Kelvin Yeung
CS506
Prof. Galletti
October 28th, 2024

Midterm 2 Write Up

Introduction

Throughout this midterm, I explored different machine learning techniques that build on the foundation of the topics covered in class. In order to properly classify the score, the rating of the Amazon review, we have to first analyze the data fields that provide relevance to training the model and then predicting the expected score.

Feature Engineering

Initially, when the midterm was first released I attempted to gauge what could be relevant and help KNN most accurately predict. Taking inspiration from the Helpfulness Ratio, denoted by $\frac{HelpfulnessNumerator}{HelpfulnessDenominator}$. This feature is the ratio of the number of users who found the review helpful over the total number of users who provided feedback. Within a KNN context, Helpfulness helps the model group reviews based on the "perceived" usefulness, where reviews with similar helpfulness rating might be grouped together due to their similarity.
High-helpfulness scores also imply well-received reviews, adding context that could improve prediction accuracy, especially in sentiment or quality-related tasks. Regarding sentiment analysis within the model, we utilized TextBlob and VADER to extract sentiment-driven features from the review text. TextBlob, a lexicon-based sentiment analysis tool, provided a polarity score ranging from -1 (negative) to 1 (positive), and helped quantify the emotional tone of the reviews. Additionally, using TextBlob's sentence-level analysis we enabled detection of "mixed sentiment" by identifying reviews that held both positive and negative connotations within a single text. Although this feature made logical sense, the reviews in train.csv raised a discrepancy. Some of the reviews contained a summary of the movie itself, which could be taken positively or negatively based on the plot. As a result, I used VADER (Valence Aware Dictionary and Sentiment Reasoner) to supplement a compound score in consumer reviews, which often include slang or informally charged language. By incorporating both tools, the model could represent reviews not only by content but also by sentiment profile, supporting the classifier in distinguishing between reviews with positive, negative, and mixed tones more effectively.

In conjunction with sentiment-analysis, we used Term Frequency-Inverse Document Frequency (TF-IDF) as a key feature extraction technique to transform review text into numerical representations. TF-IDF calculates how important a word is within a specific review across all reviews, balancing the frequency of a word with its uniqueness. Words frequently used across reviews, like "movie" receive lower weights, while meaningful terms that appear sparingly but with relevance, such as "incredible" or "disappointing," are given higher weights. By doing this, we allow the model to capture descriptive reviews and encode the text into numerical features that KNN can classify. Due to the computational stress that TF-IDF has, we limited max_features to only capture 50 words by applying dimensionality reduction via Truncated SVD on the TF-IDF matrix, further reducing the feature space and selecting only the most significant components.

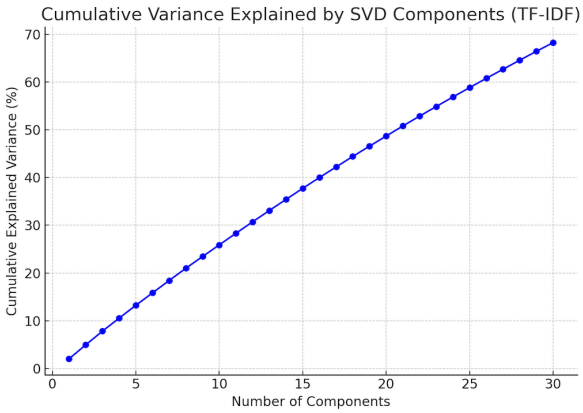

Figure 1: Variance as Correlation to Components

As for the rest of the features, like 'ReviewLength', 'UserSentimentTendency', 'UserHelpfulnessAvg', etc. It allowed us to capture User Tendencies and how effective the user is in providing feedback. For example, taking a user who consistently reviews products with negative sentiment, as captured by 'UserSentimentTendency,' may provide insight into their overall satisfaction levels. 'UserHelpfulnessAvg' enriches this by reflecting the helpfulness of a user's feedback based on how others perceive it, which can indicate the user's credibility in reviews. Together, these features offer a nuanced perspective on reviewer behavior, helping the model understand patterns not just from textual content but also from reviewer tendency.

Processing Data/Feature Selection

Before we select the features for our models, we preprocess the data to address some key issues. During processing, we ran into some inconsistent data consisting of NaN values which caused the selection to fail. As a result, we impute the rows that contain NaN values and fill them in with zero. Then we converted the sparse matrix into dense matrices for KNN to process. To address the problem of imbalance, due to the high abundance of 5 star reviews, we used SMOTE which synthetically generates samples for all minority classes and equalizes the class distribution. I noticed that SMOTE yielded the same result as undersampling and oversampling in terms of accuracy. However, class imbalance improved the model's overall predictive accuracy by learning more nuanced patterns through equal exposure to reviews.

