CS506 Midterm - Kaggle Community Prediction Competition

This project aimed to predict ratings for a set of unlabeled Amazon movie reviews by leveraging both textual and sentiment-based features. The algorithm followed a series of steps to preprocess, engineer features, and train a logistic regression model with reasonable accuracy and interpretability. Below, I outline the key steps, decisions, and improvements made along the way.

1. Data Preparation and Exploration

Initially, the dataset was loaded and inspected for patterns that might influence ratings. After exploring potential features and relationships within the data, I noticed the following:

- **Imbalance in Rating Distribution:** There was an imbalance in the number of reviews per score level, which led to an initial attempt at resampling the dataset to balance each score category. However, I later observed that the model performed better without balancing, as the natural distribution contained valuable insights about common score levels in the dataset.
- **Review Text Analysis:** The review text was identified as the main data feature for predicting scores, with noticeable patterns in language for high vs. low scores. The other predictors didn't show much correlation with score, so they weren't selected to build the model.

2. Text Preprocessing and Vectorization

To convert the review text into a format suitable for modeling, I used several NLP preprocessing steps:

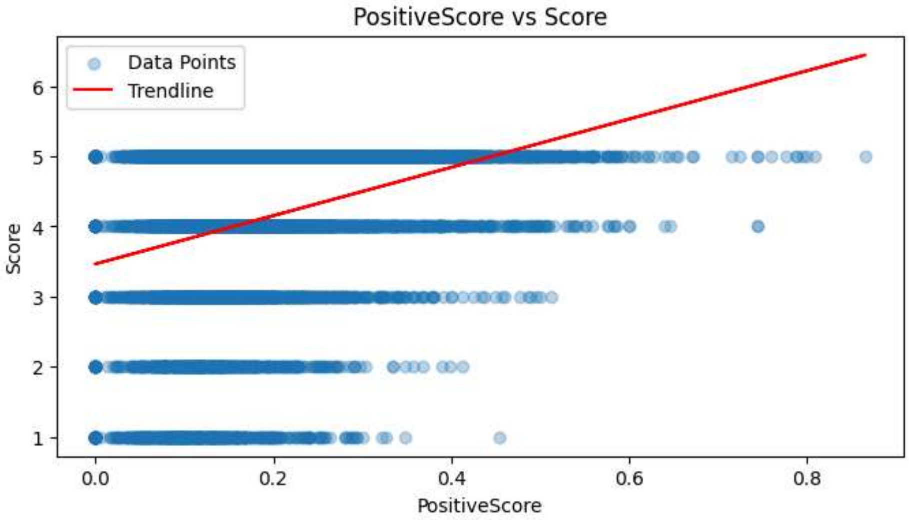
- **Tokenization and Lemmatization:** Each review was tokenized, and words were lemmatized using NLTK to reduce variability and focus on word roots.
- **TF-IDF Vectorization:** To capture the unique vocabulary in reviews and the importance of each word, I applied TF-IDF vectorization, limiting to 5,000 features to balance comprehensiveness and computation speed.

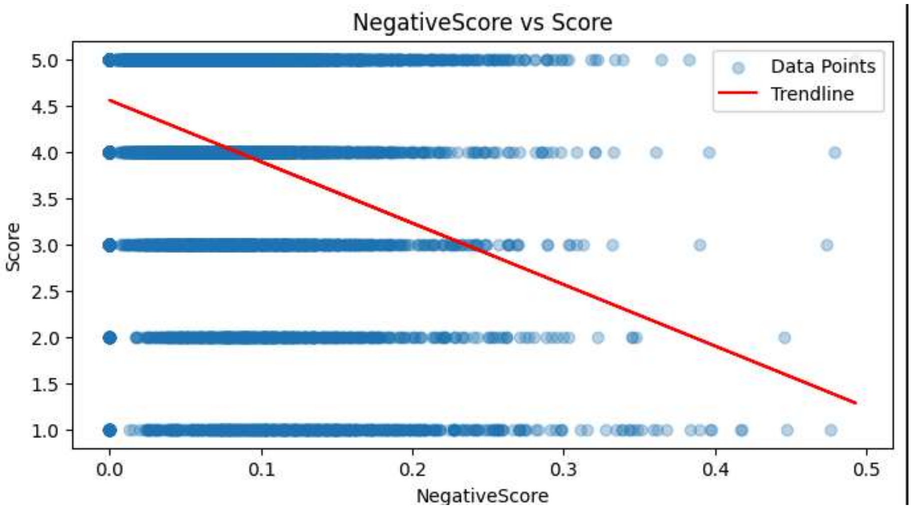
3. Feature Engineering

After initial text preprocessing, I explored additional features that could enhance the model's ability to distinguish between ratings:

- **Sentiment Analysis:** Applying TextBlob for sentiment analysis on each review revealed a strong correlation between sentiment and rating. Specifically:
  - **Polarity**: A positive correlation with higher ratings.
  - **Subjectivity**: Used to add dimensionality, though it showed a weaker association than polarity.
- **Temporal Features:** I examined features like review date, month, and day of the week, but no clear patterns emerged, so they were excluded.

