
          509: 1,
          510: 1,
          511: 1,
          512: 1,
          513: 1,
          514: 1,
          515: 1,
          516: 1,
          517: 1,
          518: 1,
          519: 1,
          520: 1,
          521: 1,
          522: 1,
          523: 1,
          524: 1,
          525: 1,
          526: 1,
          527: 1,
          528: 1,
          529: 1,
          530: 1,
          531: 1,
          532: 1,
          533: 1,
          534: 1,
          535: 1,
          536: 1,
          537: 1,
          538: 1,
          539: 1,
          540: 1,
          541: 1,
          542: 1,
          543: 1,
          544: 1,
          545: 1,
          546: 1,
          547: 1,
          548: 1,
          549: 1,
          550: 1,
          551: 1,
          552: 1,
          553: 1,
          554: 1,
          555: 1,
          556: 1,
          557: 1,
          558: 1,
          559: 1,
          560: 1,
          561: 1,
          562: 1,
          563: 1,
          564: 1,
          565: 1,
          566: 1,
          567: 1,
          568: 1,
          569: 1,
          570: 1,
          571: 1,
          572: 1,
          573: 1,
          574: 1,
          575: 1,
          576: 1,
          577: 1,
          578: 1,
          579: 1,
          580: 1,
          581: 1,
          582: 1,
          583: 1,
          584: 1,
          585: 1,
          586: 1,
          587: 1,
          588: 1,
          589: 1,
          590: 1,
          591: 1,
          592: 1,
          593: 1,
          594: 1,
          595: 1,
          596: 1,
          597: 1,
          598: 1,
          599: 1,
          600: 1,
          601: 1,
          602: 1,
          603: 1,
          604: 1,
          605: 1,
          606: 1,
          607: 1,
          608: 1,
          609: 1,
          610: 1,
          611: 1,
          612: 1,
          613: 1,
          614: 1,
          615: 1,
          616: 1,
          617: 1,
          618: 1,
          619: 1,
          620: 1,
          621: 1,
          622: 1,
          623: 1,
          624: 1,
          625: 1,
          626: 1,
          627: 1,
          628: 1,
          629: 1,
          630: 1,
          631: 1,
          632: 1,
          633: 1,
          634: 1,
          635: 1,
          636: 1,
          637: 1,
          638: 1,
          639: 1,
          640: 1,
          641: 1,
          642: 1,
          643: 1,
          644: 1,
          645: 1,
          646: 1,
          647: 1,
          648: 1,
          649: 1,
          650: 1,
          651: 1,
          652: 1,
          653: 1,
          654: 1,
          655: 1,
          656: 1,
          657: 1,
          658: 1,
          659: 1,
          660: 1,
          661: 1,
          662: 1,
          663: 1,
          664: 1,
          665: 1,
          666: 1,
          667: 1,
          668: 1,
          669: 1,
          670: 1,
          671: 1,
          672: 1,
          673: 1,
          674: 1,
          675: 1,
          676: 1,
          677: 1,
          678: 1,
          679: 1,
          680: 1,
          681: 1,
          682: 1,
          683: 1,
          684: 1,
          685: 1,
          686: 1,
          687: 1,
          688: 1,
          689: 1,
          690: 1,
          691: 1,
          692: 1,
          693: 1,
          694: 1,
          695: 1,
          696: 1,
          697: 1,
          698: 1,
          699: 1,
          700: 1,
          701: 1,
          702: 1,
          703: 1,
          704: 1,
          705: 1,
          706: 1,
          707: 1,
          708: 1,
          709: 1,
          710: 1,
          711: 1,
          712: 1,
          713: 1,
          714: 1,
          715: 1,
          716: 1,
          717: 1,
          718: 1,
          719: 1,
          720: 1,
          721: 1,
          722: 1,
          723: 1,
          724: 1,
          725: 1,
          726: 1,
          727: 1,
          728: 1,
          729: 1,
          730: 1,
          731: 1,
          732: 1,
          733: 1,
          734: 1,
          735: 1,
          736: 1,
          737: 1,
          738: 1,
          739: 1,
          740: 1,
          741: 1,
          742: 1,
          743: 1,
          744: 1,
          745: 1,
          746: 1,
          747: 1,
          748: 1,
          749: 1,
          750: 1,
          751: 1,
          752: 1,
          753: 1,
          754: 1,
          755: 1,
          756: 1,
          757: 1,
          758: 1,
          759: 1,
          760: 1,
          761: 1,
          762: 1,
          763: 1,
          764: 1,
          765: 1,
          766: 1,
          767: 1,
          768: 1,
          769: 1,
          770: 1,
          771: 1,
          772: 1,
          773: 1,
          774: 1,
          775: 1,
          776: 1,
          777: 1,
          778: 1,
          779: 1,
          780: 1,
          781: 1,
          782: 1,
          783: 1,
          784: 1,
          785: 1,
          786: 1,
          787: 1,
          788: 1,
          789: 1,
          790: 1,
          791: 1,
          792: 1,
          793: 1,
          794: 1,
          795: 1,
          796: 1,
          797: 1,
          798: 1,
          799: 1,
          800: 1,
          801: 1,
          802: 1,
          803: 1,
          804: 1,
          805: 1,
          806: 1,
          807: 1,
          808: 1,
          809: 1,
          810: 1,
          811: 1,
