

Predicting IMDb Scores

Project Title: IMDb Score Prediction

Problem Statement: Develop a machine learning model to predict the IMDb scores of movies available on Films based on their genre, premiere date, runtime, and language. The model aims to accurately estimate the popularity of movies to assist users in discovering highly rated films that align with their preferences.

Project Steps

Phase 1: Problem Definition and Design Thinking

Problem Definition: The problem is to develop a machine learning model that predicts IMDb scores of movies available on Films based on features like genre, premiere date, runtime, and language. The objective is to create a model that accurately estimates the popularity of movies, helping users discover highly rated films that match their preferences. This project involves data preprocessing, feature engineering, model selection, training, and evaluation.

Design Thinking:

1. **Data Source:** Utilize a dataset containing information about movies, including features like genre, premiere date, runtime, language, and IMDb scores.
2. **Data Preprocessing:** Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.
3. **Feature Engineering:** Extract relevant features from the available data that could contribute to predicting IMDb scores.
4. **Model Selection:** Choose appropriate regression algorithms (e.g., Linear Regression, Random Forest Regressor) for predicting IMDb scores.
5. **Model Training:** Train the selected model using the preprocessed data.
6. **Evaluation:** Evaluate the model's performance using regression metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

Project Description:

Objective: The IMDb Scores Prediction project aims to develop a machine learning model capable of predicting IMDb movie ratings with a high degree of accuracy. This predictive model will be a valuable tool for filmmakers, studios, and movie enthusiasts to anticipate the potential success of a movie.

Scope: This project will focus on building a predictive model using historical IMDb data, encompassing various movie attributes such as cast and crew information, genre, budget, and release date. The model will aim to forecast IMDb scores for both upcoming and existing movies.

Data Sources: We will gather the necessary data from IMDb's extensive database and other reliable sources, including movie industry datasets and online movie databases.

Steps:

Data cleaning is a crucial step in any data analysis or prediction project, including an IMDb prediction project. Clean data is essential for building accurate and reliable machine learning models. Here's a general outline of the data cleaning process for an IMDb prediction project:

1. Data Collection and Inspection:

- Gather the IMDb dataset or data from reliable sources.
- Inspect the data to understand its structure, features, and potential issues.

2. Handling Missing Values:

- Identify missing values in the dataset.
- Decide on an appropriate strategy for handling missing data, which may include:
 - Removing rows or columns with too many missing values.
 - Imputing missing values using mean, median, mode, or more advanced methods like regression or decision trees.

3. Dealing with Duplicate Data:

- Check for and remove duplicate entries if they exist.
- Duplicates can distort analysis and model training.

4. Handling Outliers:

- Identify outliers in numerical features that may adversely affect predictions.
- Decide on a strategy for handling outliers, such as removing them or transforming the data.

5. Data Type Conversion:

- Ensure that data types are correctly assigned to each feature. For example, dates should be in datetime format, and categorical variables should be encoded appropriately.

6. Handling Categorical Data:

- Encode categorical variables into numerical format using techniques like one-hot encoding or label encoding.

7. Descriptive Statistics:

- Begin with descriptive statistics to summarize and describe the main features of your dataset. Common descriptive statistics include:
 - Measures of central tendency (mean, median, mode)
 - Measures of variability (range, variance, standard deviation)
 - Measures of distribution shape (skewness, kurtosis)
 - Frequency distributions and histograms

8. Data Visualization:

- Create visual representations of the data to aid in understanding. Visualization techniques include histograms, box plots, scatter plots, bar charts, and more.

9. Inferential Statistics:

- Move on to inferential statistics, which involve drawing conclusions from data and making predictions. Key concepts and techniques include:

10. Statistical Software and Tools:

- Utilize statistical software packages (e.g., R, Python with libraries like NumPy, SciPy, and StatsModels) and tools (e.g., Excel, SPSS) to perform analyses.

11. Interpretation:

- Interpret the results of your statistical analysis in the context of your problem or research question. Discuss the practical significance of your findings.

12. Reporting and Visualization:

- Present your results and insights using clear and effective visualizations, tables, and narrative explanations.
- Use data visualization tools (e.g., Matplotlib, Seaborn, ggplot2) to create informative graphs and charts.

Statistical analysis is a powerful tool for extracting valuable insights from data and making data-driven decisions. It allows you to quantify uncertainty, test hypotheses, and explore relationships within your dataset, ultimately aiding in problem-solving and informed decision-making.

Dataset used:

NetflixOriginals.csv

Columns: Title, Genre, Premiere, Runtime, IMDB Score, Language, sentiment.

