IRENE CHANG - 1336784

PROJECT 02 REPORT

Abstract

In this project I'm going to explore different ways to preprocess text data from the reviews taken from different websites and build a model to predict the sentiment of these reviews. The report comes in 3 main parts. The first part discusses my approach to clean this data, and transform the texts into a useful form for model training. Parts 2, 3, and 4 are where I build the classification models using different algorithms (Logistic Regression, Neural Network and SVM) and carry out hyperparameter tuning. Lastly, in parts 5 and 6, I compare the performance of the models I have just built and deploy this model to produce the final results for this project.

1. Generating BoW features

The purpose of this task is to find the most important features, or words, for the labeling of the sentiments. To do so, I first cleaned the text data, then used TF-IDF to select the important features.

For the preprocessing of the data, I started off by removing the special characters and numbers, which apparently don't contribute to the sentiment of the written sentence in any significant way. Removing these characters (question marks (`?`), colons (`:`), exclamation marks (`!`), periods (`.`), commas (`,`), semicolons (`;`) and parentheses (`()`)) can be done by putting these characters in a list and using `str.replace()` functions to loop through and replace each of them with an empty string. Next, I removed the numbers from the texts, also by replacing them with the empty string, but to do so via regular expression (regex), I applied the function `sub()` from the package `re` on each text as followed: `re.sub(r'[0-9]', '', text)` where `r'[0-9]'` is the regex for the digits from 0 to 9. I saved the result of every transformation in a separate column called text_1, text_2, etc, so that I can redo any mistake conveniently.

From there, I proceeded to lower the cases of the words in the texts using the method `str.lower()` on the column, otherwise the algorithm will treat “word" and “Word" as 2 different features. Next comes one of the most crucial steps in text data cleaning, stemming and lemmatization. Stemming removes the last few characters of the words only, hence it doesn't perform well and may result in incorrect spellings and meanings with irregular verbs. Lemmatization, however, takes the context of the word into consideration and tries to convert it to its correct base form. In this case, I will use lemmatization to process the data since it’s more helpful. I chose to do lemmatization before checking the spellings of the words because there are still a few cases where lemmatization results in misspellings, or can't get rid of original misspellings.

I used the `lemmatize()` function from the `WordNetLemmatizer()` object of the `ntlk.stem` module. This function benefits from specifying the part of speech (POS) each word belongs to. Hence, I made a helper function that takes in a word and returns what POS it is. For each word, I used the `pos_tag()` function of `ntlk` package to retrieve its POS

Tag, the initial character of which tells us whether the word is an adjective (J), a noun (N), a verb (V) or an adverb (R). This process of lemmatization requires that we break each text down to a list of words, handle each word individually, then rejoin the words in the list using `" ".join(lemmatized_list)`. The next thing to do is use the `SpellChecker` object from `spellchecker` library to correct any misspellings in the texts, by applying the method `SpellChecker.correction(w)`. Just as the previous step, we also need to split the text into separate words for this task, then parse them together after we've done the job.

For stopwords, I chose to create 2 datasets: one with and another without stopwords, because there are words in the english stopword list from `ntlk` package that can have meaningful contribution to this classification task, such as “no", “not", “beyond", “below", etc. Yet, there are still words that may not have any significance at all (potentially “the", “and", etc). To examine the effect of stopwords on this dataset, I decided to create a set with all the words, and the one that leaves out only the words that I think are unnecessary—the, and, or, etc— and examine different models' performance on both of them. For the latter dataset, I constructed its stop-words by first going through the stop-words of the ntlk package and removing some words that I think may affect the sentiment classification, as mentioned above.

After cleaning the texts, I created 2 `TfidfVectorizer()` objects to form the bag of words for each of the 2 datasets. I chose to use tf-idf score instead of frequency score because I also experiment with the dataset that still includes a few stopwords, which can appear a lot in the text but not necessarily have significance in predicting its sentiment. Tf-idf score, however, gives strong indication of which words are important in the context of all the data, which is more suitable with the task we're handling. I also chose to use unigrams (one-words) instead of bigrams (pairs of words) because after cleaning the text, some entries are very short, some only have 1 or 2 words left. Having such few words can also mean the bigrams have a higher chance of containing words (especially nouns) that are irrelevant to the sentiment analysis.

I set the min_df parameter to be 2 because I want to exclude all the words that have the frequency of less than 2 (too rare) in the whole data. For the dataset with stop-words, I included the stop-words list in the `stop_words` parameter. I fit-transform the ‘text’ column, after which I can extract the list of words as features—my BoW. Using the `todense()` function on the sparse matrix just created, I can make a dataframe of the same length as my original data, and the columns being the words in my BoW.

2. Logistic Regression model

The process of this step includes 3 steps. Firstly, I trained a baseline model with all the default parameters. I will split the data into train and test sets (train set is roughly two-thirds of the data), train the models and obtain the train and test accuracy scores for these baseline models. Next, I do a random grid search to find the best set of parameters. Finally, I examine the scores when varying the values of some parameters in the grid search around the area that yield the best results. I chose to do a randomized grid search instead of grid search because with randomized grid search you don't have to try all combinations of parameters, yet as research has shown, you can still achieve equally good performance. I

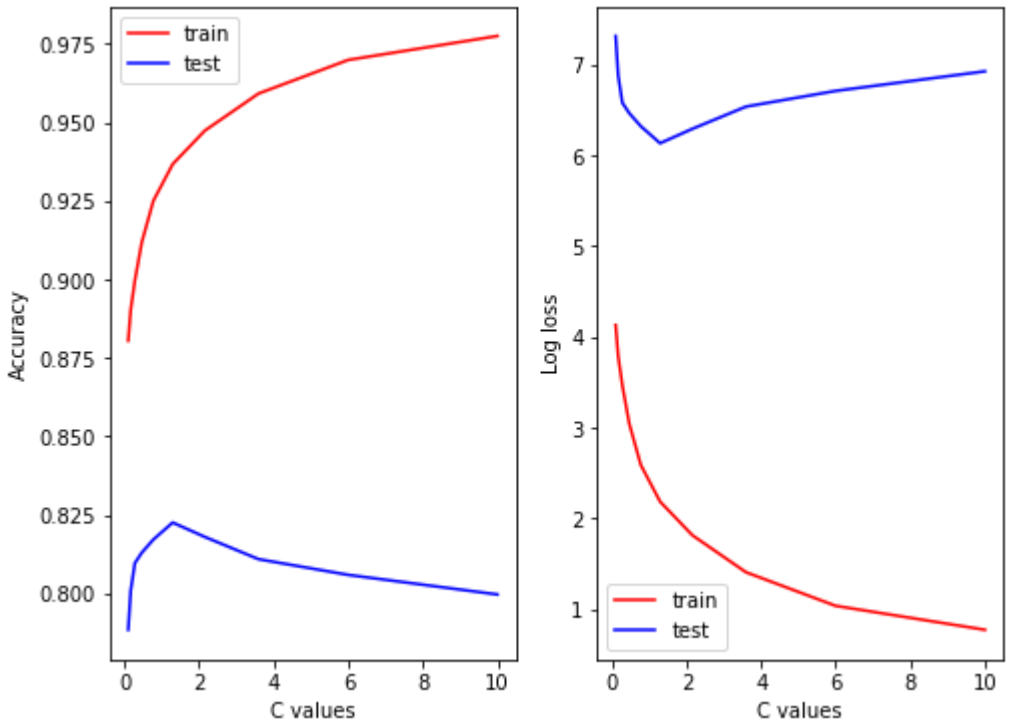
will also examine the performance of these models on the dataset with and without stop-words separately.