          812: 1,
          813: 1,
          814: 1,
          815: 1,
          816: 1,
          817: 1,
          818: 1,
          819: 1,
          820: 1,
          821: 1,
          822: 1,
          823: 1,
          824: 1,
          825: 1,
          826: 1,
          827: 1,
          828: 1,
          829: 1,
          830: 1,
          831: 1,
          832: 1,
          833: 1,
          834: 1,
          835: 1,
          836: 1,
          837: 1,
          838: 1,
          839: 1,
          840: 1,
          841: 1,
          842: 1,
          843: 1,
          844: 1,
          845: 1,
          846: 1,
          847: 1,
          848: 1,
          849: 1,
          850: 1,
          851: 1,
          852: 1,
          853: 1,
          854: 1,
          855: 1,
          856: 1,
          857: 1,
          858: 1,
          859: 1,
          860: 1,
          861: 1,
          862: 1,
          863: 1,
          864: 1,
          865: 1,
          866: 1,
          867: 1,
          868: 1,
          869: 1,
          870: 1,
          871: 1,
          872: 1,
          873: 1,
          874: 1,
          875: 1,
          876: 1,
          877: 1,
          878: 1,
          879: 1,
          880: 1,
          881: 1,
          882: 1,
          883: 1,
          884: 1,
          885: 1,
          886: 1,
          887: 1,
          888: 1,
          889: 1,
          890: 1,
          891: 1,
          892: 1,
          893: 1,
          894: 1,
          895: 1,
          896: 1,
          897: 1,
          898: 1,
          899: 1,
          900: 1,
          901: 1,
          902: 1,
          903: 1,
          904: 1,
          905: 1,
          906: 1,
          907: 1,
          908: 1,
          909: 1,
          910: 1,
          911: 1,
          912: 1,
          913: 1,
          914: 1,
          915: 1,
          916: 1,
          917: 1,
          918: 1,
          919: 1,
          920: 1,
          921: 1,
          922: 1,
          923: 1,
          924: 1,
          925: 1,
          926: 1,
          927: 1,
          928: 1,
          929: 1,
          930: 1,
          931: 1,
          932: 1,
          933: 1,
          934: 1,
          935: 1,
          936: 1,
          937: 1,
          938: 1,
          939: 1,
          940: 1,
          941: 1,
          942: 1,
          943: 1,
          944: 1,
          945: 1,
          946: 1,
          947: 1,
          948: 1,
          949: 1,
          950: 1,
          951: 1,
          952: 1,
          953: 1,
          954: 1,
          955: 1,
          956: 1,
          957: 1,
          958: 1,
          959: 1,
          960: 1,
          961: 1,
          962: 1,
          963: 1,
          964: 1,
          965: 1,
          966: 1,
          967: 1,
          968: 1,
          969: 1,
          970: 1,
          971: 1,
          972: 1,
          973: 1,
          974: 1,
          975: 1,
          976: 1,
          977: 1,
          978: 1,
          979: 1,
          980: 1,
          981: 1,
          982: 1,
          983: 1,
          984: 1,
          985: 1,
          986: 1,
          987: 1,
          988: 1,
          989: 1,
          990: 1,
          991: 1,
          992: 1,
          993: 1,
          994: 1,
          995: 1,
          996: 1,
          997: 1,
          998: 1,
          999: 1,
          1000: 1,
          1001: 1,
          1002: 1,
          1003: 1,
          1004: 1,
          1005: 1,
          1006: 1,
          1007: 1,
          1008: 1,
          1009: 1,
          1010: 1,
          1011: 1,
          1012: 1,
          1013: 1,
          1014: 1,
          1015: 1,
          1016: 1,
          1017: 1,
          1018: 1,
          1019: 1,
          1020: 1,
          1021: 1,
          1022: 1,
          1023: 1,
          1024: 1,
          1025: 1,
          1026: 1,
          1027: 1,
          1028: 1,
          1029: 1,
          1030: 1,
          1031: 1,
          1032: 1,
          1033: 1,
          1034: 1,
          1035: 1,
          1036: 1,
          1037: 1,
          1038: 1,
          1039: 1,
          1040: 1,
          1041: 1,
          1042: 1,
          1043: 1,
          1044: 1,
          1045: 1,
          1046: 1,
          1047: 1,
          1048: 1,
          1049: 1,
          1050: 1,
          1051: 1,
          1052: 1,
          1053: 1,
          1054: 1,
          1055: 1,
          1056: 1,
          1057: 1,
          1058: 1,
          1059: 1,
          1060: 1,
          1061: 1,
          1062: 1,
          1063: 1,
          1064: 1,
          1065: 1,
          1066: 1,
          1067: 1,
          1068: 1,
          1069: 1,
          1070: 1,
          1071: 1,
          1072: 1,
          1073: 1,
          1074: 1,
          1075: 1,
          1076: 1,
          1077: 1,
          1078: 1,
          1079: 1,
          1080: 1,
          1081: 1,
          1082: 1,
          1083: 1,
          1084: 1,
          1085: 1,
          1086: 1,
          1087: 1,
          1088: 1,
          1089: 1,
          1090: 1,
          1091: 1,
          1092: 1,
          1093: 1,
          1094: 1,
          1095: 1,
          1096: 1,
          1097: 1,
          1098: 1,
          1099: 1,
          1100: 1,
          1101: 1,
          1102: 1,
          1103: 1,
          1104: 1,
          1105: 1,
          1106: 1,
          1107: 1,
          1108: 1,
         
