**FUTURE SALES PREDICTION**

**Building the IMDb score prediction model by:**

* **Feature Engineering**
* **Model Training**
* **Evaluation**

**Feature Engineering**

* Feature engineering is a crucial step in building a predictive model.
* It involves selecting and transforming the relevant features (variables) that will be used to train the model.
* In the case of IMDb score prediction, relevant features could include information about the movie, its cast, crew, and other metadata.
* **Genre Encoding:**

Convert movie genres into binary variables (one-hot encoding) to represent the presence or absence of each genre in a movie. For example, if a movie belongs to the “Action” genre, the corresponding feature will be 1; otherwise, it will be 0.

* **Director and Actor Influence:**

You can create features that capture the influence of the director and lead actors. This can be based on their past movie ratings, awards, or the number of movies they’ve been a part of.

* **Release Date:**

Extract features from the release date, such as the month or season when the movie was released, to account for any seasonality in movie ratings.

* **Movie Budget:**

If available, include the movie budget as a feature. High-budget films may have different rating dynamics.

* **User Reviews and Ratings:**

Aggregate and include information on the number of user reviews and ratings. This can be an indicator of a movie’s popularity.

* **Runtime:**

Consider the length of the movie as a feature. Longer or shorter movies may have different rating patterns.

* **Awards and Nominations:**

Create features based on the number of awards and nominations received by the movie.

**Model Training**

* Now that you have your feature-engineered dataset, you can proceed to train your IMDb score prediction model. You can choose from various regression algorithms to train the model. Here’s a simplified example using Python and scikit-learn:
* **Python Code**

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

From sklearn.linear\_model import LinearRegression

From sklearn.metrics import mean\_squared\_error

# Assuming X is your feature matrix and y is the IMDb scores

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

# Initialize and train the model

Model = LinearRegression()

Model.fit(X\_train, y\_train)

# Make predictions

Y\_pred = model.predict(X\_test)

# Evaluate the model

Mse = mean\_squared\_error(y\_test, y\_pred)

**Evaluation**

* To evaluate the model, you can use various metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2). Lower MSE, RMSE, and MAE values indicate better model performance, and a higher R2 value signifies a better fit.
* Additionally, you can create visualizations like scatter plots to compare actual IMDb scores with predicted scores to see how closely they align.
* Remember to split your dataset into training and testing sets to assess the model’s generalization performance. You can also consider using techniques like cross-validation to further assess the model’s robustness
* Iterate on the feature engineering and model training steps to improve your model’s performance until you achieve satisfactory results.

**Predicting IMDb scores is a regression problem, and you can use Python with popular libraries like scikit-learn and pandas to build a predictive model.**

**Steps involved:**

* **Data preparation**

Need a dataset with movie information, including features that can be used to predict IMDb scores. Ensure you have your dataset ready. Here, I’ll use a simplified example with random data:

**Python Code:**

Import pandas as pd

# Create a sample dataset (replace this with your dataset)

Data = {

‘Budget’: [10000000, 25000000, 3000000, 15000000, 50000000],

‘Genre’: [‘Action’, ‘Drama’, ‘Comedy’, ‘Action’, ‘Drama’],

‘Director\_Rating’: [8.1, 7.9, 6.5, 8.0, 7.7],

‘Actor\_Rating’: [7.9, 8.2, 7.0, 7.8, 7.6],

‘Runtime’: [120, 140, 90, 128, 115],

‘IMDb\_Score’: [8.5, 7.2, 6.6, 8.0, 7.9]

}

Df = pd.DataFrame(data)

* **Data processing**

Need to preprocess the data, which includes encoding categorical variables and splitting it into training and testing sets.

**Python Code**

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

From sklearn.preprocessing import LabelEncoder

# Encode categorical variable ‘Genre’

Label\_encoder = LabelEncoder()

Df[‘Genre’] = label\_encoder.fit\_transform(df[‘Genre’])

# Define features and target variable

X = df.drop(‘IMDb\_Score’, axis=1)

Y = df[‘IMDb\_Score’]

# Split the data into training and testing sets

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

* **Model selection and Training**

Using a simple Linear Regression model for prediction

**Python Code**

From sklearn.linear\_model import LinearRegression

From sklearn.metrics import mean\_squared\_error, r2\_score

# Initialize and train the model

Model = LinearRegression()

Model.fit(X\_train, y\_train)

# Make predictions

Y\_pred = model.predict(X\_test)

* **Model evaluation**

Need to evaluate the model’s performance using appropriate metrics. Common regression metrics include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2).

**Python Code**

# Evaluate the model

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)

* **Prediction**

Once you have a trained model , you can use it to predict IMDb scores for new data

**Python Code**

# Example prediction

New\_data = {

‘Budget’: [20000000],

‘Genre’: [‘Action’],

‘Director\_Rating’: [7.8],

‘Actor\_Rating’: [7.9],

‘Runtime’: [130]

}

# Encode categorical variable ‘Genre’ for new data

New\_data[‘Genre’] = label\_encoder.transform(new\_data[‘Genre’])

# Predict IMDb score for new data

New\_data\_df = pd.DataFrame(new\_data)

Predicted\_score = model.predict(new\_data\_df)

Print(“Predicted IMDb Score:”, predicted\_score[0])