John Kinney
CS506
10/28/2024

Midterm Writeup

Predicting Amazon Movie Review Scores

Introduction

For this competition, participants were asked to predict the star rating (from 1 to 5) for respective Amazon movie reviews. Throughout this write-up, I will discuss my experience in this competition, including the good, the bad, and, most importantly, the ugly. I will start by discussing my process of understanding the data and eventually finding an ideal model for making these star predictions. Hopefully, through this report, I can also express the fun points of this competition as well.

Initial Exploration

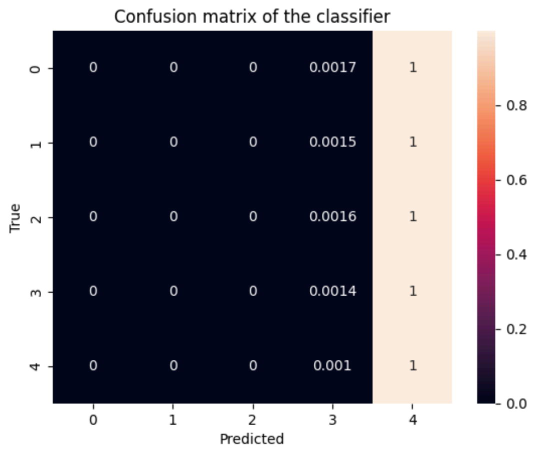

Figure 1.1: Early KNN Confusion Matrix

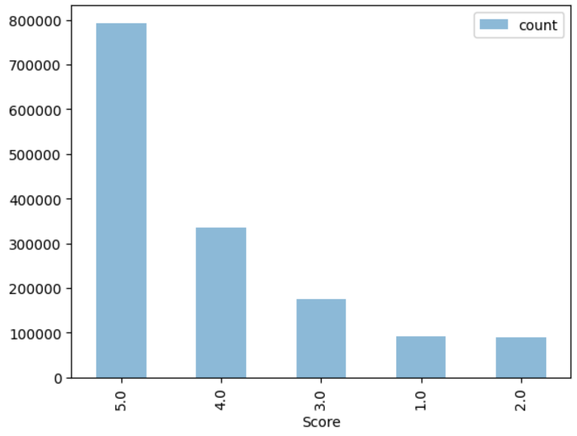

Figure 1.2: Distribution of Scores

Initially, in the contest, I started to experiment with the base code provided. This code used K-Nearest Neighbors (KNN). While this approach was a straightforward one, KNN quickly revealed its limitations to me early on. As seen in Figure 1.1, KNN was capping at an accuracy of around 0.53 as I had increased my number of neighbors, primarily selecting the score of 5. This cap unequivocally came from the dataset's imbalanced distribution (Figure 1.2), as my code was making near-constant predictions with a score of 5. KNN in this case tended to predict the majority class of score 5 due to the weight of class proximity instead of more nuanced features.

In an attempt to increase the accuracy of my predictions, I looked to add features to help capture the nuance in the difference between the score classes. By default, I was already using the helpfulness ratio, which helped, but not nearly as much as what I would find in the future. I looked to the other data fields for interesting combinations and I was able to find a very useful field, this being Text. I realized that I could use text sentiment to add more insight into how the reviews could be interpreted. Using sentiment analysis, specifically the tool TextBlob, I immediately was able to see an improvement in the trend of my predictions. I noticed that with sentiment analysis of text, I could handle the outlying points of the scores much better. For example, it became much easier to predict scores of 1 and 5 as a result of using this tool when I implemented sentiment analysis to focus on polarity (positive vs. negative sentiment) and subjectivity (degree of opinion) for each review.

Selected Model
Taking these new features into account, I decided to shift away from the KNN model, as it was very difficult to test. Unfortunately, with KNN, the model took several hours to complete a single run, which made it impractical as the deadline for the competition was drawing close. To address this, I experimented with resampling methods to lower the influence of the large quantity of 5-star scores. I tried to use grid search and random forests; however, these also only showed minimal increases for my code and were computationally costly at a time when I needed to prioritize speed and efficiency to climb in accuracy.

