PREDICTING IMDB SCORES

PHASE 4

Feature Engineering:

Release Year Features:

Extract the year from the "Release Year" column.

Genre Features:

Creating binary features for each unique genre. If a movie belongs to a particular genre, set the corresponding feature to 1; otherwise, set it to 0.

Director and Actor Features:

Average IMDb score of movies directed by a specific director. Average IMDb score of movies featuring a particular actor.

Runtime Features:

Categorize runtime into buckets (e.g., short, medium, long).

Calculating the average IMDb score for each runtime category.

Language Features:

Similar to the country features, creating binary features for each unique language spoken in the movie.

Number of Awards Features:

Creating a feature indicating the number of awards or nominations a movie has received.

Production Company Features:

Engineer features related to the production companies, such as the number of movies produced by a specific company and their average IMDb scores.

Rating Features:

Use a one-hot encoding for the movie rating (e.g., G, PG, PG-13, R, etc.).

Movie Duration Features:

Create features based on the movie's duration, such as the number of minutes or

categories like "short," "medium," and "long."

Interaction Features:

Creating interaction features that combine the effects of two or more existing features. For example, a feature that multiplies the number of user reviews by the average user rating.

Performing some of this feature engineering in Python:

```
# Extracting year and creating decade features
data['Release_Year'] =
data['ReleaseYear'].str.extract(r'(\d{4})').astype(int)
data['Decade_2000s'] =
(data['Release_Year'] >= 2000) & (data['Release_Year']
< 2010)

data['Decade_2010s'] = (data['Release_Year'] >= 2010)
&(data['Release_Year']
< 2020)

data['Decade_2020s'] = (data['Release_Year'] >= 2020)
```

Creating genre features

```
genres = data['Genre'].str.get_dummies(',')

data = pd.concat([data, genres], axis=1)
```

```
# Calculate average IMDb scores for directors
director_avg_scores =
data.groupby('Director')['IMDb'].mean().reset_index()
director_avg_scores.columns = ['Director',
'Director_Avg_IMDb'] data =
data.merge(director_avg_scores, on='Director', how='left')
```

Calculate average IMDb scores for actors

```
actor_avg_scores =
data.groupby('Actors')['IMDb'].mean().reset_index()
actor_avg_scores.columns = ['Actors',
'Actors_Avg_IMDb'] data =
data.merge(actor_avg_scores, on='Actors', how='left']
```

After feature engineering, you can proceed with model training and evaluation .

Model training:

Data Preprocessing and Feature Selection:

Preprocess the dataset by performing the feature engineering steps mentioned earlier.

Select the features you want to include in your model and define the target variable (IMDb scores). import pandas as pd

Load and preprocess your dataset (feature engineering)

Define your features and
target variable Split the Data:

Split the dataset into a training set and a testing set. This allows you to train the model on one subset and evaluate it on another form

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

Choose a Model:

```
from sklearn.linear_model import
LinearRegression model = LinearRegression()
```

Train the chosen model using the training data.

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

Model Evaluation:

Evaluate the model using appropriate regression metrics, such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2).

```
from sklearn.metrics import
mean_squared_error, r2_score y_pred
model.predict(X_test) mse =
mean_squared_error(y_test, y_pred) rmse =
mse ** 0.5
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:",
mse) print("Root Mean Squared
Error:", rmse) print("R-
squared:", r2)
```

Evaluation:

We can use regression evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2):

Data Preprocessing and Model Preparation:

Preprocess the dataset, perform feature engineering, and prepare the model, as mentioned in the previous response

Model Evaluation:

Once the model is trained and predictions are made, evaluate the model using various regression metrics.

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

Assuming you have trained the model and obtained predictions as y_pred

Calculate the Mean Squared Error (MSE)

```
mse = mean_squared_error(y_test, y_pred)
```

Calculate the Root Mean Squared Error

```
(RMSE) rmse = mse ** 0.5
```

Visualization:

By importing libraries:

import matplotlib.pyplot as plt

```
# Visualize actual IMDb scores vs.
predicted scores
 plt.scatter(y_test, y_pred)
plt.xlabel("Actual IMDb Scores")
plt.ylabel("Predicted IMDb Scores")
plt.title("Actual vs. Predicted IMDb
Scores") plt.show()
```

Model Interpretation:

Analyzing the model's coefficients (for linear models) or feature importance (for tree-based models) to understand which features have the most significant impact on IMDb scores.

Continuous Improvement:

Continuously monitor and update the model as new data becomes available. IMDb scores can change over time due to new user reviews and ratings.

Reporting and Presentation:

Summarizing the model's performance, findings, and any limitations in a report or presentation.

Additionally, the quality of the data, feature engineering, and model training process will have a significant impact on the model's performance.