PDF MARKUP - CS 506 MIDTERM

Introduction:

In this competition, our objective was to predict the scores of reviews based on various features extracted from the text data. To accomplish this I implemented a K-Nearest-Neighbors Classifier in my final model, focusing on extracting meaningful features from the reviews, applying preprocessing techniques, and performing hyperparameter tuning. Below is the outline of my project implementation, strategies, special tricks and assumptions I have made while working on the assignment

Algorithm:

I chose to use a K-Nearest Neighbors algorithm Classifier, and experimented with different hyperparameters using GridSearch to identify the optimal number of neighbors. Thanks to its simplicity and ability to handle non-lineear relations, KNN was very useful in ensuring that my code produced predictable and accurate outcomes.

My workflow consisted of:

1. Data Preparation: I loaded and sampled the csv files for efficiency.
2. Feature Engineering: Through extracting and creating new features from the review text as well as the metadata, I made sure that I would combine and utilize related functionalities and features.
3. Data Preprocessing: I handled missing values depending on whether they were numerical or not
4. Model Selection and Hyperparameter Tuning: I utilized GridSearch to find the most optimal n_neighbor value.
5. Model Evaluation: In addition to extensively testing the performance on a test set, I have also visualized the results using a confusion matrix.
6. Submission. I have used the existing code for this section to create a submission script to be submitted to Kaggle.

Feature Engineering:

A crucial part of my approach was creating features that brought out meaningful insights from the reviews. Here's what I focused on:

- **Helpfulness**: I made use of the existing HelpfulnessNumerator and HelpfulnessDenominator to compute the helpfulness score. This metric indicates how helpful a review is to others. If no helpfulness data was available, I simply set the score to 0.
- **Review Length**: I measured the length of each review text to capture verbosity, as longer reviews might correlate with higher scores.
- **Word Count**: Extracting the word count of each review gave me another useful feature.
- **Sentiment Analysis**: Using TextBlob, I analyzed the sentiment of each review, creating a ReviewSentiment score that helped capture the overall tone—positive, neutral, or negative.

- **Has Summary**: I added a new binary feature to identify whether a review included a summary, as these can often be more detailed and informative.
- **Special Trick**: The Helpfulness feature played a key role in distinguishing reviews, with highly helpful reviews often reflecting higher scores. Combining this with sentiment analysis gave me a direct way to gauge positivity or negativity, giving my model an extra advantage in terms of classification

Preprocessing and Data Handling

To maintain consistency and improve model performance, I followed these key steps:

- **Handling Missing Values**: I addressed missing data by filling in sensible defaults. For example, if HelpfulnessNumerator or HelpfulnessDenominator values were missing, I set them to 0 and 1, respectively. Missing text data was replaced with empty strings.
- **Standard Scaling**: I applied standard scaling to features like Helpfulness, Review Length, and Word Count to ensure no single feature dominated due to varying ranges.
- **Balancing the Training Set**: Using stratified sampling, I ensured a balanced distribution across different score classes, which helped prevent bias in the model.

Assumptions:

- Reviews without helpfulness data were treated as having a Helpfulness score of 0.
- Empty review text was considered to have a neutral sentiment (0.0).
- I assumed that shorter reviews might be linked to extreme scores (either very high or very low), while medium-length reviews often reflected more balanced ratings.

Model Tuning and Selection:

I used GridSearchCV to fine-tune the n_neighbors parameter for the KNN model. This process involved cross-validation to identify the optimal parameter value, which helped enhance the model's generalizability. In the end, I selected the configuration that performed best based on the grid search results.

Best Parameters Identified: n_neighbors = 11

Special Trick: I used stratified train-test splitting to ensure a balanced distribution of score classes. This approach helped prevent bias toward any particular class & led to better overall performance and more accurate classification of the reviews

Evaluation and Results:
After training my model, I evaluated its performance on a separate test set. To measure its effectiveness, I focused on accuracy and created a confusion matrix.

