COMP 551 Assignment 2 Report - Winter 2024

Group 21: Jimmy Sheng, Iris Wang, Keyu Wang

February 29, 2024

1 Abstract

In this assignment, we generated discussion around two supervised machine learning algorithms: logistic regression and multi-class regression. We first loaded the two datasets and filtered out the rare words and stop words. We also performed feature selection using simple linear regression coefficients and MI (Mutual Information) scores. Then, we implemented logistic regression and multi-class regression from scratch as required. We validated our implementations by using small perturbation and monitoring the cross-entropy. Finally, we compared these algorithms with Scikit-learn's decision tree. We used the ROC curve (Receiver Operating Characteristic Curve) for logistic regression and classification accuracy for multi-class regression. In both cases, decision tree took longer to train and had significantly lower performance than the other two algorithms.

2 Introduction

In this assignment, the two data sets we analyzed are both textual datasets. Our goal is predicting classification using frequencies of top features we selected.

1. The IMDB review dataset contains 50,000 reviews for movies, splitting into 25,000 each for training and testing data. The prediction is whether or not a review is positive (rating greater than 5) or negative (rating lower than or equal to 5), and the features are the words and their counts in the review. We first filtered out words that appeared in less than 1% and in more than 50% of all reviews. Then, we chose the 500 words with the highest absolute simple linear regression coefficients as our final features. Before executing the training process, we converted the training dataset into a sparse matrix since the data is very sparse, so using a normal matrix would be memory inefficient and extremely slow. We then implemented Logistic Regression, and used small perturbation to verify it. We also monitored the evolution of cross-entropy loss across iterations using a training and validation set to prevent over-fitting. Lastly, we performed ROC comparison with Scikit-learn's decision tree algorithm. We reached the conclusion that our implementation performs better with a higher accuracy and faster training times. We observed no sign of over-fitting since the cross-entropy of the validation set is strictly decreasing within the range of our defined max iterations.

   We also read an essay on this dataset which presents a model that combines unsupervised and supervised learning techniques to create word vectors capturing both semantic information and nuanced sentiment content, demonstrating its effectiveness over previous methods in sentiment classification tasks and introducing a substantial movie review dataset for future research.

2. The 20 news group dataset is a multi-class labelled textual dataset which includes new contents and its classification. The 5 categories we decide to work with are as suggested in the handout: *comp.graphics*, *misc.forsale*, *rec.sport.baseball*, *sci.med* and *talk.politics.guns*. First,

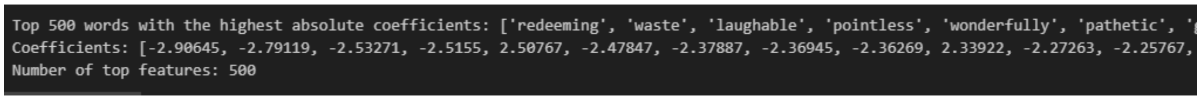
we vectorized the training and testing data using two different methods, *TfidVectorizer* and *CountVectorizer*, and filtered out words we believe are less significant to classification. Then, we computed mutual information for all the words and selected the top 80 of each class, resulting in 368 features in our final training set. Note that the final number is not 5 * 80 = 400 because there may be overlaps between classes. We then converted the prediction targets to one-hot encoding for fitting. For the training process, we implemented multi-class regression and similarly to logistic regression, we used small perturbation to verify it and monitored cross-entropy loss decrease over iterations with a validation set to prevent over-fitting. We reached the similar conclusion as in IMDB review dataset: our implementation performs better with a higher accuracy and faster training times, and we observed no sign of over-fitting within our defined maximum number of iterations.

3 Datasets

3.1 IMDB Reviews

We first loaded the already vectorized words from *labeledBow.feat*. We then excluded words that appear in less than 1% of the reviews, and in more than 50% of the reviews. The reason that filtering is necessary is because those that appear in less than 1% of the reviews are considered "rare" words, which won't affect the prediction; while those that appear in more than 50% of the reviews are too "common" to be contributing to the prediction.