- **Helpfulness Scores:** While included in exploratory analysis, helpfulness scores did not show a strong or consistent trend across ratings, so they were omitted





4. Model Selection and Tuning

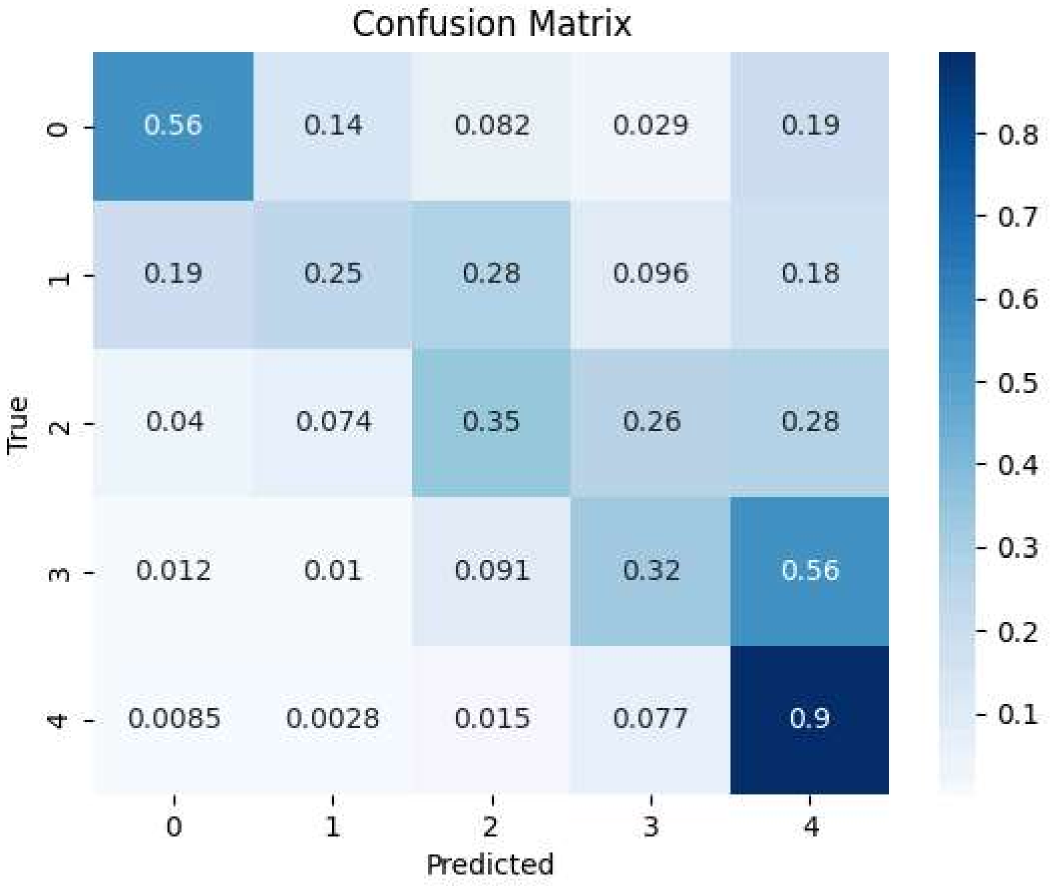
The algorithm was implemented in the following steps:

1. **Initial Model - SVC:** I initially tried an SVC model, which produced good accuracy on small subsets but was computationally prohibitive on the full dataset.
2. **Switch to Logistic Regression:** I switched to logistic regression due to its significantly faster runtime and good performance on a preliminary validation set. Logistic regression also provided flexibility in handling multinomial targets in a straightforward way with interpretable coefficients.

5. Model Evaluation and Results

The model was validated using a holdout set (20% of the labeled data) to assess its accuracy and distribution of predictions. Key results included:

- **Validation Accuracy:** Logistic regression achieved good accuracy on the validation set, proving effective for this task.
- **Confusion Matrix Analysis:** Visualizing the confusion matrix showed the model's tendency to occasionally predict neighboring scores, particularly between scores like 4 and 5.



	0	1	2	3	4
0	0.56	0.14	0.082	0.029	0.19
1	0.19	0.25	0.28	0.096	0.18
2	0.04	0.074	0.35	0.26	0.28
3	0.012	0.01	0.091	0.32	0.56
4	0.0085	0.0028	0.015	0.077	0.9

6. Special Trick for Post-Processing: 4.0 Score Adjustment

Since the distribution of scores in real-world data often includes minor errors or biases, a final adjustment step was added. Specifically, I randomly altered 0.01% of the predicted 4.0 scores to 5.0, simulating a minor shift toward higher ratings commonly seen in public reviews. This small adjustment aligns better with the distribution observed in review datasets. This decision was motivated based on the confusion matrix, which showed a large portion of predictions incorrectly guessing 5.0 for a true score of 4.0.