For the Logistic Regression model, I chose max iterations and the regularization term C as my 2 hyperparameters to tune. The baseline model that I tried on the dataset with stop-words included has similar training accuracy of 0.93, but the model without stop-words performs better on the dataset (79% accuracy) compared to the model with stop-words (77%). The C values I examined range from 10^(-3) to 10^3 because I want to explore a wide range of values with randomized grid search before narrowing the scope of each parameter. For max iterations, I examined the range from 10 and limit it at 350 because I don't want the algorithm to get more costly/time-consuming than around 300 iterations. Running randomized grid search with 3-fold crossvalidation (`RandomizedSearchCV` from sklearn, setting n_iter to 50 and cv to 3) on both dataset confirms that the model fitted on the dataset without stop-words performs better (with 80% true accuracy using the best set of parameters) compared to the model with stop-words (78% accuracy).

Thus, I go with the better performing model and use visualizations to examine how the change in C value and in max iteration value around the optimal area affect the change in accuracy score and log loss. Whereas I use train test split for the baseline model fitting and random grid search, for parameter tuning, I'm going to use the full dataset since I'm going to manually build a cross validation helper function that is able to record the accuracy as well as log loss of each fold. This function is not supported by the logistic or any other model in sklearn, but it will help us explore the metrics across the folds. For each of these C values, I used `KFold` from sklearn to shuffle and split the full train data into 4 folds, then built Logistic Regression models on each of the splits. I used the `score` and `log_loss` function to calculate the accuracy and log loss on the train and test folds and stored them in a list. After finishing fitting on all folds, I averaged the testing and training accuracy scores and log loss across different folds, and stored them in the final lists that will be used to visualize the data. The mean and standard deviation for different metrics across different alpha values are printed below:

```
C value: 0.100
Training accuracy scores for 4 folds are:  0.888 0.883 0.876 0.875 Mean:  0.880 Std:  0.005
Testing accuracy scores for 4 folds are:   0.785 0.800 0.813 0.755 Mean:  0.788 Std:  0.022
Training log loss for 4 folds are: 3.876 4.030 4.298 4.317 Mean:  4.130 Std:  0.186
Testing log loss for 4 folds are:  7.426 6.908 6.447 8.462 Mean:  7.311 Std:  0.749

C value: 0.167
Training accuracy scores for 4 folds are:  0.895 0.891 0.889 0.886 Mean:  0.890 Std:  0.003
Testing accuracy scores for 4 folds are:   0.792 0.808 0.822 0.782 Mean:  0.801 Std:  0.015
Training log loss for 4 folds are:  3.627 3.761 3.818 3.953 Mean:  3.790 Std:  0.117
Testing log loss for 4 folds are:   7.196 6.620 6.159 7.541 Mean:  6.879 Std:  0.530

C value: 0.278
Training accuracy scores for 4 folds are:  0.902 0.901 0.900 0.896 Mean:  0.900 Std:  0.002
Testing accuracy scores for 4 folds are:   0.803 0.820 0.823 0.792 Mean:  0.810 Std:  0.013
Training log loss for 4 folds are:  3.377 3.435 3.454 3.607 Mean:  3.468 Std:  0.085
Testing log loss for 4 folds are:   6.793 6.217 6.102 7.196 Mean:  6.577 Std:  0.443

C value: 0.464
Training accuracy scores for 4 folds are:  0.915 0.913 0.913 0.907 Mean:  0.912 Std:  0.003
Testing accuracy scores for 4 folds are:   0.807 0.825 0.820 0.800 Mean:  0.813 Std:  0.010
Training log loss for 4 folds are:  2.936 3.013 3.013 3.224 Mean:  3.046 Std:  0.107
Testing log loss for 4 folds are:   6.678 6.044 6.217 6.908 Mean:  6.462 Std:  0.346
```

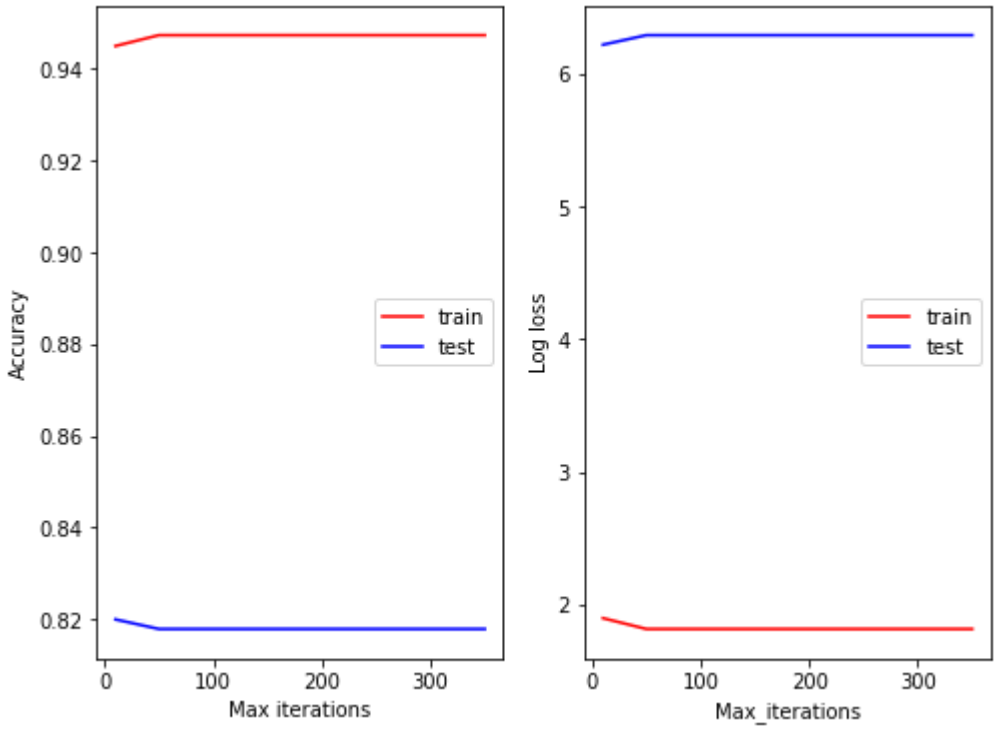
```
C value: 0.774
Training accuracy scores for 4 folds are:  0.928 0.925 0.924 0.922 Mean:  0.925 Std:  0.002
Testing accuracy scores for 4 folds are:   0.812 0.825 0.830 0.802 Mean:  0.817 Std:  0.011
Training log loss for 4 folds are:  2.475 2.590 2.610 2.686 Mean:  2.590 Std:  0.076
Testing log loss for 4 folds are:   6.505 6.044 5.872 6.850 Mean:  6.318 Std:  0.385

C value: 1.292
Training accuracy scores for 4 folds are:  0.936 0.935 0.939 0.937 Mean:  0.937 Std:  0.001
Testing accuracy scores for 4 folds are:   0.812 0.827 0.837 0.815 Mean:  0.822 Std:  0.010
Training log loss for 4 folds are:  2.207 2.245 2.111 2.187 Mean:  2.187 Std:  0.049
Testing log loss for 4 folds are:   6.505 5.987 5.641 6.390 Mean:  6.131 Std:  0.342

C value: 2.154
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450

C value: 3.594
Training accuracy scores for 4 folds are:  0.957 0.961 0.961 0.959 Mean:  0.959 Std:  0.002
Testing accuracy scores for 4 folds are:   0.803 0.818 0.830 0.792 Mean:  0.811 Std:  0.015
Training log loss for 4 folds are:  1.497 1.362 1.362 1.420 Mean:  1.410 Std:  0.055
Testing log loss for 4 folds are:   6.793 6.275 5.872 7.196 Mean:  6.534 Std:  0.503

C value: 5.995
Training accuracy scores for 4 folds are:  0.969 0.969 0.969 0.972 Mean:  0.970 Std:  0.001
Testing accuracy scores for 4 folds are:   0.798 0.815 0.818 0.792 Mean:  0.806 Std:  0.011
Training log loss for 4 folds are:  1.055 1.055 1.075 0.979 Mean:  1.041 Std:  0.037
Testing log loss for 4 folds are:   6.965 6.390 6.275 7.196 Mean:  6.706 Std:  0.385

C value: 10.000
Training accuracy scores for 4 folds are:  0.978 0.978 0.974 0.979 Mean:  0.977 Std:  0.002
Testing accuracy scores for 4 folds are:   0.797 0.813 0.812 0.777 Mean:  0.800 Std:  0.015
Training log loss for 4 folds are:  0.768 0.748 0.883 0.710 Mean:  0.777 Std:  0.064
Testing log loss for 4 folds are:   7.023 6.447 6.505 7.714 Mean:  6.922 Std:  0.509
```



