Codsoft Internship, Task_2 :MOVIE RATING PREDICTION WITH PYTHON

Import data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import random
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
data = pd.read_csv('/content/drive/MyDrive/datasets/IMDb Movies India.csv', encoding='latin1')
```

```
data.head()
```

	Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3
0		NaN	NaN	Drama	NaN	NaN	J.S. Randhawa	Manmauji	Birbal	Rajendra Bhatia
1	#Gadhvi (He thought he was Gandhi)	(2019)	109 min	Drama	7.0	8	Gaurav Bakshi	Rasika Dugal	Vivek Ghamande	Arvind Jangid
2	#Homecoming	(2021)	90 min	Drama, Musical	NaN	NaN	Soumyajit Majumdar	Sayani Gupta	Plabita Borthakur	Roy Angana
3	#Yaaram	(2019)	110 min	Comedy, Romance	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant Kapoor
4	...And Once Again	(2010)	105 min	Drama	NaN	NaN	Amol Palekar	Rajat Kapoor	Rituparna Sengupta	Antara Mali

```
data.columns
```

```
Index(['Name', 'Year', 'Duration', 'Genre', 'Rating', 'Votes', 'Director',
       'Actor 1', 'Actor 2', 'Actor 3'],
      dtype='object')
```

Clean data

```
data.shape
```

```
(15509, 10)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      15509 non-null  object 
 1   Year      14981 non-null  object 
 2   Duration  7240 non-null   object 
 3   Genre     13632 non-null  object 
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   object 
 6   Director  14984 non-null  object 
 7   Actor 1   13892 non-null  object 
 8   Actor 2   13125 non-null  object 
 9   Actor 3   12365 non-null  object 
dtypes: float64(1), object(9)
memory usage: 1.2+ MB
```

```
data.isnull().sum()
```

```
Name           0
Year         528
Duration    8269
Genre       1877
Rating      7590
Votes       7589
Director     525
Actor 1     1617
Actor 2     2384
Actor 3     3144
dtype: int64
```

Convert to the true types columns

```
#Year and duration and votes columns has object types, should be number types
data['Year'] = pd.to_numeric(data['Year'].str.replace(r'\D', ''), errors='coerce')
data['Duration'] = pd.to_numeric(data['Duration'].str.replace(r'\D', ''), errors='coerce')
data['Votes'] = pd.to_numeric(data['Votes'].str.replace(r'\D', ''), errors='coerce')
```

```
<ipython-input-138-4836e7689585>:2: FutureWarning: The default value of regex will change from True to False in a future version.
  data['Year'] = pd.to_numeric(data['Year'].str.replace(r'\D', ''), errors='coerce')
<ipython-input-138-4836e7689585>:3: FutureWarning: The default value of regex will change from True to False in a future version.
  data['Duration'] = pd.to_numeric(data['Duration'].str.replace(r'\D', ''), errors='coerce')
<ipython-input-138-4836e7689585>:4: FutureWarning: The default value of regex will change from True to False in a future version.
  data['Votes'] = pd.to_numeric(data['Votes'].str.replace(r'\D', ''), errors='coerce')
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Name      15509 non-null  object 
 1   Year      14981 non-null  float64
 2   Duration  7240 non-null   float64
 3   Genre     13632 non-null  object 
```

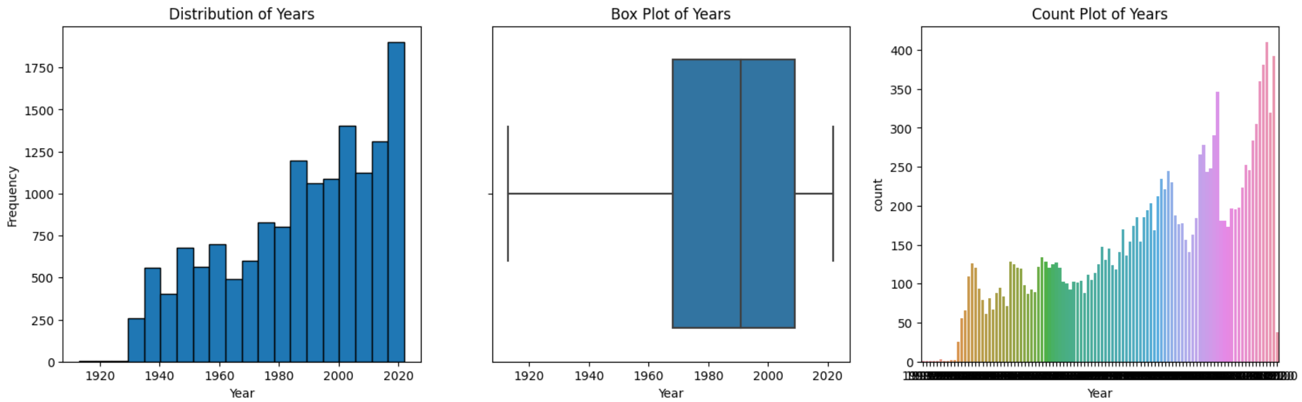
```
 4   Rating    7919 non-null   float64
 5   Votes     7920 non-null   float64
 6   Director  14984 non-null  object
 7   Actor 1   13892 non-null  object
 8   Actor 2   13125 non-null  object
 9   Actor 3   12365 non-null  object
dtypes: float64(4), object(6)
memory usage: 1.2+ MB
```

Fill the missing values for each column

