```
#task 2: MOVIE RATING PREDICTION WITH PYTHON
```

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

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

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving IMDb Movies India.csv to IMDb Movies India (4).csv
```

```
df = pd.read_csv("IMDb Movies India.csv", encoding="latin-1")
df.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,	4.4	35	Ovais Khan	Prateik	Ishita Raj	Siddhant

```
df.shape
```

```
(15509, 10)
```

```
df.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
```

```
df.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
```

```
df.duplicated().sum()
```

```
6
```

```
df.dropna(inplace=True)
df.isnull().sum()
```

```
Name        0
Year        0
Duration    0
Genre       0
Rating      0
Votes       0
Director    0
Actor 1     0
Actor 2     0
Actor 3     0
dtype: int64
```

```
df.drop_duplicates(inplace=True)
```

```
df.columns
```

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

Data Preprocessing

```
#Replacing the brackets from year column
df['Year'] = df['Year'].str.replace(r'[()]', '', regex=True).astype(int)
```

```
# Remove the min word from 'Duration' column and convert all values to numeric
df['Duration'] = pd.to_numeric(df['Duration'].str.replace(' min', ''))
```

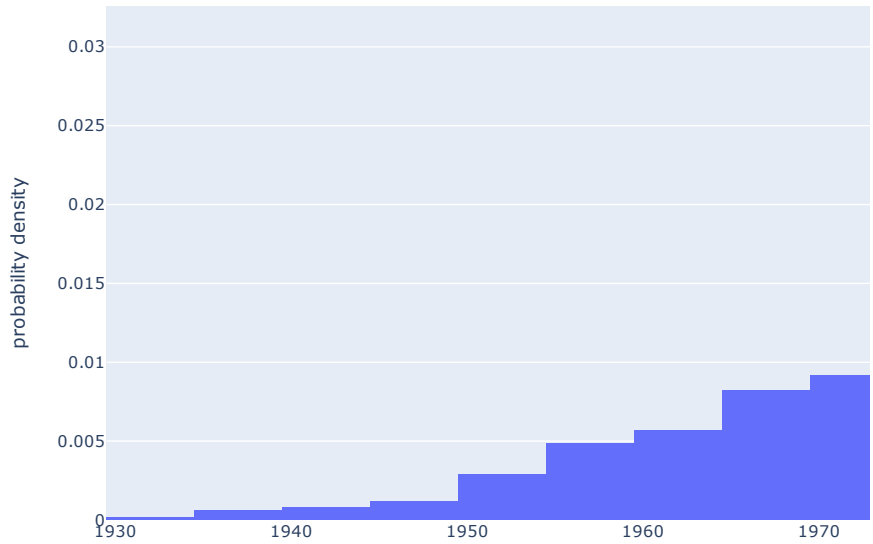
```
df['Genre'] = df['Genre'].str.split(', ')
df = df.explode('Genre')
df['Genre'].fillna(df['Genre'].mode()[0], inplace=True)
```

```
# Convert 'Votes' to numeric and replace the , to keep only numerical part
df['Votes'] = pd.to_numeric(df['Votes'].str.replace(',', ''))
```

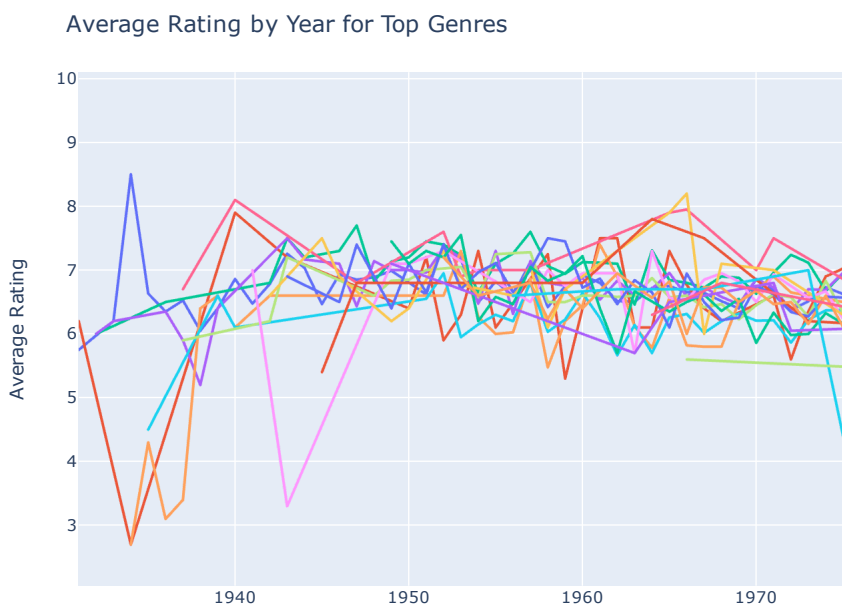
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11979 entries, 1 to 15508
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      11979 non-null  object
 1   Year      11979 non-null  int64
 2   Duration  11979 non-null  int64
 3   Genre     11979 non-null  object
 4   Rating    11979 non-null  float64
 5   Votes     11979 non-null  int64
 6   Director  11979 non-null  object
 7   Actor 1   11979 non-null  object
 8   Actor 2   11979 non-null  object
 9   Actor 3   11979 non-null  object
dtypes: float64(1), int64(3), object(6)
memory usage: 1.0+ MB
```

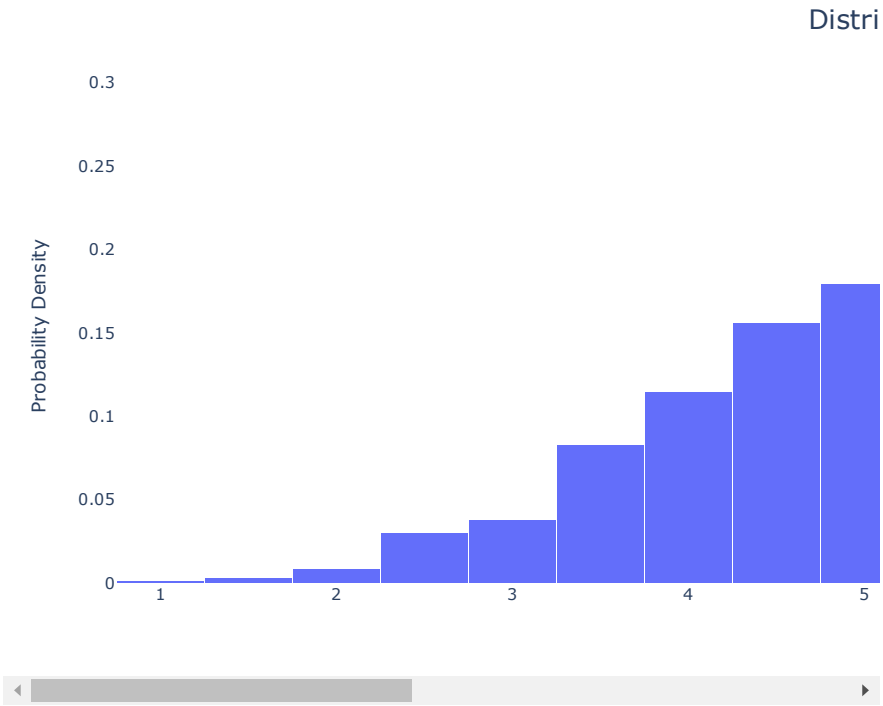
```
year = px.histogram(df, x = 'Year', histnorm='probability density', nbins = 30)
year.show()
```




```
# Group data by Year and calculate the average rating
avg_rating_by_year = df.groupby(['Year', 'Genre'])['Rating'].mean().reset_index()

# Get the top 10 genres
top_genres = df['Genre'].value_counts().head(10).index

# Filter the data to include only the top 3 genres
average_rating_by_year = avg_rating_by_year[avg_rating_by_year['Genre'].isin(top_genres)]

# Create the line plot with Plotly Express
fig = px.line(avg_rating_by_year, x='Year', y='Rating', color = "Genre")

# Updating the detals into chart like title and hue
fig.update_layout(title='Average Rating by Year for Top Genres', xaxis_title='Year', yaxis_title='Average Rating')

# Show the plot
fig.show()
```




```
#This histogram shows the distribution of ratings and its probable density

rating_fig = px.histogram(df, x = 'Rating', histnorm='probability density', nbins = 40)
rating_fig.update_layout(title='Distribution of Rating', title_x=0.5, title_pad=dict(t=20), title_font=dict(size=20), xaxis_title=
rating_fig.show()
```



Feature Engineering

```
# Importing essential libraries for model building

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.linear_model import LinearRegression

from sklearn.metrics import accuracy_score, mean_absolute_error, mean_squared_error, r2_score


# Dropping Name column because it doesn't impact the outcome
df.drop('Name', axis = 1, inplace = True)


# Grouping the columns with their average rating and then creating a new feature

genre_mean_rating = df.groupby('Genre')['Rating'].transform('mean')
df['Genre_mean_rating'] = genre_mean_rating

director_mean_rating = df.groupby('Director')['Rating'].transform('mean')
df['Director_encoded'] = director_mean_rating

actor1_mean_rating = df.groupby('Actor 1')['Rating'].transform('mean')
df['Actor1_encoded'] = actor1_mean_rating

actor2_mean_rating = df.groupby('Actor 2')['Rating'].transform('mean')
df['Actor2_encoded'] = actor2_mean_rating

actor3_mean_rating = df.groupby('Actor 3')['Rating'].transform('mean')
df['Actor3_encoded'] = actor3_mean_rating


#Keeping the predictor and target variable

X = df[[ 'Year', 'Votes', 'Duration', 'Genre_mean_rating','Director_encoded','Actor1_encoded', 'Actor2_encoded', 'Actor3_encoded']
y = df['Rating']


# Splitting the dataset into training and testing parts

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

Model Building

```
# Building machine learning model and training them
Model = LinearRegression()
Model.fit(X_train,y_train)
Model_pred = Model.predict(X_test)
```

```
 #Evaluating the performance of model with evaluation metrics

print('The performance evaluation of Logistic Regression is below: ', '\n')
print('Mean squared error: ',mean_squared_error(y_test, Model_pred))
print('Mean absolute error: ',mean_absolute_error(y_test, Model_pred))
print('R2 score: ',r2_score(y_test, Model_pred))
```

```
The performance evaluation of Logistic Regression is below:

Mean squared error:  0.4465441653985704
Mean absolute error:  0.4921902540765641
R2 score:  0.7641133663863862
```

Model Testing

```
# For testing, We create a new dataframe with values close to the any of our existing data to evaluate.

data = {'Year': [2020], 'Votes': [36], 'Duration': [105], 'Genre_mean_rating': [5.8], 'Director_encoded': [4.6], 'Actor1_encoded':
trail = pd.DataFrame(data)
```

```
 # Predict the movie rating by entered data
rating_predicted = Model.predict(trail)

# Display the predicted result from the Model
print(rating_predicted)
```

```
[4.02459319]
```