The graph shows that the model overfits the data from the very beginning. This is due to the fact that the size of the BoW is greater than the number of instances in the data. However,

reducing the numbers of words used for classification, by increasing `min_df` or using a threshold on tfidf scores, makes the model underfit. This is a trade-off I made for better prediction on unseen data. Using the best C value (2.154), we examine the max_iter parameter. Each of these `max_iter` values is used in a 4-fold cross-validated LogisticRegression model:

```
Max iteration: 10
Training accuracy scores for 4 folds are:  0.946 0.942 0.948 0.945 Mean:  0.945 Std:  0.002
Testing accuracy scores for 4 folds are:   0.808 0.830 0.840 0.802 Mean:  0.820 Std:  0.016
Training log loss for 4 folds are:  1.880 2.015 1.804 1.900 Mean:  1.900 Std:  0.076
Testing log loss for 4 folds are:   6.620 5.872 5.526 6.850 Mean:  6.217 Std:  0.538

Max iteration: 50
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450

Max iteration: 100
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450

Max iteration: 150
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450

Max iteration: 200
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450

Max iteration: 250
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450

Max iteration: 300
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450

Max iteration: 350
Training accuracy scores for 4 folds are:  0.947 0.948 0.948 0.947 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.828 0.833 0.805 Mean:  0.818 Std:  0.013
Training log loss for 4 folds are:  1.842 1.804 1.785 1.842 Mean:  1.818 Std:  0.025
Testing log loss for 4 folds are:   6.735 5.929 5.757 6.735 Mean:  6.289 Std:  0.450
```



We can see that the model converges early on and reaches its optimal value, hence we see a flat trend in the above graphs. With the optimal C value, it appears that increasing the max iteration value in the Logistic Regression model doesn't improve the performance. Thus, this shows that we can go with as few as 50 iterations and still achieve equally good results, which is great as far as cost is concerned.

3. Neural Network with MLP

The process of this step also includes 3 steps as with Logistic Regression models. First building a baseline model with all the default parameters, then using a random grid search to find the best set of parameters, and finally, varying the values of 2 parameters to investigate the accuracy and log loss.

For the baseline neural network model, I set the max_iter to 400, instead of the default of 200, random_state to 0 and I also used early_stopping in order to reduce the training time if the model has already converged.

The MLPClassifier model has a lot of categorical parameters, such as activation or learning_rate. I explored which set of parameters performs best on the data through these parameters, and I will use visualizations to explore some other less common continuous parameters like decay rate beta, depending on the best parameters achieved in the randomized grid search process. My random grid search includes 4 parameters: `activation`, `hidden_layer_sizes`, `alpha`, `learning_rate` and `solver`. For activation, I tried all 4 functions that the MLPClassifier supports (no-op activation, the logistic sigmoid function, the hyperbolic tan function, the rectified linear unit function). For hidden

layers, I tried 4 different sets of values: 1 layer of 50 neurons, 1 layer of 100 neurons, (50, 100) and (128, 64, 32, 8), the last one being a common combination in many neural network settings. For the alpha values, I used 10 linearly spaced values from 10^(-2) to 10^2, also for the purpose of exploring a wide range before narrowing down to some specific values. For learning rate and solver, I explored all the options sklearn offers.

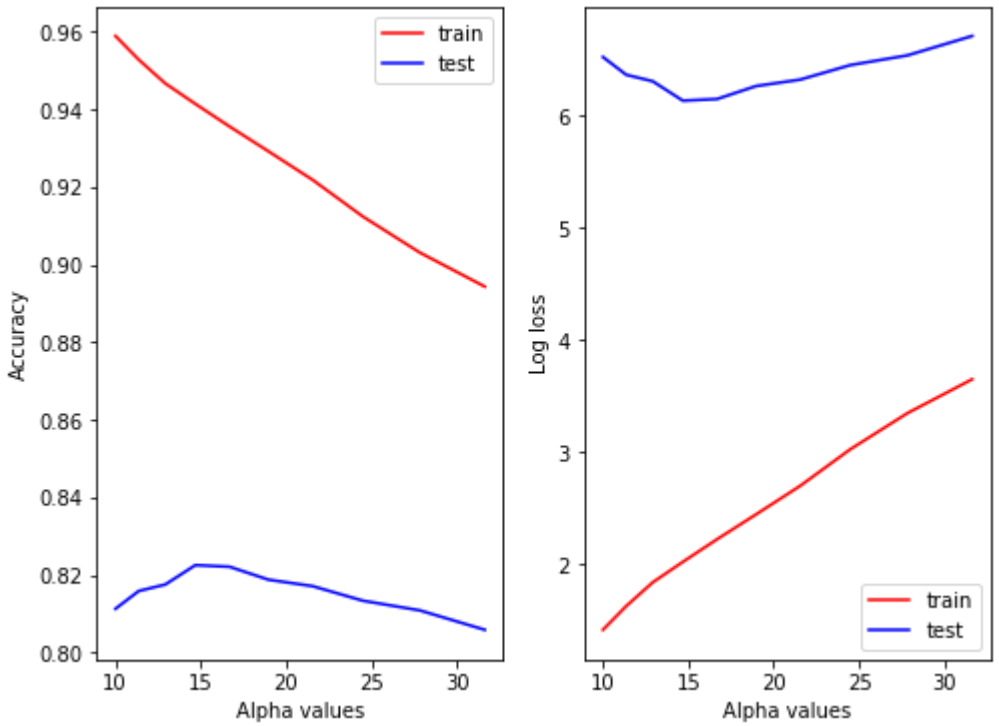
Just as before, I fitted the models on the dataset with and without the stop-words. The result shows that MLPClassifier also performs slightly better on the dataset without the stopwords after grid search, with 80% accuracy on the holdout test set compared to 79%. The training accuracy both stand at 0.96, indicating that both models overfit the data in a similar way.

