CSE 158 Assignment 2

1. DATASET

The dataset that we worked on is a subset of the Amazon Product Reviews dataset.[1] Particularly, we used the first 50,000 rows of the Video Games subset. We chose to only use the first 50,000 rows in order to reduce the time taken for our computations, while still maintaining enough of the data for our results to be meaningful.

The first thing we did was clean the data. We converted the reviewTime column, originally made up of strings that represent dates, to Python's datetime type so we can perform proper analyses with this column. We then noticed that there were three columns - vote, style, and image - that we decided to drop because they were made up of a majority of NaNs. After dropping those columns, we dropped all rows containing any NaN, which ended up being less than 0.01% of rows. We used the first 50,000 rows of this new cleaned data in our analysis.

After cleaning the data, we performed some exploratory analysis to get a better idea of the overall distribution of the dataset.

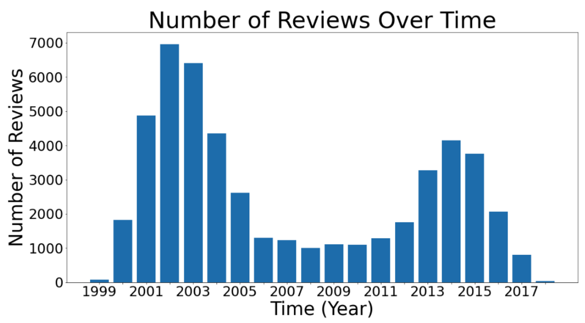


Figure 1: Graph of number of reviews over time

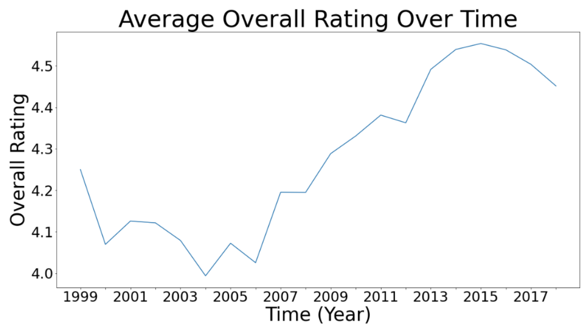


Figure 2: Graph of average overall rating over time

Within our subset of 50,000 rows, there are 13,758 unique user IDs, but only 11,924 unique names. Because there are 1834 overlapping names, we used only the user ID for user comparisons in our predictive model. The reviewer's name was therefore not used, and we dropped it as a result.

Each user also has a verification status. 30.66% of users are verified, and the average rating among them are 4.50. The average rating among unverified users is 4.13.

Our dataset contains data from all years ranging from 1999-2018. In Figure 1, we can see that the distribution of the number of reviews over time seems to be bimodal, with peaks at 2002 and 2014. The edge years, 1999 and 2018, both have significantly less data than any other year, meaning that their data is more volatile and may be less representative of the year as a whole.

The model we created predicts rating, which is an integer ranging from 1-5. Figure 2 features a graph of the average rating over time. Its general shape shows a generally downward trend from 1999-2006, and a generally upward trend from 2006-2018. Despite downward or upward trends, the average rating for any year roughly ranges from 4.0-4.6, which is quite high. The average rating overall was 4.24.

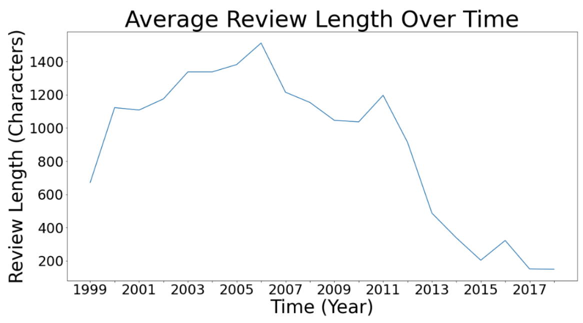


Figure 3: Graph of average review length over time

A review's text will be an important feature to include in our predictive model. Figure 3 features a graph of the average review length (in characters) over time. It appears to have an upward trend from 1999-2006, and a downward trend from 2006-2018. Interestingly, these trends are the exact reverse of the trends displayed in Figure 2, leading us to believe that review length seems to be roughly inversely correlated with overall rating. This means that the more the user writes (presumably often negative sentiments), the lower score they give.

