IMDB TOP 1000 DATASET

PRESENTED BY:

JYOTI RANJAN SAHOO

GUIDED BY:

MRS.DR RITU MAITY

IMDb

CONTENTS -:

IMDb

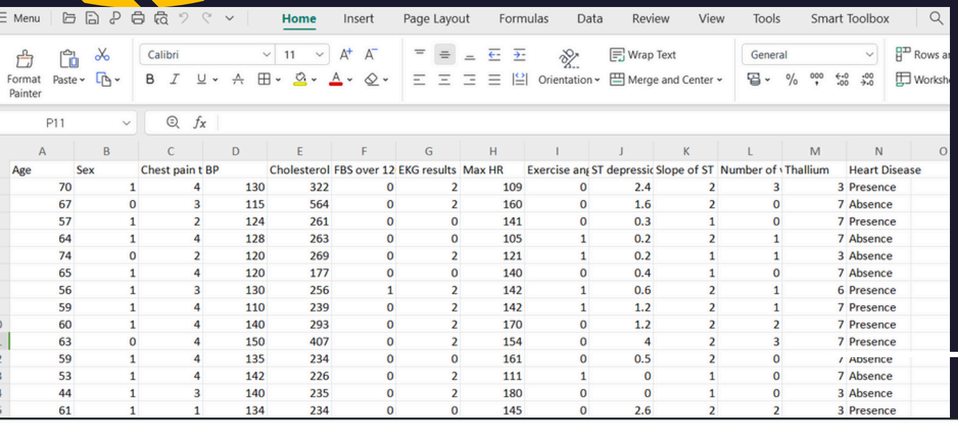
ABOUT THE DATASET-:

This dataset contains information about the top 1000 movies and TV shows as listed on IMDb. The key attributes include:

- ***Poster_Link**: URL to the movie or TV show's poster.*
- ***Series_Title**: Name of the movie or TV show.*
- ***Released_Year**: The year the movie or TV show was released.*
- ***Certificate:** Age certification for the movie or TV show.*
- ***Runtime**: Total runtime in minutes.*
- ***Genre:** Genre classification of the movie or TV show.*
- ***IMDB_Rating:** IMDb rating.*
- ***Overview:** A brief summary of the movie or TV show.*
- ***Meta_score**: Score given by Metacritic.*
- ***Director:** Name of the director.*
- ***Star1, Star2, Star3, Star4:** Names of the main stars.*
- ***No_of_votes:** Total number of votes received on IMDb.*
- ***Gross:** Box office earnings.*

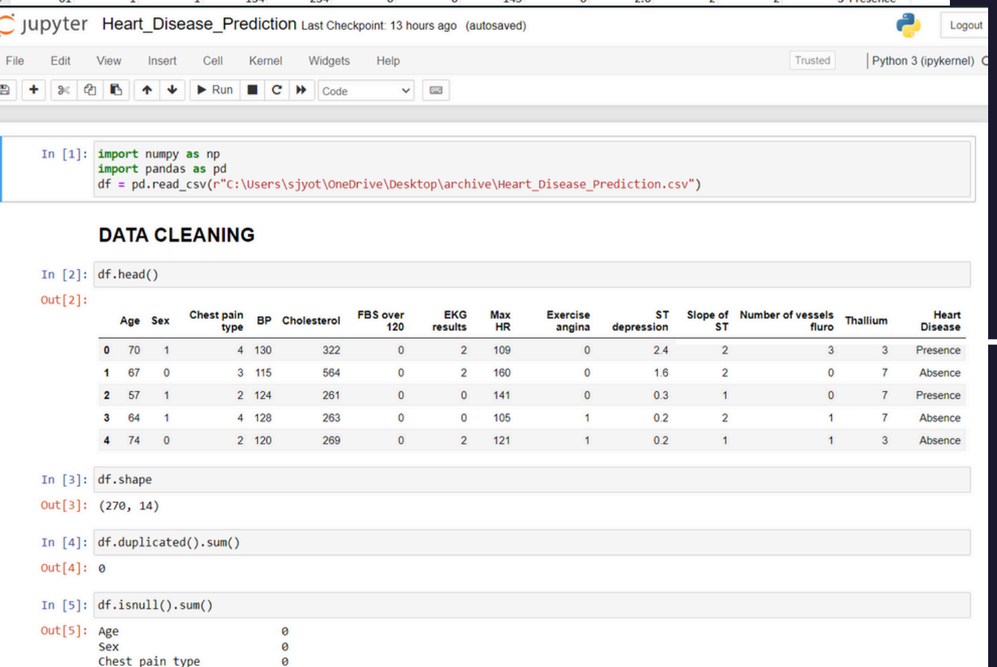
IMDb

IMPORTING THE DATASET-:



Age	Sex	Chest pain t	BP	Cholesterol	FBS over 12	EKG results	Max HR	Exercise an	ST depressic	Slope of ST	Number of \	Thallium	Heart Disease
70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence
65	1	4	120	177	0	0	140	0	0.4	1	0	7	Absence
56	1	3	130	256	1	2	142	1	0.6	2	1	6	Presence
59	1	4	110	239	0	2	142	1	1.2	2	1	7	Presence
60	1	4	140	293	0	2	170	0	1.2	2	2	7	Presence
63	0	4	150	407	0	2	154	0	4	2	3	7	Presence
59	1	4	135	234	0	0	161	0	0.5	2	0	7	Absence
53	1	4	142	226	0	2	111	1	0	1	0	7	Absence
44	1	3	140	235	0	2	180	0	0	1	0	3	Absence
61	1	1	134	234	0	0	145	0	2.6	2	2	3	Presence

Heart Disease Dataset view in excel format



jupyter Heart_Disease_Prediction Last Checkpoint: 13 hours ago (autosaved)

```
In [1]: import numpy as np
        import pandas as pd
        df = pd.read_csv(r"C:\Users\sjyot\OneDrive\Desktop\archive\Heart_Disease_Prediction.csv")
```

DATA CLEANING

```
In [2]: df.head()
```

Out[2]:

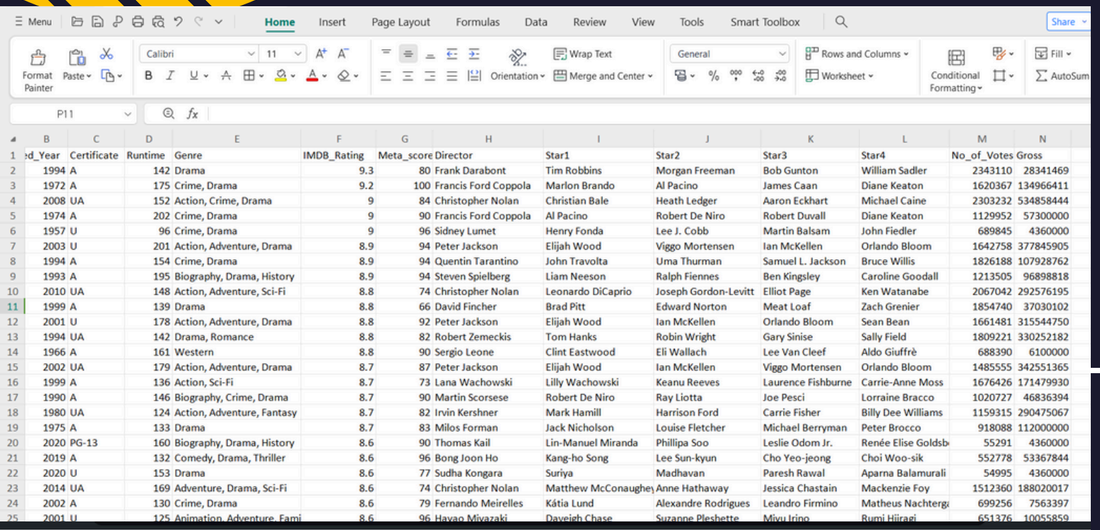
	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium	Heart Disease
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence

```
In [3]: df.shape
Out[3]: (270, 14)

In [4]: df.duplicated().sum()
Out[4]: 0

In [5]: df.isnull().sum()
Out[5]: Age                  0
        Sex                  0
        Chest pain type      0
```

Importing the Heart Disease Dataset in Jupyter Notebook

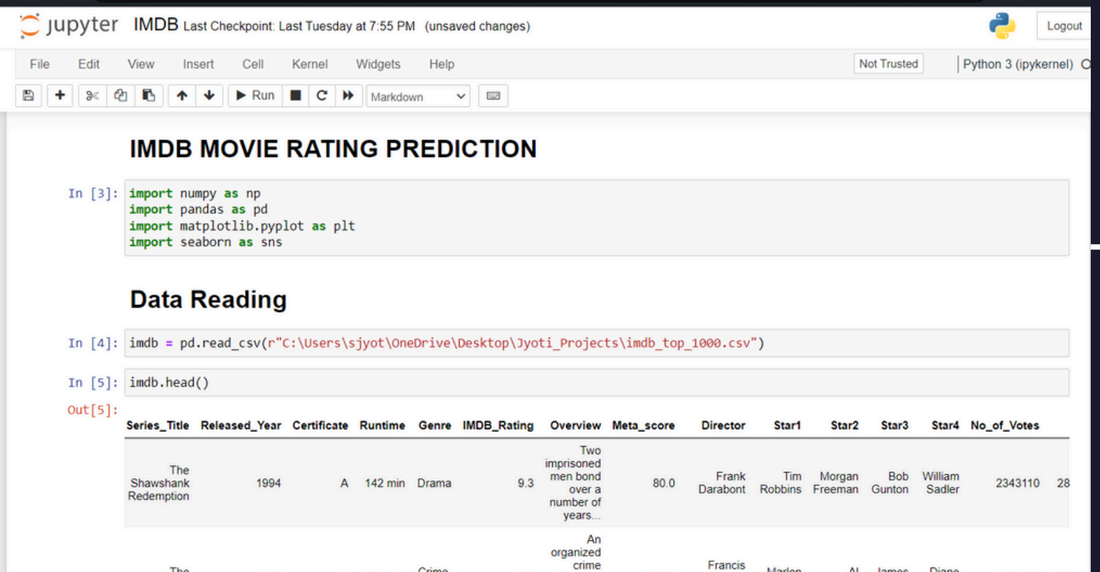
IMDb

IMPORTING THE DATASET-:



	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ed_Year	Certificate	Runtime	Genre	IMDB_Rating	Meta_score	Director	Star1	Star2	Star3	Star4	No_of_Votes	Gross
2	1994	A	142	Drama	9.3	80	Frank Darabont	Tim Robbins	Morgan Freeman	Bob Gunton	William Sadler	2343110	28341469
3	1972	A	175	Crime, Drama	9.2	100	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Diane Keaton	1620367	134966411
4	2008	UA	152	Action, Crime, Drama	9	84	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Michael Caine	2303232	534858444
5	1974	A	202	Crime, Drama	9	90	Francis Ford Coppola	Al Pacino	Robert De Niro	Robert Duvall	Diane Keaton	1129952	57300000
6	1957	U	96	Crime, Drama	9	96	Sidney Lumet	Henry Fonda	Lee J. Cobb	Martin Balsam	John Fiedler	689845	4360000
7	2003	U	201	Action, Adventure, Drama	8.9	94	Peter Jackson	Elijah Wood	Viggo Mortensen	Ian McKellen	Orlando Bloom	1642758	377845905
8	1994	A	154	Crime, Drama	8.9	94	Quentin Tarantino	John Travolta	Uma Thurman	Samuel L. Jackson	Bruce Willis	1826188	107928762
9	1993	A	195	Biography, Drama, History	8.9	94	Steven Spielberg	Liam Neeson	Ralph Fiennes	Ben Kingsley	Caroline Goodall	1213505	96898818
10	2010	UA	148	Action, Adventure, Sci-Fi	8.8	74	Christopher Nolan	Leonardo DiCaprio	Joseph Gordon-Levitt	Elliot Page	Ken Watanabe	2067042	292576195
11	1999	A	139	Drama	8.8	66	David Fincher	Brad Pitt	Edward Norton	Meat Loaf	Zach Grenier	1854740	37030102
12	2001	U	178	Action, Adventure, Drama	8.8	92	Peter Jackson	Elijah Wood	Ian McKellen	Orlando Bloom	Sean Bean	1661481	315544750
13	1994	UA	142	Drama, Romance	8.8	82	Robert Zemeckis	Tom Hanks	Robin Wright	Gary Sinise	Sally Field	1809221	330252182
14	1966	A	161	Western	8.8	90	Sergio Leone	Clint Eastwood	Eli Wallach	Lee Van Cleef	Aldo Giuffrè	688390	6100000
15	2002	UA	179	Action, Adventure, Drama	8.7	87	Peter Jackson	Elijah Wood	Ian McKellen	Viggo Mortensen	Orlando Bloom	1485555	342551365
16	1999	A	136	Action, Sci-Fi	8.7	73	Lana Wachowski	Lilly Wachowski	Keanu Reeves	Laurence Fishburne	Carrie-Anne Moss	1676426	171479930
17	1990	A	146	Biography, Crime, Drama	8.7	90	Martin Scorsese	Robert De Niro	Ray Liotta	Joe Pesci	Lorraine Bracco	1020727	46836394
18	1980	UA	124	Action, Adventure, Fantasy	8.7	82	Irvin Kershner	Mark Hamill	Harrison Ford	Carrie Fisher	Billy Dee Williams	1159315	290475067
19	1975	A	133	Drama	8.7	83	Milos Forman	Jack Nicholson	Louise Fletcher	Michael Berryman	Peter Brocco	918088	112000000
20	2020	PG-13	160	Biography, Drama, History	8.6	90	Thomas Kail	Lin-Manuel Miranda	Phillipa Soo	Leslie Odom Jr.	Renée Elise Goldsb	55291	4360000
21	2019	A	132	Comedy, Drama, Thriller	8.6	96	Bong Joon Ho	Kang-ho Song	Lee Sun-kyun	Cho Yeo-jeong	Choi Woo-sik	552778	53367844
22	2020	U	153	Drama	8.6	77	Sudha Kongara	Suriya	Madhavan	Paresh Rawal	Aparna Balamurali	54995	4360000
23	2014	UA	169	Adventure, Drama, Sci-Fi	8.6	74	Christopher Nolan	Matthew McConaughey	Anne Hathaway	Jessica Chastain	Mackenzie Foy	1512360	188020017
24	2002	A	130	Crime, Drama	8.6	79	Fernando Meirelles	Kátia Lund	Alexandre Rodrigues	Leandro Firmino	Matheus Nachterg	699256	7563397
25	2001	U	125	Animation, Adventure, Fami	8.6	96	Hayao Miyazaki	Daveigh Chase	Suzanne Pleshette	Miyu Irino	Rumi Hiiragi	651376	10055859