```
#year
#How fill the missing values in Year column ?
data['Year'].max(),data['Year'].min()
#Fill with the mode (most year frequently) / with the mean/ with the median
```

```
(2022.0, 1913.0)
```

```
plt.figure(figsize=(18,5))
plt.subplot(1,3,1)
plt.hist(data['Year'], bins=20, edgecolor='black')
plt.title('Distribution of Years')
plt.xlabel('Year')
plt.ylabel('Frequency')
plt.subplot(1,3,2)
sns.boxplot(x=data['Year'])
plt.title('Box Plot of Years')
plt.subplot(1,3,3)
sns.countplot(x=data['Year'])
plt.title('Count Plot of Years')
plt.show()
```

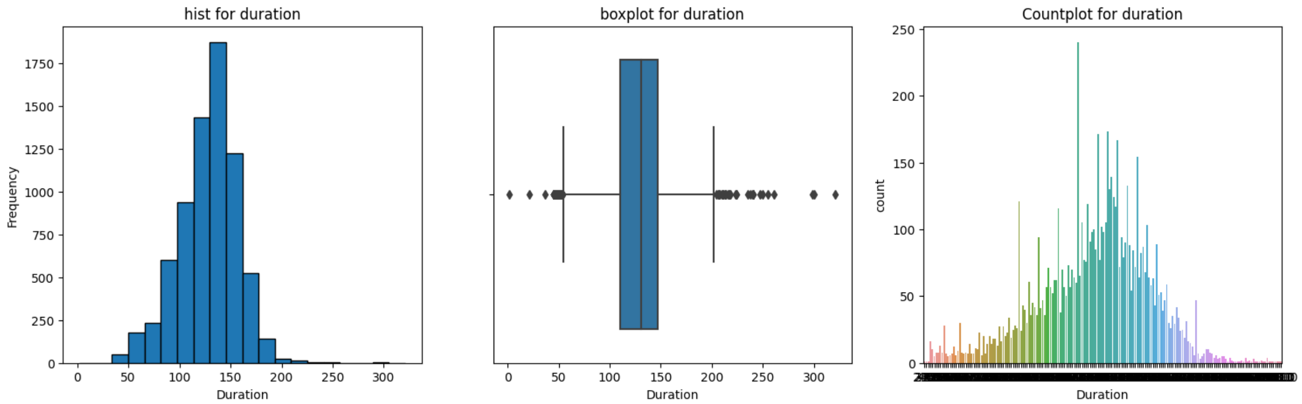



```
#based on the result we decided to fill the missing with mode
mode_year=data['Year'].mode()
data['Year']=data['Year'].fillna(2019)


data['Year'].isnull().sum()
```

    0

```
#Duration
plt.figure(figsize=(18,5))
plt.subplot(1,3,1)
plt.hist(data['Duration'].dropna(),bins=20,edgecolor='Black')
plt.title('hist for duration')
plt.xlabel('Duration')
plt.ylabel('Frequency')
plt.subplot(1,3,2)
sns.boxplot(x=data['Duration'].dropna())
plt.title('boxplot for duration')
plt.subplot(1,3,3)
sns.countplot(x=data['Duration'].dropna())
plt.title('Countplot for duration')
plt.show()
```




```
#After seeing this results we decided to fill the duration meissing with mean

data['Duration']=data['Duration'].fillna(data['Duration'].mean())
```

```
data['Genre']
#we see that in genre column that usualy it have 3 genre, soo we need to expand it into 3 Columns
```

```
0                  Drama
1                  Drama
2         Drama, Musical
3        Comedy, Romance
4                  Drama
              ...
15504             Action
15505      Action, Drama
15506             Action
15507             Action
15508      Action, Drama
Name: Genre, Length: 15509, dtype: object
```