I chose to go with the better performing model, which uses lbfgs solver, constant learning rate, 1 hidden layer of 100 neurons, alpha value of 12.915, and relu activation function. I will use visualizations to examine how the change in `alpha` value and in `tol` value around the optimal area affect the change in accuracy score and log loss.

For each of these alpha values, I used `KFold` from sklearn to shuffle and split the full train data into 4 folds, then built MLPClassifier models on each of the splits. I used the `score` and `log_loss` function to calculate the accuracy and log loss on the train and test folds and stored them in a list. After finishing fitting on all folds, I averaged the testing and training accuracy scores and log loss across different folds, and stored them in the final lists that will be used to visualize the data. The mean and standard deviation for different metrics across different alpha values are printed below:

```
Alpha value: 10.000
Training accuracy scores for 4 folds are:  0.957 0.960 0.960 0.959 Mean:  0.959 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.818 0.830 0.792 Mean:  0.811 Std:  0.014
Training log loss for 4 folds are:  1.497 1.382 1.382 1.420 Mean:  1.420 Std:  0.047
Testing log loss for 4 folds are:   6.735 6.275 5.872 7.196 Mean:  6.519 Std:  0.496

Alpha value: 11.365
Training accuracy scores for 4 folds are:  0.953 0.953 0.953 0.953 Mean:  0.953 Std:  0.000
Testing accuracy scores for 4 folds are:   0.803 0.823 0.833 0.803 Mean:  0.816 Std:  0.013
Training log loss for 4 folds are:  1.631 1.631 1.631 1.631 Mean:  1.631 Std:  0.000
Testing log loss for 4 folds are:   6.793 6.102 5.757 6.793 Mean:  6.361 Std:  0.449

Alpha value: 12.915
Training accuracy scores for 4 folds are:  0.947 0.947 0.947 0.946 Mean:  0.947 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.825 0.833 0.807 Mean:  0.818 Std:  0.012
Training log loss for 4 folds are:  1.842 1.823 1.823 1.880 Mean:  1.842 Std:  0.024
Testing log loss for 4 folds are:   6.735 6.044 5.757 6.678 Mean:  6.303 Std:  0.416

Alpha value: 14.678
Training accuracy scores for 4 folds are:  0.942 0.939 0.942 0.942 Mean:  0.941 Std:  0.001
Testing accuracy scores for 4 folds are:   0.810 0.828 0.840 0.812 Mean:  0.823 Std:  0.012
Training log loss for 4 folds are:  1.996 2.092 1.996 2.015 Mean:  2.024 Std:  0.040
Testing log loss for 4 folds are:   6.562 5.929 5.526 6.505 Mean:  6.131 Std:  0.428

Alpha value: 16.681
Training accuracy scores for 4 folds are:  0.936 0.935 0.937 0.935 Mean:  0.936 Std:  0.001
Testing accuracy scores for 4 folds are:   0.812 0.830 0.832 0.815 Mean:  0.822 Std:  0.009
Training log loss for 4 folds are:  2.226 2.245 2.187 2.245 Mean:  2.226 Std:  0.024
Testing log loss for 4 folds are:   6.505 5.872 5.814 6.390 Mean:  6.145 Std:  0.306

Alpha value: 18.957
Training accuracy scores for 4 folds are:  0.932 0.928 0.930 0.927 Mean:  0.929 Std:  0.002
Testing accuracy scores for 4 folds are:   0.810 0.830 0.832 0.803 Mean:  0.819 Std:  0.012
Training log loss for 4 folds are:  2.360 2.475 2.418 2.533 Mean:  2.447 Std:  0.064
```

```
Testing log loss for 4 folds are:   6.562 5.872 5.814 6.793 Mean:  6.260 Std:  0.426

Alpha value: 21.544
Training accuracy scores for 4 folds are:  0.924 0.921 0.923 0.920 Mean:  0.922 Std:  0.002
Testing accuracy scores for 4 folds are:   0.810 0.828 0.827 0.803 Mean:  0.817 Std:  0.011
Training log loss for 4 folds are:  2.629 2.744 2.667 2.763 Mean:  2.701 Std:  0.055
Testing log loss for 4 folds are:   6.562 5.929 5.987 6.793 Mean:  6.318 Std:  0.369

Alpha value: 24.484
Training accuracy scores for 4 folds are:  0.916 0.913 0.913 0.907 Mean:  0.912 Std:  0.003
Testing accuracy scores for 4 folds are:   0.807 0.825 0.822 0.800 Mean:  0.813 Std:  0.010
Training log loss for 4 folds are:  2.897 2.993 3.013 3.204 Mean:  3.027 Std:  0.111
Testing log loss for 4 folds are:   6.678 6.044 6.159 6.908 Mean:  6.447 Std:  0.357

Alpha value: 27.826
Training accuracy scores for 4 folds are:  0.907 0.904 0.903 0.898 Mean:  0.903 Std:  0.003
Testing accuracy scores for 4 folds are:   0.810 0.822 0.820 0.792 Mean:  0.811 Std:  0.012
Training log loss for 4 folds are:  3.224 3.320 3.339 3.511 Mean:  3.348 Std:  0.104
Testing log loss for 4 folds are:   6.562 6.159 6.217 7.196 Mean:  6.534 Std:  0.412

Alpha value: 31.623
Training accuracy scores for 4 folds are:  0.898 0.894 0.893 0.892 Mean:  0.894 Std:  0.002
Testing accuracy scores for 4 folds are:   0.797 0.817 0.823 0.787 Mean:  0.806 Std:  0.015
Training log loss for 4 folds are:  3.531 3.646 3.703 3.723 Mean:  3.651 Std:  0.075
Testing log loss for 4 folds are:   7.023 6.332 6.102 7.368 Mean:  6.706 Std:  0.511
```

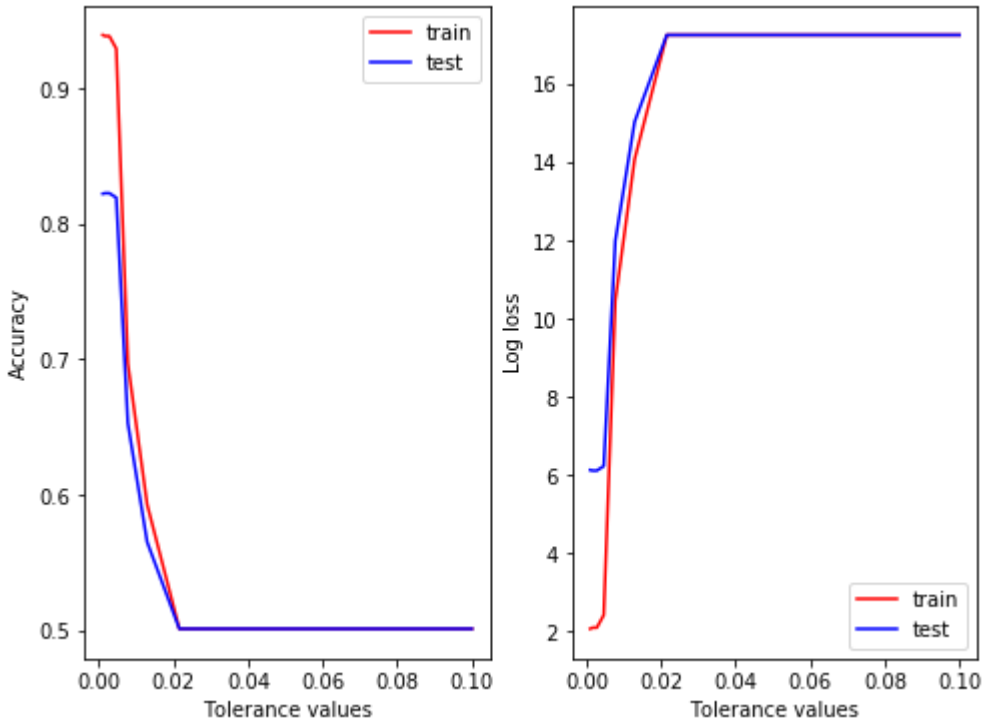


The graph shows that the model overfits the data from the very beginning, the reason for which I suspect is the same as with the Logistic Regression model—due to the large size of the BoW. The model also seems to underfit if we keep increasing the alpha value. The best alpha value is around 15, precisely 14.678. Using this best alpha value, we examine the `tol` parameter. The definition of tolerance, as explained in sklearn documentation:

> "*When the loss or score is not improving by at least* `tol` *for* `n_iter_no_change` *consecutive iterations, unless* `learning_rate` *is set to 'adaptive', convergence is considered to be reached and training stops*"

The result of tuning this parameter is as followed:

```
Tolerance value: 0.0010
Training accuracy scores for 4 folds are:  0.939 0.936 0.942 0.943 Mean:  0.940 Std:  0.003
Testing accuracy scores for 4 folds are:   0.810 0.830 0.840 0.810 Mean:  0.823 Std:  0.013
Training log loss for 4 folds are:  2.092 2.226 2.015 1.976 Mean:  2.077 Std:  0.095
Testing log loss for 4 folds are:   6.562 5.872 5.526 6.562 Mean:  6.131 Std:  0.449

Tolerance value: 0.0017
Training accuracy scores for 4 folds are:  0.939 0.936 0.941 0.940 Mean:  0.939 Std:  0.002
Testing accuracy scores for 4 folds are:   0.810 0.830 0.843 0.808 Mean:  0.823 Std:  0.015
Training log loss for 4 folds are:  2.092 2.226 2.034 2.072 Mean:  2.106 Std:  0.072
Testing log loss for 4 folds are:   6.562 5.872 5.411 6.620 Mean:  6.116 Std:  0.502

Tolerance value: 0.0028
Training accuracy scores for 4 folds are:  0.938 0.936 0.941 0.942 Mean:  0.939 Std:  0.002
Testing accuracy scores for 4 folds are:   0.810 0.830 0.843 0.808 Mean:  0.823 Std:  0.015
Training log loss for 4 folds are:  2.149 2.226 2.034 2.015 Mean:  2.106 Std:  0.086
Testing log loss for 4 folds are:   6.562 5.872 5.411 6.620 Mean:  6.116 Std:  0.502

Tolerance value: 0.0046
Training accuracy scores for 4 folds are:  0.938 0.913 0.938 0.931 Mean:  0.930 Std:  0.010
Testing accuracy scores for 4 folds are:   0.810 0.828 0.835 0.805 Mean:  0.820 Std:  0.012
Training log loss for 4 folds are:  2.149 3.013 2.149 2.399 Mean:  2.427 Std:  0.353
Testing log loss for 4 folds are:   6.562 5.929 5.699 6.735 Mean:  6.231 Std:  0.430

Tolerance value: 0.0077
Training accuracy scores for 4 folds are:  0.877 0.913 0.500 0.499 Mean:  0.697 Std:  0.198
Testing accuracy scores for 4 folds are:   0.770 0.832 0.503 0.507 Mean:  0.653 Std:  0.150
Training log loss for 4 folds are:  4.260 3.013 17.269 17.308 Mean:  10.462 Std:  6.840
Testing log loss for 4 folds are:   7.944 5.814 17.154 17.039 Mean:  11.988 Std:  5.164

Tolerance value: 0.0129
Training accuracy scores for 4 folds are:  0.508 0.864 0.500 0.499 Mean:  0.593 Std:  0.157
Testing accuracy scores for 4 folds are:   0.480 0.770 0.503 0.507 Mean:  0.565 Std:  0.119
Training log loss for 4 folds are:  17.001 4.682 17.269 17.308 Mean:  14.065 Std:  5.419
Testing log loss for 4 folds are:   17.960 7.944 17.154 17.039 Mean:  15.024 Std:  4.103

Tolerance value: 0.0215
Training accuracy scores for 4 folds are:  0.508 0.497 0.500 0.499 Mean:  0.501 Std:  0.004
Testing accuracy scores for 4 folds are:   0.480 0.513 0.503 0.507 Mean:  0.501 Std:  0.013
Training log loss for 4 folds are:  17.001 17.385 17.269 17.308 Mean:  17.241 Std:  0.145
Testing log loss for 4 folds are:   17.960 16.809 17.154 17.039 Mean:  17.241 Std:  0.434

Tolerance value: 0.0359
Training accuracy scores for 4 folds are:  0.508 0.497 0.500 0.499 Mean:  0.501 Std:  0.004
Testing accuracy scores for 4 folds are:   0.480 0.513 0.503 0.507 Mean:  0.501 Std:  0.013
Training log loss for 4 folds are:  17.001 17.385 17.269 17.308 Mean:  17.241 Std:  0.145
Testing log loss for 4 folds are:   17.960 16.809 17.154 17.039 Mean:  17.241 Std:  0.434

Tolerance value: 0.0599
Training accuracy scores for 4 folds are:  0.508 0.497 0.500 0.499 Mean:  0.501 Std:  0.004
Testing accuracy scores for 4 folds are:   0.480 0.513 0.503 0.507 Mean:  0.501 Std:  0.013
Training log loss for 4 folds are:  17.001 17.385 17.269 17.308 Mean:  17.241 Std:  0.145
Testing log loss for 4 folds are:   17.960 16.809 17.154 17.039 Mean:  17.241 Std:  0.434

Tolerance value: 0.1000
Training accuracy scores for 4 folds are:  0.508 0.497 0.500 0.499 Mean:  0.501 Std:  0.004
Testing accuracy scores for 4 folds are:   0.480 0.513 0.503 0.507 Mean:  0.501 Std:  0.013
Training log loss for 4 folds are:  17.001 17.385 17.269 17.308 Mean:  17.241 Std:  0.145
Testing log loss for 4 folds are:   17.960 16.809 17.154 17.039 Mean:  17.241 Std:  0.434
```



