Jake Goodman (Net ID: jakeg5) & Michael McClanahan (Net ID: mjm31)

STAT-452: Practical Statistical Learning (Online MCS-DS | Fall 2021)

Project 3: Movie Review Sentiment Analysis

**1. Team Members and Contributions**

The team for this project consisted of two members: Jake Goodman (Net ID: jakeg5) and Michael McClanahan (Net ID: mjm31). Each member contributed equally to the project. Both members performed and equal amount of research to develop the code necessary for the *mymain.R* file. Both members also contributed equally to the project report. Jake created the Rmarkdown file explaining how the vocabulary was constructed, and documented and discussed the results. Michael wrote the introduction, methods outlining model implementation and training processes, and a discussion of model interpretability.

**2. Introduction**

**3. Technical Details**

**4. Model Implementation**

As we implemented the model, one key input was already put together which was the vocabulary list. The procedure for obtaining that list can be found in the corresponding “vocab\_creation\_details” html document.

With the vocabulary list already put together, the model implementation process was relatively simple. We first read in and processed the training data by removing any of its html tags and converting the character instances to lower-case. We then created the DocumentTerm matrix while using the previously defined vocabulary list which was kept constant across all 5 splits. We then trained a Ridge regression model with this new DocumentTerm matrix and the corresponding training data.

With the model trained, we could then read in and process the test data, which was done in the same way as the training data. Similarly to the training data, we constructed a DocumentTerm matrix on the test data with the previously defined vocabulary list. With this DocumentTerm matrix from the test data, we could then make predictions on the test data from the previously trained model. When calling the predict function, *lambda.min* was used as the value for the penalty parameter. The probabilities from these predictions were then stored in a new text file so that AUC could be calculated to gauge the model performance.

This process was repeated for each of the 5 splits.

**5. Results**

*5.1 Model Accuracy*

As noted in the table below, our model’s average test WMAE was below the necessary threshold of 1610 across all folds.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test AUC and Processing Time by Split Number** | | | | | |
| Metric | 1 | 2 | 3 | 4 | 5 |
| Test AUC | 0.9601 | 0.9632 | 0.9629 | 0.9635 | 0.9627 |
| Processing Time (seconds) | 26.9400 | 28.0327 | 27.3694 | 26.8944 | 27.2591 |

*5.2 Processing Runtimes*

Using a Dell Precision 5550 laptop with an Intel Core i7 vPRO 2.70 GHz processor and 32 GB SSD memory, we saw a total runtime, for all 5 test/train splits, of 137.06 seconds. This does not include processing time for vocabulary construction. Average processing time for each split (training and prediction/evaluation) was 27.41 seconds.

**6. Discussion**

During this project, a list of movie reviews were processed in order to develop a relatively brief list of words (<= 1000) to be able to train a model in order to make predictions on the review’s sentiment based on the review itself. The approach was deemed successful if our predictions could produce an AUC value of >= 0.96 across all 5 splits of data while having a vocabulary list of 1000 or fewer words.

With the vocabulary list we constructed and the model we implemented, we were able to achieve the necessary success criterion. From a vocabulary list perspective, we had just 997 terms present and from an accuracy perspective, we had an AUC value of >= 0.96 for each split.

An interesting finding from this project is that we were able to achieve this level of accuracy from just using the first split. It’d be interesting to see how the performance differed or stayed the same depending on which split was used and/or if the whole dataset was instead used. Additionally, it was interesting to see that we didn’t lose much accuracy when trimming the vocabulary list from 2000 to 1000 as each split still yielded an AUC value of >= 0.96. It’d be interesting to see how much further we could trim the vocabulary list down while still meeting the accuracy requirements.

While we achieved the necessary model accuracy, there were still some limitations of the model. For example, we used Lasso to do some variable selection which yielded some non-essential words from being put in our vocabulary list. If we were to take the variable selection a step further, we could apply Lasso repeatedly to better understand which variables are consistently being picked and which ones might be non-essential. From there, we could further refine our vocabulary list by only choosing the ones that were consistently chosen based on some defined frequency threshold. This kind of next step will be kept in mind if any future work is done with this project.

**7. References**