1. Introduction

The goal of this project was to build a model that predicts a target variable from a large-scale dataset containing both numerical and text-based features as more accurate as possible. In order to achieve this, I use a linear classifier due to its effectiveness with high-dimensional, sparse data. The project involved various stages of data preprocessing, feature engineering, and model training.

2. Feature Selection

1. HelpfulnessNumerator: Number of users who found a review helpful.
2. HelpfulnessDenominator: Number of users who rated the helpfulness of a review.
3. Time: Timestamp of the review.
4. Helpfulness: Ratio of HelpfulnessNumerator to HelpfulnessDenominator.
5. Summary_Length: Character length of the review summary.
6. Text: Full review text.

These features were chosen because they provide a mix of numerical and text-based inputs, allowing for a holistic analysis of the data. The assumption here was that both numerical statistics and textual information would be valuable in predicting the target variable.

Handling Textual Data

To use both the Summary and Text columns, a new feature called ‘Combined_Text’ was created by concatenating these two columns. This was done to ensure that information from both short and detailed review fields was captured. The creation of ‘Combined_Text’ assumed that combining different sources of textual information would provide a more comprehensive representation of the review content, potentially improving the model's performance.

Creating Interaction Features

In addition to the standard numerical features, I create a new interaction feature called ‘Length_Helpfulness’ which created by multiplying ‘Summary_Length’ and Helpfulness. This feature aims to capture potential interactions between how detailed a

review is and how helpful users found it. The assumption was that longer summaries could have different impacts based on their perceived helpfulness, thus influencing the target variable.
Creating interaction features was one of the key techniques used to enhance the model's ability to capture non-linear relationships in the data, which are not directly observable from individual features.

TF-IDF Vectorization for Text Features

To convert the text-based feature 'Combined_Text' into numerical data suitable for machine learning, TF-IDF vectorization was applied. It helps assign higher importance to words that are frequent within a document but less common across the entire dataset. The assumption here was that using a mix of unigrams and bigrams would provide better context, especially in cases where two-word phrases carry more semantic meaning than individual words.

Combining Numerical and Text Features

Once the text was vectorized using TF-IDF, the text-based features were combined with the numerical features into a single sparse matrix. This matrix contained both text-based and numerical inputs, providing a unified representation of the data. This combined feature set served as the input for the linear SGD classifier.
Combining the features in this way ensured that the model could simultaneously consider textual semantics and numerical patterns, leading to a more robust prediction.

Use SciPy for Sparse Data Handling

Memory Efficiency: SciPy's sparse matrix functions ensure that only non-zero elements are stored, which is crucial when dealing with high-dimensional data, such as TF-IDF features.
Scalability: Operations on sparse matrices are computationally efficient, making it feasible to train models on large datasets without excessive memory usage or slow computations.

Seamless Integration: The use of csr_matrix and hstack allowed for seamless integration of numerical and text-based features into a single input matrix, enabling the model to leverage both types of information effectively.

3.Model Choice: SGD Classifier

The SGD Classifier was selected as the final model due to its advantages:

Scalability**:** SGD is well-suited for large datasets and high-dimensional data, making it a natural choice for the given problem.

Flexibility**:** The classifier supports multiple loss functions, allowing for experimentation with different objective functions to find the best fit for the data.

Compatibility with Sparse Data**:** SGD works well with sparse matrices, which is critical given the use of TF-IDF for text vectorization.

The final model, based on the optimized **SGDClassifier**, effectively utilized a combination of numerical and text-based features to achieve good predictive performance. By applying advanced preprocessing steps, interaction features, TF-IDF vectorization, and rigorous hyperparameter tuning, the model was able to generalize well across different subsets of the data.