**Predicting Review Helpfulness Using Machine Learning Models**

**WEI DAI**

**1. Introduction**

Online reviews have become a critical part of the decision-making process for consumers and businesses. With many reviews available, it is essential to identify which reviews are most helpful for businesses to improve products and services and for consumers to focus on relevant feedback.

In this project, we aim to predict whether a review will be considered helpful based on key features extracted from the review. Specifically, our research question is:  
How do various product features, such as star ratings and review length, correlate with the helpfulness score of a review?

To address this question, we employed machine learning models, focusing on features that included:

**The content of the review (TF-IDF features)**: Captured through a textual vectorization method that measures word importance.

**Star ratings (ScoreIndexed)**: The numerical rating of the product provided by the reviewer.

**Review length**: The total number of words in the review, which indicates the level of detail provided.

**Interaction between rating and review length (Score-Length Interaction)**: This feature captures how the length of the review interacts with the star rating to influence helpfulness.

Our goal is to build predictive models using these features to classify reviews as "helpful" or "not helpful." We evaluate the performance of different machine learning models in this task, including Logistic Regression, Random Forest, and Support Vector Machines (SVM).

**2. Methodology**

**2.1 Data Collection**

The dataset used in this project comes from Amazon.com and contains a comprehensive set of product reviews. However, I get the dataset from https://www.kaggle.com/datasets/arhamrumi/amazon-product-reviews instead of obtaining it directly from Amazon. The dataset includes 568,454 records and 10 attributes, stored in CSV format. The following fields are available in the dataset:

**Id**: Unique identifier for each review.

**ProductId**: Identifier for the product being reviewed.

**UserId**: Identifier for the user who submitted the review.

**ProfileName**: Name of the user who submitted the review.

**HelpfulnessNumerator**: Number of users who found the review helpful.

**HelpfulnessDenominator**: Number of users who voted on the helpfulness of the review.

**Score**: The product rating given by the reviewer (1-5 stars).

**Time**: The timestamp when the review was posted.

**Summary**: A brief summary of the review.

**Text**: The full text of the review.

The primary fields used in this analysis are the Score, HelpfulnessNumerator, HelpfulnessDenominator, and Text, which were cleaned and transformed into useful features for predicting the helpfulness of the review. The dataset provides a rich source of information for analyzing the correlation between review characteristics and helpfulness.

**2.2 Data Preprocessing**

**Text Cleaning:**1. Removed digits from the review text.

2.Removed non-word characters (punctuation, special characters).

3.Converted all text to lowercase to standardize the format.

**Tokenization:** Split the cleaned text into individual words (tokens) using the Tokenizer from PySpark.

**Stopwords Removal:** Removed common stopwords (e.g., "and," "the," "is") using the StopWordsRemover to reduce noise and focus on meaningful words.

**TF-IDF Vectorization:**Transformed the tokenized and filtered words into numerical features using Term Frequency-Inverse Document Frequency (TF-IDF) with HashingTF and IDF. This captures the importance of words in a review relative to their frequency across all reviews.

**Helpfulness Ratio:** Created a new feature, *Helpfulness Ratio*, by dividing the HelpfulnessNumerator by the HelpfulnessDenominator, indicating how helpful the review was based on user votes.

**Binary Target Creation:** Created a binary target label (*HelpfulBinary*) by checking if the helpfulness ratio was greater than 0.5. Reviews with a ratio greater than 0.5 were labeled as "helpful" (1), and others as "not helpful" (0). I filtered out reviews where HelpfulnessDenominator is 0.

**Feature Engineering:**1. Created a new feature called *Review Length*, which represents the number of words in each review after tokenization and stopword removal.

2.Used StringIndexer to convert the star rating (Score) into a numerical value (*ScoreIndexed*).

3.Created an interaction feature (*Score-Length Interaction*) that multiplies the review length by the product rating (*ScoreIndexed*) to capture interactions between these two variables. In this case, a review's helpfulness might depend not just on the length of the review or the rating separately but on how these two features interact For instance: a longer review with a high rating might be more helpful because it provides detailed positive feedback. Conversely, a longer review with a low rating might be even more helpful because it could offer a detailed critique, which can be more insightful for consumers. On the other hand, a short review might be less helpful regardless of the rating.

**Feature Assembly:** Assembled all key features (TF-IDF vectors, *ScoreIndexed*, *Review Length*, and *Score-Length Interaction*) into a single feature vector using VectorAssembler.

**Feature Scaling:** Applied StandardScaler to scale all features, ensuring that they contribute equally during model training.

**2.3 Handling Class Imbalance:** The dataset exhibited a class imbalance, with more reviews labeled as helpful than not helpful. To address this, class weights were adjusted in the models.

**2.4 Model Selection**

We evaluated three machine learning models:

**Logistic Regression** with hyperparameters for regularization (regParam) and the elastic net mixing parameter (elasticNetParam).

**Random Forest Classifier** with hyperparameters for the number of trees (numTrees), maximum depth (maxDepth), and minimum instances per node (minInstancesPerNode).

**Support Vector Machine (SVM)** with hyperparameters for the maximum number of iterations (maxIter) and regularization (regParam).

