**PREDICTION OF IMDB SCORES**

**INTRODUCTION:**

Predicting IMDb scores for movies and TV shows is a fascinating application of data science and machine learning. The IMDb (Internet Movie Database) score is a numeric representation of a movie or TV show's overall quality, as rated by users and critics. In this data science project, we aim to develop a predictive model using Python to estimate IMDb scores based on various features associated with the content, such as cast, crew, genre, release year, and more.

The primary goal of this project is to create a model that can accurately forecast IMDb scores for new and existing movies or TV shows. Accurate predictions can provide valuable insights to filmmakers, studios, and streaming platforms by helping them understand the factors that influence audience perception and reception.

This step-by-step guide outlines the essential phases involved in the process:

1. Data Collection: Gathering a comprehensive dataset that includes both IMDb scores and relevant features for movies or TV shows.

2. Data Preprocessing: Cleaning and preparing the dataset by addressing issues like missing values, data encoding, and splitting the data into training and testing sets.

3. Feature Engineering: Creating and selecting relevant features that have a potential impact on IMDb scores. This might involve text analysis of movie summaries, exploring the significance of directors and actors, or considering the effect of release year.

4. Model Selection: Choosing an appropriate machine learning algorithm for regression tasks, as IMDb score prediction is essentiallya regression problem. Common choices include Linear Regression, Random Forest, Support Vector Regression, and ensemble methods.

5. Model Training: Training the selected model on the training data and ensuring it learns the relationships between the features and IMDb scores.

6. Hyperparameter Tuning: Optimizing the model's hyperparameters to improve its predictive performance.

7. Model Evaluation: Assessing the model's accuracy and generalization by evaluating it on a separate test dataset.

8. Feature Importance Analysis (Optional): Exploring which features have the most significant impact on IMDb score predictions can provide valuable insights into the factors that influence audience ratings.

**GIVEN DATA SET:**

****

585 ROWS X 6 COLUMNS

**NECESSARY STEPS TO FOLLOW:**

**1.IMPORT LIBRARIES:**

Start by importing the necessary libraries:

Program:

Import pandas as pd

Import numpy as np

Import matplotlib as plt

from Sklearn.model\_selection import train\_test\_split

from sklear.preprocessing import standardScalar

**2.LOAD THE DATASET:**

Load your dataset into a pandas Dataframe.You can typically find the imdb scores easily as it would be easy to plot graphs and also work with the frames by removing unnecessary data and high efficiency.

Program:

df=pd.read\_csv(‘C:\Users\srini\OneDrive\Desktop\NetflixOriginals.csv’)

pd.read()

**3.EXPLORATORY DATA ANALYSIS(EDA):**

Performing EDA to understand your data better this includes checking the missing values and performing numerical operations. After these steps we can create real time visualization by creating insights using 2d and 3d graphs.

Program:

# Check for missing values

print(df.isnull().sum())

# Explore statistics

print(df.describe())

# Visualize the data (e.g., histograms, scatterplots, etc)

**4. Feature Engineering:**

Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling date/time data, or scaling numerical features.

Program:

# Example: One-hot encoding for categorical variables

df=pd.dummies(df,columns=[‘Avg.IMDB scores’, ‘Avg.Runtime’])

**5.Split the data:**

Split your dataset into training and testing sets. This helps you evaluate your model’s performance later.

X = df.drop(‘IMDB scores’, axis=1)

Y = df[‘price’] #target variable

X\_train, X\_test, y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=42)

**6.Feature scaling:**

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean=0 and std=1) is a common choice.

Program:

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

#### Challenges involved in loading and preprocessing a IMDb scores dataset(Netflixoriginals dataset)

Predicting IMDb scores for movies involves several challenges when it comes to loading and preprocessing data. Here are some key challenges:

Data Collection: Acquiring a comprehensive dataset of movies with relevant features like cast, crew, genre, budget, and release year can be challenging. IMDb's data may have limitations due to licensing and privacy concerns.

Missing Data: Movie datasets often have missing values for various attributes, such as budget, box office earnings, or user reviews. Handling missing data is crucial for accurate predictions.

Data Cleaning: Noisy or inconsistent data, such as misspelled movie titles or duplicate entries, needs to be cleaned to ensure data quality.

Scaling and Normalization: Different features may have varying scales and units. Standardizing or normalizing the data is necessary to ensure that features contribute equally to the model.

Handling Categorical Data: Encoding categorical features, like genres or movie ratings, into numerical values is crucial. Techniques like one-hot encoding or label encoding are often used.

Outliers: Identifying and dealing with outliers in the data can be challenging, as extreme values can skew the predictions.

#### How to overcome the challenges in loading and preprocessing a IMDb scores dataset (Netflix originals)

There are a number of things that can be done to overcome the challenges of loading and preprocessing a IMDb scores dataset(Netflixoriginals), including:

#### Use a data preprocessing library:

There are a number of libraries available that can help with data preprocessing tasks, such as handling missing values, encoding categorical variables, and scaling the features.

#### Carefully consider the specific needs of your model:

The best way to preprocess the data will depend on the specific machine learning algorithm that you are using. It is important to carefully consider the requirements of the algorithm and to preprocess the data in a way that is compatible with the algorithm

#### Validate the preprocessed data:

It is important to validate the preprocessed data to ensure that it is in a format that can be used by the machine learning algorithm and that it is of high quality. This can be done by inspecting the data visually or by using statistical methods.

# Loading the dataset:

* Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
* The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks:

#### Identify the dataset:

The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

#### Load the dataset:

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

#### Preprocess the dataset:

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.

Identify the

dataset

Loading the

dataset

Preprocess the

dataset

Load the dataset

PROGRAM

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection

import train\_test\_split from sklearn.preprocessing

import StandardScaler from sklearn.metrics

import r2\_score, mean\_absolute\_error,mean\_squared\_error

from sklearn.linear\_model import LinearRegression from sklearn.linear\_model import Lasso

from sklearn.ensemble import RandomForestRegressor from sklearn.svm import SVR

import xgboost as xg

%matplotlib inline import warnings

warnings.filterwarnings("ignore")

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init .py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np\_minversion} and

<{np\_maxversion}"

**To load the dataset**

dataset = pd.read\_csv('E:/Netflixoriginals.csv')

**Output**

****

# Preprocessing the dataset:

Data preprocessing is the process of cleaning, transforming, and integrating data in order to make it ready for analysis This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

VISUALIZATION

netflix %>%

ggplot(aes(x = `IMDB.Score`)) +

geom\_dotplot(binwidth = 0.1,

fill = "#2d2d2d",

color = "#e9ecef") +

labs(title = "IMDB Score Distribution") +

theme\_minimal() +

theme(

legend.position = "none",

plot.title = element\_text(

family = "Bebas Neue",

size = 25,

colour = "#E50914"

),

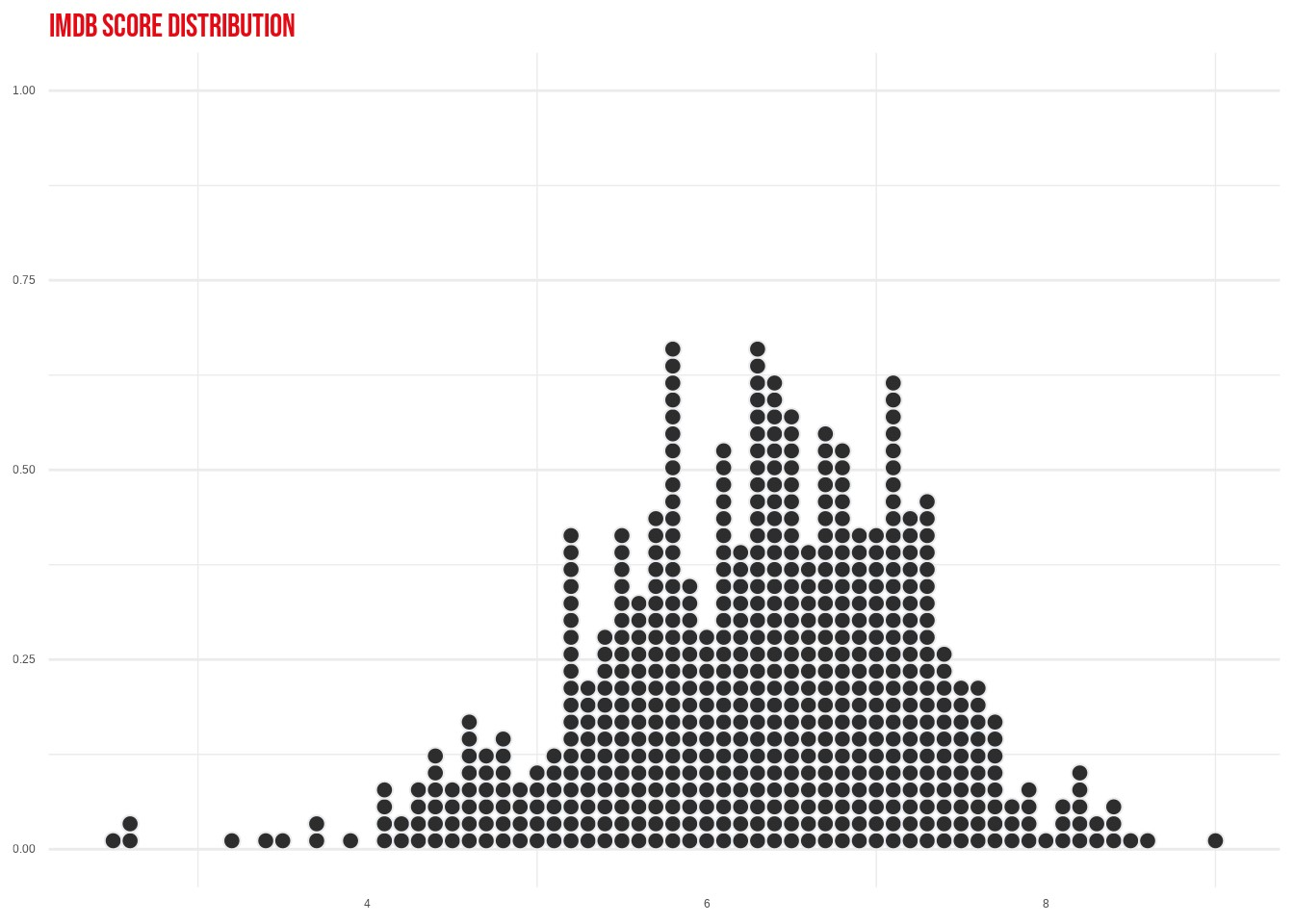
axis.title.x = element\_blank(),

axis.title.y = element\_blank(),

panel.grid.major.x = element\_blank()

)

**Output**



ggplot(aes(x=`IMDB.Score`, y= Runtime)) +

geom\_point() +

geom\_smooth(method = lm,colour = "#E50914") +

labs(title = "Runtime vs IMDB Rating") +

theme\_minimal() +

scale\_fill\_manual(values = c( "#2d2d2d", "#E50914")) +

theme(

legend.position = "none",

plot.title = element\_text(

family = "Bebas Neue",

size = 25,

colour = "#E50914"

),

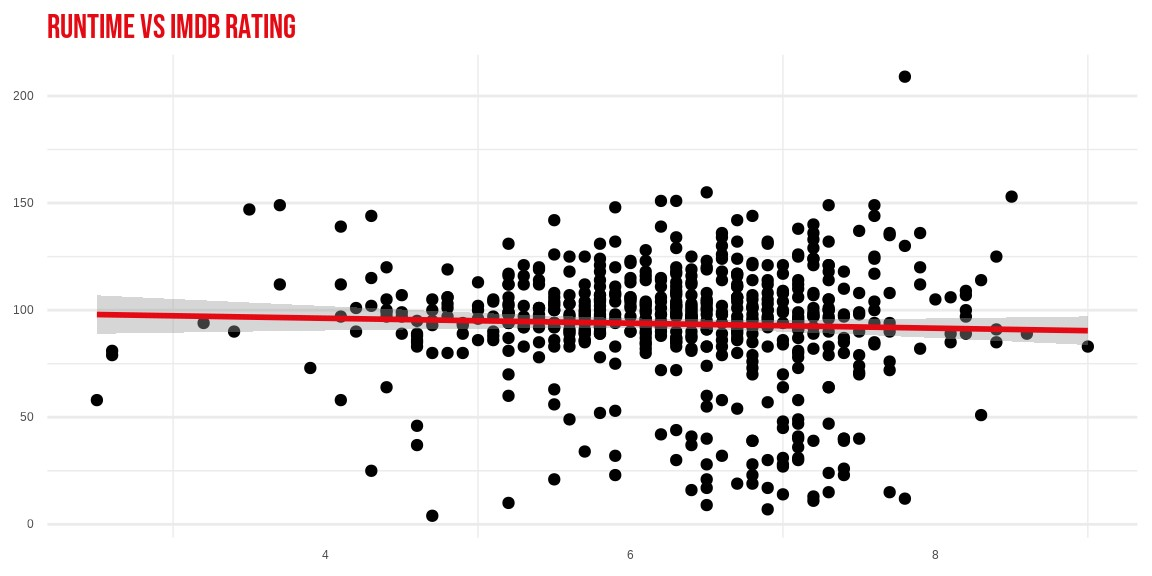
axis.title.x = element\_blank(),

axis.title.y = element\_blank(),

panel.grid.major.x = element\_blank())

## `geom\_smooth()` using formula 'y ~ x'

**Output**



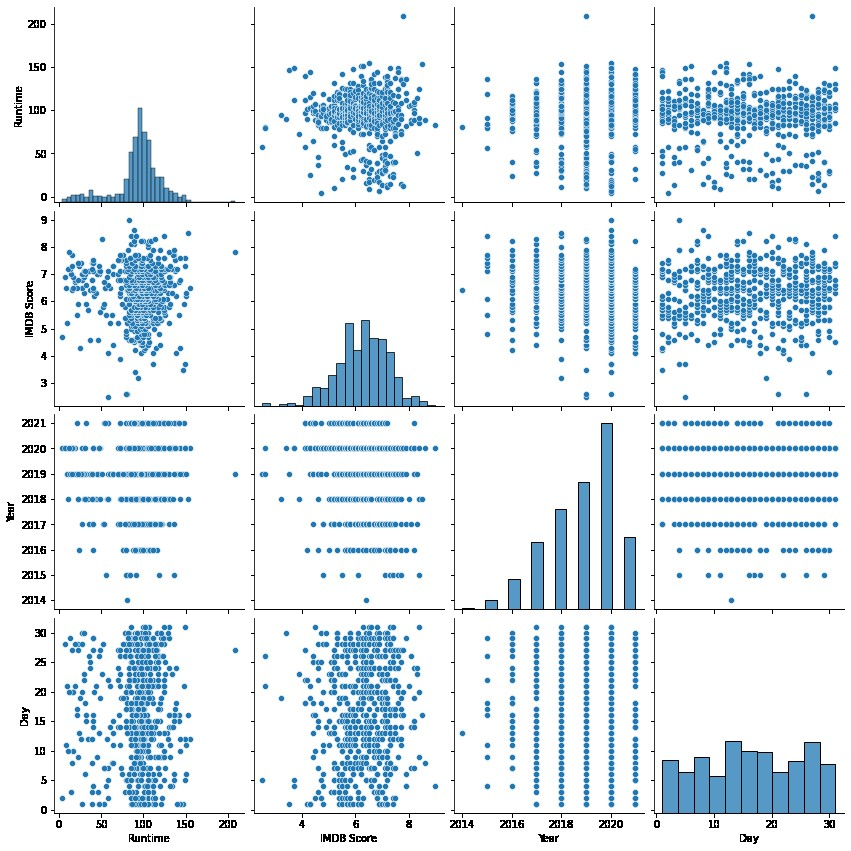
Visualazing correlation using heatmap

sns.heatmap(df.corr(),annot=True)



sns.pairplot(movies, size = 3)

plt.show()



**Program:**

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset (make sure you have a CSV file or another format)

# Replace 'movies\_dataset.csv' with the actual dataset file.

data = pd.read\_csv('movies\_dataset.csv')

# Assuming your dataset has a column named 'IMDb\_Score'

imdb\_scores = data['IMDb\_Score']