IMDB Dataset view in excel format



jupyter IMDB Last Checkpoint: Last Tuesday at 7:55 PM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run | Markdown

IMDB MOVIE RATING PREDICTION

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Reading

In [4]:
```
imdb = pd.read_csv(r"C:\Users\sjyot\OneDrive\Desktop\Jyoti_Projects\imdb_top_1000.csv")
```

In [5]:
```
imdb.head()
```

Out[5]:

	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Star4	No_of_Votes	
0	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins	Morgan Freeman	Bob Gunton	William Sadler	2343110	28
1	The				Crime		An organized crime		Francis	Marlon	Al	James	Diane		

Importing the IMDB Dataset in Jupyter Notebook

IMDb

DATA CLEANING-:

Data Cleaning Steps:

- *data reading(for ".data" and ".csv" extension write "pd.read_csv" and for ".xlsx" extensionwrite "pd.read_excel")*
- *renaming columns(if required)*
- *check null values or missing values (var.isnull().sum())*
- *check data types (var.dtypes)*
- *check unique values of each column*
- *print describe and use the values from it to replace whereever you find problems in anyattribute or column of dataset*
- *change datatype after replacing if required*
- *Matplotlib and Seaborn*

IMDb

DATA CLEANING IN JUPYTER NOTEBOOK

```
In [4]: imdb.isnull().sum()
```

```
Out[4]: Poster_Link        0
        Series_Title       0
        Released_Year      0
        Certificate      101
        Runtime            0
        Genre              0
        IMDB_Rating        0
        Overview           0
        Meta_score       157
        Director           0
        Star1              0
        Star2              0
        Star3              0
        Star4              0
        No_of_Votes        0
        Gross            169
        dtype: int64
```

```
In [5]: imdb.dtypes
```

```
Out[5]: Poster_Link       object
        Series_Title      object
        Released_Year     object
        Certificate       object
        Runtime           object
        Genre             object
```

```
In [6]: for i in imdb.columns:
            print(i,':','\n',imdb[i].unique())
```

```
'https://m.media-amazon.com/images/M/MV5BMTMxNTMwODM0NF5BMl5BanBnXkFtZTcwODAyMTk2Mw@@._V1_UX67_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BMWMwMGQzZTItY2JlNC00OWZiLWIyMDctNDk2ZDQ2YjRjMWQ0XkEyXkFqcGdeQXVyNzkwMjQ5Nz
98_CR1,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BMWU4N2FjNzYtNTVkNC00NzQ0LTg0MjAtYTJlMjFhNGUxZDFmXkEyXkFqcGdeQXVyNjc1NTYyMj
67_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BNzA5ZDNlZWMtM2NhNS00NDJjLTk4NDItYTRmY2EwMWZlMTY3XkEyXkFqcGdeQXVyNzkwMjQ5Nz
67_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBlZi00MTRlLWFjM2ItYzViMjE3YzI5MjljXkEyXkFqcGdeQXVyNzkwMjQ5Nz
98_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BNDE4OTMxMTctNmRhYy00NWE2LTg3YzItYTk3M2UwOTU5Njg4XkEyXkFqcGdeQXVyNjU0OTQ0OT
67_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@._V1_UX67_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BMmEzNTkxYjQtZTc0MC00YTVjLTg5ZTEtZWMwOWVlYzY0NWIwXkEyXkFqcGdeQXVyNzkwMjQ5Nz
67_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BN2EyZjM3NzUtNWUzMi00MTgxLWI0NTctMzY4M2VlOTdjZWRiXkEyXkFqcGdeQXVyNDUzOTQ5Mj
67_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BNWIwODRlZTUtY2U3ZS00Yzg1LWJhNzYtMmZiYmEyNmU1NjMzXkEyXkFqcGdeQXVyMTQxNzMzND
98_CR0,0,67,98_AL_.jpg'
'https://m.media-amazon.com/images/M/MV5BOTQ5NDI3MTI4MF5BMl5BanBnXkFtZTgwNDQ4ODE5MDE@._V1_UX67_CR0,0,67,98_AL_.jpg'
```

```
In [7]: data = imdb.describe(include = 'all')
```

```
In [8]: data
```

Out[8]:

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Di
count	1000	1000	1000	899	1000	1000	1000.000000	1000	843.000000	
unique	1000	999	100	16	140	202	NaN	1000	NaN	
top	https://m.media-amazon.com/images/M/MV5BMDFkYT...	Drishyam	2014	U	100 min	Drama	NaN	Two imprisoned men bond over a number of years...	NaN	Hitc

```
In [9]: imdb.Certificate.replace(np.NaN,data['Certificate'][2],inplace = True)
        imdb.Gross.replace(np.NaN,data['Gross'][2],inplace = True)
        imdb.Meta_score.replace(np.NaN,data['Meta_score'][4],inplace = True)
        imdb = imdb.drop(imdb[imdb['Released_Year'] == 'PG'].index)
        imdb["Runtime"] = imdb["Runtime"].str.replace(' min', '')
        imdb['Gross'] = imdb['Gross'].str.replace(',', '')

        # Convert the column to integer
        imdb["Runtime"] = imdb["Runtime"].astype(int)
        imdb['Gross'] = imdb['Gross'].astype(int)
```

```
10]: imdb['Released_Year'] = imdb['Released_Year'].astype(int)
     imdb['Meta_score'] = imdb['Meta_score'].astype(int)
```

```
11]: imdb=imdb.drop(['Overview','Poster_Link'], axis=1)
     imdb
```

11]:

	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Meta_score	Director	Star1	Star2	Star3	Star4
0	The Shawshank Redemption	1994	A	142	Drama	9.3	80	Frank Darabont	Tim Robbins	Morgan Freeman	Bob Gunton	William Sadler
1	The Godfather	1972	A	175	Crime, Drama	9.2	100	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Diane Keaton
2	The Dark Knight	2008	UA	152	Action, Crime, Drama	9.0	84	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Michael Caine
3	The Godfather: Part II	1974	A	202	Crime, Drama	9.0	90	Francis Ford Coppola	Al Pacino	Robert De Niro	Robert Duvall	Diane Keaton
4	12 Angry Men	1957	U	96	Crime, Drama	9.0	96	Sidney Lumet	Henry Fonda	Lee J. Cobb	Martin Balsam	John Fiedler
...	...	...	...	...	...	...	...	...	...	...	...	...

```
In [12]: imdb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 999 entries, 0 to 999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Series_Title   999 non-null    object
 1   Released_Year  999 non-null    int32
 2   Certificate    999 non-null    object
 3   Runtime        999 non-null    int32
 4   Genre          999 non-null    object
 5   IMDB_Rating    999 non-null    float64
 6   Meta_score     999 non-null    int32
 7   Director       999 non-null    object
 8   Star1          999 non-null    object
 9   Star2          999 non-null    object
 10  Star3          999 non-null    object
 11  Star4          999 non-null    object
 12  No_of_Votes    999 non-null    int64
 13  Gross          999 non-null    int32
dtypes: float64(1), int32(4), int64(1), object(8)
memory usage: 101.5+ KB
```

```
In [13]: imdb.isnull().sum()
```

```
Out[13]: Series_Title     0
         Released_Year    0
```

IMDb

DATA VISUALIZATION-:

*Libraries like **Matplotlib** and **Seaborn** are used in this process.*

***Matplotlib steps**-:*

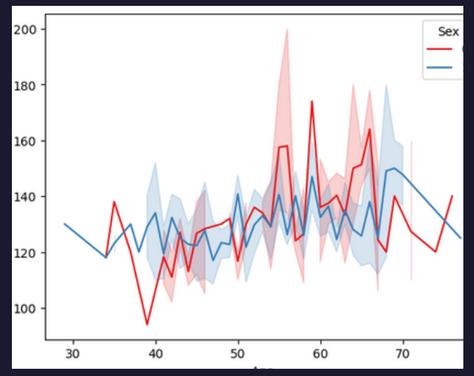
- *Import the library: import matplotlib.pyplot as plt*
- *Create data to be visualized.*
- *Use plotting functions like plt.plot(), plt.scatter(), plt.bar(), etc.*
- *Customize the plot with titles, labels, and legends.*
- *Display the plot using plt.show().*

Seaborn steps-:

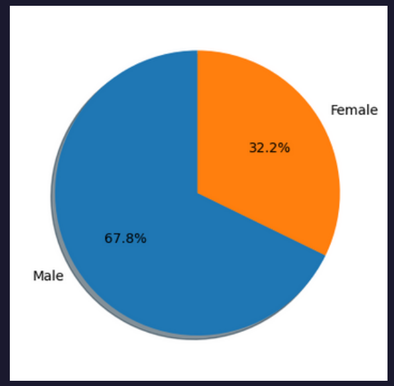
- *Import the library: import seaborn as sns*
- *Create data to be visualized.*
- *Use Seaborn functions like sns.lineplot(), sns.barplot(), sns.heatmap(), etc.*
- *Customize the plot as needed.*
- *Display the plot using Matplotlib's plt.show()*

