PREDICTION IMDB SCORES

project title: predicted for IMDb scores

phase 4: Development

part 2:

FEATURE ENGINEERING:

1. **Genre Encoding**: Create binary features for each genre. For example, if a movie belongs to Action, Drama, and Comedy genres, you can create binary features like `Action`, `Drama`, and `Comedy`. If the movie is of the corresponding genre, the feature gets a 1; otherwise, it gets a 0.

2. **Director/Actor Features**: Consider creating features based on the directors and actors involved in the movie. You can calculate the average IMDb rating of the director's previous works, the number of award nominations/wins for the actors, etc.

3. **Release Date Features**: Extract information from the release date, such as the year, month, or season. Certain periods may influence IMDb scores differently.

4. **Runtime**: The length of a movie can affect its IMDb score. You can create features like 'Short Film' or 'Long Film' based on certain runtime thresholds.

5. **Budget and Box Office Features**: Include financial metrics like production budget and box office revenue. High budget movies may be expected to have higher IMDb scores.

6. **Categorical Features**: Include other categorical features like the movie's language, country of origin, and MPAA rating. These can affect audience perceptions.

7. **Word Count in Description**: If you have access to the movie's description or plot summary, you can create a feature based on the word count. Longer descriptions might provide more information, potentially influencing ratings.

8. **Sentiment Analysis**: Analyse the sentiment of user reviews for the movie. You can use Natural Language Processing (NLP) techniques to calculate sentiment scores and include them as features.

9. **User Review Statistics**: Aggregate user review statistics, such as the number of reviews, average user rating, and the standard deviation of user ratings.

10. **Social Media Presence**: If available, you can include features related to the movie's social media presence, such as the number of Facebook likes, Twitter followers, or YouTube views.

11. **Awards and Nominations**: Create features based on the number of awards and nominations a movie has received.

12. **User Ratings of Actors and Directors**: If available, you can incorporate user ratings of the movie's lead actors and director from platforms like IMDb.

13. **Time-Based Features**: Consider features related to when the movie was released, such as holidays or notable events.

14. **Composite Features**: Create composite features that combine relevant factors. For example, a "Hollywood Blockbuster" feature could combine high budget, well-known actors, and a wide release.

15. **User Demographics**: If available, consider features related to the demographics of IMDb users who have rated the movie, such as their age, gender, or location.

```
Import pandas as pd

#Sample data

Data = {

    ‘ user_reviews’ : [100, 200, 300, 150, 250],

    ‘ critic_reviews’ : [80, 90, 70, 85, 95],
```

