Project 3: Movie Review Sentiment Analysis

CS 598 Practical Statistical Learning, UIUC Fall 2023

- Ryan Fogle (rsfogle2@, UIN: 652628818)
- Sean Enright (seanre2@, UIN: 661791377)

Section 1: Technical Details

In this project, we perform sentiment analysis on movie review text in order to produce a binary review classification. Our dataset consists of labeled IMDB movie reviews. We have developed a model that uses the input review text to classify the review as being either positive or negative.

Our implementation uses Python.

Data Preprocessing

The data preprocessing pipeline is executed on both the train and test datasets separately.

Data Parsing The dataset contains columns for `id`, `score`, `sentiment`, and `review`. We parse the train and test data files with `pandas.read_csv()`, selecting the `review` column for prediction and the `sentiment` column and class labels. The former is only available in the train set. Observations are indexed by the `id` column. The `review` column is also stripped of any remaining HTML tags from the source material.

Vocabulary For both training and prediction, we restrict the vocabulary to a set of 874 words and phrases, listed in `myvocab.txt`. Only words or phrases found in this list are used for the prediction task. This greatly reduces the memory demand of our classifier and reduces execution time, as it significantly reduces the size of the model design matrix.

The details of our process for generating this vocabulary are found in `vocab.ipynb`.

Count Vectorizer The `CountVectorizer` class of `sklearn` is used to tokenize the input reviews and generate a matrix of n-grams for each review.

We restrict our vocabulary to consist of n-grams between lengths 1 and 4 that are found in the provided vocabulary list by providing the `vocabulary` argument in `CountVectorizer`. Any n-grams from the reviews that are not found in the vocabulary file or are found in the stop word list are ignored. Additionally, input review text is converted to lowercase before tokenization.

The result is a design matrix of size $n \times |V| + 1$, where n is the number of input reviews, $|V|$ is the vocabulary size, and an offset column is used. Each entry indicates the count of that n-gram for a given review.

TF-IDF Conversion The n-gram counts are converted into term-frequency times inverse document-frequency (TF-IDF) to weigh them within the context of the document (review), and amongst other documents. This does not change the dimensions of the design matrix, but does change the interpretation of each entry.

Prediction

The final step in the prediction pipeline is a logistic regression model. We use `sklearn`'s `LogisticRegressionCV` class to build a cross-validated logistic regression model with an L_2 (ridge) penalty and fit it with the training set. The cross-validation search space includes a range of logarithmically spaced regularization weights that have been selected to produce accurate results in the five training/test splits.

To make our final model, we refit the training data with the weight that produced the highest mean AUROC in cross-validation. This refitting happens by default with the `LogisticRegressionCV` class.

Finally, the test set goes through the same preprocessing steps above, and is input into the model to generate sentiment predictions. The probability of each test observation being a positive review is then written to file.

Section 2: Performance Metrics

In our testing our data meets the thresholds given in the report.

Evaluation Metric

AUROC is used to gauge the performance of our classifier with the generated vocabulary list.

Results

Our prediction gave a minimum AUROC of 0.9629. A table of result data is included in the appendix.

Execution Time

Generation of the vocabulary took 3 minutes and 44 seconds. This included data preprocessing, model fitting with cross validation, refitting, and selection of the vocabulary.

The classification task took a total of 1 minute and 41 seconds for all five splits. This includes data preprocessing, model training, and prediction. The mean run time per fold was 20.2 seconds.

Computer System

For the evaluation of this report, we used a Ryzen 5600X with 32GB of RAM for all 5 training/test splits.

Appendix

Table of Results

Fold	AUROC	Execution Time (s)
1	0.9641	20.3
2	0.9630	20.0
3	0.9636	20.3
4	0.9643	20.5
5	0.9630	19.8