DATA VISUALISATION IN JUPYTER NOTEBOOK

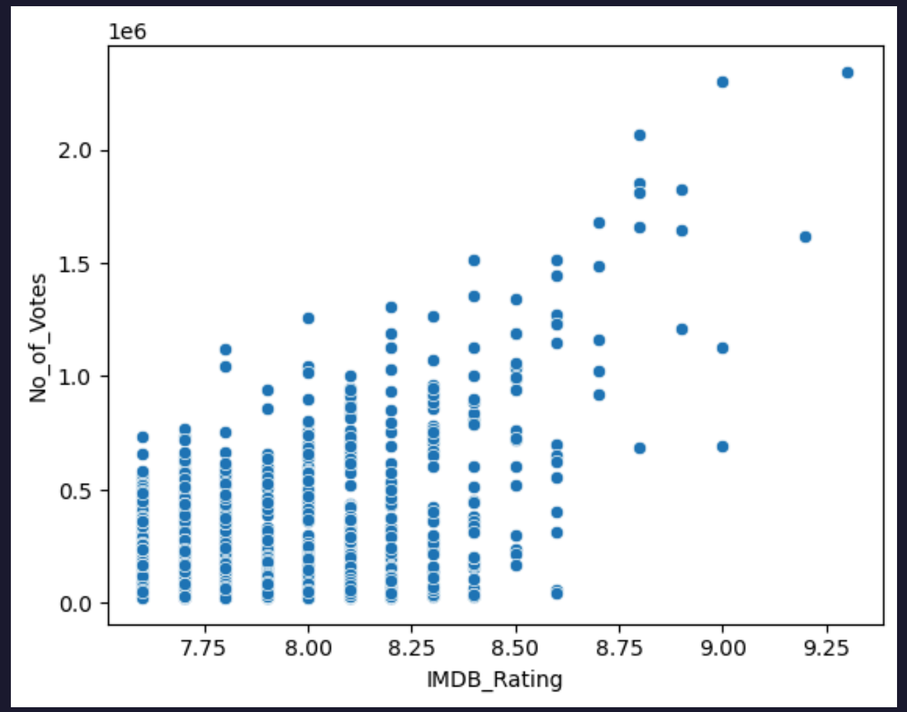
IMDb



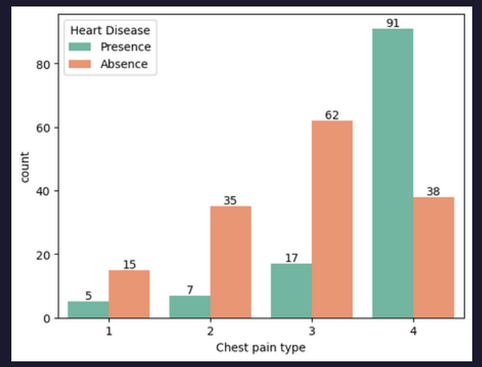
Line plot



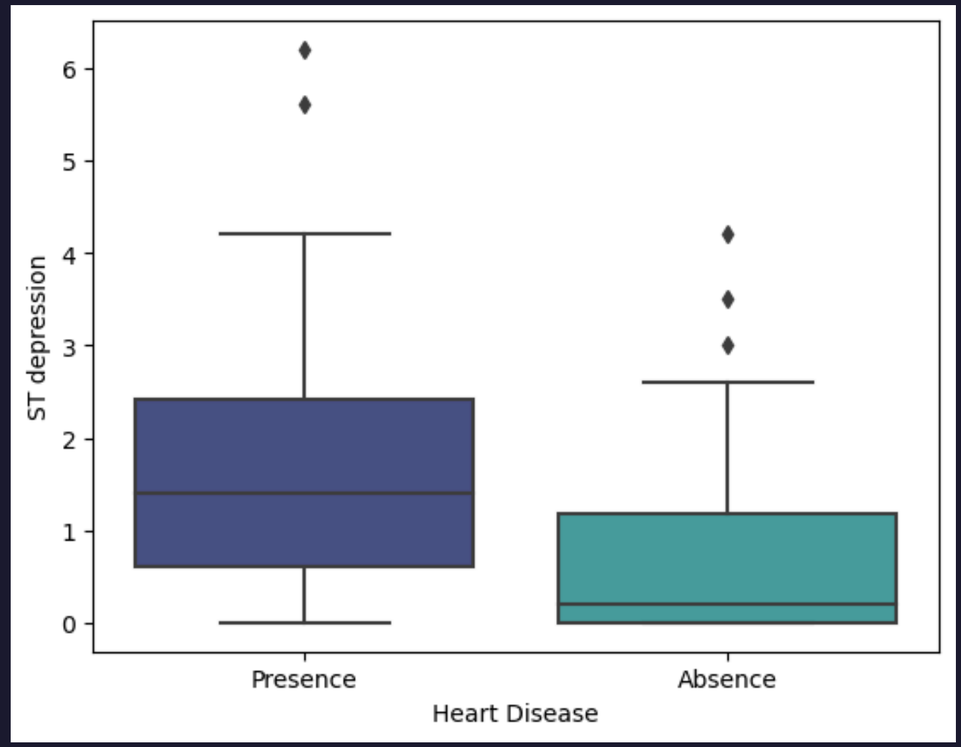
Pie Chart



Scatter plot



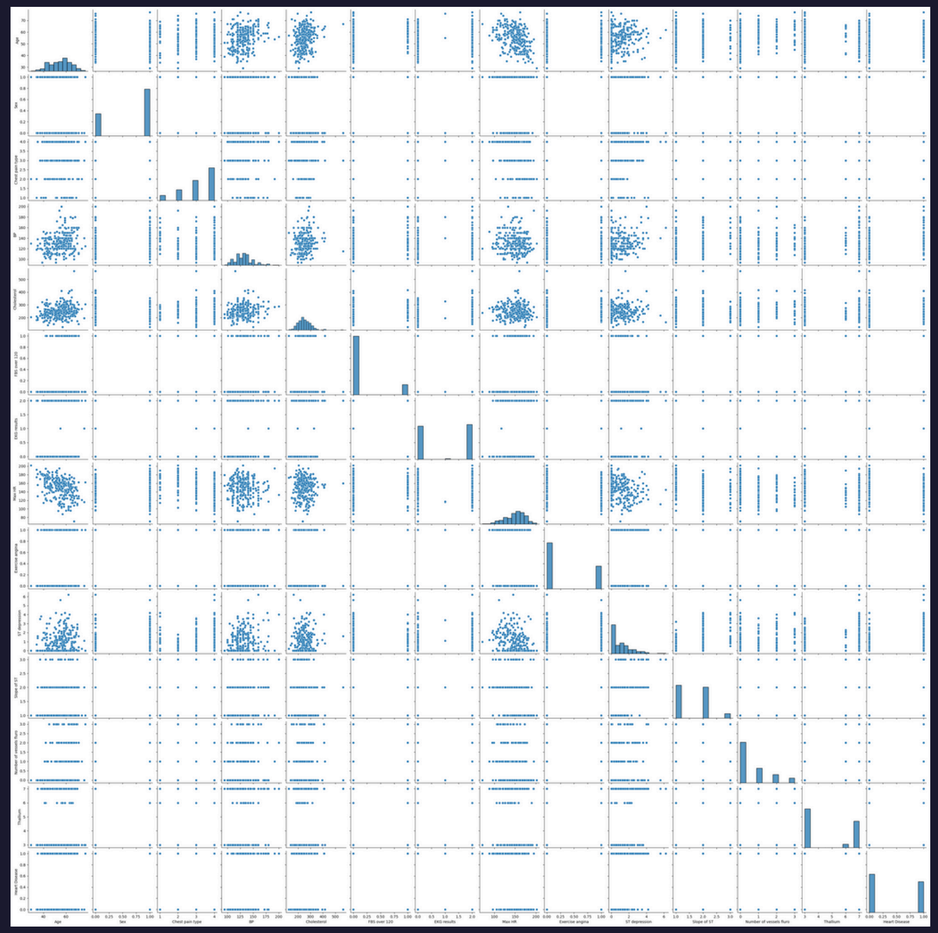
Bar plot



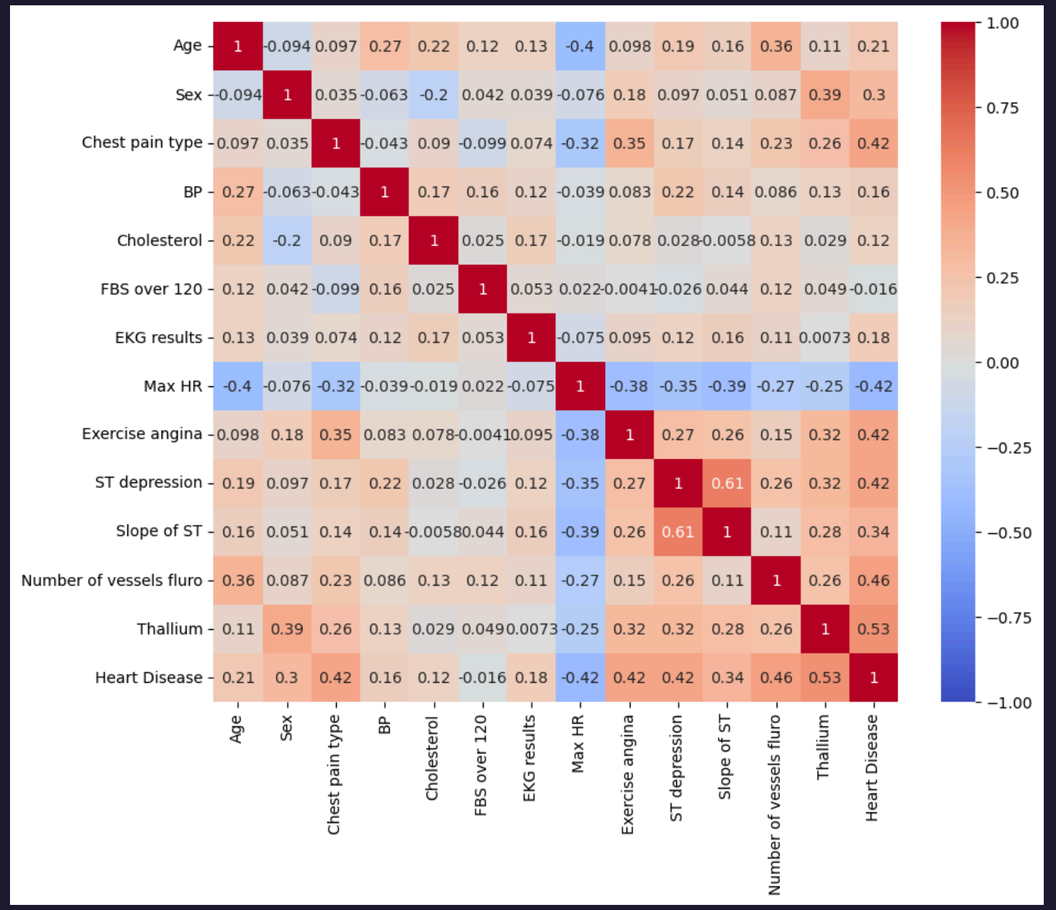
Box plot

IMDb

CONTD..
PAIR PLOT AND HEATMAP



Pair-plot



Heatmap

DIVIDING THE DATASET INTO INPUT AND OUTPUT FOR PREDICTION

```
In [18]: df.dtypes
Out[18]: Age                       int64
         Sex                       int64
         Chest pain type           int64
         BP                        int64
         Cholesterol               int64
         FBS over 120              int64
         EKG results               int64
         Max HR                    int64
         Exercise angina           int64
         ST depression           float64
         Slope of ST               int64
         Number of vessels fluro   int64
         Thallium                  int64
         Heart Disease            object
         dtype: object
```

```
In [44]: X = df.drop(columns=['Heart Disease'])
         y = df['Heart Disease']
```

These are all my columns with their respective datatypes

As there is a categorial column Heart Disease which have only two unique values i.e. either true or false so we have considered the column Heart Disease as the Output Column and all other columns as input columns.

In [42]: x

Out[42]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro	Thallium
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
265	52	1	3	172	199	1	0	162	0	0.5	1	0	7
266	44	1	2	120	263	0	0	173	0	0.0	1	0	7
267	56	0	2	140	294	0	2	153	0	1.3	2	0	3
268	57	1	4	140	192	0	0	148	0	0.4	2	0	6
269	67	1	4	160	286	0	2	108	1	1.5	2	3	3

270 rows × 13 columns

Input Columns

```
[43]: y
t[43]: 0       Presence
       1        Absence
       2       Presence
       3        Absence
       4        Absence
                 ...
       265      Absence
       266      Absence
       267      Absence
       268      Absence
       269     Presence
       Name: Heart Disease, Length: 270, dtype: object
```

Output Columns

IMDb

SELECTING MODEL-CLASSIFICATION/REGRESSION