Then, using Simple Linear Regression we implemented, we managed to identify the top 500 words with the highest absolute coefficients, and used those as our final features for training.



```
Top 500 words with the highest absolute coefficients: ['redeeming', 'waste', 'laughable', 'pointless', 'wonderfully', 'pathetic', '
Coefficients: [-2.90645, -2.79119, -2.53271, -2.5155, 2.50767, -2.47847, -2.37887, -2.36945, -2.36269, 2.33922, -2.27263, -2.25767,
Number of top features: 500
```

Figure 1: Top features found by their absolute coefficient

We then built sparse matrices for training and testing using these top 500 features. The reason is that our data is very sparse (with a lot of zeros). Therefore such processing of data will benefit in memory efficiency and training speed.

3.2 20 news

In order to understand the dataset better, we plotted the histogram of the distribution of all categories as well as the 5 we're looking to dive deeper into as suggested in the handout.

We tried two methods of vectorizers: *TfidVectorizer* and *CountVectorizer*. The difference between these two methods is that *TfidVectorizer* calculates frequencies while *CountVectorizer* calculates occurrences. After experimenting with both methods and comparing the accuracy while holding all other factors the same, we reached the conclusion that *TfidVectorizer* will always give a better accuracy. We believe since *TfidVectorizer* uses frequency instead of occurrences, which provides more information on the dataset.

During the filtering process, after experimenting with different thresholds for features to be included, we decided to go with a minimum frequency of 0.5% and maximum frequency of 30%. For the similar reason as above, using features within this percentages provided with us the best accuracy.

Then we proceed to selecting top features. As instructed, we made our selection using mutual information. We tried implementing our own *news_mi* function and compared it to the sklearn version. Since sklearn version does the same thing but faster, we continued using sklearn's mutual information function.

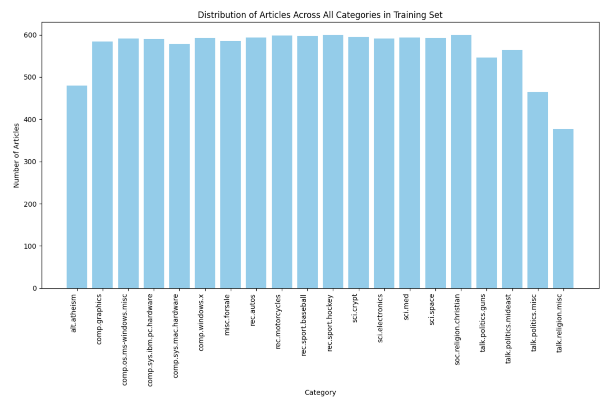


Figure 2: Class distribution of ALL categories

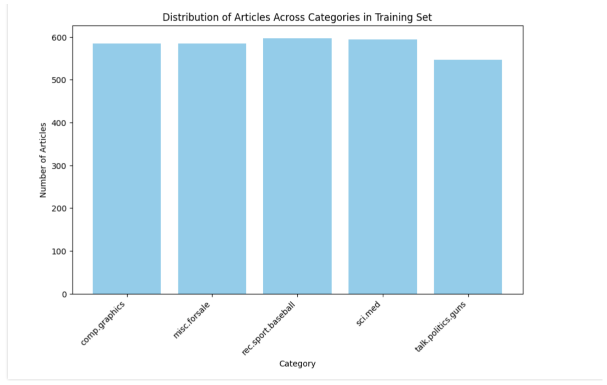


Figure 3: Class distribution of 5 categories

4 Results

4.1 Top 20 features in IMDB using Simple Linear Regression

As shown in figure 4, the 10 features to the left of the line: $coefficient = 0$ are for negative reviews while those to the right are for positive reviews. From the meaning of these words, we can clearly see that the top words for positive reviews are definitely more on the good side and on the contrary, the top words for negative reviews are clearly "not encouraging" words. Therefore we can conclude that the features selected using Simple Linear Regression does make sense for calling a movie good or bad.