```

```
23: 1,  
24: 2})
```

In [10]:

```
set(X['sex'])
```

Out[10]:

```
{'F', 'I', 'M'}
```

In [11]:

```
from sklearn.preprocessing import LabelEncoder  
enc = LabelEncoder()  
X['sex'] = enc.fit_transform(X['sex'])
```

In [12]:

```
set(X['sex'])
```

Out[12]:

```
{0, 1, 2}
```

In [13]:

```
df.head()
```

Out[13]:

	sex	length	diameter	height	weight	sweight	vweight	shweight	rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, random_state = 0, test_size = 0.25)
```

In [16]:

```
len(X_train)
```

Out[16]:

```
3132
```

In [17]:

```
len(X_test)
```

Out[17]:

```
1045
```

In [18]:

```
X_train.head()
```

Out[18]:

	sex	length	diameter	height	weight	sweight	vweight	shweight
940	1	0.460	0.345	0.105	0.4490	0.1960	0.0945	0.1265
2688	2	0.630	0.465	0.150	1.0270	0.5370	0.1880	0.1760
1948	2	0.635	0.515	0.165	1.2290	0.5055	0.2975	0.3535
713	2	0.355	0.265	0.085	0.2010	0.0690	0.0530	0.0695
3743	0	0.705	0.555	0.195	1.7525	0.7105	0.4215	0.5160