# Create a histogram to visualize IMDb scores

plt.figure(figsize=(10, 6))

plt.hist(imdb\_scores, bins=20, edgecolor='k', alpha=0.7)

plt.xlabel('IMDb Score')

plt.ylabel('Number of Movies')

plt.title('Distribution of IMDb Scores')

plt.grid(True)

# Show the histogram

plt.show()

**#Step 1: Loading the dataset:**

Data=pd.read\_csv('E:Netflixoriginals.csv')

**#Step 2: Exploratory Data Analysis:**

import matplotlib.pyplot as plt

import seaborn as sns

# Assuming 'df' is your IMDb dataset loaded into a DataFrame

# Count the number of movies in each genre

genre\_counts = df['Genre'].value\_counts()

# Sort the genres by count in descending order

sorted\_genres = genre\_counts.index

n\_largest\_genre = genre\_counts.values

# Create a bar plot

plt.figure(figsize=(12, 7))

sns.barplot(x=sorted\_genres, y=n\_largest\_genre, color='blue')

plt.xticks(rotation=90)

plt.xlabel('Genres')

plt.ylabel('Number of Movies in Genre')

plt.title('Number of Movies in Each Genre')

plt.show()

**#Step 3: Feature Engineering:**

print("Feature Engineering")

# Assuming you have already loaded your IMDb dataset into the 'data' DataFrame

# Separate features and target variable

X = data.drop('IMDb\_score', axis=1) # Assuming 'IMDb\_score' is your target variable

y = data['IMDb\_score']

# Define which columns should be one-hot encoded (categorical)

categorical\_cols = ['Genre', 'Director', 'Actors'] # Add relevant categorical columns from your IMDb dataset

# Define preprocessing steps using Column Transformer and Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler, OneHotEncoder

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), ['Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population', 'Avg. Area Income']), # Numeric features

('cat', OneHotEncoder(), categorical\_cols) # Categorical features

])

**# Step 4: Data Splitting**

print("Data Splitting")

# Assuming you have already defined X and y for your IMDb dataset

from sklearn.model\_selection import train\_test\_split

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Print the shapes of the resulting datasets

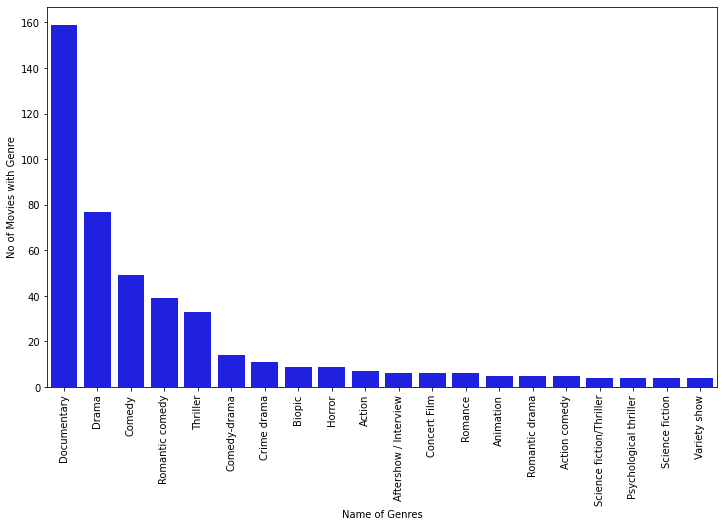
print("X\_train shape:", X\_train.shape)

print("X\_test shape:", X\_test.shape)

print("y\_train shape:", y\_train.shape)

print("y\_test shape:", y\_test.shape)

**Output:**

****

**CONCLUSION:**

In the quest to predict IMDb scores through the model, we have

embarked on a critical journey that begins with loading and

preprocessing the dataset.We have traversed through essential

steps, starting with importing the necessary libraries to facilitate

data manipulation and analysis.

* Understanding the data's structure, characteristics, and any

potential issues through exploratory data analysis (EDA) is

essential for informed decision-making.

* Data preprocessing emerged as a pivotal aspect of this process. It

involves cleaning, transforming, and refining the dataset to ensure

that it aligns with the requirements of data science.

* With these foundational steps completed, our dataset is now

primed for the subsequent stages of building and training a IMDB

score predictions model.