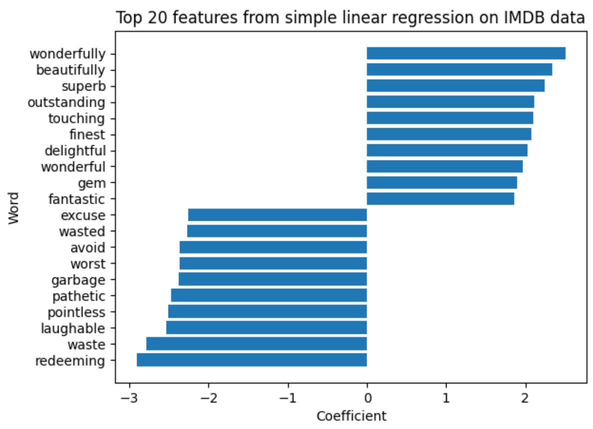


Figure 4: Top 10 positive and 10 negative features

4.2 Convergence of Regression plots

Learning rate and number of iterations on both datasets are set to reach the goal that

1. Learning is completed in reasonable time.
2. Overfitting doesn't happen.

Using logistic regression of 0.1 learning rate and 4000 iterations, we obtained the result as shown in Figure 5. We observe that as iteration increases, both training and testing loss converges. It's clear that the more iteration the better. Since the validation loss is strictly decreasing, we can conclude that no over-fitting happened.

4.3 Convergence of Regression plots

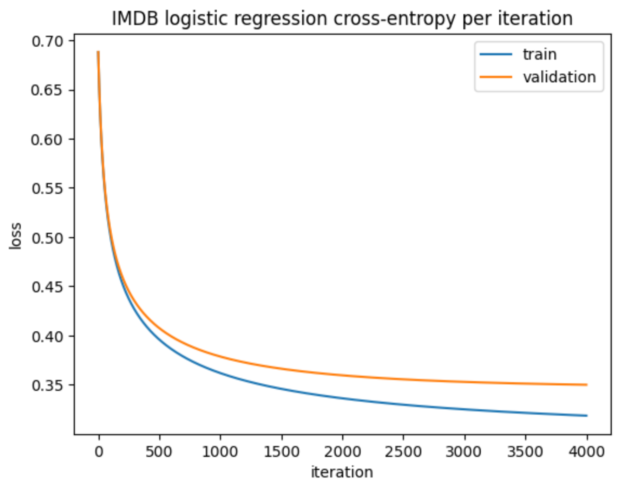


Figure 5: IMDB convergence using logistic regression

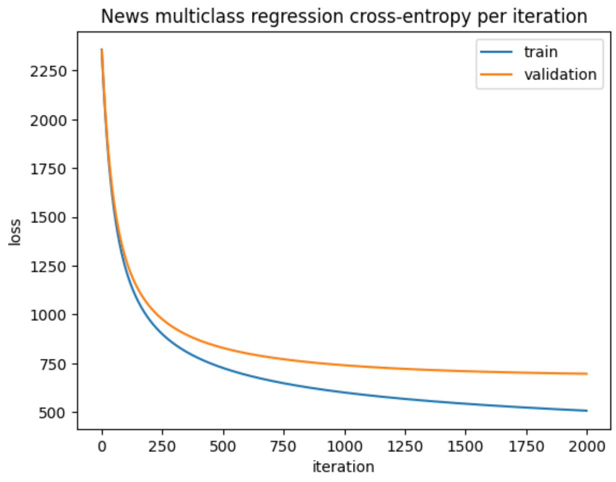


Figure 6: 20 News convergence using multi-class regression

Using multiclass regression of 0.005 learning rate and 2000 iterations, we obtained the result shown in Figure 6. Similar to above for logistic regression on IMDB dataset, both training and testing loss converges without over-fitting.