```
data[['Genre_1','Genre_2','Genre_3']]=data['Genre'].str.split(',',expand=True)
```

```
data=data.drop('Genre',axis=1)
```

```
print('missing values in Genre_1 = ',data['Genre_1'].isnull().sum())
print('missing values in Genre_2 = ',data['Genre_2'].isnull().sum())
print('missing values in Genre_3 = ',data['Genre_3'].isnull().sum())
```

```
missing values in Genre_1 =  1877
missing values in Genre_2 =  9308
missing values in Genre_3 =  12269
```

```
#We see that the missing values in genre_1 column is the small, so we keep it and fill it, and we remove the genre 2 and genre 3
data=data.drop('Genre_2',axis=1)
data=data.drop('Genre_3',axis=1)
```

```
data.info()
```

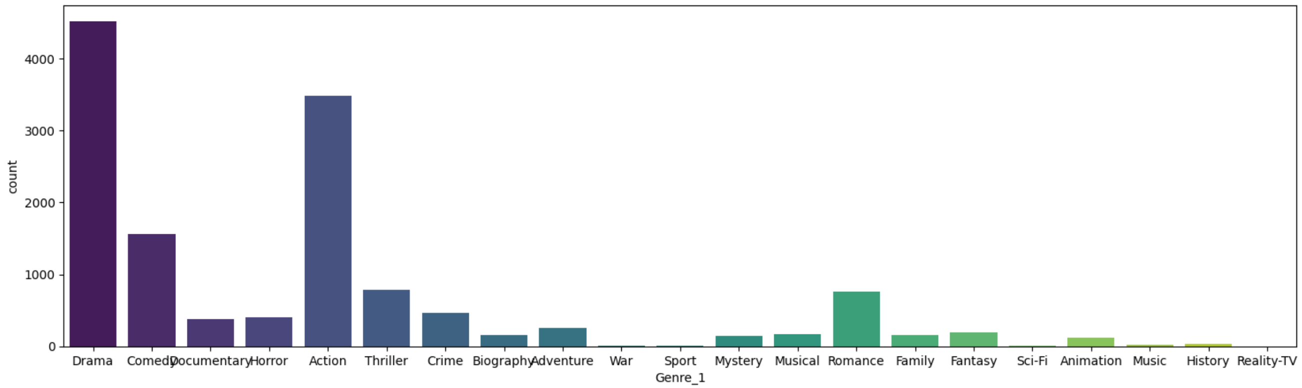
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15509 entries, 0 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      15509 non-null  object
 1   Year      15509 non-null  float64
 2   Duration  15509 non-null  float64
 3   Rating    7919 non-null   float64
 4   Votes     7920 non-null   float64
 5   Director  14984 non-null  object
 6   Actor 1   13892 non-null  object
 7   Actor 2   13125 non-null  object
 8   Actor 3   12365 non-null  object
 9   Genre_1   13632 non-null  object
dtypes: float64(4), object(6)
memory usage: 1.2+ MB
```

```
data=pd.read_csv('/content/drive/MyDrive/datasets/IMDb Movies India_V2.csv')
```

```
data['Genre_1'].isnull().sum()
```

```
1877
```

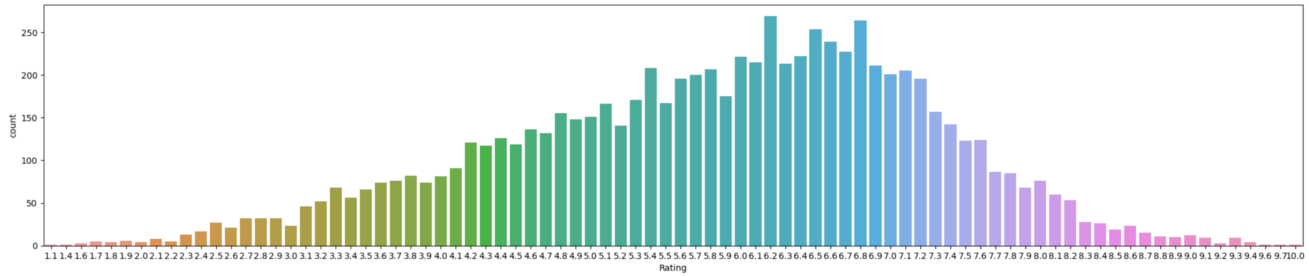
```
plt.figure(figsize=(18,5))
sns.countplot(x=data['Genre_1'], palette='viridis')
plt.show()
```




```
#we see that the Drama genre is most frequently
data['Genre_1']=data['Genre_1'].fillna('Drama')
```

```
plt.figure(figsize=(26,5))
sns.countplot(x=data['Rating'])
```

```
<Axes: xlabel='Rating', ylabel='count'>
```