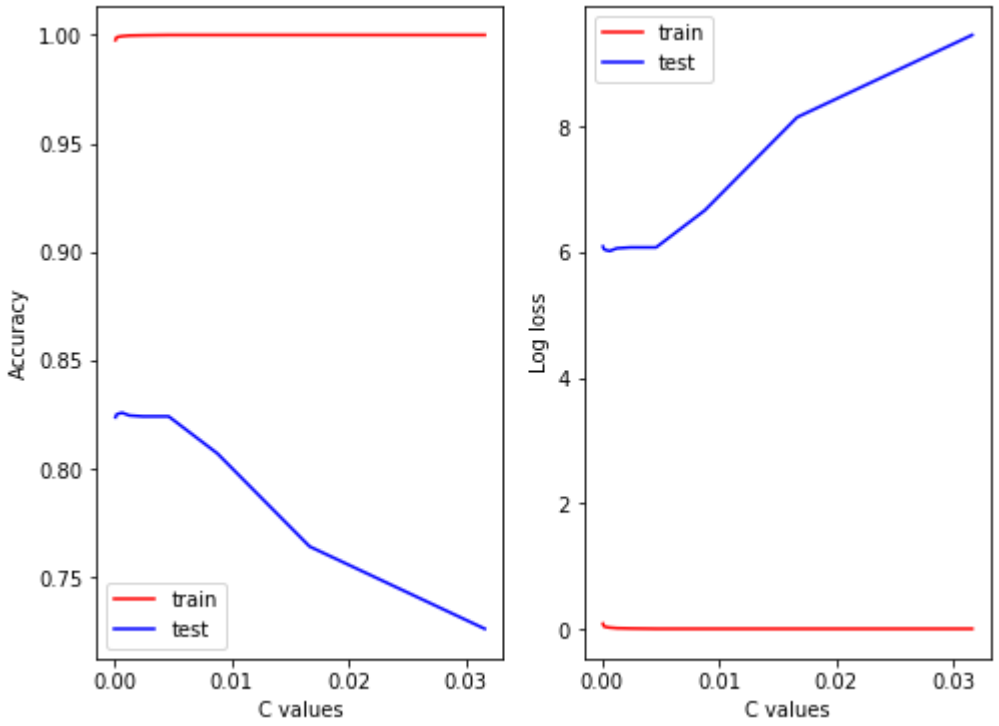
From the graph, we can see that large tolerance values don't yield as good results as smaller ones. The reason is that when we set tolerance so high, as the model moves towards convergence, the score improvement won't be as great as a few percentage points at a time, and thus the model will stop very early at accuracy around 50%, while the optimal solution has not been reached. By decreasing the tol value, we allow more space for improvement before the model's accuracy can no longer improve, thus both accuracy and log loss scores improve significantly.

4. SVM

For the baseline model, I go with all the default parameters. The baseline model on the dataset with stopwords has training accuracy of 0.992 and testing accuracy of 0.792; the baseline model on the dataset without stopwords has training accuracy of 0.991 and testing accuracy of 0.794. We can see that the performances of these 2 baseline models don't differ substantially. We go on to tune a range of parameters with random grid search as we did with the previous models.

The parameters I use to tune are the kernel type, degree of the polynomial kernel function, kernel coefficient value and the regularization term C. The `kernel` parameter includes the 5 options in sklearn: 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'. For `degree`, I explored the values from 1 to 5. For `C` and `gamma`, I explored the values 0.001, 0.01, 0.1, 1, 10, 100. The grid search result for the dataset with stopwords is 0.8030 for testing accuracy and 0.9994, and the model on the dataset without stopwords has test accuracy of 0.7929 and train accuracy of 0.9496. Even though the former has a slightly better test accuracy, it also seems to overfit the model more since the training accuracy is much higher, almost perfectly fitting the data. Thus, I decided to not decide which model is better yet, but to explore both models further through varying the parameter C on each of them. Below is the result of the model on the dataset **with stopwords.**

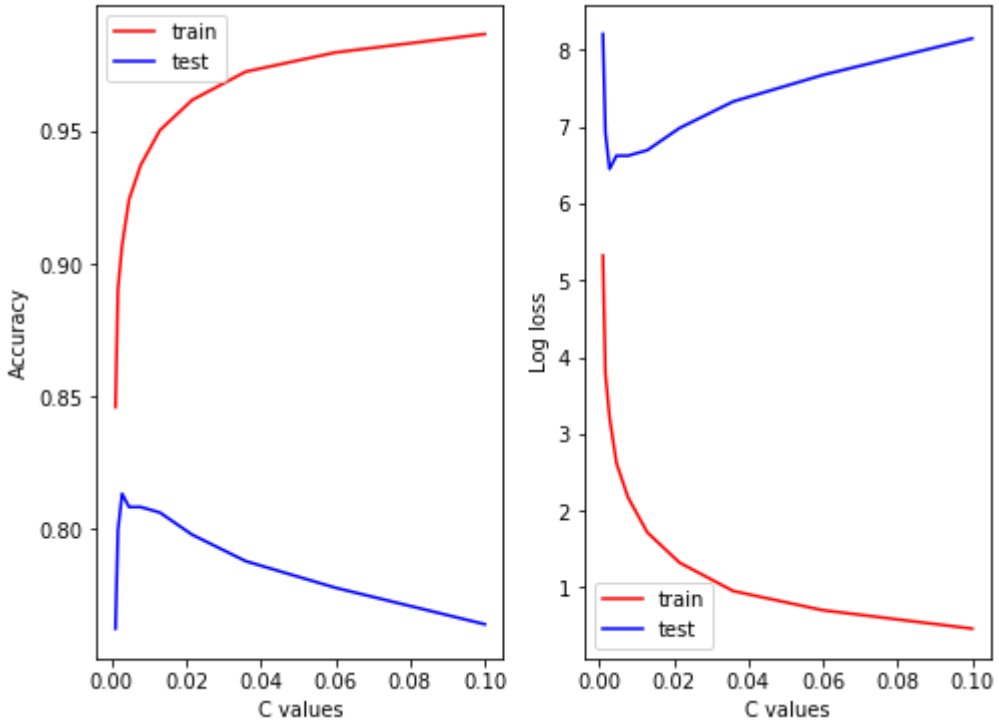
```
C value: 0.0001
Training accuracy scores for 4 folds are:  0.998 0.996 0.998 0.998 Mean:  0.998 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.822 0.847 0.822 Mean:  0.824 Std:  0.015
Training log loss for 4 folds are:  0.058 0.134 0.077 0.058 Mean:  0.082 Std:  0.031
Testing log loss for 4 folds are:   6.735 6.159 5.296 6.159 Mean:  6.088 Std:  0.514

C value: 0.0002
Training accuracy scores for 4 folds are:  0.999 0.998 0.998 0.999 Mean:  0.999 Std:  0.001
Testing accuracy scores for 4 folds are:   0.805 0.827 0.847 0.822 Mean:  0.825 Std:  0.015
Training log loss for 4 folds are:  0.019 0.058 0.058 0.019 Mean:  0.038 Std:  0.019
Testing log loss for 4 folds are:   6.735 5.987 5.296 6.159 Mean:  6.044 Std:  0.513

C value: 0.0004
Training accuracy scores for 4 folds are:  0.999 0.999 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.802 0.828 0.848 0.823 Mean:  0.825 Std:  0.017
Training log loss for 4 folds are:  0.019 0.038 0.038 0.019 Mean:  0.029 Std:  0.010
Testing log loss for 4 folds are:   6.850 5.929 5.238 6.102 Mean:  6.030 Std:  0.573

C value: 0.0007
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.828 0.847 0.823 Mean:  0.826 Std:  0.015
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.735 5.929 5.296 6.102 Mean:  6.016 Std:  0.512

C value: 0.0013
Training accuracy scores for 4 folds are:  0.999 1.000 1.000 0.999 Mean:  1.000 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.825 0.845 0.823 Mean:  0.825 Std:  0.014
Training log loss for 4 folds are:  0.019 0.000 0.000 0.019 Mean:  0.010 Std:  0.010
Testing log loss for 4 folds are:   6.735 6.044 5.354 6.102 Mean:  6.059 Std:  0.489
C value: 0.0024
Training accuracy scores for 4 folds are:  1.000 1.000 1.000 0.999 Mean:  1.000 Std:  0.000
Testing accuracy scores for 4 folds are:   0.807 0.825 0.848 0.817 Mean:  0.824 Std:  0.015
Training log loss for 4 folds are:  0.000 0.000 0.000 0.019 Mean:  0.005 Std:  0.008
Testing log loss for 4 folds are:   6.678 6.044 5.238 6.332 Mean:  6.073 Std:  0.532

C value: 0.0046
Training accuracy scores for 4 folds are:  1.000 1.000 1.000 1.000 Mean:  1.000 Std:  0.000
Testing accuracy scores for 4 folds are:   0.807 0.820 0.843 0.827 Mean:  0.824 Std:  0.013
Training log loss for 4 folds are:  0.000 0.000 0.000 0.000 Mean:  0.000 Std:  0.000
Testing log loss for 4 folds are:   6.678 6.217 5.411 5.987 Mean:  6.073 Std:  0.456

C value: 0.0088
Training accuracy scores for 4 folds are:  1.000 1.000 1.000 1.000 Mean:  1.000 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.792 0.815 0.817 Mean:  0.807 Std:  0.010
Training log loss for 4 folds are:  0.000 0.000 0.000 0.000 Mean:  0.000 Std:  0.000
Testing log loss for 4 folds are:   6.735 7.196 6.390 6.332 Mean:  6.663 Std:  0.344

C value: 0.0167
Training accuracy scores for 4 folds are:  1.000 1.000 1.000 1.000 Mean:  1.000 Std:  0.000
Testing accuracy scores for 4 folds are:   0.788 0.733 0.755 0.780 Mean:  0.764 Std:  0.022
Training log loss for 4 folds are:  0.000 0.000 0.000 0.000 Mean:  0.000 Std:  0.000
Testing log loss for 4 folds are:   7.311 9.211 8.462 7.599 Mean:  8.146 Std:  0.747

C value: 0.0316
Training accuracy scores for 4 folds are:  1.000 1.000 1.000 1.000 Mean:  1.000 Std:  0.000
Testing accuracy scores for 4 folds are:   0.730 0.707 0.727 0.742 Mean:  0.726 Std:  0.013
Training log loss for 4 folds are:  0.000 0.000 0.000 0.000 Mean:  0.000 Std:  0.000
Testing log loss for 4 folds are:   9.326 10.132 9.441 8.923 Mean:  9.455 Std:  0.435
```