4.4 ROC of logistic regression and sk-learn decision trees on IMDB data

As ROC curve shown in Figure 7, our Logistic Regression outperforms sklearn's Decision Tree. This is showcased by our Logistic Regression has significantly larger (0.93) AUROC compared to sklearn's Decision Tree (0.72), which larger AUROC means more accurate prediction and less mistakes.

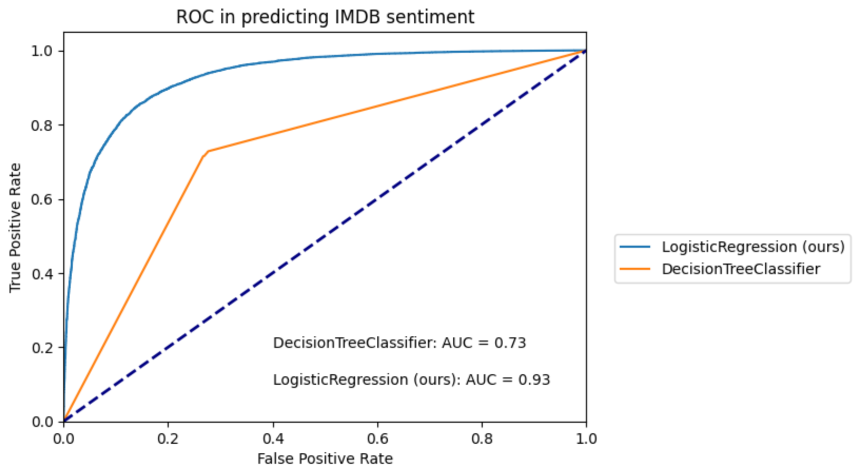


Figure 7: ROC and AUROC of Logistic Regression and sklearn's Decision Tree

4.5 AUROC of logistic regression and DT in function of training data size

As plot below shows in Figure 8, AUROC increases slightly for increasing training set size for both Logistic Regression and Decision Tree.

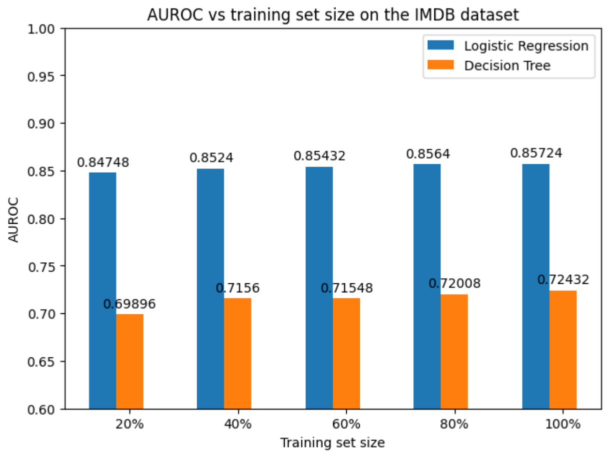


Figure 8: AUROC of Linear Regression and DT w.r.t. training set size

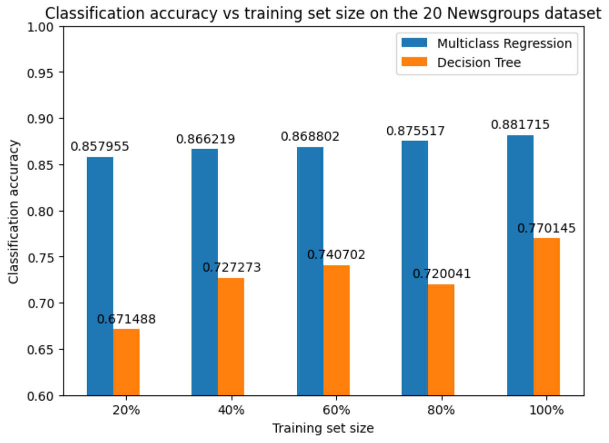


Figure 9: AUROC of Multi-class Regression and DT w.r.t. training set size

4.6 Classification accuracies of multiclass regression and DT in function of training data size

As shown in figure 9, AUROC increases sightly for increasing training set size for multiclass regression, while for Decision Tree, a small drop happened when training set size increased from 60% to 80 %.