Predicting IMDb scores using either Gradient Boosting or Neural Networks:

Design Plan for Predicting IMDb Scores using Gradient Boosting:

1. Data Collection and Preprocessing:

Gather IMDb dataset including features like movie genre, director, actors, budget, release date, etc.

Handle missing data, encode categorical variables, and normalize numerical features.

2. Feature Selection:

Analyze feature importance to select relevant features for the model.

3. Model Selection:

Choose Gradient Boosting algorithms such as XGBoost, LightGBM, or CatBoost due to their effectiveness in handling complex relationships in data.

4. Data Splitting:

Split the dataset into training and testing sets (typically 80-20 or 70-30 ratio).

5. Model Training:

Train the Gradient Boosting model on the training dataset.

Tune hyperparameters using techniques like Grid Search or Random Search for better accuracy.

6. Evaluation:

Evaluate the model using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

Validate the model on the test dataset to ensure its generalizability.

7. Optimization:

Fine-tune the model further if necessary for better accuracy.

Python Code for gradient boosting:

```
import pandas as pd

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.metrics import mean_squared_error

data=pd.read_csv("IMDB.csv")

X = data.drop('IMDb_Score', axis=1)

y = data['IMDb_Score']

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

clf = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3,
random_state=42)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
```

Design Plan for Predicting IMDb Scores using Neural Networks:

1. Data Collection and Preprocessing:

Gather IMDb dataset including features like movie genre, director, actors, budget, release date, etc.

Handle missing data, encode categorical variables, and normalize numerical features.

2. Feature Selection:

Analyze feature importance to select relevant features for the model.

3. Model Selection:

Choose a neural network architecture suitable for regression tasks, like feedforward neural networks or recurrent neural networks (RNNs).

4. Data Splitting:

Split the dataset into training and testing sets (typically 80-20 or 70-30 ratio).

5. Model Design and Training:

Design the neural network architecture with appropriate input, hidden, and output layers.

Choose activation functions, loss functions (mean squared error for regression), and optimizer (e.g., Adam, RMSprop).

Train the neural network on the training dataset.

6. Evaluation:

Evaluate the neural network model using the same metrics as Gradient Boosting models.

Validate the model on the test dataset.

7. Optimization:

Experiment with different architectures, activation functions, and regularization techniques to optimize the neural network.

Adjust hyper parameters like learning rate and batch size for better performance.

8. Prediction:

Use the trained neural network to predict IMDb scores for new or unseen data.

Additional Considerations:

Ensemble Methods (Optional): You can also explore ensemble methods where predictions from both Gradient Boosting and Neural Network models are combined for potentially higher accuracy.

Cross-Validation: Implement cross-validation techniques like k-fold cross-validation to ensure the model's robustness and reliability.

Remember that the choice between Gradient Boosting and Neural Networks might also depend on the size and complexity of your dataset. Experimentation and iterative refinement are key to achieving the best prediction accuracy.

IMDB Score Prediction

Data cleaning and pre-processing:

Data Cleaning:

1. Replace the missing values:

Missing values should be replaced in the data set in order to perform further calculations. Here we make use of python's "fillna()" method which is used to fill the null values. The python code used is given below:

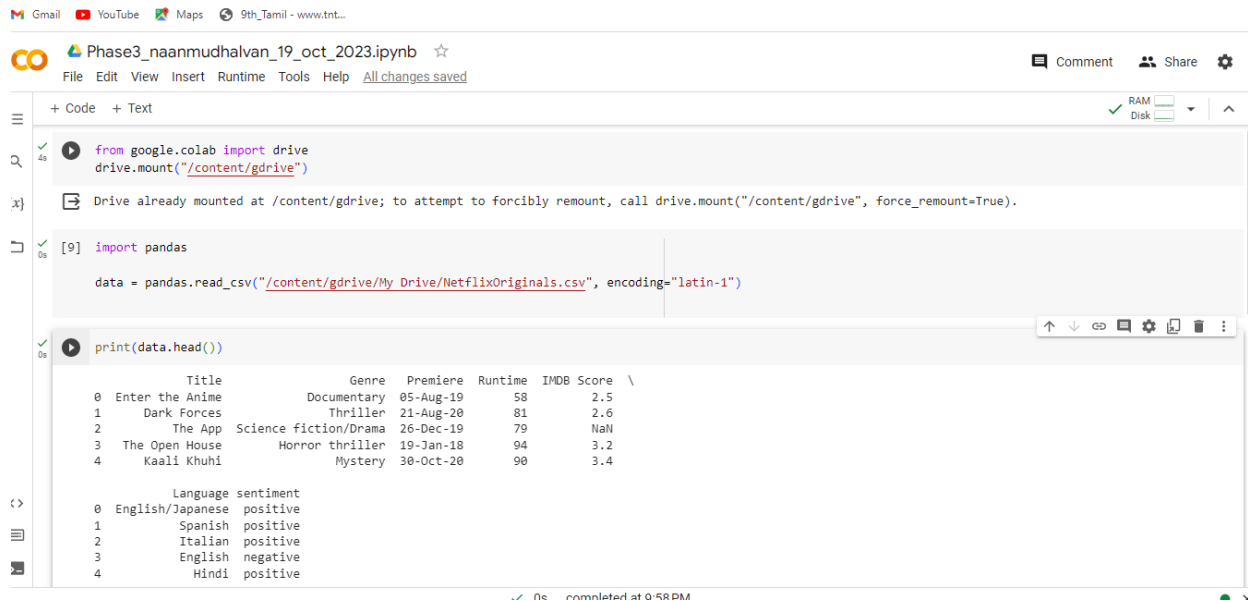
```
import pandas

data=pandas.read_csv("NetflixOriginals.csv")

data["IMDB Score"].fillna(5,inplace=True)
```

The execution of the code is shown below:

i) Before fillna()



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

import pandas

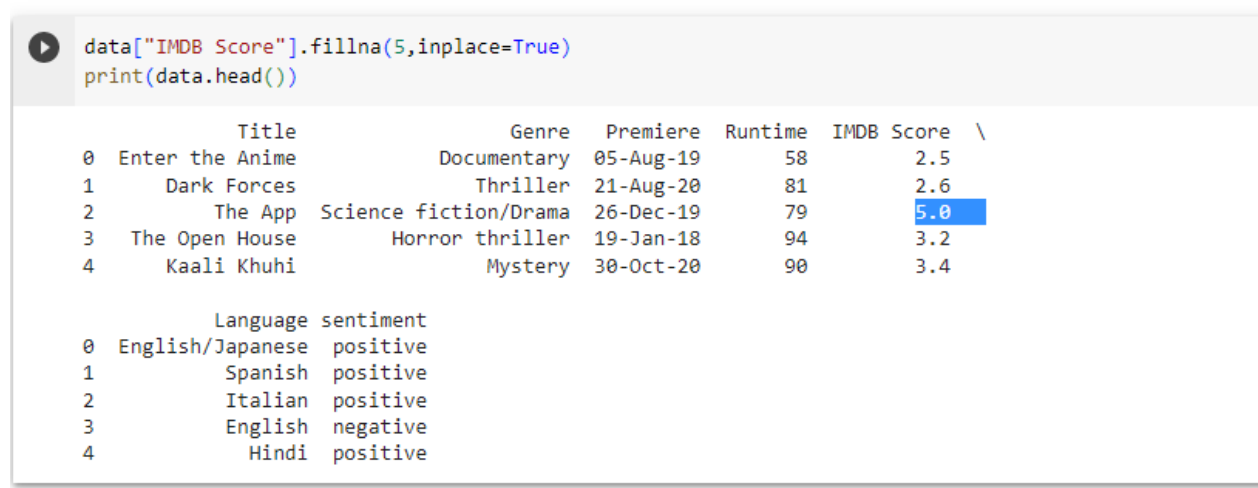
data = pandas.read_csv("/content/gdrive/My Drive/NetflixOriginals.csv", encoding="latin-1")

print(data.head())
```

	Title	Genre	Premiere	Runtime	IMDB Score	\
0	Enter the Anime	Documentary	05-Aug-19	58	2.5	
1	Dark Forces	Thriller	21-Aug-20	81	2.6	
2	The App	Science fiction/Drama	26-Dec-19	79	NaN	
3	The Open House	Horror thriller	19-Jan-18	94	3.2	
4	Kaali Khuhi	Mystery	30-Oct-20	90	3.4	

	Language	sentiment
0	English/Japanese	positive
1	Spanish	positive
2	Italian	positive
3	English	negative
4	Hindi	positive

ii) After fillna()



