Algorithm Implementation Writeup

1. Introduction

The algorithm implemented is for classifying Amazon Movie review scores based on text data. This classification task involves using a combination of feature engineering, vectorisation, dimensionality reduction, and ensemble modelling.

2. Preprocessing and Feature Engineering

2.1 Dataset Loading and Initial Processing

The model utilises the two given datasets: a training set and a testing set. The training set includes text data and a target variable (Score), while the testing set includes only text data. To handle potential missing values in the text data, both Text and Summary fields were filled with empty strings to avoid errors during vectorisation.

2.2 Feature Engineering

Feature engineering involved adding multiple new features:

- Helpfulness Ratio: Calculated as the ratio of HelpfulnessNumerator to HelpfulnessDenominator. This ratio is for capturing the usefulness of each review based on user feedback.
- Text Length Features: The length of the Text and Summary fields was added to capture verbosity.
- Word Count Features: Word counts for Text and Summary were computed, as they can correlate with detailed or brief reviews.

I used these features to capture nuances in user reviews, such as perceived helpfulness, verbosity, and engagement. Missing values in these features were handled by filling them with zeros.

2.3 Text Vectorization using TF-IDF (Term Frequency-Inverse Document Frequency)

To capture textual patterns, I used a TfidfVectorizer on the Text field with a maximum of 8,000 features. TF-IDF scores were calculated for the most relevant terms in the reviews, representing each review in a way that highlights important terms while reducing redundancy. I had limited my model to 5,000 features for a previous trial, but I thought an increase in features with the same reduction would capture more of the important values.

2.4 Dimensionality Reduction with Truncated SVD

To reduce the high-dimensional TF-IDF matrix, I applied a Truncated Singular Value Decomposition (SVD), which reduced the feature space to 50 components. This mitigated the risk of overfitting, compressed the text data, and captured the most significant semantic patterns.

I ensured reproducibility of the transformation results, which allows consistent feature representation across model training and evaluation. I also experimented with various component

counts and found 50 as a balance between maintaining text information and computational efficiency.

2.5 Combining Numerical and Text Features

The selected numerical features (HelpfulnessNumerator, HelpfulnessDenominator, Time, and Helpfulness) were combined with the text features from SVD. This combination created a single feature array for each review.

3. Thought Process and Patterns Observed

Throughout the development, I identified several patterns, which informed the selection of features and model choices:

- Helpfulness Ratio Impact: Reviews with higher helpfulness scores tended to align closely with extreme sentiment (high or low ratings). Including this feature helped the model differentiate between average and more opinionated reviews.
- Text Length and Verbosity: I observed that reviews with longer Text lengths were often associated with more detailed and polarised opinions, which influenced the use of Text_length and Summary_length features.
- Word Count and Rating Consistency: Reviews with low word counts or character counts often received low scores, perhaps due to their lack of detail.

By identifying these patterns, I focused on features that would capture the unique aspects of each review's sentiment. Numerical feature patterns, especially those related to helpfulness, length, and timing, were integrated with text-based patterns to create a comprehensive feature set for the final model.

4. Model Experimentation and Selection

After preprocessing and feature engineering, I experimented with various models, including XGBoost and Gradient Boosting. Although both models are known for their performance in structured data, they did not perform as well as Random Forest on this specific task. Possible reasons could be:

- Feature Interaction Complexity: The text data, even with dimensionality reduction, created complex feature interactions, which the Random Forest model handled better without significant hyperparameter tuning.
- Computational Efficiency: Random Forest with 400 estimators was relatively faster to train and validate, particularly when the dataset contained both text-based and numeric features. Some of my trials with Gradient boosting grid search took over an hour to run.

The Random Forest Classifier ultimately provided a good balance of interpretability, performance, and computational efficiency, making it the best choice for this model.

5. Model Training

A RandomForestClassifier was used as the main model due to its ability to handle mixed feature types.

5.1 Training and Validation Split

The data was split into training (75%) and validation (25%) sets, to ensure the model's generalisability. The choice of a validation set was essential for estimating model performance on unseen data before the final testing phase.

5.2 Model Parameters

A RandomForestClassifier with 400 estimators was selected, which improved stability while controlling computational complexity. The n_jobs=-1 parameter was used to parallelise the model fitting process, allowing for efficient processing.
I experimented with different values for n_estimators and found that 400 offered a good balance between accuracy and speed. Setting n_jobs=-1 allowed the model to fully utilise CPU cores, significantly speeding up training without compromising accuracy.

6. Model Evaluation

Model performance was evaluated using validation accuracy and a confusion matrix:

- Accuracy: Validation accuracy provides a quantitative measure of model performance.
- Confusion Matrix: A confusion matrix normalised by the true label counts was plotted to visualise the classifier's ability to distinguish between classes.

These metrics provided insight into the model's strengths and weaknesses, particularly its performance across different review scores.

7. Conclusion

The final model was designed to capture meaningful review characteristics, balancing text and numerical feature representations. Using both traditional feature engineering and modern machine learning techniques, the model classifies review scores with a 58.6% accuracy.