2. PREDICTIVE TASK

As mentioned previously, a rating is an integer ranging from 1-5, which represents a user's score attached to a review of an item. We will be predicting a review's overall rating, given all of its other information. This information includes attributes, such as a user's verification status, review text, and summary text.

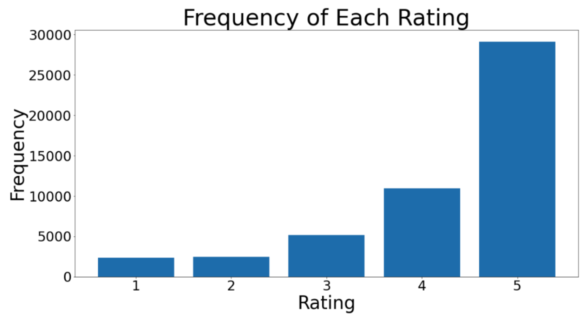


Figure 4: Graph of frequency of each rating

Next, we have to decide which metric to use to assess the validity of our model's predictions. To do so, we should take a look at the distribution of ratings. Figure 4 shows the distribution to be left-skewed and is quite imbalanced as a result. Because the distribution is imbalanced, we will use F1-score, which is the harmonic mean of precision and recall. Another commonly used metric, such as accuracy, would not work as well for our model because it weights false positives and false negatives similarly. It makes more sense in our case to weigh false positives and false negatives differently, hence the F1-score.

Although we know what metric we will use for comparisons, we need to have a baseline to compare to. We needed a simple classification model for our baseline model, so we decided on a logistic regression model for comparison. Our baseline model only utilized the review text and summary text by using the TfidfVectorizer, and resulted in an F1-score of 0.6501.

Next, we should consider which columns to use and how to transform them into features. We are given two text columns- one being the review and one being the summary. We could use a Bag of Words model, which is a set of vectors of word counts, but this does not take context into account. Instead, we chose to use TF-IDF to convert these columns into a sparse matrix, which gives more weight to words in few documents and less weight to words in most documents. TF-IDF can take additional time and memory, but our subset of data allows these costs to be more manageable.

3. MODEL

The model that we chose to implement is the logistic regression model that we used as our baseline, but this time including a temporal feature that we created using the reviewTime column, as well as the verified column. Since the objects in the reviewTime column are datetime objects, we can get the weekday in which the review was created. The weekday column that came from this consists of numerical values ranging from 0 to 6, each representing a specific weekday such as Saturday, or Sunday. We ran it through the pipeline that we created, as the additional column doesn't require any further parsing. After running the model, we found that the F1-score was slightly higher than the baseline value, with the score being 0.6519. Any further optimizations of the model would be tedious, considering that the logistic regression does not have a lot of hyperparameters that we can manipulate.

We considered using a SGDClassifier as it is often used for text classification, but it gave us a slightly lower F1-score of 0.6479. We also tried a Naive Bayes classifier, but it gave us a much lower F1-score of 0.5862. An advantage of the SGDClassifier is that it is less prone to overfitting while a logistic regressor is vulnerable to it.

Additionally, if we were to attempt to include all of the data from the original dataset, we would most likely have issues with memory, as the actual dataset is significantly larger than the number of rows that we took to create the baseline and model. Utilizing the train test split module from sklearn allowed us to split our data so that we would not encounter any issues with the model overfitting to the training data. Furthermore, we could also pull more rows from the entire dataset, which contains more data that the trained model would not have seen.

4. LITERATURE

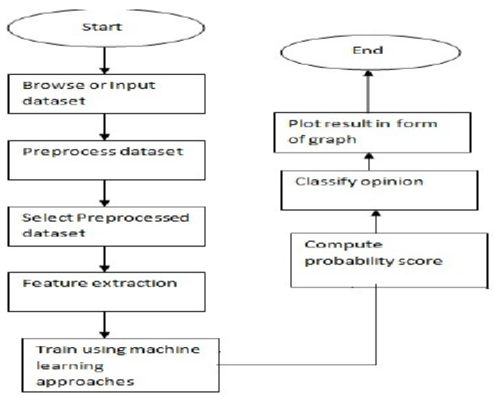
The subset of data we chose to work with came from the Amazon review dataset released in 2014. This data was and is still used for a flurry of machine learning/data science projects such as rating predictions, sentiment analysis for products, product recommendation, and much more.