As our Output Column is Categorical So we have to use Classifier models

So Some of the Classifier techniques are -:

- *Logistic Regression*
- *Gaussian Naive Bayes (GNB)*
- *Support Vector Regression (SVR)*
- *K-Nearest Neighbors (KNN)*

IMDb

STANDARD SCALER AND PCA
(GETTING THE DATASET READY FOR CLASSIFICATION)

Feature Scaling

```
In [48]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)

In [49]: le = LabelEncoder()
         y=le.fit_transform(y)
```

StandardScaler is used to standardize features by removing the mean and scaling to unit variance, ensuring each feature contributes equally to the model

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms data into a set of orthogonal components, capturing the most variance. It simplifies complex datasets while preserving essential information.

Feature Extraction

```
In [50]: from sklearn.decomposition import PCA

In [51]: pca = PCA(n_components=9)
         X_train_trf = pca.fit_transform(X_train)
         X_test_trf = pca.transform(X_test)
```

IMDb

CALCULATING R2 SCORE AND MEAN SQUARED ERROR-:

***R^2 Score**: The R^2 score, or coefficient of determination, measures how well the regression predictions approximate the real data points, indicating the proportion of the variance in the dependent variable explained by the independent variables.*

***Mean Squared Error (MSE):** Mean Squared Error quantifies the average squared difference between the observed actual outcomes and the predicted values, serving as a measure of the quality of an estimator.*

IMDb

Logistic Regression and Gaussian Naive Bayes (GNB)

Model Building

In [52]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
```

In [53]:
```
lr = LogisticRegression()
```

In [54]:
```
lr.fit(X_train_trf,y_train)
```

Out[54]:
```
LogisticRegression
LogisticRegression()
```

In [55]:
```
y_pred = lr.predict(X_test_trf)
```

Accuracy Prediction

In [56]:
```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
```

In [57]:
```
acc = accuracy_score(y_pred,y_test)
pre = precision_score(y_pred,y_test,average = 'macro')
rec = recall_score(y_pred,y_test,average = 'macro')
f1 = f1_score(y_pred,y_test,average = 'macro')
print('f1',f1,'acc=',acc,'pre',pre,'rec',rec)
```

```
f1 0.8472385428907168 acc= 0.8555555555555555 pre 0.8425925925925926 rec 0.8540669856459331
```

LOGISTIC REGRESSION

Predicted label

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train_trf,y_train)
```

```
GaussianNB
GaussianNB()
```

```
gpred = gnb.predict(X_test_trf )
gpred
```

```
array([1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 0], dtype=int64)
```

```
acc = accuracy_score(gpred,y_test)
pre = precision_score(gpred,y_test,average = 'macro')
rec = recall_score(gpred,y_test,average = 'macro')
f1 = f1_score(gpred,y_test,average = 'macro')
print('f1',f1,'acc=',acc,'pre',pre,'rec',rec)
```

```
f1 0.8237367802585194 acc= 0.8333333333333334 pre 0.8194444444444444 rec 0.8301435406698565
```

Gaussian Naive Bayes (GNB)

IMDb

In [64]:
```
knn = KNeighborsClassifier()
```

In [65]:
```
knn.fit(X_train_trf,y_train)
y_pred2 = knn.predict(X_test_trf)
```

In [66]:
```
acc = accuracy_score(y_pred2,y_test)
pre = precision_score(y_pred2,y_test,average = 'macro')
rec = recall_score(y_pred2,y_test,average = 'macro')
f1 = f1_score(y_pred2,y_test,average = 'macro')
print('f1',f1,'acc=',acc,'pre',pre,'rec',rec)
```

```
f1 0.7503632281072514 acc= 0.7666666666666667 pre 0.7453703703703703 rec 0.7607982504100601
```

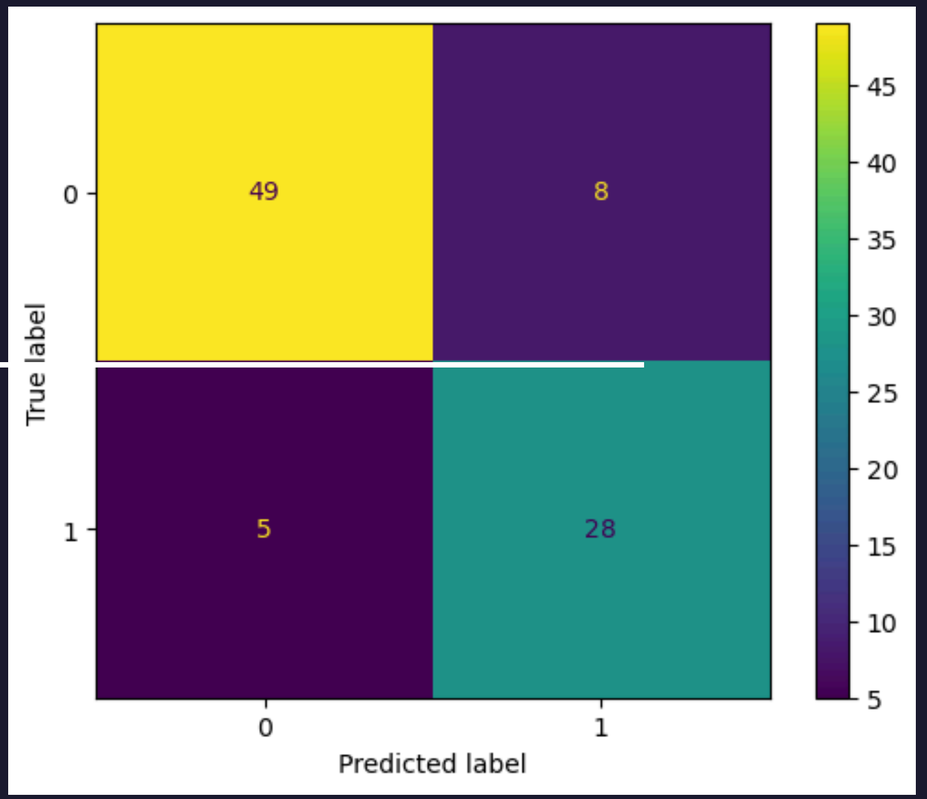
In [67]:
```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train_trf,y_train)
y_pred3 = svc.predict(X_test_trf)
acc = accuracy_score(y_pred3,y_test)
pre = precision_score(y_pred3,y_test,average = 'macro')
rec = recall_score(y_pred3,y_test,average = 'macro')
f1 = f1_score(y_pred3,y_test,average = 'macro')
print('f1',f1,'acc=',acc,'pre',pre,'rec',rec)
```

```
f1 0.812987012987013 acc= 0.8222222222222222 pre 0.8101851851851851 rec 0.8167016806722689
```

K-Nearest Neighbours (KNN) and Support Vector Regression (SVR)

