**Building the IMDb score prediction model**

Sure, I can provide an outline for building an IMDb score prediction model by continuing with feature engineering, model training, and evaluation. This project assumes you're working with a dataset of movies, including various features, and you want to predict IMDb scores. Here's a step-by-step guide:

* Feature engineering
* Model training
* Evaluation.

Feature Engineering:

**Data Preprocessing:**

Handle missing data: Decide whether to impute or remove missing values in your dataset.

Data encoding: Convert categorical features to numerical using techniques like one-hot encoding or label encoding.

Feature scaling: Normalize or standardize numerical features to ensure they're on the same scale.

**Feature Selection:**

Analyze feature importance to select the most relevant features for prediction.

Consider using techniques like Recursive Feature Elimination (RFE), feature correlation analysis, or domain knowledge to choose the right features.

**Feature Transformation:**

Create new features that might be more informative.

For example,

you could calculate the director's average IMDb score or the movie's genre distribution.Use techniques like PCA for dimensionality reduction if you have many features.

**Model Training:**

**Split Data:**

Split your dataset into training, validation, and test sets. Common splits are 70% training, 15% validation, and 15% testing, but the ratio can vary depending on the dataset's size and characteristics.

**Select a Model:**

* Choose a machine learning model for regression. Common choices include linear regression, decision trees, random forests, gradient boosting, and neural networks.
* Consider trying different models to see which one performs the best.

**Train the Model:**

* Train the selected model using the training dataset.
* Tune hyperparameters using the validation dataset to improve the model's performance.

### **Evaluation:**

* Model Evaluation
* Test the Model
* Interpret Results
* Fine-tuning and Deployment

**Model Evaluation :**

* Evaluate the model's performance on the validation dataset using appropriate regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2).
* Adjust the model or hyperparameters as needed.

**Test the Model:**

Once satisfied with the model's performance on the validation dataset, test it on the test dataset to estimate its real-world performance.

**Interpret Results**

Interpret the model's predictions and assess the importance of each feature in the model.

Visualize the results, such as plotting actual IMDb scores against predicted scores.

**Fine-tuning and Deployment**

* If necessary, make final adjustments to the model based on test results and domain expertise.
* Once the model meets your requirements, prepare it for deployment in a production environment.

***Sample code for prediction:***

import pandas as pd

column\_names = ['user\_id', 'item\_id', 'rating', 'timestamp']

path = 'https://media.geeksforgeeks.org/wp-content/uploads/file.tsv'

df = pd.read\_csv(path, sep='\t', names=column\_names)

df.head()

| **user\_id** | **item\_id** | **rating** | **timestamp** |
| --- | --- | --- | --- |
| **0** | 0 | 50 | 5 | 881250949 |
| **1** | 0 | 172 | 5 | 881250949 |
| **2** | 0 | 133 | 1 | 881250949 |
| **3** | 196 | 242 | 3 | 881250949 |
| **4** | 186 | 302 | 3 | 891717742 |

data = pd.merge(df, movie\_titles, on='item\_id')

data.head()

|  | **user\_id** | **item\_id** | **rating** | **timestamp** | **title** |
| --- | --- | --- | --- | --- | --- |
| **0** | 0 | 50 | 5 | 881250949 | Star Wars (1977) |
| **1** | 290 | 50 | 5 | 880473582 | Star Wars (1977) |
| **2** | 79 | 50 | 4 | 891271545 | Star Wars (1977) |
| **3** | 2 | 50 | 5 | 888552084 | Star Wars (1977) |
| **4** | 8 | 50 | 5 | 879362124 | Star Wars (1977) |

import matplotlib.pyplot as plt

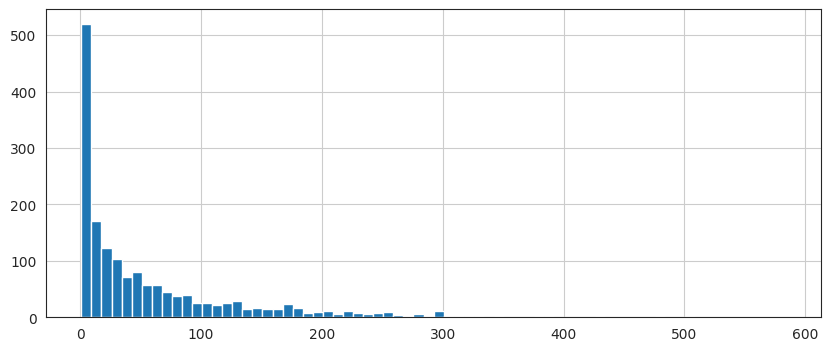
import seaborn as sns

sns.set\_style('white')

%matplotlib inline

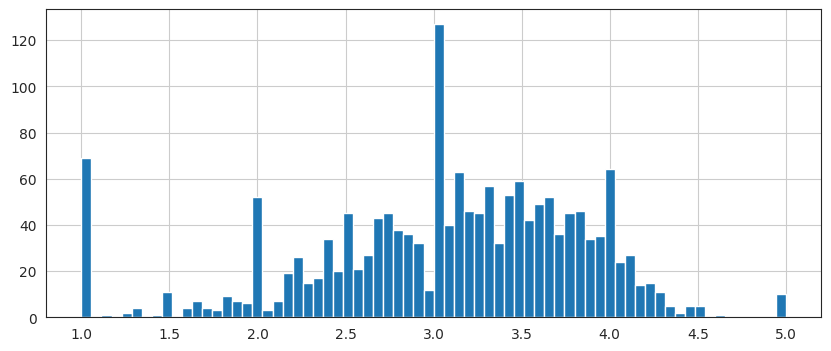
plt.figure(figsize =(10, 4))

ratings['num of ratings'].hist(bins = 70)

****

plt.figure(figsize =(10, 4))

ratings['rating'].hist(bins = 70)

****

moviemat = data.pivot\_table(index ='user\_id',

      columns ='title', values ='rating')

moviemat.head()

ratings.sort\_values('num of ratings', ascending = False).head(10)

|  | **rating** | **num of ratings** |
| --- | --- | --- |
| **title** |  |  |
| **Star Wars (1977)** | 4.359589 | 584 |
| **Contact (1997)** | 3.803536 | 509 |
| **Fargo (1996)** | 4.155512 | 508 |
| **Return of the Jedi (1983)** | 4.007890 | 507 |
| **Liar Liar (1997)** | 3.156701 | 485 |
| **English Patient, The (1996)** | 3.656965 | 481 |
| **Scream (1996)** | 3.441423 | 478 |
| **Toy Story (1995)** | 3.878319 | 452 |
| **Air Force One (1997)** | 3.631090 | 431 |
| **Independence Day (ID4) (1996)** | 3.438228 | 429 |

**Conclusion:**

Remember to iterate through these steps, experimenting with different features, models, and hyperparameters as needed to achieve the best IMDb score prediction performance. Additionally, it's essential to keep track of the model's performance over time and update it as new data becomes available or the model's accuracy changes.