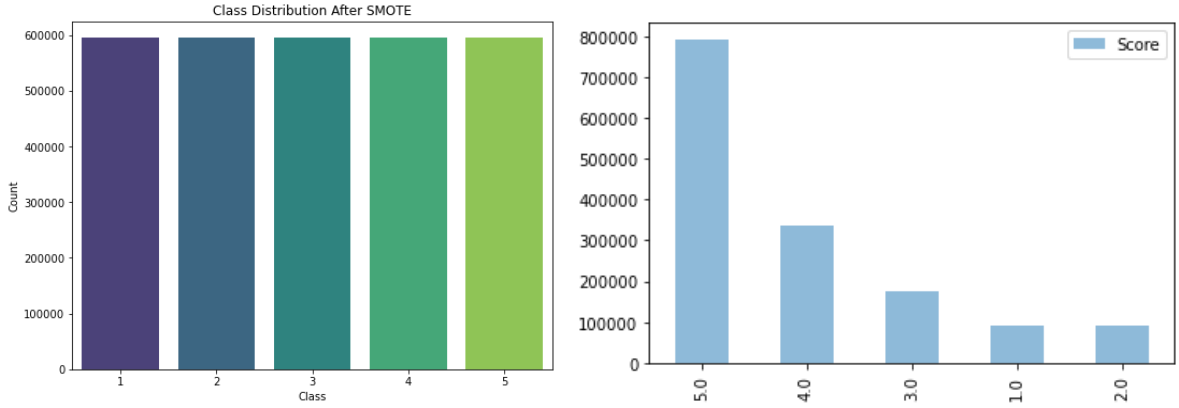

Figure 2: Class Distribution After SMOTE (shown on left), Class Distribution in train.csv (shown on right)

To minimize computation load and memory leaks, after we split the data for training and testing, we prepare the data set by selectively downsampling our features, especially the TF-IDF features, additionally we also take a smaller sample size of train.csv to speed up processing time. During feature selection, we kept the original features and retained 15 of the most informative SVD TF-IDF features. To identify the most informative TF-IDF features, we use SelectKBest and rank features based on their correlation with the target variable, thus selecting only the top TF-IDF components that provide the most variance. By selecting features with higher variance and greater correlation, it allows KNN to focus on the features that provide the most discriminative power, improving model accuracy. Finally, by using StandardScaler we normalize the data so all data points contribute equally.

Modeling and Results

For our classification model, I decided to stick to KNN classification and through fine-tuning n, number of neighbors, I found that a higher n yield a significantly higher accuracy due to the vast amount of data points that exist in the plane. By running n values from n = 5 to n = 100, I opted to use n = 100 due to the accuracy and balance of the computational limit. After the submission ended, I also experimented running another model on Random Forest. Random Forest was slightly more effective with high-dimensional data because it selects a random subset of features at each decision split. It reduces overfitting and focuses on some features. Overall, it did not yield much effect as I never fine-tuned the parameters making the best result of my accuracy to be 0.5777409138893078

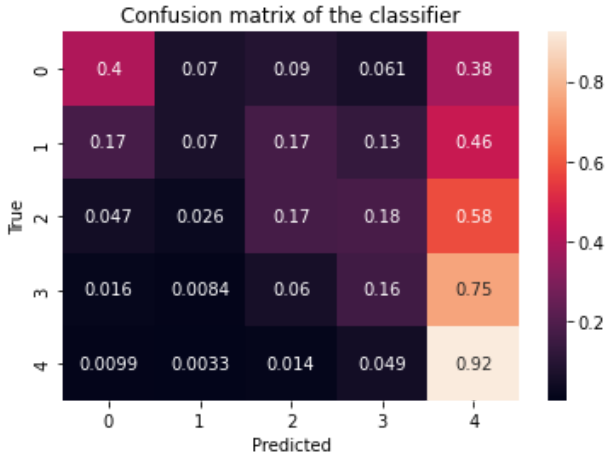

Figure 3: Confusion Matrix of classifier

Based on the confusion matrix, we can see that the model mostly predicts 5 stars correctly with a .92 accuracy. However, based on the predicted result of class 4, we can see that an overwhelming majority of predictions that were not 5 stars were classified as 5 stars. Although this is supposed to be mitigated by oversampling and undersampling, it is evident that more work needed to be done on that aspect. The second best prediction to actual rating is 1 star, however with a .4 accuracy that is also a weak prediction.