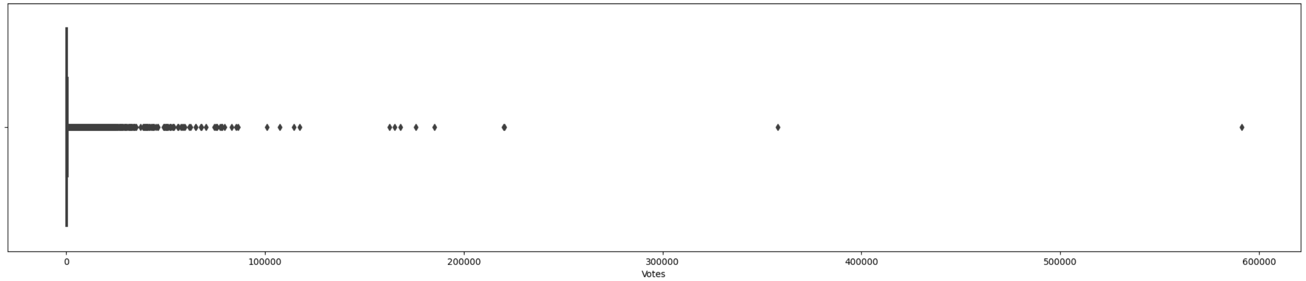

```
data['Rating']=data['Rating'].fillna(round(random.uniform(5.0,7.0)/0.1)*0.1)
```

```
data['Rating'].isnull().sum()
```

```
0
```

```
plt.figure(figsize=(26,5))
sns.boxplot(x=data['Votes'])
```

```
<Axes: xlabel='Votes'>
```




```
#based on the boxplot results we will fill the missing values
data['Votes']=data['Votes'].fillna(data['Votes'].mean())
```

```
#Director
data.dropna(subset=['Director'], inplace=True)
```

```
# for actor columns we will keep the column which has less missing values, so it is the column 'Actor_1'
data=data.drop('Actor 2',axis=1)
data=data.drop('Actor 3',axis=1)
```

```
#Actor_1
data.dropna(subset=['Actor 1'], inplace=True)
```

```
data.to_csv('/content/drive/MyDrive/datasets/IMDb Movies India_VF.csv',index=False)
```

```
data=pd.read_csv('/content/drive/MyDrive/datasets/IMDb Movies India_VF.csv')
```

```
data.head()
```

```
#Convert to categorical variables
#Identify the categorical columns
categ_col=['Name','Director','Actor 1','Genre_1']
#use label encoding for categorical columns
for column in categ_col:
  data[column]=data[column].astype('category').cat.codes
```

```
#remove duplicated values
d=data.duplicated().sum()
data=data.drop_duplicates()
d=data.duplicated().sum()
d
```

```
     0
```

```
data.shape
```

```
     (13887, 8)
```

```
#split data into train and test
train,test=[],[]
train=data[0:int(len(data)*0.8)]
test=data[int(len(data)*0.8):len(data)-1]
```

```
train['Actor 1']
```

```
#split data into features and target
X_train,y_train,X_test,y_test=[],[],[],[]
y_train=train['Rating']
X_train=train.drop('Rating',axis=1)
y_test=test['Rating']
X_test=test.drop('Rating',axis=1)
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
     (11109, 7) (11109,)
     (2777, 7) (2777,)
```

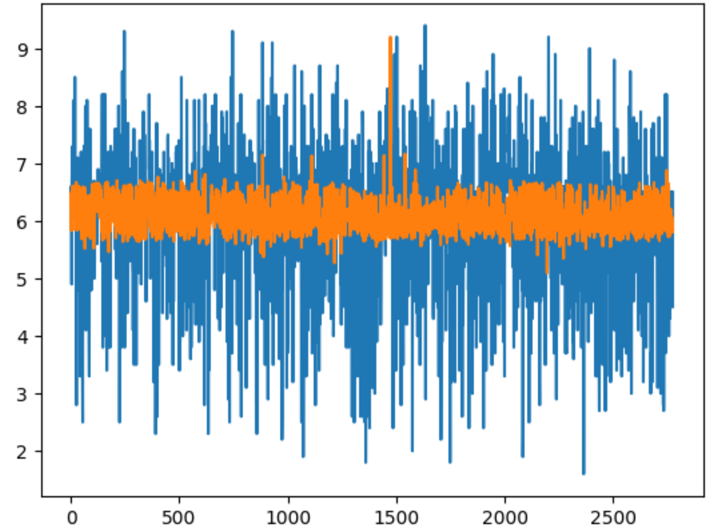
Linear regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error

