Pavana Manoj

Professor Galletti

CS 506

28 October 2024

Midterm Writeup

The purpose of this competition was to predict star ratings associated with Amazon movie reviews utilizing the given metadata and review text in the train.csv file. Using the Pandas library imports in the starter code, I uploaded the train.csv file and test.csv file provided to memory. I used ChatGPT for a starting point and based on its suggestion, used Random Forest Classifier modeling to predict star ratings. I looked into the RandomForestClassifier (RFC) documentation on the Scikit-Learn website and found it to be a method well-suited in handling different types of data effectively. Within the code, I imported the Random Forest Classifier from the Scikit-Learn library, initialized it with specific parameters (like the number of trees and maximum depth), and then trained it using the features I created from the review data. By combining the results from multiple "trees," the RFC model learns patterns in the reviews and makes more accurate predictions.

I also used ChatGPT to gather ideas on feature engineering techniques I could potentially implement for the best rating prediction accuracy. Then, based on these suggestions, I researched these techniques such as helpfulness metrics (mentioned in the starter code), review length, and date-based features. The first technique I used within the add_features_to method was the helpfulness ratio included within the starter code. The ratio was calculated using two of the metadata fields, HelpfulnessNumerator and HelpfulnessDenominator, in the train file:

Helpfulness = HelpfulnessNumerator/HelpfulnessDenominator. This feature helps improve the model's prediction accuracy by showing how useful a review seems to other people. Reviews rated as more helpful often match certain star ratings, which helps the model differentiate between high-quality and low-quality reviews. The second technique I used in the add_features_to method was adding features based on the review dates. I took the Time field, which is stored as a long number (a Unix timestamp), and turned it into Year and Month features. This helped the model pick up on patterns over time, like changes in how people review older compared to newer movies. Recognizing these patterns helps the model make more accurate predictions, as it learns from trends in when reviews were posted (Towards Data Science).

The other features I used included text length, summary word count, helpfulness categories, and helpfulness difference. Text length gives the model an idea of how detailed a review is. Longer reviews usually mean stronger opinions, which often connect with very high or very low ratings and help the model distinguish opinionated reviews from more neutral ones. Summary word count is useful because shorter summaries tend to be more neutral, while longer summaries might show a stronger positive or negative opinion, making it easier for the model to predict the correct rating (Medhat, Hassan, and Korashy). I also categorized helpfulness into simple groups like "Low," "Medium," and "High," which lets the model see if certain helpfulness levels directly correlate with specific star ratings (OpenAI). Finally, helpfulness difference shows how many people found a review useful as opposed to not, helping the model identify well-received reviews and improve its rating predictions (OpenAI).

To check how well the model was performing, I used a confusion matrix from the starter code to see how accurately it predicted each star rating. The Random Forest model reached an accuracy of 0.553, which was not bad considering we're working with a dataset containing over a million movie reviews, and some ratings were harder to predict accurately than others.

Based on what I've read about review trends, longer reviews tend to have very high or very low ratings, like 1 or 5 stars, while more helpful reviews are often linked with middle or higher ratings. This insight suggested that features like review length and helpfulness could help improve prediction accuracy (Mudambi and Schuff). It can be assumed that this pattern was followed in our dataset, and that the text feature performed similarly in helping the model make accurate predictions.

Besides Pandas, I used other libraries (some provided in the starter code, some not provided), like Matplotlib and Seaborn for making charts to better understand the data, and Scikit-Learn's GridSearchCV to help adjust the model settings for improved performance. While the Random Forest model worked well overall, I wasn't able to reach the 68% accuracy required for full credit. Perhaps with more research or time, I might have found ways to improve it further.

Works Cited

"Feature Engineering on Time Series Data: Transforming Signal Data of a Smartphone Accelerometer for Predictive Modeling." Towards Data Science, Medium, 2021, https://medium.com/towards-data-science/feature-engineering-on-time-series-data-transforming-signal-data-of-a-smartphone-accelerometer-for-72cbe34b8a60.

Medhat, Wesam, Ahmed Hassan, and Hoda Korashy. "Sentiment Analysis Algorithms and Applications: A Survey." Ain Shams Engineering Journal, vol. 5, no. 4, 2014, pp. 1093-1113. ScienceDirect, https://www.sciencedirect.com/science/article/pii/S2090447914000550.

Mudambi, Susan M., and David Schuff. "What Makes a Helpful Review? A Study of Customer Reviews on Amazon.com." MIS Quarterly, vol. 34, no. 1, 2010, pp. 185-200. JSTOR, https://misq.org/what-makes-a-helpful-review-a-study-of-customer-reviews-on-amazon-com.html.

OpenAI. ChatGPT (October 28 Version) [Large language model]. 2024. https://chat.openai.com/.