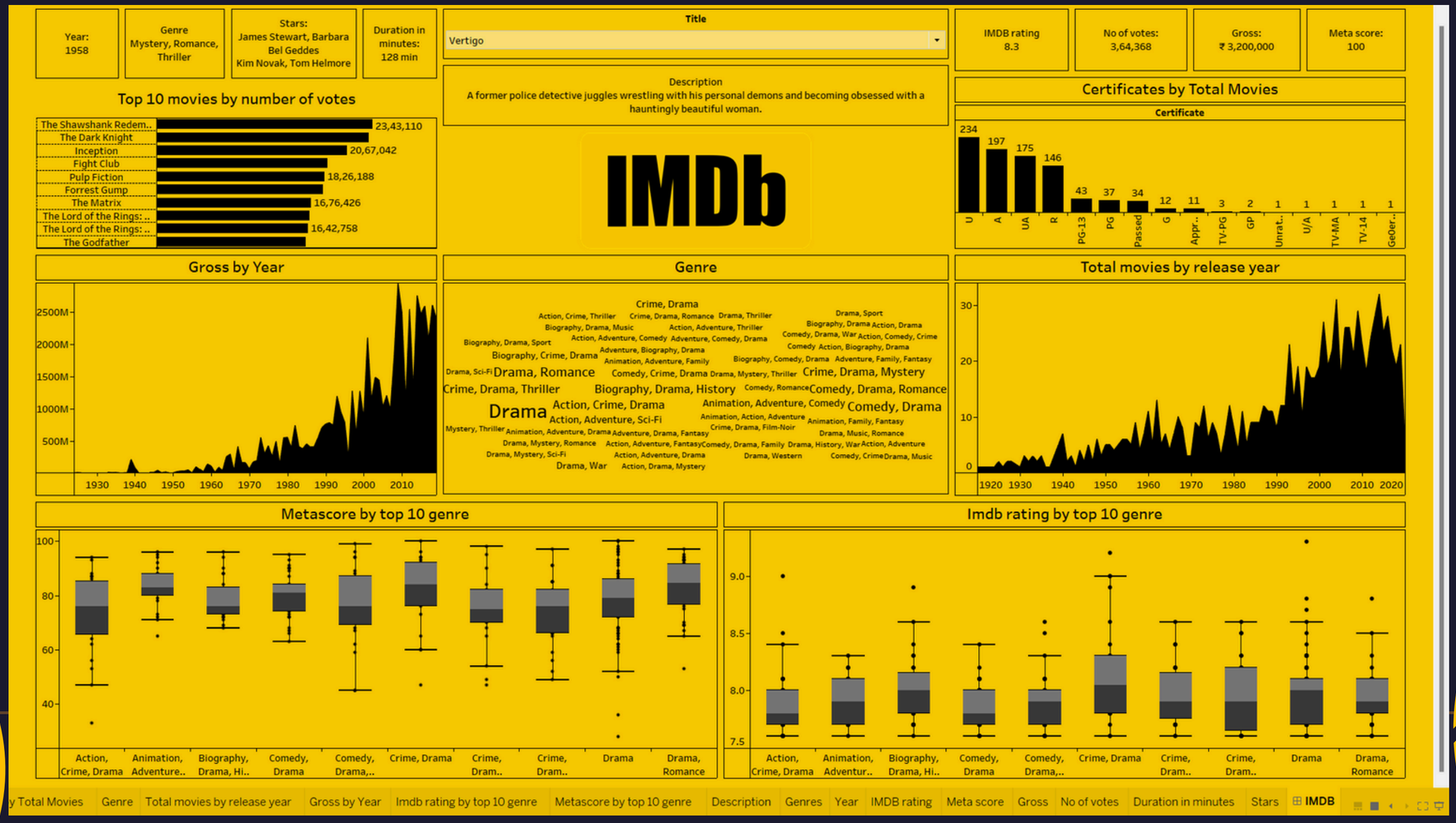
IMDb

A confusion matrix is a table used to evaluate the performance of a classification algorithm, displaying the counts of true positives, true negatives, false positives, and false negatives. It provides insight into the accuracy and errors of the model across different classes.



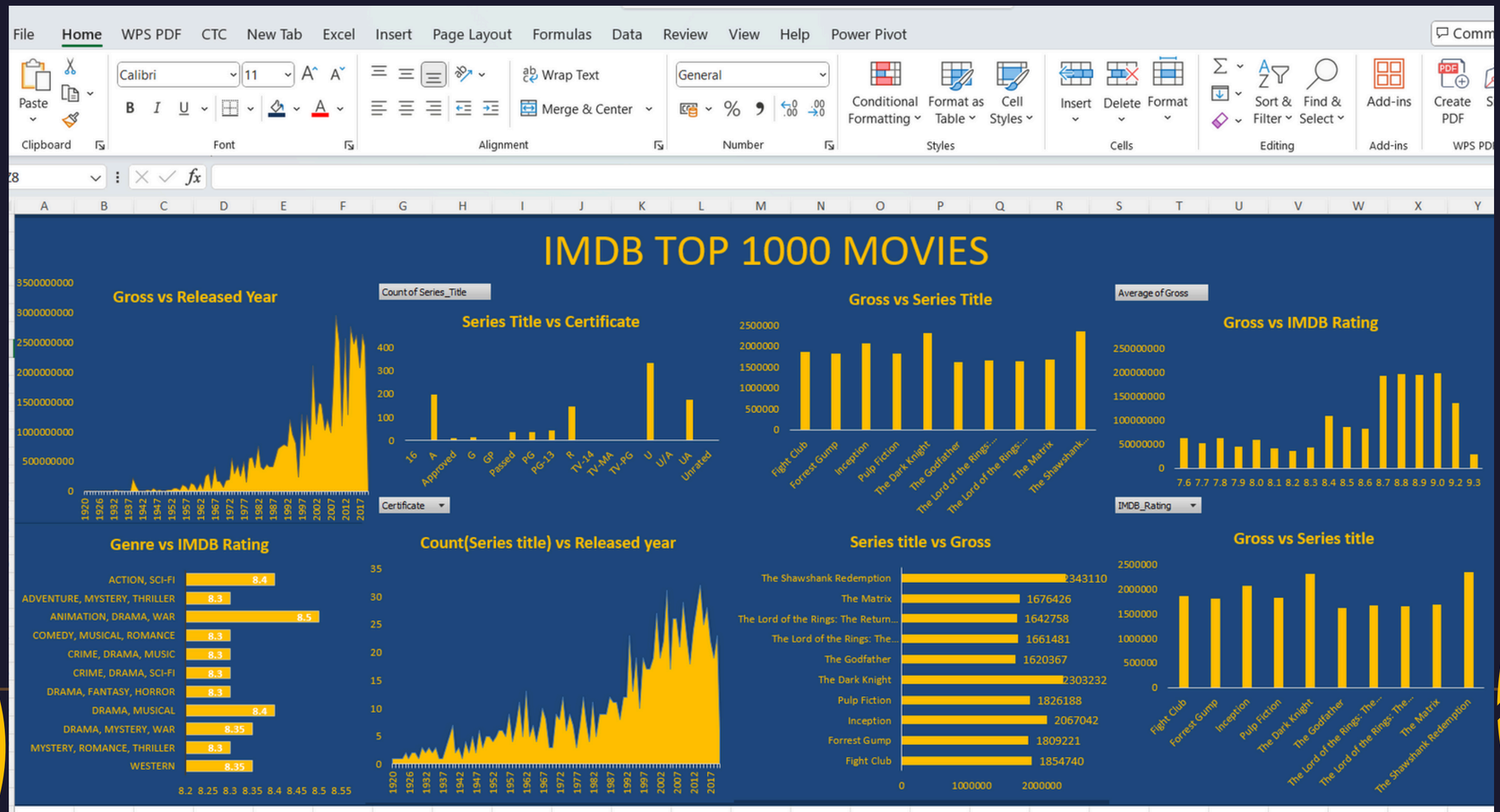
IMDb

TABLEU DASHBOARD



IMDb

EXCEL DASHBOARD



IMDb

CONCLUSION-:

We have tried several techniques but the best performance is given by Logistic Regression with a f1 score of 0.84 , accuracy score of 0.85 , precision score of 0.84 and recall score of 0.85

We have also created interactive dashboards in Excel and Tableu for better data analysis and Visualisation.