#Build the model
model_LR=LinearRegression()
#fit the model
model_LR.fit(X_train,y_train)
#prediction
y_pred=model_LR.predict(X_test)
#metrics
```

```
#metrics
mse=mean_squared_error(y_test,y_pred)
mae=mean_absolute_error(y_test,y_pred)
print('MSE=',mse,'\nMAE=',mae)
#plotting result
plt.plot(y_test.values,label='real values')
plt.plot(y_pred,label='predicted values')
```

```
MSE= 1.120708812087992
MAE= 0.7581457498583789
[<matplotlib.lines.Line2D at 0x7d225de2f310>]
```

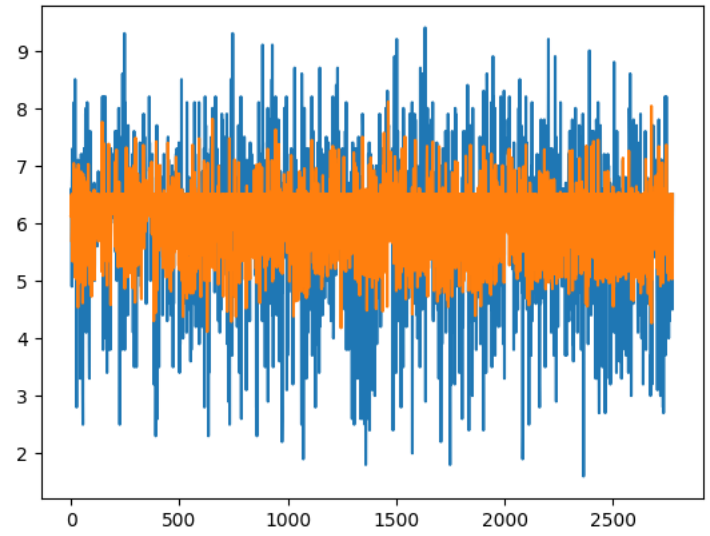


Random forest regressor

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score


#Build the model
model_RF=RandomForestRegressor()
#fit the model
model_RF.fit(X_train,y_train)
#prediction
y_pred_RF=model_RF.predict(X_test)
#metrics
mse_RF=mean_squared_error(y_test,y_pred_RF)
mae_RF=mean_absolute_error(y_test,y_pred_RF)
print('MSE=',mse_RF,'\nMAE=',mae_RF)
#plotting result
plt.plot(y_test.values,label='real values')
plt.plot(y_pred_RF,label='predicted values')
```

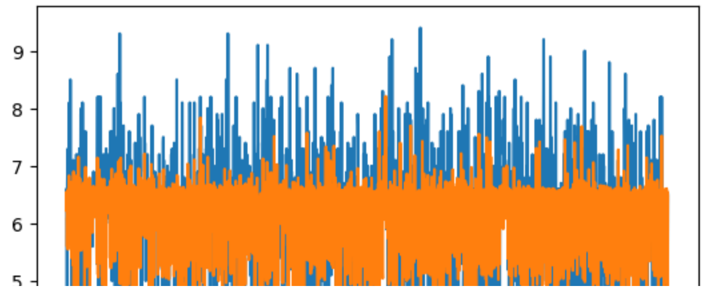
```
MSE= 0.8436727310046812
MAE= 0.5300864241987756
[<matplotlib.lines.Line2D at 0x7d225ddf6110>]
```



```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV

#Build the model
model_GB=GradientBoostingRegressor()
#fit the model
model_GB.fit(X_train,y_train)
#prediction
y_pred_GB=model_GB.predict(X_test)
#metrics
mse_GB=mean_squared_error(y_test,y_pred_RF)
mae_GB=mean_absolute_error(y_test,y_pred_RF)
print('MSE=',mse_RF,'\nMAE=',mae_RF)
#plotting result
plt.plot(y_test.values,label='real values')
plt.plot(y_pred_GB,label='predicted values')
```

```
MSE= 0.8436727310046812
MAE= 0.5300864241987756
[<matplotlib.lines.Line2D at 0x7d225d5bead0>]
```



Comparaison between the 3 algorithms useds

```
plt.figure(figsize=(16,6))
#LR
plt.plot(y_test.values,label='real values')
plt.plot(y_pred,label='predicted values',color='red')
#RF
plt.plot(y_pred,label='predicted values',color='black')
#GB
plt.plot(y_pred_GB,label='predicted values',color='green')
plt.legend()
```