**PREDICTING IMDb SCORES USING MACHINE LEARNING**

**TEAM MEMBER:** SRI LAKSHMI T

Phase 4 submission document

Project Title: IMDb Scores Prediction

Phase 4: Development Part 2

**Topic** : In this part you will continue building your project.

Continue building the IMDb score prediction model by:

* Feature engineering
* Model training
* Evaluation.

**Introduction:**

* Predicting IMDb scores for movies or TV shows typically involves using machine learning models and features such as cast, crew, genre, user reviews, and more.
* You can use regression algorithms to build a predictive model.
* The quality of your predictions depends on the quality and

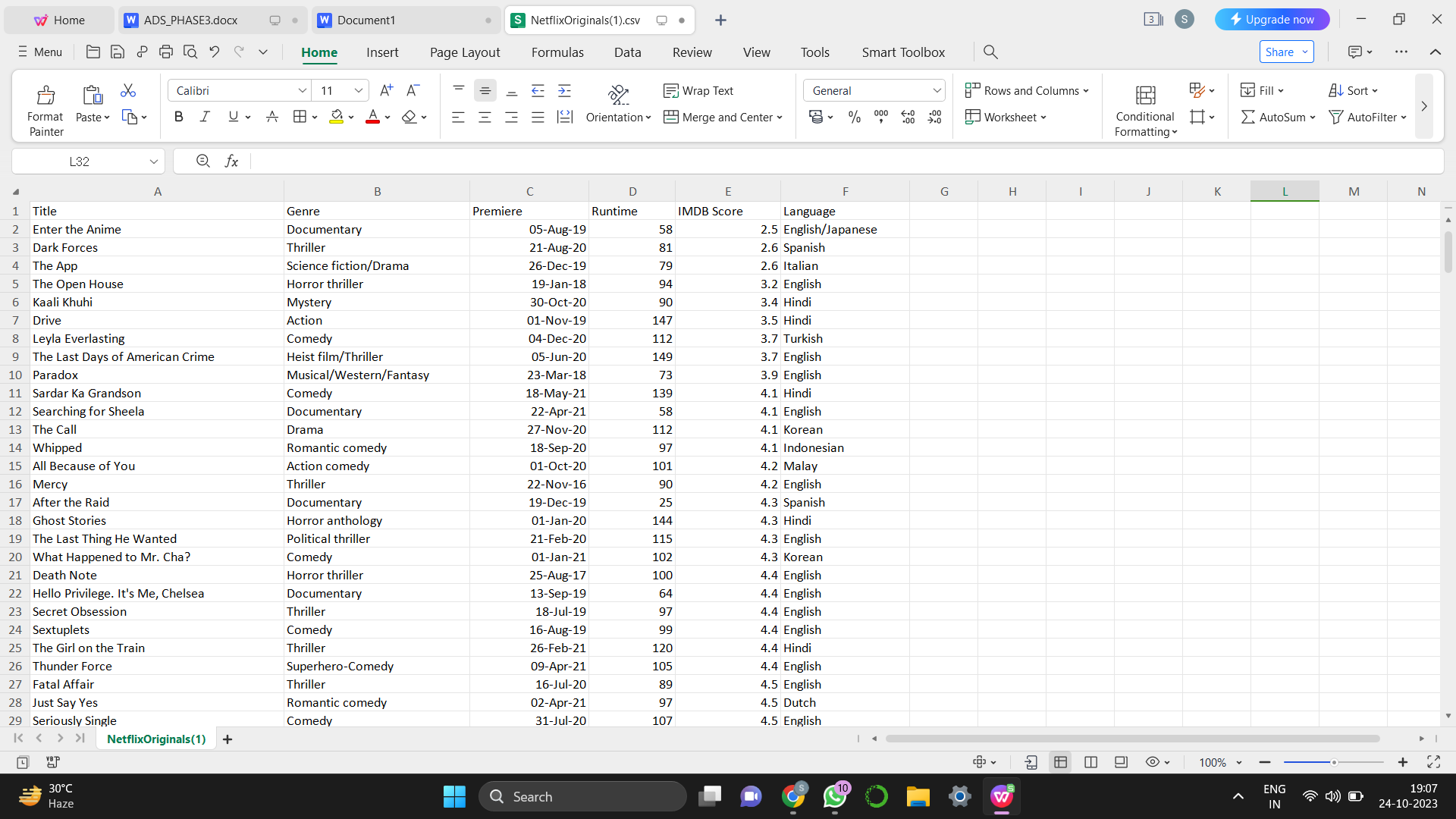
quantity of data, as well as the choice of features and model.



**Data Source :** A Good Data for Predicting IMDb Scores using machine

learning model should be Accurate , complete , accessible

**Dataset Link :** (https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores)



**Feature Engineering :**

Feature engineering is a process in machine learning and data analysis that involves creating new, derived features from the existing data to improve the performance of a predictive model. It involves transforming and enhancing the raw data by extracting useful information or creating new representations that can capture the underlying patterns and relationships.

**Here are some common techniques used in feature engineering:**

1. Encoding Categorical Variables: Convert categorical variables into numerical representations that machine learning models can understand. This can be done using techniques like one-hot encoding, ordinal encoding, or label encoding.

2. Handling Missing Values: Deal with missing data by imputing values based on strategies like mean, median, mode, or using more complex methods like k-nearest neighbors or predictive models.

3. Scaling and Normalization: Scale numerical features to bring them to a similar range or distribute them normally. Common techniques include standardization (subtracting mean and dividing by standard deviation) or min-max scaling (scaling between a specified range).

4. Binning: Group continuous numeric features into bins or intervals to convert them into categorical variables. This can help capture non-linear relationships or handle outliers.

5. Feature Interaction: Create new features by combining or interacting existing features. For example, combining height and weight to calculate BMI or creating interactions between different categorical variables.

6. Time-based Features: Extract relevant information from date or time variables, such as day of the week, month, or time duration since a particular event.

7. Textual Features: Process and extract features from text data, such as word counts, TF-IDF vectors, n-grams, or word embeddings. These transformations can be useful for natural language processing tasks.

8. Feature Selection: Use techniques like correlation analysis, mutual information, or regularization methods to select the most informative features and eliminate redundant or irrelevant ones. This helps reduce overfitting and improve model performance.

9. Domain-Specific Features: Incorporate domain knowledge by creating features that capture specific characteristics or patterns related to the problem at hand. This often requires expertise in the particular application area.

10. Feature Extraction from Images or Audio: If the data involves images or audio, techniques like convolutional neural networks (CNNs) or Mel-frequency cepstral coefficients (MFCC) can be used to extract meaningful features.

It's important to note that feature engineering is an iterative process and requires experimentation to find the most effective features for a given problem. Additionally, it is essential to balance the complexity of features with the size and quality of the dataset to avoid overfitting or introducing unnecessary noise.

**Procedure :**

To build an IMDb score prediction model, feature engineering plays a crucial role in extracting relevant information from the available data. Here's a general outline of the feature engineering process:

1. Data Preparation: Start by collecting and cleaning the IMDb dataset, removing any duplicates or missing values, and handling outliers appropriately.

2. Textual Features:

   - Title: Extract information from the movie title, such as the number of words or characters, presence of certain keywords (e.g., "adventure," "romance").

   - Plot Summary: Analyze the plot summary using techniques like tokenization, stop-word removal, and stemming/lemmatization. Generate features like bag-of-words, TF-IDF, or word embeddings to capture the essence of the movie's storyline.

   - Genre: Convert categorical genres into binary features (one-hot encoding).

3. Numeric Features:

   - Runtime: Transform the movie's runtime into a numerical feature by converting it into minutes or seconds.

   - Production Year: Extract the production year from the release date and convert it into a numerical feature.

   - Budget and Revenue: Utilize financial information such as budget and revenue (if available) as numeric features.

4. Cast and Crew:

   - Actors and Directors: Consider the involvement of popular actors and directors as binary features, indicating whether they are associated with the movie.

   - Awards and Nominations: Extract information about awards and nominations received by the cast and crew as numerical features.

5. External Data:

Incorporate external data sources like sentiment analysis on reviews or social media mentions to capture the public sentiment around the movie.

1. User Interaction:

Leverage existing user ratings and reviews to create features like average user rating, counts, or sentiment analysis of reviews.

1. Scaling and Normalization:

Scale and normalize the features to ensure they're on a similar scale.

1. Feature Selection:

Utilize techniques like correlation analysis, mutual information, or feature importance scores to select the most relevant features for the prediction model.

1. Model Building:

Choose a machine learning algorithm suitable for regression tasks, such as linear regression, decision trees, random forests, or gradient boosting. Train and fine-tune the model using appropriate evaluation metrics (e.g., mean squared error) and cross-validation techniques.

Remember, feature engineering is an iterative process that requires experimentation with different techniques and domain knowledge to create the most informative features for the IMDb score prediction model.

**Model training :**

Model training is the process of teaching a machine learning model to learn patterns and relationships from the available data. Here's a general outline of the steps involved in model training:

1. Data Preparation: Prepare your dataset for training by splitting it into two sets: a training set and a validation/test set. The training set is used to teach the model, while the validation/test set is used to evaluate its performance.

2. Feature Selection/Extraction: Select the relevant features that will be used as inputs to the model. This may involve applying feature engineering techniques as discussed earlier to extract informative representations from the data.

3. Model Selection: Choose an appropriate machine learning algorithm or model architecture based on the nature of your problem. This could be a linear regression model, decision tree, support vector machine, neural network, or any other algorithm suitable for your specific task.

4. Model Initialization: Initialize the model with random or predefined parameters. This step is crucial for models that require initial weights or biases.

5. Training Loop: Iterate over the training data multiple times and update the model's parameters to minimize the difference between predicted outputs and true labels. This process is known as optimization or learning. The most commonly used optimization algorithms are gradient descent and its variations.

6. Loss Function: Define a loss function that quantifies the discrepancy between predicted outputs and true labels. The goal is to minimize this loss during the training process. The choice of loss function depends on the nature of the problem (e.g., mean squared error for regression, cross-entropy for classification).

7. Backpropagation: For models with trainable parameters (e.g., neural networks), perform backpropagation to compute the gradients of the loss with respect to the model's parameters. This allows for updating the parameters in the direction that reduces the loss.

8. Hyperparameter Tuning: Adjust the hyperparameters of the model, such as learning rate, regularization strength, batch size, and number of hidden layers, to optimize performance. This can be done through techniques like grid search, random search, or more advanced methods like Bayesian optimization.

9. Validation: Periodically evaluate the model's performance on the validation/test set during training to monitor its progress and detect any overfitting or underfitting issues. Adjust the model or hyperparameters accordingly.

10. Final Evaluation: Once the model training is complete, assess its performance on an independent test set to get an unbiased estimate of its generalization ability. This step helps determine how well the model will perform on unseen data.

11. Model Deployment: After the model has been trained and tested successfully, it can be deployed to make predictions on new and unseen data.

Remember that the model training process often requires iterations and experimentation to achieve the best performance.

**Procedure :**

To build an IMDb score prediction model, you can follow these steps for model training:

1. Data Preparation: Preprocess the IMDb dataset, handle missing values, convert categorical variables to numerical representations (one-hot encoding, label encoding), and split the data into training and test sets.

2. Feature Engineering: Create relevant features from the available data as discussed earlier. This step helps in capturing important information for predicting IMDb scores.

3. Model Selection: Choose a suitable machine learning algorithm for regression, such as linear regression, decision trees, random forests, support vector machines, or gradient boosting models. Consider the characteristics of your dataset and the assumptions of each algorithm to make an informed choice.

4. Model Training: Fit your chosen algorithm to the training data using the features and IMDb scores. This process involves estimating the model's parameters and finding the best fit to the training data.

5. Optimization and Hyperparameter Tuning: Optimize your model by fine-tuning its hyperparameters. This process involves adjusting parameters like learning rate, regularization strength, depth of trees, or the number of hidden layers to improve performance on the validation set. Techniques like grid search, random search, or Bayesian optimization can be used for hyperparameter tuning.

6. Model Evaluation: Evaluate your trained model using the test set. Calculate evaluation metrics appropriate for regression tasks, such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared (coefficient of determination). Assessing these metrics helps understand how well your model performs in predicting IMDb scores.

7. Model Refinement: Analyze the model's performance and iteratively refine it by repeating steps 4 to 6. This involves adjusting the feature engineering process, trying different algorithms, or re-tuning hyperparameters to improve prediction accuracy.

8. Final Model Deployment: Once you're satisfied with the model's performance, deploy it to make predictions on new, unseen data. Be mindful of any necessary preprocessing steps needed for the input data to match the format used during training.

It's essential to note that the process described above is a general guideline, and the specific steps may vary based on the chosen algorithm, dataset characteristics, and domain knowledge. Experimentation, fine-tuning, and careful evaluation are the key to building an accurate IMDb score prediction model

**Evaluation :**

Evaluation is a crucial step in the machine learning workflow to assess the performance and effectiveness of a trained model. It helps determine how well the model generalizes to new, unseen data and provides insights into its strengths and weaknesses. Here are some common evaluation techniques:

1. Train-Test Split: Divide the available dataset into two subsets, the training set and the test set. Train the model on the training set, and then evaluate its performance on the test set. This helps estimate how well the model will perform on unseen data.

2. Cross-Validation: Split the dataset into multiple subsets (folds). Train the model on a combination of these subsets and evaluate it on the remaining fold. This process is repeated several times, with each fold serving as a test set once, and then average the evaluation results across all folds. Cross-validation provides a more robust estimate of the model's performance compared to a single train-test split.

3. Evaluation Metrics: Choose appropriate evaluation metrics based on the problem type. For classification tasks, metrics like accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) are commonly used. For regression tasks, metrics like mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared (coefficient of determination) are typically used. The choice of metric depends on the specific problem requirements and can help measure the model's performance accurately.

4. Confusion Matrix: For classification tasks, a confusion matrix provides a detailed breakdown of model predictions and actual labels for each class. It helps understand the model's performance in terms of true positives, true negatives, false positives, and false negatives. From the confusion matrix, additional metrics such as precision, recall, and F1-score can be derived.

5. ROC Curve and Precision-Recall Curve: These curves provide a graphical representation of the performance of a binary classifier across different probability thresholds. The ROC curve shows the trade-off between sensitivity (recall) and specificity (1 - false positive rate), while the precision-recall curve illustrates the trade-off between precision and recall.

6. Bias and Fairness Analysis: Evaluate the model's fairness and bias by examining its performance across different subgroups (e.g., demographic groups). This helps identify potential disparities and biases in predictions, which is important for ensuring ethical and fair deployment of the model.

7. Error Analysis: Analyze specific instances where the model makes mistakes or produces unexpected results. This helps gain insights into areas where the model may be underperforming or where the training data may contain discrepancies or biases.

Remember that evaluation is an iterative process, and it may involve refining the model, adjusting hyperparameters, or exploring different feature engineering techniques based on the evaluation results. Additionally, it's important to carefully consider the limitations and scope of the evaluation metrics chosen to ensure they align with the specific problem and goals of the model.

**Evaluation in machine learning :**

In machine learning, evaluation refers to the process of assessing the performance and quality of a trained model. It involves measuring how well the model can generalize its predictions to unseen data. The evaluation process helps to determine if the model is effectively learning patterns and making accurate predictions.

**Here are some common evaluation techniques used in machine learning:**

1. Training and Test Sets: The most basic evaluation technique involves splitting the available dataset into two sets: a training set and a test set. The model is trained on the training set and then evaluated on the test set. The performance on the test set provides an estimate of how well the model generalizes to unseen data. This approach helps identify if the model is overfitting (performing well on training data but not on test data) or underfitting (performing poorly on both training and test data).

2. Cross-Validation: Cross-validation is a more advanced evaluation technique that helps overcome limitations in the train-test split method, especially when the dataset is small. It involves dividing the data into multiple subsets or "folds." The model is trained on a combination of these folds (usually k-1 folds) and evaluated on the remaining fold. This process is repeated k times, with each fold serving as the test set once. The performance results are then averaged to provide a more reliable estimate of the model's performance.

3. Evaluation Metrics: Different evaluation metrics are used based on the type of machine learning task. For classification tasks, metrics like accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC) are commonly used. For regression tasks, metrics like mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared are often used. The choice of evaluation metrics depends on the specific problem and the desired performance criteria.

4. Validation Set: In addition to the test set, a separate validation set can be used during the model training process to evaluate the model's performance at different stages. This helps in monitoring and fine-tuning the model to prevent overfitting.

5. Noisy Data Analysis: Evaluation also involves analyzing the model's performance on noisy or mislabeled data. This helps in understanding the model's robustness and potential limitations in real-world scenarios.

It's important to remember that the evaluation process should be performed on data that the model has not seen during training. This ensures a fair assessment of the model's performance on unseen data and its generalization ability. Additionally, evaluation results should be interpreted with domain knowledge and context in mind to make meaningful conclusions about the model's effectiveness and reliability.

**Procedure :**

To build an IMDb score prediction model using evaluation, you can follow these steps:

1. Data Preparation: Preprocess the IMDb dataset by handling missing values, converting categorical variables into numerical representations (such as one-hot encoding or label encoding), and splitting the data into training and test sets. Make sure to shuffle the data to remove any inherent ordering.

2. Feature Selection/Engineering: Select relevant features from the dataset that can help in predicting IMDb scores. This can include attributes such as movie genre, director, cast, runtime, release year, etc. Perform any necessary feature engineering steps like scaling or normalization to ensure consistency in the data.

3. Model Selection: Choose an appropriate model for IMDb score prediction. This can include regression algorithms like linear regression, decision trees, random forests, support vector machines (SVM), or gradient boosting models. Consider the characteristics of your dataset and the requirements of your problem when selecting a model.

4. Model Training: Train the selected model on the training dataset using the chosen features and IMDb scores as the target variable. Adjust the model's hyperparameters as necessary based on experimentation or cross-validation results. Fit the model to the data to estimate its parameters and optimize its performance.

5. Model Evaluation: Evaluate the trained model using the test dataset. Calculate evaluation metrics appropriate for regression tasks, such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or R-squared (coefficient of determination). Assess how well the model predicts IMDb scores and compare its performance against baseline or benchmark models.

6. Hyperparameter Tuning: Fine-tune the model's hyperparameters to improve its performance. This can involve techniques like grid search, random search, or Bayesian optimization. Iterate on different combinations of hyperparameters to find the best configuration that maximizes the evaluation metrics.

7. Model Refinement: Analyze the model's performance and potential shortcomings. Assess any biases, errors, or areas of improvement. Iterate on the feature engineering process, try different algorithms, adjust hyperparameters, or consider additional data sources if necessary. This iterative refinement process helps improve the model's prediction capabilities.

8. Final Model Deployment: Once you are satisfied with the model's performance, deploy it to make predictions on new, unseen data. Use the entire dataset (including the previously held-out test set, if desired) to train the final model before deployment, to utilize all available data for training.

Remember that evaluation is an ongoing process, and it's important to continuously monitor and assess the model's performance as new data becomes available. This helps ensure that the model remains accurate and effective over time.