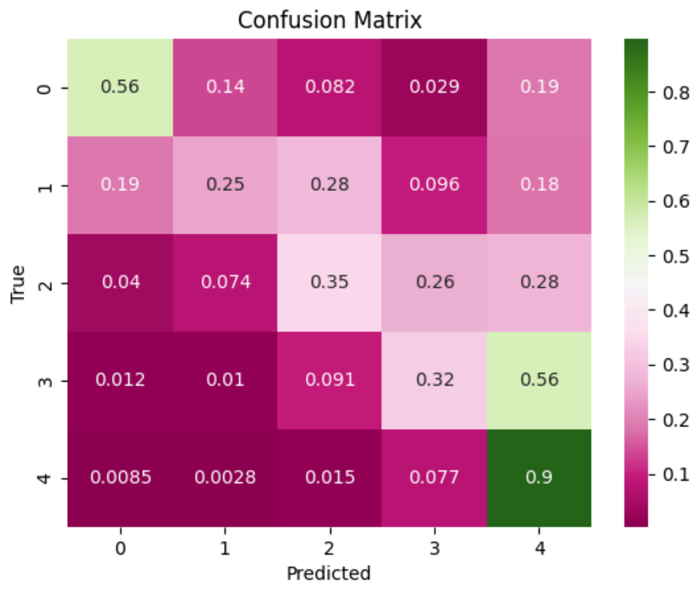

Figure 1.3: Late Logistic Regression Confusion Matrix

By the competition's closing time, I chose to use Logistic Regression. The reasoning behind this is that in my recent coursework, I have worked with logistic regression so I was somewhat familiar with it. On paper logistic regression is really good at efficiency, classification (especially binary, but I used multiclass in the context of this competition given the variety of scores), and imbalanced datasets. Given that I had very limited time left, I did my best to create a logistic regression model that took advantage of the sentiment features of polarity and subjectivity. In addition to this, I decided to pair these features with Term Frequency-Inverse Document Frequency (TF-IDF). This pairing of TF-IDF and sentiment analysis can place a higher emphasis on terms for reviews and reduce the weighting of general terms that are less relevant. Not only this but through my new approach with Logistic Regression, my previous time of executing and gathering interpretable data had shifted from several hours to less than half of an hour. This led to much easier finetuning and tweaking. In addition to using Logistic Regression, this additional pairing is what I believe allowed me to increase my prediction accuracy greatly in such a short period (Figure 1.3).
‘

Although I was able to increase my accuracy, in my own time I would like to go beyond and develop this project further. There are still underlying flaws present within my model. As seen above in Figure 1.3, my model became much better at predicting, though it still struggles between 1 and 5, specifically scores 2, 3, and 4. I believe this is because while I did make use of sentiment analysis, there are still other data fields that can be used to help establish the nuanced differences between the absolutes of the distribution (Positives and Negatives, Best and Worst, 5 and 1, etc.). I am very fortunate to have been able to take part in this opportunity. This experience has deepened my understanding of data preprocessing, feature engineering, and most importantly the selection of a proper model.

Works Cited

Scikit-Learn: Logistic Regression. Scikit-Learn, https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed 28 Oct. 2024.

Scikit-Learn: TF-IDF Vectorizer. Scikit-Learn, https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. Accessed 28 Oct. 2024.

TextBlob: Quickstart. TextBlob, https://textblob.readthedocs.io/en/dev/quickstart.html. Accessed 28 Oct. 2024.

"What is TF-IDF? Featuring Josh Moody." *YouTube*, uploaded by 97th Floor, 4 Jul. 2017, https://www.youtube.com/watch?v=sCbQGBofiMk. Accessed 28 Oct. 2024.