Writeup: Product Review Scores

1. Introduction and Problem Statement

The objective of this assignment is to classify product review scores based on various features derived from Amazon product reviews. Given a dataset of reviews with attributes such as ProductId, UserId, Helpfulness, Summary, and Time, the goal is to predict the Score that each review would receive. This project required creative feature engineering, model evaluation, and intelligent selection of methods, without the use of deep learning techniques.
In addition to achieving good accuracy, the focus was on exploring and deriving meaningful features to gain insight into the data and improve the model's performance and accuracy. This writeup presents the approach taken to preprocess the data, select and engineer features, train models, and evaluate their performance.

2. Data Exploration and Initial Observations

To better understand the dataset, I conducted an initial exploratory data analysis (EDA), which revealed the following:

- There was a noticeable imbalance in review scores, with certain scores (e.g., 4.0) being much more prevalent than others.
- The Text and Summary fields were observed to vary significantly in length, potentially indicating different levels of review detail, which can impact the score.
- HelpfulnessNumerator and HelpfulnessDenominator provided information about the perceived helpfulness of reviews, which could be transformed into a normalized feature for the model.

There were several challenges identified through EDA:

- Some fields had missing values that needed to be addressed, either by imputation or by removal.
- Text data requires transformation, such as extracting sentiment or summarizing word counts, to make it useful for modeling and predicting.
- With a higher prevalence of certain review scores, balancing the data or choosing suitable evaluation metrics was essential.

3. Feature Engineering and Data Processing

A major focus of this project was to engineer relevant features from the raw data, as these new features could provide additional context to the model. The following features were created based on the observations:

- Helpfulness Ratio: Calculated as the ratio of HelpfulnessNumerator to HelpfulnessDenominator, capturing the proportion of users who found a review helpful. Missing values were replaced with 0 to ensure consistency.
- Text Length Features:
    - Text_Length: The character length of the Text field.
    - Word_Count: The number of words in the Text field.
    - Avg_Word_Length: Average word length in Text, providing insight into the complexity of language used.

- Sentiment Scores: Using TextBlob, polarity scores were computed for both Text and Summary to capture the sentiment. Although installing TextBlob presented challenges, these scores added meaningful sentiment information about each review.
- Timestamp Features: Extracted Review_Year and Review_Month from Time to represent temporal trends, which might reflect seasonal patterns in review scores.
- User and Product Mean Scores: Calculated average scores for each ProductId and UserId to capture potential reviewer or product bias.

Some techniques were also required for data cleaning and scaling:

- Imputation: Missing values were imputed using the mean for numerical columns.
- Feature Scaling: Since algorithms like K-Nearest Neighbors (KNN) are sensitive to feature scales, StandardScaler was applied to standardize numerical features.

4. Algorithm Selection and Model Training

To achieve the best classification accuracy, multiple machine learning models were tested, beginning with K-Nearest Neighbors (KNN) due to its simplicity and effectiveness with scaled data. The following steps were taken:

- Initial Model: KNN was chosen as the baseline due to its straightforward approach to classification and sensitivity to feature scaling. A value of k=3 provided a reasonable tradeoff between overfitting and underfitting.

These are some of the method that could have been applied for better accuracy as suggested by ChatGPT:

- Hyperparameter Tuning: Additional models, such as Random Forest, were evaluated using GridSearchCV to tune parameters like n_estimators and max_depth. This step provided insight into whether a more complex model could outperform KNN.
- Offline Evaluation and Cross-Validation: Cross-validation was used to evaluate model performance reliably, mitigating the risk of overfitting on a single train-test split.

5. Evaluation Metrics and Results

- Accuracy: The primary metric used for evaluation was accuracy, calculated on both the test set and through cross-validation.
- Confusion Matrix: A confusion matrix was examined to understand where the model struggled. The matrix highlighted common misclassifications, particularly between scores 4.0 and 5.0, which is likely due to similar review patterns.

Results

The KNN model achieved an accuracy of approximately 53% on the test set. Although not optimal, this result reflected the challenges posed by class imbalance and the limited predictive power of certain features.

Features related to Helpfulness, Text_Length, and User_Mean_Score ranked highly, which shows that helpfulness and verbosity were important indicators of review scores.

6. Challenges, Assumptions, and Special Tricks

- Imputation Assumptions: Missing values were imputed with the mean, assuming that this would maintain the distributional properties of the features.
- Feature Selection Consistency: Assumed that engineered features would remain consistent across train, test, and submission sets. Adjustments were made to ensure feature order and alignment.
- Special Tricks:
    - Feature Engineering from Text: Extracted multiple features from the Text and Summary fields to capture sentiment and verbosity, providing additional context for the model.
    - Use of Temporal Features: Review_Year and Review_Month were incorporated to capture any potential temporal effects.
    - Rather than using overly complex algorithms, the focus remained on interpretable features and straightforward preprocessing steps.

7. Future Work and Improvements (credits to ChatGPT)

While the model achieved a reasonable accuracy, several avenues for improvement remain:

- Ensemble Models: Using ensemble techniques could further improve accuracy by combining the strengths of different models.
- Advanced Text Processing: Techniques like TF-IDF or word embeddings could provide a more nuanced representation of review text.
- Class Balancing: Methods such as SMOTE or weighted class adjustments could better handle class imbalance and potentially improve performance.

8. Conclusion

In this assignment, the primary focus was on thoughtful feature engineering, effective data preprocessing, and offline evaluation to build a classification model for product review scores. Although accuracy could be improved, the insights gained through feature engineering and the iterative process of model selection provided a deeper understanding of the data. In my future work, I will explore advanced text representation and ensemble methods, with the aim of refining the model's predictive performance.