Predicting Amazon Movie Review Star Ratings

## Introduction

This project aims to predict Amazon movie review star ratings based on review text and metadata. The assignment focuses on applying traditional machine learning methods (excluding deep learning or neural networks) to classify the star ratings with high accuracy. Following the instructions, the project uses offline evaluation methods to assess models and leverages feature engineering techniques to derive meaningful insights from the available data. This report describes the data preprocessing, feature engineering, model selection, and optimization techniques used, as well as any specific “special tricks” and assumptions made throughout the process.

## Data Preprocessing

To handle the large volume of text data efficiently and extract useful features, we performed several preprocessing steps:

- **Handling Missing Values:** We filled missing values in the Text and Summary fields with empty strings to avoid data loss. We also removed any rows with missing values in the Score column, as these would make training and evaluation problematic.
- **Feature Engineering:**
    - Review and Summary Length: As we noticed that review length could correlate with rating tendencies (e.g., longer reviews might indicate strong opinions), we

    calculated the character length of each review (review_length) and summary (summary_length).
    - **Helpfulness Ratio:** The dataset includes helpfulness scores, but we chose to standardize this by creating a ratio of HelpfulnessNumerator to HelpfulnessDenominator. This feature reflects perceived helpfulness, which could relate to review credibility.
- **TF-IDF Transformation:** We used TF-IDF (Term Frequency-Inverse Document Frequency) to convert text data into numerical format, capturing the importance of words relative to their usage across the dataset. A maximum of 5000 features was set to balance computational efficiency and model performance. Due to memory constraints, we processed text data in chunks.

**Special Trick #1:** Chunked Processing for TF-IDF Transformation Given the large dataset, using a standard TF-IDF transformation would cause memory overload. Instead, we processed the data in chunks. By breaking the data into manageable parts, we fit and transformed the text data iteratively, which allowed us to complete this step within Colab's memory limits.

## Model Selection and Offline Evaluation

After preprocessing, we focused on two machine learning algorithms covered in class: Random Forest and Support Vector Machine (SVM). Here's a breakdown of our approach:

**Random Forest Classifier:**

Random Forest was chosen for its robustness with large feature spaces and interpretability. We used 100 estimators, as this value provided a reasonable balance between accuracy and computation time. This model was trained and evaluated using offline validation to assess its performance before final submission.

**Offline Evaluation:** The dataset was split into 80% training and 20% validation sets. This offline evaluation allowed us to assess model performance without overfitting to the test data.

**Support Vector Machine (SVM):**

SVM was used with a linear kernel, as it is efficient for high-dimensional text data. Despite being computationally demanding, SVM often performs well with text features such as TF-IDF, making it a good choice for this assignment.

Special Trick #2: Row Alignment in Feature Engineering During data preprocessing, we realized that row mismatches could occur between text_features and non_text_features after filtering out rows with missing scores. We applied a filtering technique to align the rows in both datasets accurately, ensuring consistent dimensions during model training.

## Observations and Assumptions

Several key assumptions and observations guided the modeling choices and preprocessing steps:

**Assumption on Review Length:** We assumed that longer reviews likely indicate a more engaged reviewer. This led us to include review_length and summary_length as engineered features.

**Helpfulness Ratio Assumption:** By converting helpfulness scores to a ratio, we aimed to standardize this feature, assuming that more helpful reviews might correlate with more reliable star ratings.

**Offline Model Evaluation Assumption:** Given the assignment's focus on offline evaluation, we refrained from making predictions directly on the test set until we were confident in the model's performance on the validation set.

## Results and Future Work

After evaluating both models, Random Forest showed a balance between accuracy and computation time, making it the preferred model for our final submission. Key metrics such as accuracy, precision, recall, and F1-score were calculated, showing that Random Forest performed well on the validation set, especially compared to SVM.

**Special Trick #3:** Optimizing with Model Selection Based on Efficiency Due to the computational demands of SVM, we chose Random Forest as the final model, which provided competitive accuracy with faster training and inference times. This allowed us to focus on further feature engineering rather than hyperparameter tuning alone.

**Future Work:** Potential improvements could include experimenting with more complex text features, such as n-grams, or exploring alternative feature representations like word embeddings. Additionally, more advanced ensemble methods and hyperparameter optimization could further enhance performance.

## Conclusion

This project demonstrated effective methods for predictive modeling with large text datasets, focusing on TF-IDF transformations, offline model evaluation, and practical feature engineering. By aligning engineered features with the text data and optimizing for efficiency, we developed a reproducible and interpretable model suitable for the competition. This approach not only met the assignment's requirements but also provided insights into handling large-scale text classification tasks without deep learning.

## References

Scikit-Learn Documentation. (n.d.). Retrieved from https://scikit-learn.org/stable/

Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Kaggle. (n.d.). Amazon Movie Reviews Dataset. Retrieved from https://www.kaggle.com/