Each model was trained using the cleaned and engineered features, and hyperparameter tuning was conducted using 5-fold cross-validation to optimize the models.

**2.5 Evaluation Metrics**

The models were evaluated using the following metrics:

**Accuracy**: The proportion of correctly classified reviews.

**Precision**: The proportion of predicted "helpful" reviews that were actually helpful.

**Recall**: The proportion of actual helpful reviews that were correctly identified.

**F1 Score**: The harmonic mean of precision and recall, providing a balanced measure of model performance.

**Confusion Matrix**: To visualize the true positives, false positives, true negatives, and false negatives for each model.

**Results**

**Model Performance**:

**Logistic Regression**:

Accuracy: 0.7538, Precision: 0.7540, Recall: 0.7538, F1 Score: 0.7538

Confusion matrix:

[[ 480. 13690.]

[ 167. 44806.]]

**Random Forest**:

Accuracy: 0.7666, Precision: 0.8170, Recall: 0.7666, F1 Score: 0.6715

Confusion matrix:

[[3.7200e+02 1.3798e+04]

[7.0000e+00 4.4966e+04]]

**SVM**:

Accuracy: 0.7604, Precision: 0.5782, Recall: 0.7604, F1 Score: 0.6569

Confusion matrix:

[[ 0. 14170.]

[ 0. 44973.]]

**Best Hyperparameters**:

Logistic Regression: regParam = 0.01, elasticNetParam = 0.0

Random Forest: numTrees = 10, maxDepth = 20, minInstancesPerNode = 1

SVM: maxIter = 50, regParam = 0.01

**4. Discussion**

**4.1 Model Comparisons**

The Random Forest model outperformed Logistic Regression and SVM in terms of accuracy. However, its F1 score was lower due to an imbalance between precision and recall. Logistic Regression and SVM, while achieving lower accuracy, had better F1 scores due to their balanced handling of both classes.

**4.2 Feature importance:**

**Random Forest Model:**

TF-IDF Features: 0.0 (This suggests that the textual content of reviews, represented by TF-IDF, did not contribute significantly to the Random Forest model's performance in this particular case.)

ScoreIndexed (Star Rating): 0.0001964846790571074

ReviewLength: 0.0001456189964397172

ScoreLengthInteraction: 0.000013908128265083569 (Interaction between score and length contributes very little to the model's performance.)

**2. Logistic Regression Model:**

TF-IDF Features: 0.011442100193983798 (Higher than Random Forest, indicating that text content is slightly more important for Logistic Regression.)

ScoreIndexed (Star Rating): 0.003916559379116645

ReviewLength: 0.0056936361639672904

ScoreLengthInteraction: 0.007654054156566802

**3. SVM Model:**

TF-IDF Features: 0.0012034916087930094

ScoreIndexed (Star Rating): 3.600980202301592e-05

ReviewLength: 0.00018184021876386817

ScoreLengthInteraction: 0.0007544985812825008

**Analysis:**

**TF-IDF (Text Content):**

In the Random Forest model, TF-IDF does not appear to contribute at all, with an importance score of 0.0. However, it plays a slightly more important role in Logistic Regression and SVM, particularly in Logistic Regression where its coefficient is 0.01144. This suggests that for these models, the actual content of the review has a small but non-negligible impact on predicting helpfulness.

**ScoreIndexed (Star Rating):**

Across all models, the star rating has a low impact on the predictions. The coefficients and feature importances are very small, suggesting that the star rating alone is not a strong predictor of whether a review is helpful.

**Review Length:**

Review length shows a similar trend. The impact of review length is modest in Logistic Regression (0.00569) but nearly negligible in Random Forest and SVM.

**ScoreLengthInteraction:**

The interaction feature between star rating and review length contributes very little in all models. The feature importance and coefficients are quite small, indicating that this combined feature does not have a significant impact on the model's predictions.

**4.3 Limitations**

Despite moderate accuracy, all models struggled with identifying helpful reviews. It is reflected in the low feature importance and small coefficients. TF-IDF features, star rating, and review length contributed only marginally to model predictions. It is suggesting that these features alone may not be sufficient to capture the complex factors influencing review helpfulness. Additionally, the class imbalance likely affected the models' ability to accurately predict helpful reviews, even with class weight adjustments. The interaction between review length and star rating also had minimal impact. It is indicating that more advanced feature engineering, such as sentiment analysis or topic modeling, might be needed.

**5. Conclusion**

In this project, we explored the task of predicting review helpfulness using machine learning models. While Random Forest achieved the highest accuracy, the feature importance scores indicated that key features such as review length, star rating, and text content had limited influence on model predictions. Logistic Regression and SVM performed similarly, though none of the models demonstrated a strong ability to predict helpful reviews. It is likely due to the relatively weak impact of the available features.

Future work should focus on enhancing feature engineering, incorporating sentiment analysis to capture the tone of reviews, or leveraging techniques like oversampling (e.g., SMOTE) to address class imbalance. Additionally, experimenting with more complex models such as deep learning could help to uncover deeper patterns and improve the models' ability to predict helpful reviews with greater accuracy.