In [19]:

```
from sklearn.naive_bayes import GaussianNB
```

In [20]:

```
clf = GaussianNB()
```

In [21]:

```
# train
clf.fit(X_train, y_train)
```

Out[21]:

```
GaussianNB()
```

In [22]:

```
y_pred = clf.predict(X_test)
```

In [23]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [24]:

```
accuracy_score(y_test, y_pred) * 100
```

Out[24]:

```
26.02870813397129
```

In [25]:

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
3	0.50	1.00	0.67	7
4	0.30	0.62	0.40	13
5	0.27	0.42	0.33	40
6	0.32	0.43	0.36	63
7	0.26	0.36	0.30	114
8	0.27	0.29	0.28	139
9	0.25	0.30	0.27	152
10	0.21	0.24	0.23	139
11	0.26	0.42	0.32	121
12	0.50	0.01	0.02	93
13	0.00	0.00	0.00	51
14	0.00	0.00	0.00	32
15	0.00	0.00	0.00	22
16	0.00	0.00	0.00	16
17	0.00	0.00	0.00	12
18	0.00	0.00	0.00	6
19	0.00	0.00	0.00	10
20	0.00	0.00	0.00	8
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	1
23	0.00	0.00	0.00	2

24	0.00	0.00	0.00	1
27	0.00	0.00	0.00	0
29	0.00	0.00	0.00	1
accuracy			0.26	1045
macro avg	0.13	0.17	0.13	1045
weighted avg	0.24	0.26	0.22	1045

```
D:\Program Files\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Und
efinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Program Files\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Und
efinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels wi
th no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Program Files\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Und
efinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Program Files\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Und
efinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels wi
th no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Program Files\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Und
efinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Program Files\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: Und
efinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels wi
th no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [26]:

```
from sklearn.linear_model import LinearRegression
```

In [27]:

```
reg = LinearRegression()
```

In [28]:

```
reg.fit(X_train, y_train)
```

Out[28]:

```
LinearRegression()
```

In [29]:

```
y_pred = reg.predict(X_test)
```

In [30]:

```
y_pred
```

Out[30]:

```
array([13.10451425,  9.66747548, 10.35605247, ...,  9.95962005,
        12.59111443, 12.18516586])
```

In [31]:

```
from sklearn.metrics import mean_absolute_error
```

In [32]:

```
mean_absolute_error(y_test, y_pred)
```

Out[32]:

1.5955158378194023

In [33]:

```
from sklearn.metrics import r2_score
```

In [34]:

```
r2_score(y_test, y_pred)
```

Out[34]:

0.5354158501894077

In []:

In []:

```
# Practical No. 4
# Data Science and Visualization

'''Use Netflix Movies and TV Shows dataset from Kaggle and perform following operation :
1. Make a visualization showing the total number of movies watched by children
2. Make a visualization showing the total number of standup comedies
3. Make a visualization showing most watched shows.
4. Make a visualization showing highest rated show.
Make a dashboard (DASHBOARD A) containing all of these above visualizations.'''
```

In [2]:

```
import pandas as pd
```

In [3]:

```
df = pd.read_csv('C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DS\\Data Set\\netflix_titles.csv')
```

In [4]:

```
df.shape
```

Out[4]:

```
(7787, 13)
```

In [5]:

```
categories = df['listed_in']
```

In [6]:

```
total_child = sum(df['listed_in'].str.contains('Child'))
```

In [7]:

```
total_child
```

Out[7]:

```
532
```

In [8]:

```
standup_comedies = sum(df['listed_in'].str.contains('Stand'))
```

In [9]:

```
standup_comedies
```

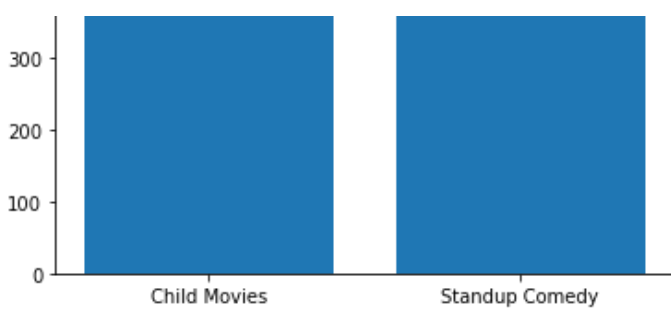
Out[9]:

```
381
```

In [10]:

```
import matplotlib.pyplot as plt
plt.bar(['Child Movies', 'Standup Comedy'],
        [total_child, standup_comedies])
plt.show()
```





In [11]:

```
set(df['type'])
```

Out[11]:

```
{'Movie', 'TV Show'}
```

In [12]:

```
tv_shows = df[df['type'] == 'TV Show'] # Boolean filtering
```

In [13]:

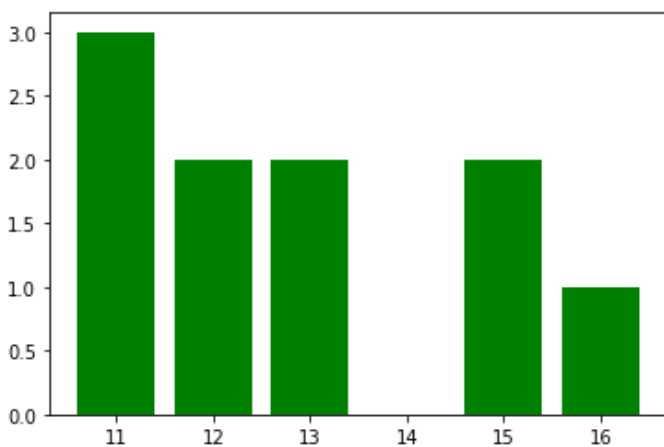
```
seasons13 = tv_shows[tv_shows['duration'] == '13 Seasons']
seasons15 = tv_shows[tv_shows['duration'] == '15 Seasons']
seasons16 = tv_shows[tv_shows['duration'] == '16 Seasons']
seasons12 = tv_shows[tv_shows['duration'] == '12 Seasons']
seasons11 = tv_shows[tv_shows['duration'] == '11 Seasons']
```

In [14]:

```
plt.bar([11,12,13,15,16],
        [len(seasons11),len(seasons12),len(seasons13),
         len(seasons15),len(seasons16)],
        color='green')
```

Out[14]:

<BarContainer object of 5 artists>



In [15]:

```
from collections import Counter
ratings = Counter(df['rating'])
```

In [16]:

```
ratings = dict(ratings)
```

In [17]:

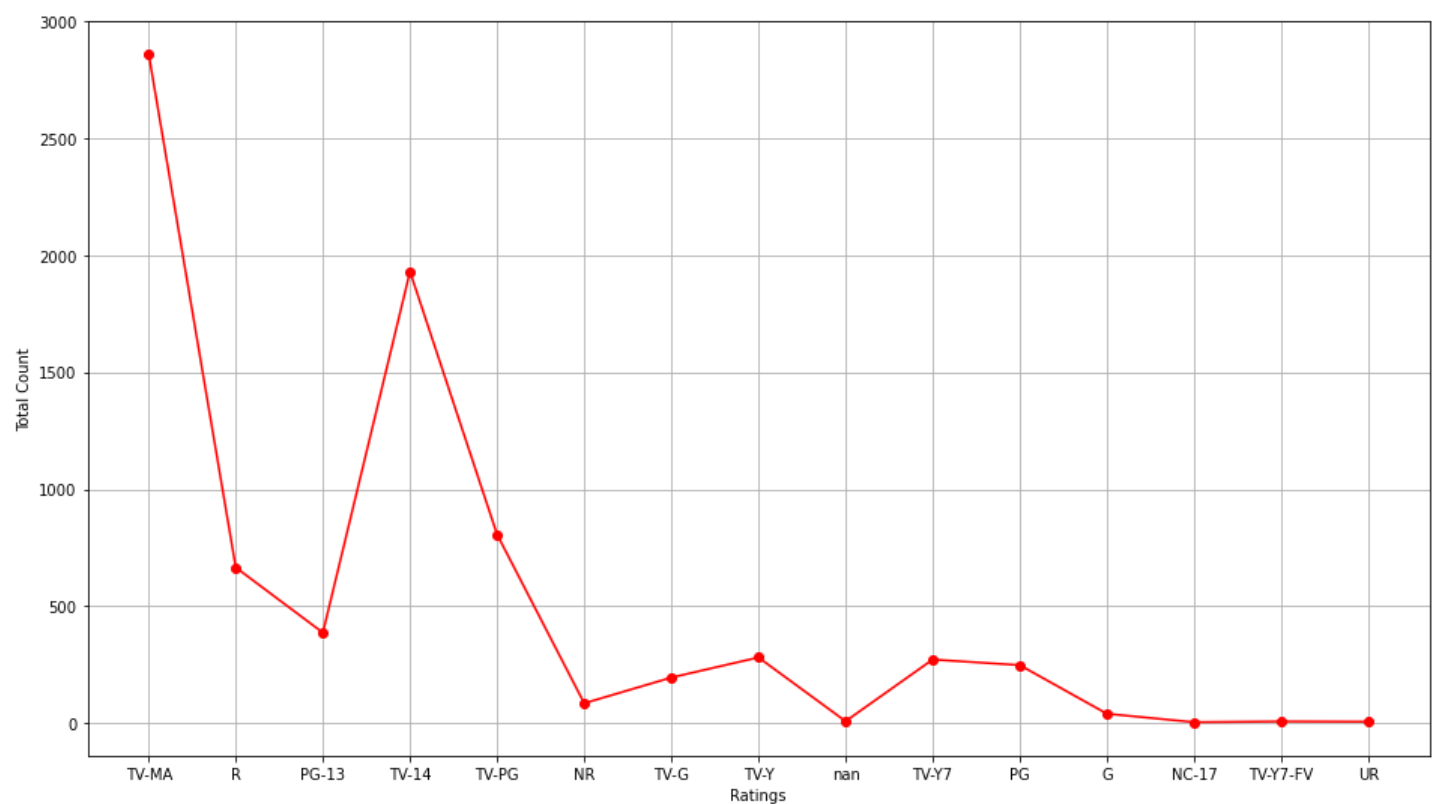
```
ratings
```

Out[17]:

```
{ 'TV-MA': 2863,
  'R': 665,
  'PG-13': 386,
  'TV-14': 1931,
  'TV-PG': 806,
  'NR': 84,
  'TV-G': 194,
  'TV-Y': 280,
  nan: 7,
  'TV-Y7': 271,
  'PG': 247,
  'G': 39,
  'NC-17': 3,
  'TV-Y7-FV': 6,
  'UR': 5}
```

In [18]:

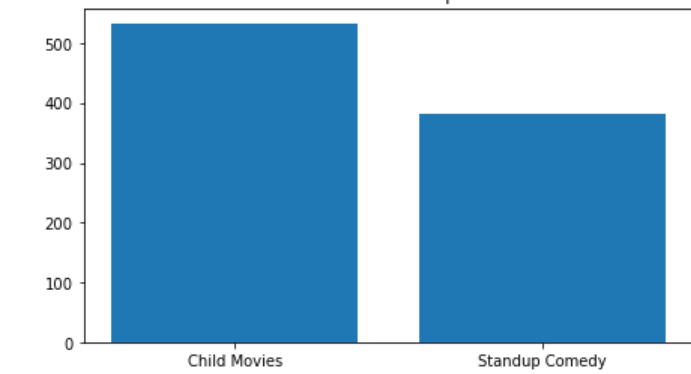
```
plt.figure(figsize=(16,9))
plt.plot(ratings.keys(), ratings.values(), color = 'red', marker='o')
plt.xlabel('Ratings'); plt.ylabel('Total Count')
plt.grid()
```



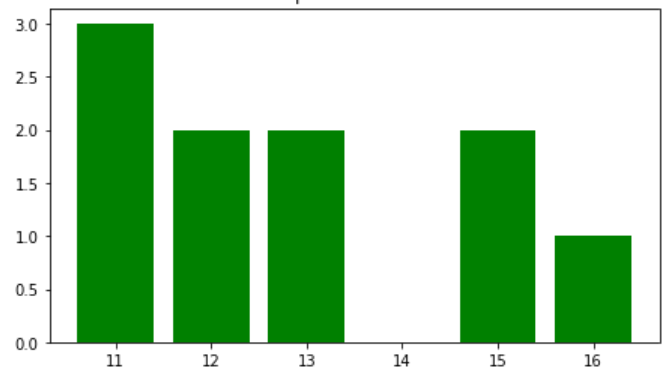
In [19]:

```
plt.figure(figsize=(16,9))
#plot1
plt.subplot(2,2,1)
plt.title('Child movies vs. Standup Comedies')
plt.bar(['Child Movies', 'Standup Comedy'],
        [total_child, standup_comedies])
#plot2
plt.subplot(2,2,2)
plt.title('Popular TV Shows')
plt.bar([11,12,13,15,16],
        [len(seasons11), len(seasons12), len(seasons13),
         len(seasons15), len(seasons16)],
        color='green')
#plot3
plt.subplot(2,1,2)
plt.title('Ratings')
plt.plot(ratings.keys(), ratings.values(), color = 'red', marker='o')
plt.xlabel('Ratings'); plt.ylabel('Total Count')
plt.grid()
```

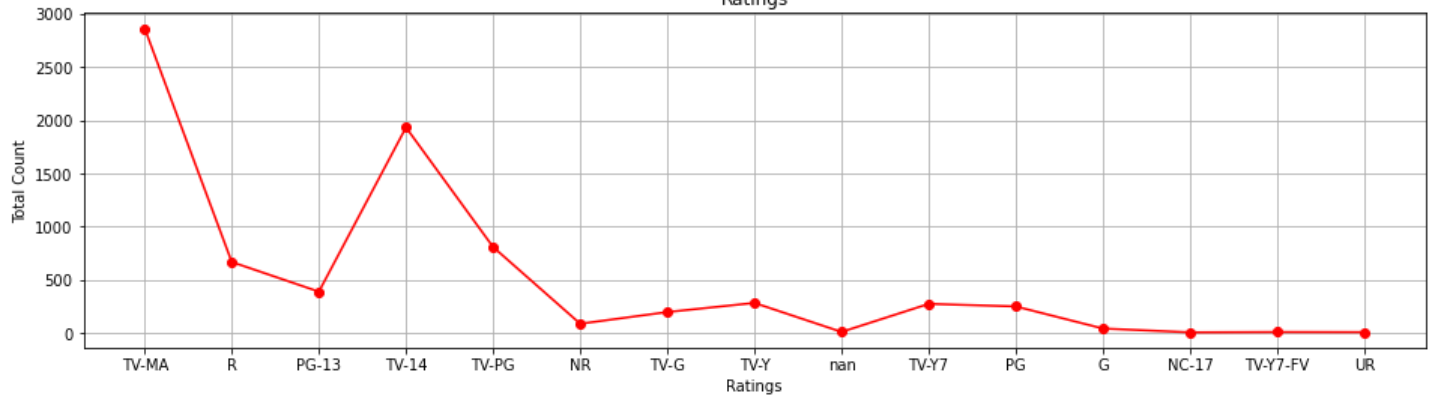
Child movies vs. Standup Comedies



Popular TV Shows



Ratings



In []: