COMP 551 Assignment 2

Daphne Hegedus, Kelly Rombough, Karla Gonzalez

March 13, 2020

1 Abstract

In this project, our task was to categorize text data from two different datasets. The first dataset, which we will refer to as IMDB, consists of media reviews to be classified as positive or negative. The second dataset, which we will refer to as Newsgroups, consists of newsgroups posts to be classified according to which of 20 newsgroups it was posted in. To do this, we used various models from the scikit-learn library and used grid search with cross validation to tune the hyper-parameters. With this approach, we obtained the highest accuracy (88%) on the IMDB dataset using SVM, Logistic Regression, and Linear SVC. On the Newsgroups dataset, we obtained the highest accuracy (70%) using SVM, Naive Bayes, and Linear SVC.

2 Introduction

Text classification is a popular topic in the field of machine learning, and it has many practical applications such as search engine optimization, gauging consumer reactions from reviews, and moderating the content of posts on social media. Given its prevalence, there are many techniques used for text classification today, each with their own benefits and drawbacks. In this project, text classification is performed on two datasets using seven different machine learning models: Logistic Regression, Decision Trees, Support Vector Machines, Linear Support Vector Classification, AdaBoost, and Random Forests. Grid search was used to determine the optimal parameters for each model, then the models were evaluated based on the best overall accuracy for the aforementioned datasets. The test data was left out until the models were trained, at which point those models were used to predict the class of the testing instances. The accuracies discussed in this report are the accuracies found on the testing data unless otherwise stated. In order to run our code, you just need to run LoadData.py which will complete the entire train-test-validation process.

3 Discussion of Related Work

A 2001 paper by Rennie and Rifkin compares the accuracy of Naive Bayes and SVM on the 20 Newsgroups dataset. They found that when the training set was small, SVM significantly outperformed Naive Bayes, but with larger training sets their accuracies were almost equal. Our 20 Newsgroups findings agree with this, as we used a large training set of 11,314 samples (60% of the data) and got 70% accuracy with both Naive Bayes and SVM after optimization.

Also, a 2005 paper written by Zelikovitz and Hirsh explore how background text can hugely improve the text classification performance by treating background text as unlabeled data and using techniques such as Expectation Maximization and Naive Bayes to iterate through the labels. This is particularly interesting and relevant to the project due to the amount of information given in text-based datasets. Adding this parameter while taking into consideration particular feature selection and extraction may lead to high classification accuracy.

4 Datasets

IMDB

The IMDB dataset has 50,000 review samples and two categories, positive and negative. The reviews are evenly split between positive and negative, and the train-test split is 50/50. All IMDB reviews include a rating from 1 to 10, so in this dataset, negative reviews have a rating of 1,2,3, or 4 and positive reviews have a rating of 7,8,9, or 10. Since no reviews with ratings 5 or 6 were included, we predicted that it would be clear to the model as to how to differentiate negative from positive. This observation may be the reason why such a high accuracy was achieved. In order to check confirm the theory, the instances were classified based on the number rating. This classification was used to evaluate whether the more extreme ratings (ie. 1 and 10) were more accurately classified. The data was loaded by simply importing the train and test folders into dataframes and labelling the x and y columns appropriately. No data cleaning was required since there are no missing values or unnecessary features.

Newsgroups

The Newsgroups dataset has 18,846 post samples and 20 categories, with a 60/40 train-test split. Within the 20 categories, there are clusters of similar categories including five about computers, four about science, and three about religion. It was expected that categories within the same cluster would frequently get confused with each other, whereas categories in different clusters would rarely get confused. The data was loaded using a scikit-learn function. The only data preprocessing required was document/input parsing which was included in the default train subset.

5 Approach

GridSearchCV was used to tune hyper-parameters for each model instead of RandomizedSearch. While the randomized option can speed up fitting the model without much loss in accuracy, time was not a constraint and so the full permutation of the parameter grid was run. This project had the approach of pipelining our data preparation, which consisted of a CountVectorizer to obtain the number of occurrences and a Transformer to turn those counts into frequencies. The two datasets were classified using seven different models from the scikit-learn library. The following section describes the aforementioned models and their parameters.

1 **Logistic Regression** is a linear classification model that has a probabilistic interpretation. It classifies instances as class 0 or 1 based on a threshold applied to a sigmoid "squashing" function. It is widely used to achieve a model with low variance that is simple yet efficient. The hyper-parameters tuned in this project are the regularization strength and the solver ie. the function used to update the weights at each iteration.

2 **Naive Bayes**: We decided to include Naive Bayes in this project as a benchmark, as it is commonly used as a benchmark in scikit-learn tutorials. This provided us with a good starting point to make sure our models were working as expected in comparison to implementations in scikit-learn documentation. The hyper-parameters that we tuned are the strength, the loss function used, and whether or not to use idf (as described in the tutorial).

3 **Decision Trees** are extremely interpretable methods that do not need data normalization. However, they can overfit very easily since their depth is not limited, which is the case with both of our data sets. They commonly serve as the weak learners in boosting methods to improve accuracy. The hyper-parameters tuned here are the measurement of disorder in the nodes (gini index vs. entropy), and the method used to choose the split at each node (best vs. random).

4 **Random Forests** is a boosting method that involves random feature sub-sampling which reduces the dependence between decision trees. This can be seen in the fact that the boosting methods (AdaBoost and Random Forests) both outperformed the Decision Trees on our datasets. The hyper-parameters

that we tuned are the measurement of disorder in the tree nodes (gini index vs. entropy), and the number of trees in the forest.

5 **Linear Support Vector Classification** is a more flexible and scalable implementation of SVM, especially for datasets with more than a few thousand instances. Because our data sets are about this size, we would expect their performance to be comparatively similar as the underlying mechanism is the same. This holds true for our results. The hyper-parameters that we tuned are the regularization strength, the loss function used, and the maximum number of iterations.

6 **AdaBoost** takes weak learners (in our case Decision Tree Classifiers) and sequentially combines them to create a stronger learner. The hyper-parameters that we tuned are the number of weak learners allowed and the max depth of the trees.

7 **Support Vector Machine / Perceptron**: We decided to include this model in order to see which loss function would perform best - perceptron loss (perceptron) or hinge loss (SVM). In both datasets we found that the SVM had better performance, most likely due to the addition of regularization in SVM hinge loss. The other hyper-parameter that we tuned was the regularization strength.

6 Results

After using grid search to optimize parameters for each of our seven chosen models, we computed the accuracy score of these models on our test data, which is given by the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The overall accuracy and best parameters for each model with each dataset are displayed in a table in the Appendix. Important findings are summarized below. The plot below shows these scores, which range from 41% to 70% for Newsgroups and from 71% to 88% for IMDB. The Linear SVC models and SVM models had comparable performance on each data set. Across the board, Logistic Regression, Linear SVC / SVM had the highest accuracy. On the IMDB data set, Logistic Regression and SVM had the same accuracy - 88%. It can also be seen that AdaBoost and Random Forests outperformed Decision Trees, probably due to Decision Trees' tendency to overfit on the training data.

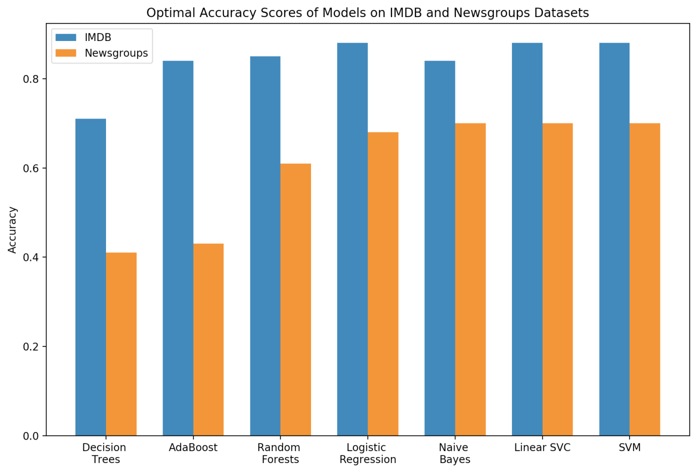

Figure 1: Bar graphs showing the optimal accuracy scores for both datasets for the seven models that we trained. These are the scores obtained from running the model on the test sets after tuning hyper-parameters and doing cross validation.

We also plotted confusion matrices for both datasets. We found the matrices for the Newsgroups dataset to be the most interesting since there were 20 categories and therefore more potential confusion. As you can see in the figure below, there was relatively little confusion with the logistic regression model, which is expected as it had one of the highest accuracies. However, on the right we see much higher rates of confusion with decision trees, which was our lowest performing model. For example, we see that label 9 and label 10 frequently get confused with each other. These labels correspond to 'rec.sport.baseball' and 'rec.sport.hockey'. Since they share the same parent category of 'sport', it makes intuitive sense that a poorly tuned model would mix them up. In the same fashion, we see a faint square in the top left corner of the decision tree matrix. This indicated that categories 1 through 5 are often confused, which correspond to all of the subcategories that fall under 'computers'.

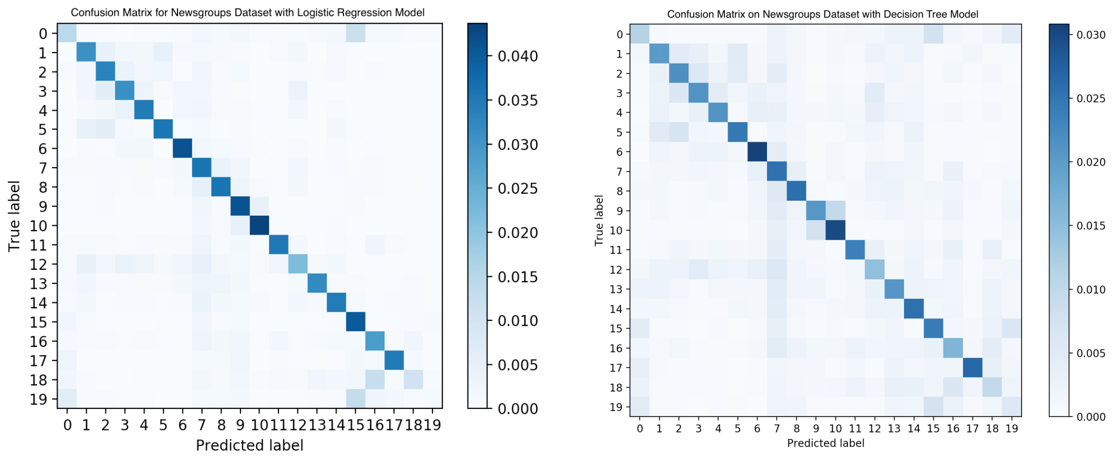

Figure 2: Left: Confusion matrix for Newsgroups dataset with logistic regression model. Right: Confusion matrix for Newsgroups dataset with decision tree model. Note that there is more confusion with the decision tree model, which had much lower accuracy. Note that the scale on the right has a lower maximum, which is to say that there is more misclassification than the left plot.

Below is a plot of the training score and the Cross Validation score dependent on the training size for the Newsgroup in both the decision tree and the SVM model. As expected, it can be seen that the decision tree model seems to have overfit to the training data, as the CV score is relatively low and the training score stays constant at nearly 100%. On the other hand, the SVM model's training score does decrease slightly and the CV score increases at a significant rate.

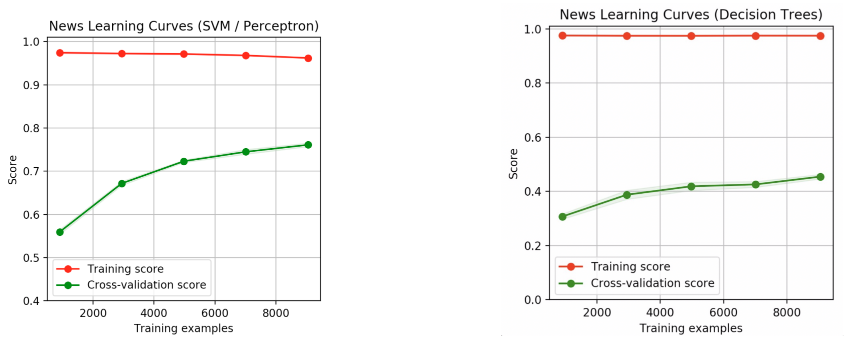

Figure 3: Left: Learning Curve for SVM model on the News data set. Right: Learning Curve for SVM model on the News data set. Note that the CV score of the SVM was much higher (≈75%) than the Decision Tree model (≈42%). It can also be seen that the Decision tree training score never dropped below 98%, which is indicative of over fitting.

The plot below displays the F1 scores for the IMDB dataset when classified by their number rating from 1-10, excluding 5 and 6. The most substantial finding from this experiment was that the more "extreme" ratings (ie. 1 and 10) are correctly classified more often. This makes sense as these instances most likely had more expressive words, either positive or negative respectively with less ambiguity between these sentiments.

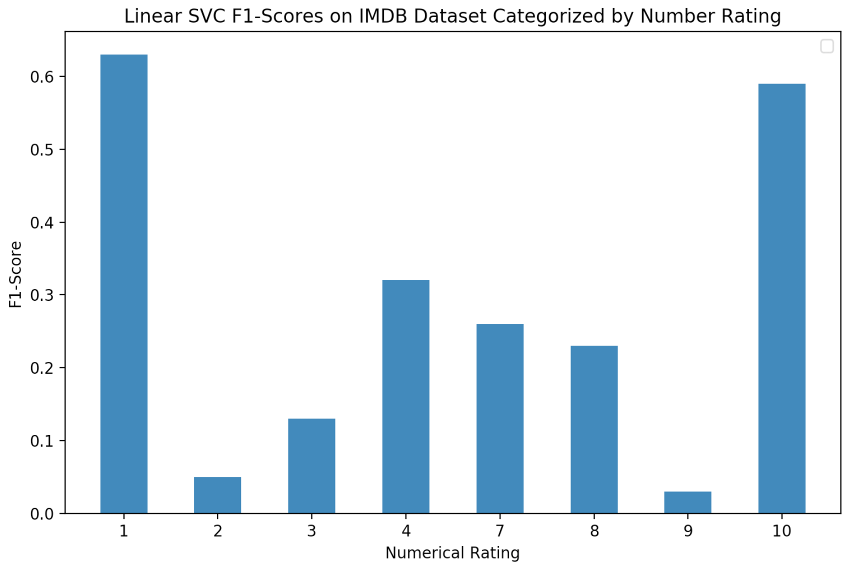

Figure 4: The F1 breakdown of scores for the IMDB training data set classified by the number rating.

7 Discussion & Conclusion

Overall, the Linear SVC and Logistic Regression models show the best accuracy out of the models on both data sets. One interesting thing to note is that the AdaBoost and Random Forests models outperform the Decision Tree model on both datasets. This is to be expected as these boosting methods build on top of the Decision Tree weak learners to build a combined strong learner. Also, Linear SVC and SVM show comparable accuracy. This makes sense as they have the same underlying architecture, but Linear SVC is known to be scalable to larger data sets of greater than a few thousand instances. In our case, the dataset aren't extremely large, so both models perform similarly. On last finding was that the Logistic and SVM models had the same accuracy (88%) on the Newsgroup. However, the Logisitic model took less time to fit a single model, which allowed us to test more parameters. This can be explained by the fact that Logistic Regression has a running time of $O(n)$ while SVM's running time is $O(n^2)$ (Joachims).

For future work, it would be beneficial to run some experiments varying aspects of the data to determine which models are sensitive to certain variables. For example, we could add random noise to the samples to test the robustness of each model. We could also truncate the samples to see which models perform best on short text fragments. Lastly, it would also be interesting to try to reduce the number of parameters that we tune in GridSearch, in order to speed up the running time. For example, the regularization strength for the Logistic Regression model does not appear to affect the accuracy so we could stop tuning that parameter altogether.

8 Statement of Contributions

Kelly: Authored writeup. Wrote script to generate figures. Analyzed results to draw conclusions.
Daphne: Wrote the Gridsearch and CV script to generate the best parameters and run predict on the test data (FitAndPredict). Implemented the Pipeline to vectorize and transform the data (LoadData). Helped with writeup.
Karla: Wrote and edited script to generate figures. Wrote discussion for related papers. Edited and contributed to the write up.

9 Works Cited

1. Joachims, Thorsten. Training Linear SVMs in Linear Time. Training Linear SVMs in Linear Time, www.cs.cornell.edu/people/tj/publications/joachims_06a.pdf. Accessed 12 March 2020.

2. Rennie, Jason, and Ryan Rifkin. Improving Multiclass Text Classification with the Support Vector Machine. Improving Multiclass Text Classification with the Support Vector Machine, dspace.mit.edu/bitstream/handle/1721.1/7241/AIM-2001-026.pdf?sequence=2. Accessed 12 March 2020.

3. Zelikovitz and Hersh (2005), Improving Text Classification Using EM with Background Text. Accessed March 12 2020

10 Appendix

News Accuracy and Best Parameters

MODEL	PARAMETERS	ACCURACY
Logistic Regression	C=1500, solver='liblinear'	0.68
Naive Bayes	alpha=0.01, norm='l2', use_idf=True	0.70
Linear SVC	C=1, loss='hinge', max_iter=100	0.70
SVM / Perceptron	alpha=0.0001, loss='hinge'	0.70
Decision Trees	criterion='gini', splitter='random'	0.41
AdaBoost	max_depth=2, n_estimators=50	0.43
Random Forests	criterion='gini', n_estimators=200	0.61

IMDB Accuracy and Best Parameters

MODEL	PARAMETERS	ACCURACY
Logistic Regression	C=10, solver='liblinear'	0.88
Naive Bayes	alpha=0.1, norm='l1', use_idf=True	0.84
Linear SVC	C=1, loss='hinge', max_iter=100	0.88
SVM / Perceptron	alpha=0.0001, loss='hinge'	0.88
Decision Trees	criterion='gini', splitter='random'	0.71
AdaBoost	max_depth=2, n_estimators=75	0.84
Random Forests	criterion='gini', n_estimators=200	0.85