```
data["IMDB Score"].fillna(5,inplace=True)
print(data.head())
```

	Title	Genre	Premiere	Runtime	IMDB Score	\
0	Enter the Anime	Documentary	05-Aug-19	58	2.5	
1	Dark Forces	Thriller	21-Aug-20	81	2.6	
2	The App	Science fiction/Drama	26-Dec-19	79	5.0	
3	The Open House	Horror thriller	19-Jan-18	94	3.2	
4	Kaali Khuhi	Mystery	30-Oct-20	90	3.4	

	Language	sentiment
0	English/Japanese	positive
1	Spanish	positive
2	Italian	positive
3	English	negative
4	Hindi	positive

2. Converting categorical data to numerical data:

The categorical data such as male/female, positive/negative should be converted into numerical values. For example, 1-for male, 2-for female.

We use label encoder to do this conversion. The python code for this is:

```
from sklearn.preprocessing import LabelEncoder
```



```
label_encoder=LabelEncoder()
data=label_encoder.fit_transform(data["sentiment"])
print(data)
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
data=label_encoder.fit_transform(data["sentiment"])
print(data)
```

```
[1 1 1 0 1 1 1 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1 1 0 1 0 0 0
0 1 0 0 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1
0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0
0 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 1
0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0
0 1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 1 0 0 1
1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1
1 1 0 0 0 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 0
0 1 1 1 0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1
1 1 0 1 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1 1 1 0 0 0 1 1
1 1 0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 1 1 0 1
1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 0 1 0 0
0 0 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0
1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1
1 0 0 1 1 0 0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0
0 1 1 1 0 0 0 0 0 1 0 1 1 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1 0]
```

```
file_data=pandas.get_dummies(data,columns=["sentiment"])
print(file_data)
```

```
file_data=pandas.get_dummies(data,columns=["sentiment"])
print(file_data)
```

```
0 1
0 0 1
1 0 1
2 0 1
3 1 0
4 0 1
.. ..
579 0 1
580 0 1
581 1 0
582 0 1
583 1 0
```

```
[584 rows x 2 columns]
```

Otherwise, we can simply make use of replace() method:

```
data.replace({"positive":1,"negative":0},inplace=True)
print(data.head())
```



```
data.replace({"positive":1,"negative":0},inplace=True)  
print(data.head())
```

	Title	Genre	Premiere	Runtime	IMDB Score	\
0	Enter the Anime	Documentary	05-Aug-19	58	2.5	
1	Dark Forces	Thriller	21-Aug-20	81	2.6	
2	The App	Science fiction/Drama	26-Dec-19	79	5.0	
3	The Open House	Horror thriller	19-Jan-18	94	3.2	
4	Kaali Khuhi	Mystery	30-Oct-20	90	3.4	

	Language	sentiment
0	English/Japanese	1
1	Spanish	1
2	Italian	1
3	English	0
4	Hindi	1

3. Removal of outliers:

Outliers are the values that does not match the value range in the dataset. For example, if $A=[1,3,4,2,6,8,7,100]$, then 100 is the outlier since it is a out of range value in 'A'.

The removal of outliers from IMDB data st is very important because it may affect our prediction value.

The following python code removes the outliers from our dataset.

```
import numpy as np  
Q1=data['IMDB Score'].quantile(0.25)  
Q3=data['IMDB Score'].quantile(0.75)  
IQR=Q3-Q1  
lower_bound=Q1-1.5*IQR  
upper_bound=Q3+1.5*IQR  
outliers=data[(data['IMDB Score']<lower_bound)|(data['IMDB Score']>upper_bound)]  
print("Outliers in IMDB Scores:",outliers)
```

+ Code + Text

```
[42] Q1=data['IMDB Score'].quantile(0.25)
      Q3=data['IMDB Score'].quantile(0.75)
      IQR=Q3-Q1
```

```
lower_bound=Q1-1.5*IQR
upper_bound=Q3+1.5*IQR
outliers=data[(data['IMDB Score']<lower_bound)|(data['IMDB Score']>upper_bound)]
print("Outliers in IMDB Scores:",outliers)
```

		Title	Genre	Premiere \
0		Enter the Anime	Documentary	05-Aug-19
1		Dark Forces	Thriller	21-Aug-20
3		The Open House	Horror thriller	19-Jan-18
4		Kaali Khuhi	Mystery	30-Oct-20
5		Drive	Action	01-Nov-19
6		Leyla Everlasting	Comedy	04-Dec-20
7		The Last Days of American Crime	Heist film/Thriller	05-Jun-20
583		David Attenborough: A Life on Our Planet	Documentary	04-Oct-20

	Runtime	IMDB Score	Language	sentiment
0	58	2.5	English/Japanese	1
1	81	2.6	Spanish	1
3	94	3.2	English	0
4	90	3.4	Hindi	1
5	147	3.5	Hindi	1
6	112	3.7	Turkish	1
7	149	3.7	English	0

completed at 11:06 PM

```
data_cleaned=data[(data['IMDB Score']>=lower_bound)&(data['IMDB Score']<=upper_bound)]
print(data_cleaned)
```

+ Code + Text

```
data_cleaned=data[(data['IMDB Score']>=lower_bound)&(data['IMDB Score']<=upper_bound)]
print(data_cleaned)
```

		Title	Genre	\
2		The App	Science fiction/Drama	
8		Paradox	Musical/Western/Fantasy	
9		Sardar Ka Grandson	Comedy	
10		Searching for Sheela	Documentary	
11		The Call	Drama	
..		
578		Ben Platt: Live from Radio City Music Hall	Concert Film	
579		Taylor Swift: Reputation Stadium Tour	Concert Film	
580		Winter on Fire: Ukraine's Fight for Freedom	Documentary	
581		Springsteen on Broadway	One-man show	
582		Emicida: AmarElo - It's All For Yesterday	Documentary	

	Premiere	Runtime	IMDB Score	Language	sentiment
2	26-Dec-19	79	5.0	Italian	1
8	23-Mar-18	73	3.9	English	0
9	18-May-21	139	4.1	Hindi	1
10	22-Apr-21	58	4.1	English	0
11	27-Nov-20	112	4.1	Korean	0
..
578	20-May-20	85	8.4	English	0
579	31-Dec-18	125	8.4	English	1
580	09-Oct-15	91	8.4	English/Ukrainian/Russian	1
581	16-Dec-18	153	5.0	English	0
582	08-Dec-20	89	8.6	Portuguese	1

completed at 11:10 PM

Data preprocessing:

In order to perform IMDB score prediction, we need to split the data into training and testing. The following code is used to split the data:

```
from sklearn.model_selection import train_test_split
x = data_cleaned[['Runtime', 'sentiment']]
y = data_cleaned['IMDB Score']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

[576 rows x 7 columns]

```
✓ [11] from sklearn.model_selection import train_test_split
    Os x = data_cleaned[['Runtime', 'sentiment']]
        y = data_cleaned['IMDB Score']
        X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Feature Engineering

Feature engineering is a crucial step in improving the performance of machine learning models for predicting IMDB scores or any other kind of regression problem. IMDB scores are typically associated with movie ratings, so the features you engineer can be related to various aspects of the movies.

Below python code divides the movies into long, medium and short based on movie's runtime.

```
data = {'Movie': ['Movie1', 'Movie2', 'Movie3'],
        'Runtime': [90, 125, 160]}
df = pandas.DataFrame(data)
bins = [0, 100, 150, float('inf')]
labels = ['Short', 'Medium', 'Long']
df['Runtime_Category'] = pd.cut(df['Runtime'], bins=bins, labels=labels)
category_mapping = {'Short': 1, 'Medium': 2, 'Long': 3}
df['Runtime_Category'] = df['Runtime_Category'].map(category_mapping)
print(df)
```

```

data = {'Movie': ['Movie1', 'Movie2', 'Movie3'],
        'Runtime': [90, 125, 160]}
df = pandas.DataFrame(data)
bins = [0, 100, 150, float('inf')]
labels = ['Short', 'Medium', 'Long']
df['Runtime_Category'] = pd.cut(df['Runtime'], bins=bins, labels=labels)
category_mapping = {'Short': 1, 'Medium': 2, 'Long': 3}
df['Runtime_Category'] = df['Runtime_Category'].map(category_mapping)
print(df)

```

	Movie	Runtime	Runtime_Category
0	Movie1	90	1
1	Movie2	125	2
2	Movie3	160	3

Model Training:

The training process for predicting IMDB scores involves the following:

Data Splitting:

```

from sklearn.model_selection import train_test_split

X = data[['sentiment', 'Runtime']] # Features
y = data['IMDB Score'] # Target variable

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

```

Model Selection:

```

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, random_state=42)

```

Model Training:

```

odel.fit(X_train, y_train)

```



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



RandomForestRegressor

RandomForestRegressor(random_state=42)

Evaluation:

The below python code for evaluating the data of IMDB:

```
from sklearn.metrics import mean_squared_error, r2_score
```

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
```



```
from sklearn.metrics import mean_squared_error, r2_score
```

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
```

```
Mean Squared Error: 1.6862085227621486
R-squared (R2) Score: -0.34047099938390657
```

Conclusion:

Thus the prediction of IMDB Scores is made successfully using the python libraries.