4.7 Top 20 features from logistic regression on IMDB data

As shown in Figure 10, the positive and negative features definitely are description of "good movies" and "bad movies" respectively. Comparing the top 20 features selected from logistic regression and Simple Linear Regression, we notice that some but not all words coincide. We believe it's because logistic regression is more in-depth than simple linear regression.

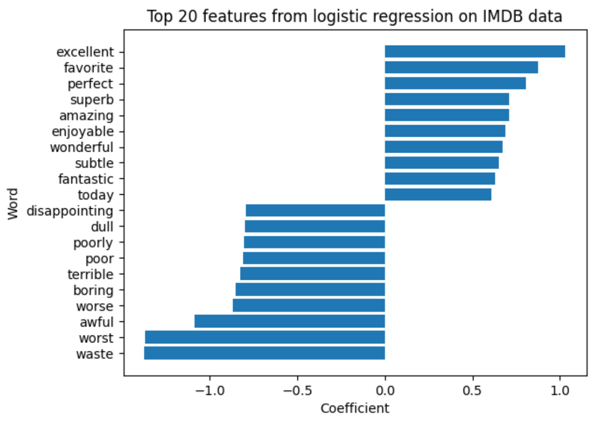


Figure 10: Top 20 features from logistic regression on IMDB data

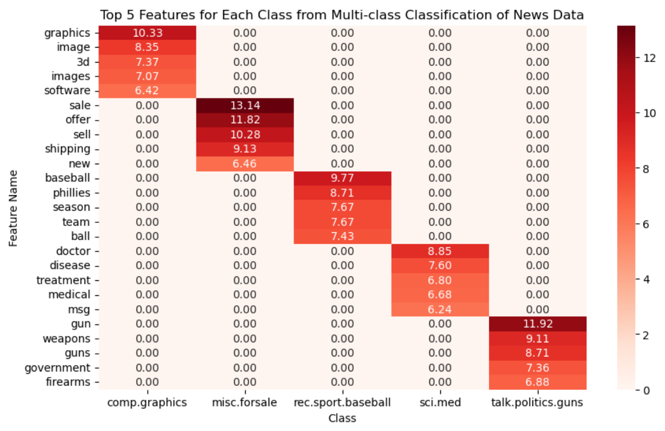


Figure 11: Heatmap for Top 5 features for each class

4.8 Top 5 features for each class from multi-class classification on News data

Looking at top 5 features for each class as shown in Figure 11, it's obvious that those with darker colour is more significant in terms of meaning with respect to their classification. Four out of five top labels in the 5 classes are words exactly in the class name itself.

5 Creativity

5.1 Use of Sparse Matrix

As mentioned above in data processing section, we initially encountered a long runtime issue with both algorithms, due to the big training sizes. We resolved it by introducing sparse matrices where it reduce computational time & storage by ignoring the ”0” entries in matrices. Great improvements was achieved.

5.2 Explore effects of different learning rates

To gain a deeper learning on how to choose hyper-parameters for gradient descent, we proceed by trying out different values for the multi-class dataset, as it is more complicated & has higher loss than the logistic regression in general. We have 2 major findings:

- When lr = 0.1, overfit occurs, as shown in Figure 12
- The classification accuracy stays similar (around 87%) for all learning rates & max iteration tried. Possible reason can be that the model might have converged to a similar solution regardless of the learning rate used. If the loss curves are similar and the model converges quickly, the resulting accuracies may not differ significantly. Also that the dataset with 5 classes may not be complex enough to benefit significantly from different learning rates.

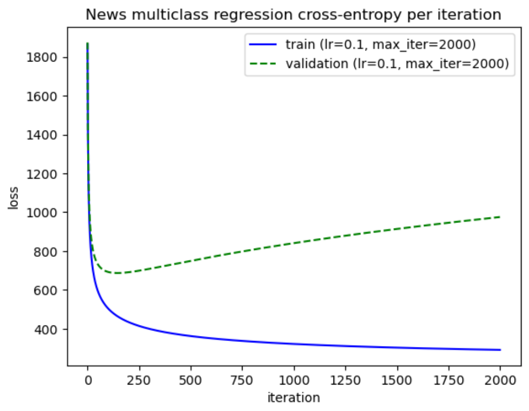


Figure 12: Model overfits when learning rate = 0.1 (too large)

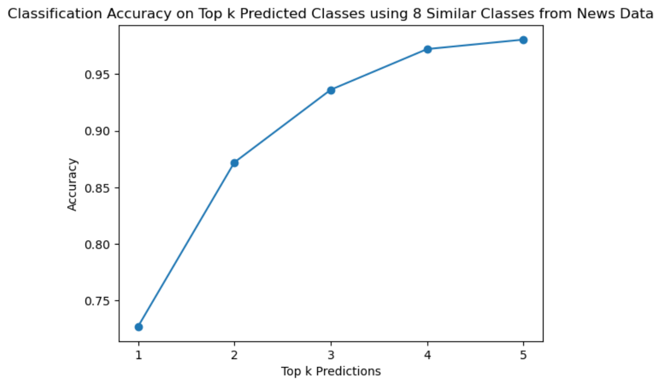


Figure 13: Accuracy increases from 73% to 87% when increasing k from 1 to 2 for similar classes

5.3 Explore more than 5 classes on News Data

Using the knowledge gained from the above section about learning rates, with our particular interest in the result from the heatmap of top features from our Multi-class Regression, and wanted to explore on the behaviours of the algorithm being used on more number of classes, and more importantly, classes with more similarities. We proceeded this tasks by 4 experiments:

- Experiment 1: Increase class size to 8;
- Experiment 2: Increase class size to 8 & purposely choose the more similar classes (namely `categories = 'comp.os.ms-windows.misc','comp.sys.ibm.pc.hardware','comp.sys.mac.hardware', 'comp.windows.x','talk.politics.guns','talk.politics.mideast','talk.politics.misc', 'talk.religion.misc']`);

- Experiment 3: Increase class size to 12;
- Experiment 4: Increase class size to 20.

A crucial finding is that actively adjusting the word filter thresholds, so that we make sure that the **there are not too little or too many features given the sample size** (since sample size increases as we use more classes), and that it does not produce over-fitting. How we formulate the training dataset is important to the model performance. For example, we decreased the lower threshold from 0.05% to 0.03% when having more classes.

When using more similar classes, for example in Experiment 2, that the model accuracy greatly improved as we use higher `k` values, which is as we expected, as shown in Figure13

Another finding to be note is from experiment 4, where we used all 20 classes, and we had a hard time finding ideal model with high accuracy AND no over-fitting. From all of our trials, even though the prediction accuracy is high (achieved 87% when `k = 3` as shown in Figure 14), it seems hard to not have over-fitting due to the our machine's computation limit. For example, we ideally should have a smaller learning rate & larger number for max iteration for doing gradient descent to converge to a small number, but it takes too long to converge to a small loss. Other reasons of over-fitting can be due to the fact that the classes have features that are too similar to each other, or that the training set is not big enough for 20 classes. Further enhancements to the feature selection process, such as regularization, might also yield better performance.