We can see that overfitting gets worse very rapidly as we increase the value for C. We compare this performance to the model on the dataset **without stopwords:**

```
C value: 0.0010
Training accuracy scores for 4 folds are:  0.857 0.880 0.828 0.819 Mean:  0.846 Std:  0.024
Testing accuracy scores for 4 folds are:   0.778 0.810 0.763 0.698 Mean:  0.762 Std:  0.041
Training log loss for 4 folds are:  4.931 4.145 5.948 6.255 Mean:  5.320 Std:  0.837
Testing log loss for 4 folds are:  7.656 6.562 8.174 10.419 Mean:  8.203 Std:  1.406

C value: 0.0017
Training accuracy scores for 4 folds are:  0.895 0.896 0.887 0.884 Mean:  0.890 Std:  0.005
Testing accuracy scores for 4 folds are:   0.800 0.818 0.822 0.758 Mean:  0.800 Std:  0.025
Training log loss for 4 folds are:  3.627 3.588 3.914 4.010 Mean:  3.785 Std:  0.181
Testing log loss for 4 folds are:   6.908 6.275 6.159 8.347 Mean:  6.922 Std:  0.871

C value: 0.0028
Training accuracy scores for 4 folds are:  0.911 0.909 0.904 0.903 Mean:  0.907 Std:  0.003
Testing accuracy scores for 4 folds are:   0.813 0.827 0.828 0.785 Mean:  0.813 Std:  0.017
Training log loss for 4 folds are:  3.070 3.128 3.300 3.358 Mean:  3.214 Std:  0.119
Testing log loss for 4 folds are:   6.447 5.987 5.929 7.426 Mean:  6.447 Std:  0.600

C value: 0.0046
Training accuracy scores for 4 folds are:  0.926 0.926 0.926 0.920 Mean:  0.924 Std:  0.002
Testing accuracy scores for 4 folds are:   0.802 0.828 0.815 0.788 Mean:  0.808 Std:  0.015
Training log loss for 4 folds are:  2.571 2.571 2.571 2.763 Mean:  2.619 Std:  0.083
Testing log loss for 4 folds are:   6.850 5.929 6.390 7.311 Mean:  6.620 Std:  0.515

C value: 0.0077
Training accuracy scores for 4 folds are:  0.938 0.937 0.939 0.934 Mean:  0.937 Std:  0.002
Testing accuracy scores for 4 folds are:   0.798 0.828 0.820 0.787 Mean:  0.808 Std:  0.017
Training log loss for 4 folds are:  2.130 2.168 2.111 2.283 Mean:  2.173 Std:  0.067
Testing log loss for 4 folds are:   6.965 5.929 6.217 7.368 Mean:  6.620 Std:  0.574

C value: 0.0129
Training accuracy scores for 4 folds are:  0.955 0.946 0.951 0.949 Mean:  0.950 Std:  0.003
Testing accuracy scores for 4 folds are:   0.803 0.828 0.810 0.783 Mean:  0.806 Std:  0.016
```

```
Training log loss for 4 folds are:  1.554 1.861 1.708 1.746 Mean:  1.717 Std:  0.110
Testing log loss for 4 folds are:   6.793 5.929 6.562 7.483 Mean:  6.692 Std:  0.556

C value: 0.0215
Training accuracy scores for 4 folds are:  0.962 0.962 0.959 0.964 Mean:  0.962 Std:  0.002
Testing accuracy scores for 4 folds are:   0.797 0.813 0.798 0.783 Mean:  0.798 Std:  0.011
Training log loss for 4 folds are:  1.305 1.324 1.420 1.247 Mean:  1.324 Std:  0.062
Testing log loss for 4 folds are:   7.023 6.447 6.965 7.483 Mean:  6.980 Std:  0.367

C value: 0.0359
Training accuracy scores for 4 folds are:  0.972 0.972 0.972 0.973 Mean:  0.972 Std:  0.001
Testing accuracy scores for 4 folds are:   0.778 0.807 0.795 0.772 Mean:  0.788 Std:  0.014
Training log loss for 4 folds are:  0.959 0.979 0.959 0.921 Mean:  0.955 Std:  0.021
Testing log loss for 4 folds are:   7.656 6.678 7.081 7.886 Mean:  7.325 Std:  0.475

C value: 0.0599
Training accuracy scores for 4 folds are:  0.979 0.979 0.979 0.980 Mean:  0.980 Std:  0.000
Testing accuracy scores for 4 folds are:   0.780 0.792 0.788 0.752 Mean:  0.778 Std:  0.016
Training log loss for 4 folds are:  0.710 0.710 0.710 0.691 Mean:  0.705 Std:  0.008
Testing log loss for 4 folds are:   7.599 7.196 7.311 8.577 Mean:  7.671 Std:  0.544

C value: 0.1000
Training accuracy scores for 4 folds are:  0.987 0.987 0.986 0.987 Mean:  0.987 Std:  0.001
Testing accuracy scores for 4 folds are:   0.765 0.785 0.780 0.727 Mean:  0.764 Std:  0.023
Training log loss for 4 folds are:  0.461 0.441 0.499 0.461 Mean:  0.465 Std:  0.021
Testing log loss for 4 folds are:   8.117 7.426 7.599 9.441 Mean:  8.145 Std:  0.790
```