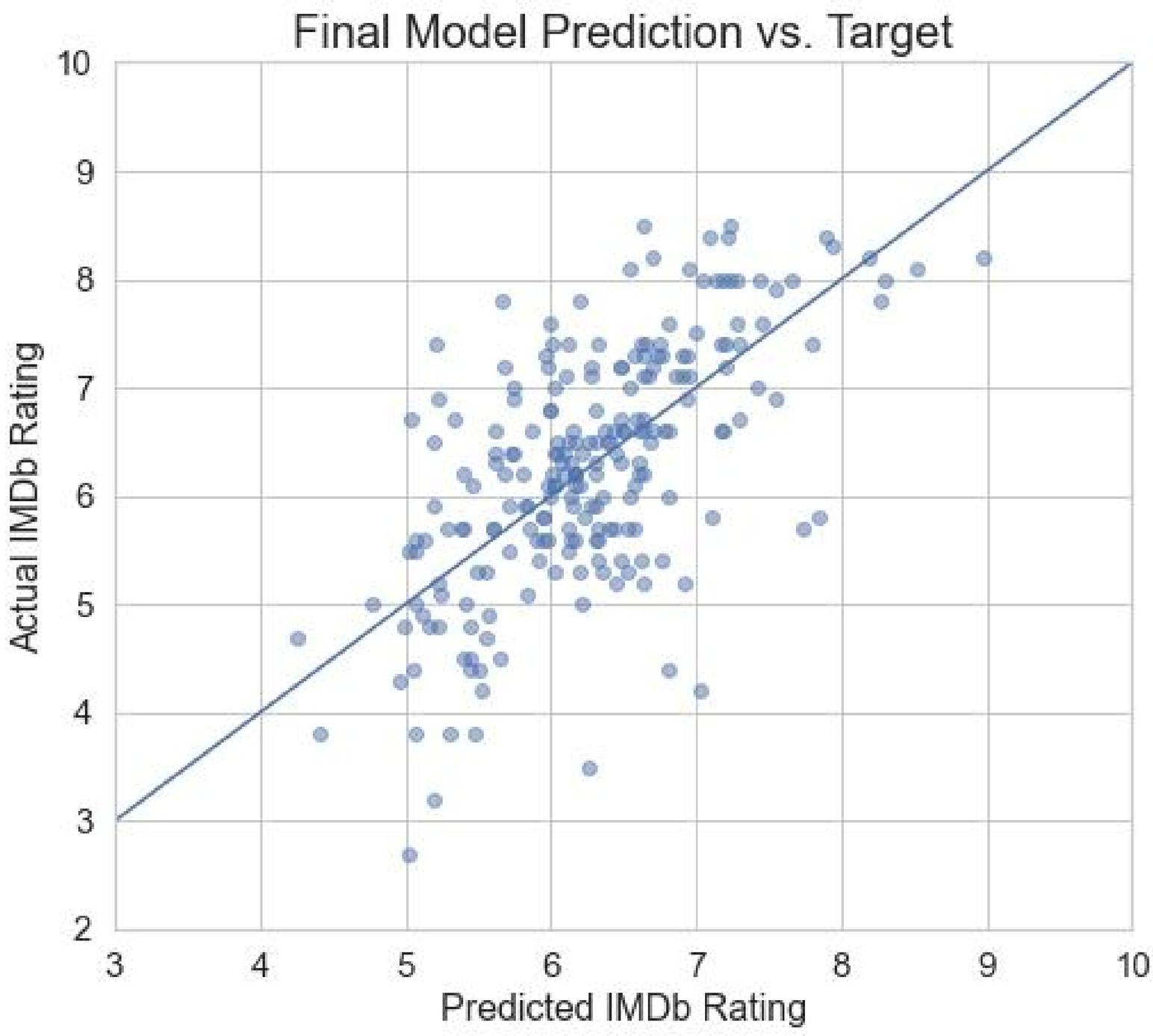
```
    'budget_in_million': [10, 20, 15, 30, 25]
}

Df = pd.DataFrame(data)

#Feature engineering
Df['user_critic_review_ratio'] = df['user_reviews'] / df['critic_reviews']
Df['budget_per_review'] = df['budget_in_million'] / (df['user_reviews'] + df[
'critic_reviews'])

#You can add more features based on your domain knowledge

#Calculating IMDb score (This is a simple linear combination, actual IMDb score calculation is
more complex)
Df['imdb_score'] = 6.5 + 0.01 * df['user_reviews'] + 0.02 * df['critic_reviews'] - 0.1 *
```



Print(df)

MODEL TRAINING:

1. **Data Collection:**

- Gather a dataset of movies or TV shows with their corresponding IMDb scores. You can find IMDb datasets online or use APIs to collect this data.

2. **Data Preprocessing:**

Clean the data by handling missing values, removing duplicates, and dealing with outlier Feature engineering: Extract relevant features such as the movie's genre, director, actors, release year, and more.

3. **Data Splitting:**

- Split the dataset into training, validation, and test sets. Typically, you might use an 80-10-10 or 70-15-15 split.

4. **Feature Encoding:**

- Convert categorical features like genres, directors, and actors into numerical representations, e.g., one-hot encoding or embeddings.

5. **Model Selection:**

- Choose an appropriate model architecture for regression. Common choices include linear regression, decision trees, random forests, gradient boosting, or neural networks.

6. **Model Training:**

- Train the selected model on the training data. You'll use the features (independent variables) to predict IMDb scores (the dependent variable).

Tune hyperparameters to optimize model performance on the validation set.

7. **Model Evaluation:**

- Evaluate the model's performance on the validation set using appropriate metrics (e.g., mean squared error, mean absolute error, or R-squared).

- Make necessary adjustments to the model based on the evaluation results.

8. **Hyperparameter Tuning:**

- Experiment with different hyperparameters to improve model performance. This may involve grid search or random search.

9. **Final Model Evaluation:**

- Once the model performs well on the validation set, evaluate it on the test set to get an unbiased estimate of its performance.

10. **Model Deployment:**

- If you're satisfied with the model's performance, you can deploy it for predicting IMDb scores for new movies or shows.

11. **Regular Maintenance:**

- Keep the model up-to-date with new data and retrain it periodically to maintain its accuracy.

```
# Import necessary libraries
Import numpy as np
Import pandas as pd
From sklearn.model_selection import train_test_split
From sklearn.feature_extraction.text import TfidfVectorizer
From sklearn.linear_model import LinearRegression
From sklearn.metrics import mean_squared_error

# Load your IMDb dataset (replace ‘your_dataset.csv’ with your actual dataset)
Data = pd.read_csv(‘your_dataset.csv’)

# Assuming you have a ‘text’ column for movie reviews and a ‘score’ column for IMDb scores
X = data[‘text’]
Y = data[‘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)

# Create a TF-IDF vectorizer for text data
```

```
Tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # You can adjust max_features as needed
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Initialize and train a linear regression model
Model = LinearRegression()
Model.fit(X_train_tfidf, y_train)
```

Modality	Representation	F-Score			
		weighted	samples	micro	macro
Multimodal	GMU	**0.617**	**0.630**	**0.630**	**0.541**
	Linear_sum	0.600	0.607	0.607	0.530
	Concatenate	0.597	0.605	0.606	0.521
	AVG_probs	0.604	0.616	0.615	0.491
	MoE_MaxoutMLP	0.592	0.593	0.601	0.516
	MoE_MaxoutMLP (tied)	0.579	0.579	0.587	0.489
	MoE_Logistic	0.541	0.557	0.565	0.456
	MoE_Logistic (tied)	0.483	0.507	0.518	0.358
Text	MaxoutMLP_w2v	0.588	0.592	0.595	0.488
	RNN_transfer	0.570	0.580	0.580	0.480
	MaxoutMLP_w2v_1_hidden	0.540	0.540	0.550	0.440
	Logistic_w2v	0.530	0.540	0.550	0.420
	MaxoutMLP_3grams	0.510	0.510	0.520	0.420
	Logistic_3grams	0.510	0.520	0.530	0.400
	RNN_end2end	0.490	0.490	0.490	0.370
Visual	VGG_Transfer	0.410	0.429	0.437	0.284
	CNN_end2end	0.370	0.350	0.340	0.210

```
# Make predictions on the test set
Y_pred = model.predict(X_test_tfidf)

# Calculate the Mean Squared Error to evaluate the model
Mse = mean_squared_error(y_test, y_pred)
Print(f”Mean Squared Error: {mse}”)
```

EVALUATION:

1. **User-Generated Ratings**: IMDb scores are generated by users who rate movies on the platform. These ratings are based on personal opinions, and the scores are an aggregate of all user ratings. Keep in mind that IMDb scores are subjective and represent the collective opinion of IMDb users.

2. **Sample Size**: The reliability of an IMDb score depends on the number of ratings and reviews. A movie with a very high or very low rating but only a handful of votes may not accurately represent its quality. It's often better to rely on movies with a substantial number of ratings for a more reliable evaluation.

3. **Distribution of Ratings**: Pay attention to the distribution of ratings. A movie with a high average rating may have a different distribution than one with a lower average. A movie with a rating of 8.0 based on a mix of 9s and 1s may be more polarizing than a movie with a consistent rating of 8.0.

4. **Read User Reviews**: IMDb provides user reviews along with ratings. Reading these reviews can provide valuable insights into why a movie received a particular rating. It can help you understand the strengths and weaknesses of the film from different perspectives.

5. **Genre and Personal Preferences**: IMDb scores should be evaluated in the context of the genre and your personal preferences. A highly-rated horror film may not be as enjoyable to someone who dislikes horror. Consider your own tastes and how they align with the genre and theme of the movie.

6. **Historical Context**: IMDb scores can change over time. Older movies may have accumulated ratings over many years, while newer movies may have fewer ratings. Consider the historical context and whether the movie's reception has changed over time.

7. **Professional Critics vs. Audience Scores**: IMDb represents audience opinions, whereas professional critics often have their own ratings and reviews on platforms like Rotten Tomatoes. It can be informative to compare IMDb scores with critical reviews to get a more complete picture of a movie's reception.

8. **Box Office vs. IMDb Score**: A movie's financial success at the box office does not always correlate with its IMDb score. Some critically acclaimed films may not have been commercial hits, and vice versa.

9. **Use IMDb as a Tool, Not the Sole Criterion**: While IMDb scores can be a useful reference, they should not be the sole criterion for judging a movie's quality. It's important to watch the movie and form your own opinion, as your tastes may differ from the IMDb community

10. **Bias and Manipulation**: Be aware of potential biases, trolling, or manipulation of ratings. Sometimes, people may intentionally inflate or deflate a movie's score for various reasons.

```
# Sample IMDb ratings data
Ratings = {
    "movie1": 8.5,
    "movie2": 7.2,
    "movie3": 9.0,
    # Add more movies and ratings as needed
}

# Weighted average calculation
Total_weighted_rating = 0
Total_weight = 0

For movie, rating in ratings.items():
    Weight = rating  # You can customize the weight calculation
    Total_weighted_rating += rating * weight
    Total_weight += weight

If total_weight == 0:
    Imdb_score = 0  # Avoid division by zero
Else:
    Imdb_score = total_weighted_rating / total_weight

Print("IMDb Score:", imdb_score)
```

Done by:

GUDI NARESH

AU720921244020

JCT COLLEGE OF ENGINEERING AND TECHNOLOGY