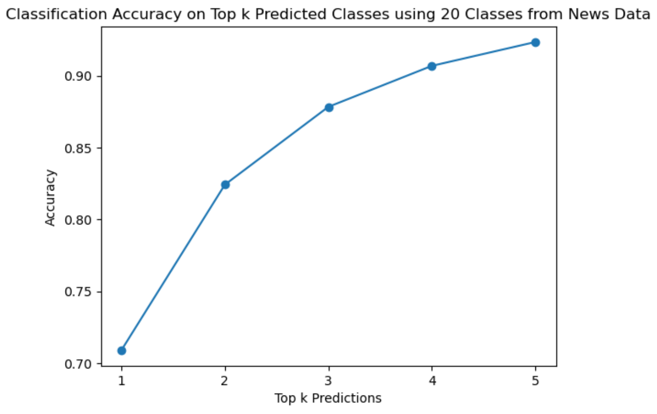


Figure 14: Accuracy of 20-class classification

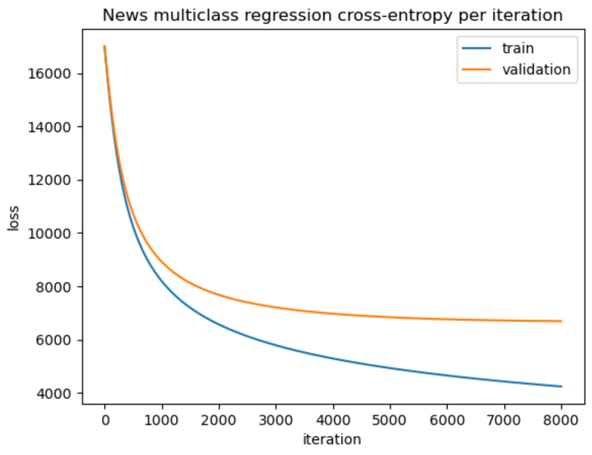


Figure 15: Over-fitting when setting lr=0.001

6 Discussion and Conclusion

6.1 Discussion

Discussion In this study, we explored the application of logistic regression and multi-class regression models to perform classification tasks on two distinct datasets: the IMDB reviews dataset and the 20 Newsgroups datase. The choice of these models was driven by their simplicity and efficiency in handling text-based data, which is inherently high-dimensional and sparse.

For the IMDB reviews dataset, logistic regression was employed to classify reviews into positive or negative sentiments. The classification was based on feature words selected through a preprocessing step, which included filtering, stop words removal, and transforming into sparse matrix. This approach allowed the model to focus on the most relevant features, improving its performance over training and testing

Similarly, for the 20 Newsgroups dataset, we performed multi-class classification. Given the more complex nature of this task—classifying documents into one of twenty different categories—the models were adjusted to handle 5 classes to out favor. Despite the increased complexity, our approach once again outperformed the decision tree baseline. This success can be attributed to the models' ability to manage the high feature space characteristic of text data, where decision trees might suffer from over fitting or fail to capture the subtleties in the data.

Throughout the study, we also explored the impact of different numbers of iterations on model performance. Monitoring small perturbations and cross-entropy allowed us to observe how models react to changes in their training environment, offering insights into their stability and robustness. This aspect of the study highlighted the importance of iteration tuning in achieving optimal model performance, especially in the context of dataset specifics and model complexity.

6.2 Conclusion

The findings of this study reinforce the effectiveness of linear and logistic regression models in handling text classification tasks, outperforming decision tree classifiers in both binary and multi-class settings. The superior performance of these models can be largely attributed to their ability to deal with high-dimensional and sparse data, which is typical in text analytics. Furthermore, the study's exploration of iteration numbers and the monitoring of model responses to small perturbations and cross-entropy provided valuable insights into the models' behavior and stability, underscoring the importance of careful hyperparameter tuning.

For further explorations, we would consider further improvements in feature selection, which could be achieved by exploring more sophisticated engineering techniques, such as word embeddings (Word2Vec, GloVe) or advanced NLP models (BERT, GPT) to capture semantic relationships between words more effectively. We could also explore more in hyperparameter tuning to ensure an even more optimal testing result.

7 Statement of Contributions

All 3 of us attempted task 1. Jimmy and Iris compared their result of IMDB word filtering results. Jimmy finished 20news vectorization and processing. Jimmy introduced sparse matrix to Iris and Keyu. Jimmy then wrote task 2. Keyu and Jimmy wrote task 3. Iris wrote the report, along with some plots generations. Keyu worked on the creativity part and we all read and edited the report.

8 Citations

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*