- **Accuracy**: The final accuracy I achieved on the test set was 51.4%.

- **Confusion Matrix**: I visualized the confusion matrix to analyze misclassifications and identify patterns. This visualization revealed which scores were frequently misclassified, providing valuable insights that I could use to refine the model further if needed.

Conclusion

My approach to the competition centered on extracting meaningful features from the text data and applying effective preprocessing techniques to enhance performance. By carefully tuning the KNN model, I achieved a robust classification that balanced accuracy and interpretability.

Key Takeaways:

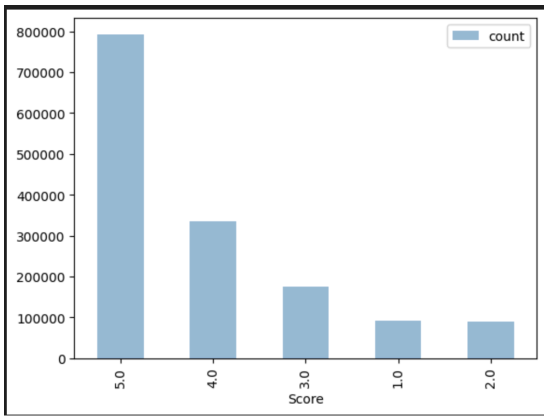
- **Feature Engineering**: Focusing on features like Helpfulness and sentiment analysis significantly improved the model's accuracy.
- **Standard Scaling and Stratified Sampling**: These techniques ensured that the model maintained consistent performance across different review scores.
- **Hyperparameter Tuning**: Utilizing grid search allowed me to identify the best configuration for KNN, resulting in a well-rounded model.

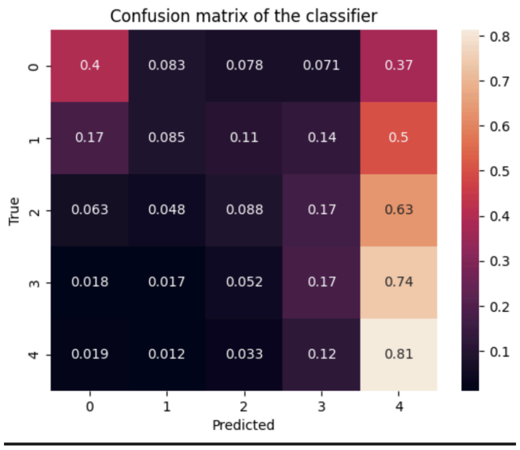
MODEL CHOICE:

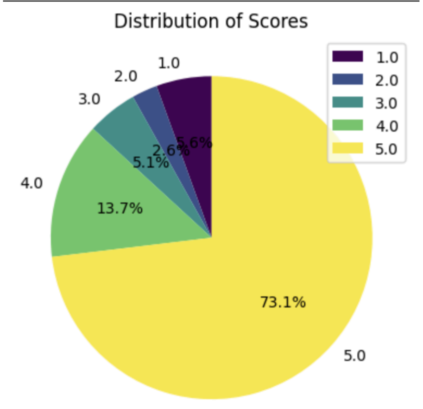
I went with K-Nearest Neighbors (KNN) because it's a straightforward yet powerful model that aligns well with this type of data. KNN excels at handling non-linear patterns, which I believe was ideal since review scores often don't have a clear linear relationship with features of the text. While models like SVM or Random Forests could offer more accuracy, they require far more tuning and can be difficult to interpret. With KNN, I could easily visualize and understand why certain reviews were classified the way they were, making it much more intuitive. Plus, by using GridSearch to fine-tune the `n neighbors` parameter, I struck a solid balance between accuracy and model simplicity, avoiding unnecessary complexity while still capturing the nuances in the review data.

RESULTS

My model reached an accuracy of 51.29%


count
Score


Confusion matrix of the classifier
True
Predicted


Distribution of Scores