Joseph Gross
CS 462 Machine Learning
Prepared for Dr. G. G. Md. Nawaz Ali
11/26/2024

Sentiment Analysis of Pharmaceutical Reviews
Project Report

The primary goal of this project was to build an effective predictive model of the numerical user rating of a pharmaceutical product based on its review. The dataset was retrieved from UC Irvine featuring 215,000 review instances[1] and consisting of drug reviews from drugs.com along with meta information such as user rating of the drug, name, date, drug name and counts of useful information(review likes/feedback). The best developed model had 76% accuracy predicting the review rating of a drug on a scale from 0-9 (dependent) given the original X input of its review (independent).

Feature engineering began with the dropping of task-irrelevant features from the dataset, (date, name, drug, etc.) leaving the review string as the only X feature and the rating as the y class. Holes were previously dropped from the dataset. The mean y was approximately 7, the standard deviation was approximately 3.27. Because the intent was to do sentiment analysis with the various words from the review as X features, the review had to be vectorized. This meant creating a vectorizer that reduced the string input to something called Term Frequency-Inverse Document Frequency. TFIDF is the resulting measure of vectorizing review paragraphs and exists as a sparse matrix where each row is an instance, each column is a term in the vocabulary, and each entry is a numerical value denoting the frequency of the word's appearance relative to the size of that review, essentially a weight. Although exploratory analysis proves difficult with so many instances within the TFIDF (X), the most common words in the matrix and the most correlated with y were both charted. 80% of the TFIDF was split to serve as the input for machine learning models which learned to predict sentiment in the form of rating on a scale from 1-10 like the initial y and tested against the remaining 20% test instances. Initial two component Principal Component Analysis in an attempt to visualize the dataset found nothing of value.

Due to the size of the dataset exceeding 215,000 instances, the CuML library was used to provide task acceleration on a CUDA capable Nvidia gpu. CuML features nearly identical syntax to Scikit Learn and shares many of its common algorithms. However, using CuML required GPU data formats; CuPY replaced numpy and CuDF replaced pandas both again with

[1] https://archive.ics.uci.edu/dataset/462/drug+review+dataset+drugs+com

very similar syntax. The increased performance comes at some cost, switching to gpu calculations has mild overhead and loses some of the flexibility from Scikit; some parameters and less common algorithms are missing.

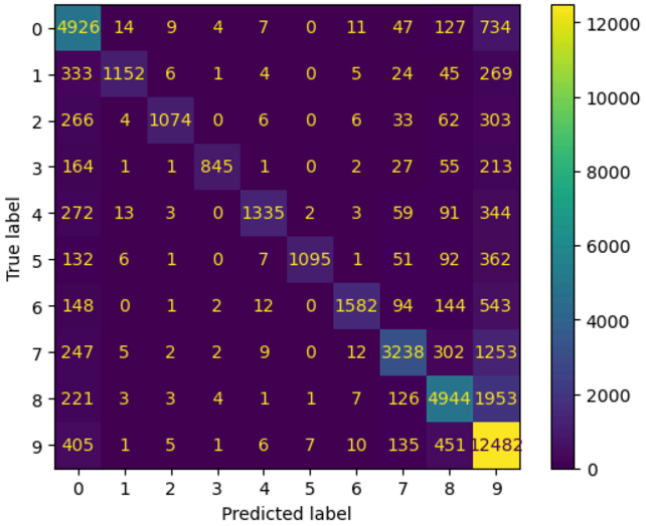
One idea reinforced over the course of this project is that it is far more efficient to begin with the most computationally efficient parameters and work up from there rather than to take shots in the dark. Computational efficiency in the vectorizer was optimized by setting the max number of feature words, filters for the minimum and maximum appearances of a word to remove those features that were irrelevant on either side as well as filtering out common English words like articles. After the vectorization, if the algorithm used was likely to be computationally intensive, data transformation had to be done to truncate the number of features using Truncated Singular Value Decomposition. This form of dimensionality reduction is designed to decompose matrices, approximating a high dimension matrix into a lower one by taking the top k(100 in this case) values and their corresponding vectors to focus on only the major components. Significantly better results were obtained increasing the number of max_features and then truncating rather than just using a lower max_features.

Models were instantiated first with minimal intense parameters like k, estimators and depth and worked upwards to find which parameters had more efficient increases in accuracy for computation time. Random forest with 60 estimators and 100 max depth proved the best algorithm with 76% accuracy, the lowest MAE (0.893 for 0-9 scale of y) and a 0.523 R^2 score. Gradient Boosting (XGboost) on a truncated dataset of bigrams (TFIDF matrix of two word combinations) proved the second most effective algorithm with 77% accuracy and an abysmal -.05 R^2 score. Logistic Regression performed on a truncated dataset of bigrams proved the third best with 47% accuracy and a 0.253 R^2 score. The 76% accuracy of the random forest, given the fact that misclassifications often still occur close to the true class, likely proves it a performative model for rating drugs based on review data. The poor R^2 scores in all models is likely reflecting the fact that TF-IDF matrices are high-dimensional and sparse, capturing word frequencies rather than the nuanced relationships needed for regression. This would result in models struggling to explain the variance in continuous targets like ratings, even if they perform well in classification.

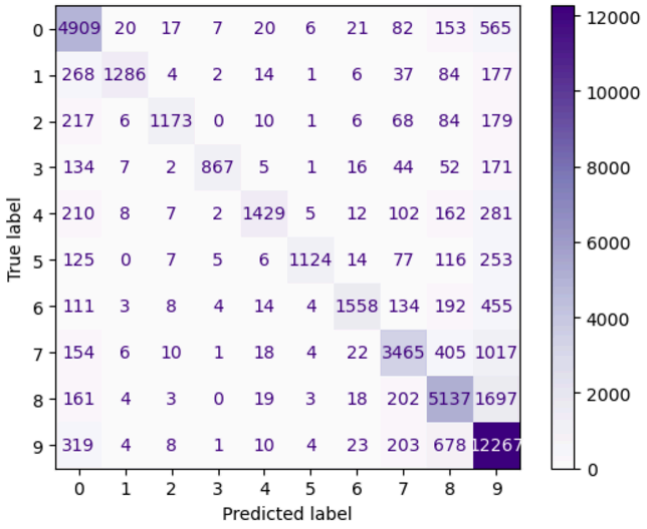
In application, these models could be used to review forum data to gather public opinion of a product. As a future portfolio project I intend to deploy it on data crawled from Reddit and the druglibs dataset also available on UC Irvine.

Top Results by Algorithm

Random Forest



XGBoost



Other Ineffective Algorithms:

1. MultinomialNB - 42% Acc.
2. KNN - 34% Acc.
3. Linear SVC - 38% Acc.

Logistic Regression