Similar datasets have been studied in the past such as book reviews from GoodReads or movie reviews from IMDB. These have been analyzed in a similar fashion to Amazon product reviews because they consist of mostly the same data. A rating out of 5 or 10, a text review, and usually a ranking system to place the highly rated reviews to the top for everyone to see.

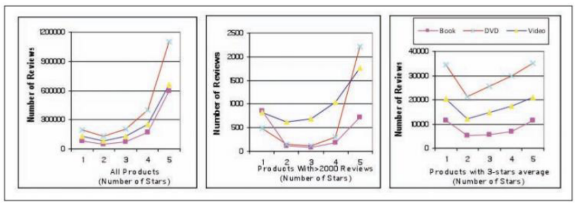
Review datasets have been primarily used with sentiment analysis to gauge its ranking among other products. We however, decided to predict the rating of products which is also useful, but not used as much in the real world.

From the research paper 'Product Review Based on Machine Learning Algorithms'[2] we will dive into state-of-the-art methods used to study product review data. Here we learn that before the advent of machine learning and the internet, the purchasing of any items mostly relied on peer feedback from friends, families, or trusted sources. But now with the mountain of data we get from the online industry, we have new ways to find if a product is worthy of purchase or not.

In this paper, they make clear that sentiment analysis is the best way to determine how a product shows up in a customer's feed. They cover several approaches in other papers relating to n-grams, Vector of Feature Intensities, and more. The approach they took started by first pre-processing data to get rid of redundant words in the reviews. An example of this is changing "administer" to work. This cleaning of data lets the model run more efficiently by having a smaller dictionary of words to keep. They also removed stop words and special characters. They then ran a unigram model to separate components in a review and attach them to descriptors. Which then they can label most of them as positive or negative. For example, if the word "great" gets attached to an adjective descriptor, it is rated as positive sentiment. They then used machine learning algorithms such as Naive Bayes and SVM to compute a probability and classify a text as positive, negative, or neutral. Here they included a graph showing their process in determining sentiment among text accurately.


Start
Browse or Input dataset
Preprocess dataset
Select Preprocessed dataset
Feature extraction
Train using machine learning approaches
Compute probability score
Classify opinion
Plot result in form of graph
End

On another point, in the article, Hu et al. (2009)[3] we can see from their data analysis of the distribution of product ratings for 3 different Amazon categories.



According to the paper, this type of distribution indicates two types of bias. First being a purchasing bias meaning that only people interested in the product end up buying it which tends to have the customer leaving more positive reviews. Another bias is people only leaving reviews if they had a very positive experience or very negative experience, leaving the neutral experience underrepresented. Resampling could be a potential solution to this problem that is faced in review datasets.

5. RESULTS

From what we saw in our model, including the weekday values that we created from the reviewTime column, as well as the verified column increased the F1 score accuracy slightly compared to the baseline. We interpreted this as the verification of a user and the time when a user posted a review having little to no effect on the predictability of their rating. In particular, after reviewing the model with the inclusion of either the verification of a user or the weekday that the review was posted, we found that the weekday had a smaller effect on the prediction of rating compared to the verification of a user. Furthermore, we determined classifiers such as SGD and Naive Bayes were not a good fit for predicting the rating, simply because there was not a large enough number of independent variables to warrant a better performance from the more complex classifiers.

6. REFERENCES

[1] **Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering**
R. He, J. McAuley
WWW, 2016
pdf
Image-based recommendations on styles and substitutes
J. McAuley, C. Targett, J. Shi, A. van den Hengel
SIGIR, 2015
pdf
[2]
https://www.researchgate.net/publication/355346616_Product_Review_Based_on_Machine_Learning_Algorithms
[3] Hu, N., Pavlou, P. & Zhang, J. (2009). Overcoming the J-shaped Distribution of Products

Reviews. Communications of the ACM, 52,
144-147. doi: 10.1145/1562764.1562800