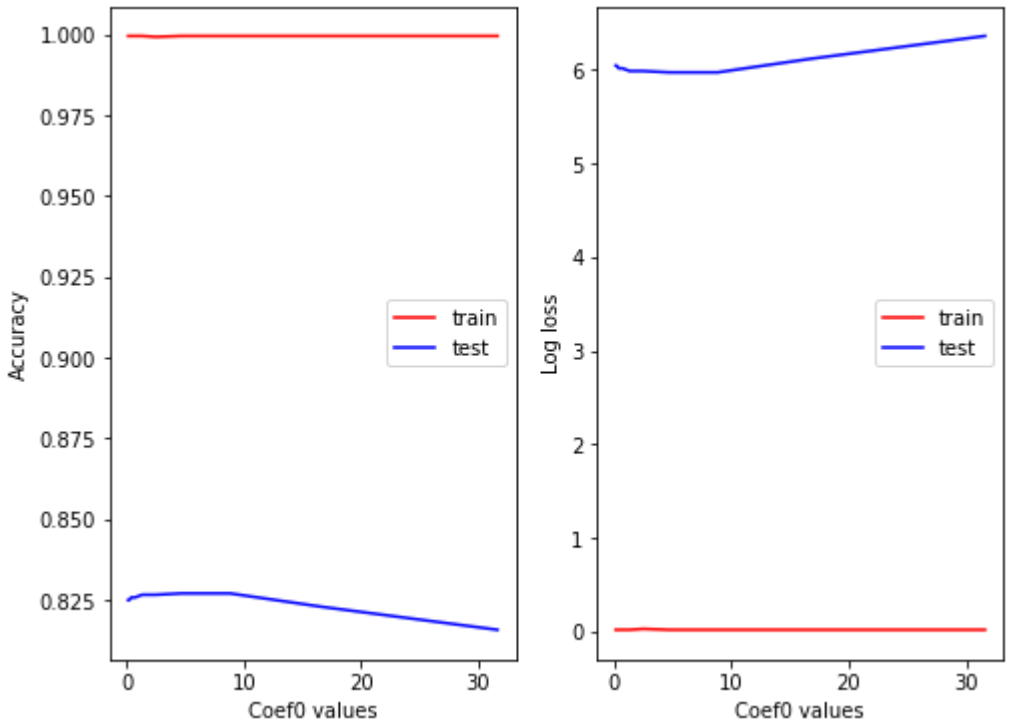
This model also suffers from overfitting as C value increases. However, the graph for this second model captures the optimal performance of the model, as well as the underfitting earlier on, and after the model has reached its peak. For the first model, the range of values we explored don't capture the initial underfitting of the model, even though the first few C values are even smaller than the values in the second model. This interesting observation may be due to the fact that we use degree 2 polynomials to fit the data, while for this model, we fit it using degree 1 polynomials. This aligns with a common characteristic for some certain datasets that the higher the polynomial degree of a model, the higher the chance of

overfitting it is. Hence, such models need a strong regularization term to control the overfitting, which corresponds to a smaller C value.

After investigating this behavior on C, I chose the model with all stopwords included because for small enough C value, this model outperforms in terms of accuracy and log loss. Next, with the best set of parameters achieved (kernel='poly', gamma=100, degree=2, C= 0.00068129), we explore the parameter `coef0`, the independent term in kernel function, only used for 'poly' kernel, which is what we are having. The formula for the polynomial kernel projection in svm is $(a \times b + r)^d$ where a, b are 2 points of different classes that we are trying to separate, d is the degree and r is the `coef0` parameter. Roughly speaking, as the degree grows towards infinity, powers of points with coordinates smaller than 1 will be shrunk towards 0, and powers of points with large coordinates will shoot up to infinity, creating a very big separation. Hence this constant term can be used to "scale" the data so that no such big distinction exists. The results of tuning this parameter are the following:

```
Coef0 value: 0.1000
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.827 0.845 0.823 Mean:  0.825 Std:  0.014
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.735 5.987 5.354 6.102 Mean:  6.044 Std:  0.490

Coef0 value: 0.1896
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.827 0.845 0.823 Mean:  0.825 Std:  0.014
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.735 5.987 5.354 6.102 Mean:  6.044 Std:  0.490

Coef0 value: 0.3594
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.807 0.828 0.845 0.823 Mean:  0.826 Std:  0.014
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.678 5.929 5.354 6.102 Mean:  6.016 Std:  0.472

Coef0 value: 0.6813
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.828 0.847 0.823 Mean:  0.826 Std:  0.015
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.735 5.929 5.296 6.102 Mean:  6.016 Std:  0.512

Coef0 value: 1.2915
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.830 0.848 0.823 Mean:  0.827 Std:  0.016
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.735 5.872 5.238 6.102 Mean:  5.987 Std:  0.535

Coef0 value: 2.4484
Training accuracy scores for 4 folds are:  0.999 0.999 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.805 0.828 0.848 0.825 Mean:  0.827 Std:  0.015
Training log loss for 4 folds are:  0.019 0.038 0.038 0.019 Mean:  0.029 Std:  0.010
Testing log loss for 4 folds are:   6.735 5.929 5.238 6.044 Mean:  5.987 Std:  0.531

Coef0 value: 4.6416
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.802 0.830 0.855 0.822 Mean:  0.827 Std:  0.019
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.850 5.872 5.008 6.159 Mean:  5.972 Std:  0.661

Coef0 value: 8.7992
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:  0.802 0.832 0.848 0.827 Mean:  0.827 Std:  0.017
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
```

```
Testing log loss for 4 folds are:   6.850 5.814 5.238 5.987 Mean:  5.972 Std:  0.578

Coef0 value: 16.6810
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.800 0.825 0.842 0.825 Mean:  0.823 Std:  0.015
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.908 6.044 5.469 6.044 Mean:  6.116 Std:  0.514

Coef0 value: 31.6228
Training accuracy scores for 4 folds are:  0.999 1.000 0.999 0.999 Mean:  0.999 Std:  0.000
Testing accuracy scores for 4 folds are:   0.800 0.820 0.827 0.817 Mean:  0.816 Std:  0.010
Training log loss for 4 folds are:  0.019 0.000 0.038 0.019 Mean:  0.019 Std:  0.014
Testing log loss for 4 folds are:   6.908 6.217 5.987 6.332 Mean:  6.361 Std:  0.339
```



There are some values of coef0 that lift the model's performance by some decimal points, but not significantly. Increasing the value of this independent term also hurts the performance. This suggests that as this parameter becomes too large, the data is “scaled" too much, thus the classes become less separated so it's harder to classify the data.

5. Evaluating the models

Below is a summary of the best performing model for each algorithm, each cell shows the average cross validated value for each metrics, along with its standard deviation:

	Train accuracy	Test accuracy	Train log loss	Test log loss
Logistic Regression	0.937 Std: 0.001	0.822 Std: 0.010	2.187 Std: 0.049	6.131 Std: 0.342
Neural Network	0.940 Std: 0.003	0.823 Std: 0.013	2.077 Std: 0.095	6.131 Std: 0.449
SVM	0.999 Std: 0.000	0.827 Std: 0.015	0.029 Std: 0.010	5.987 Std: 0.531

Out of all the models we explored, the neural network model using the dataset excluding the stopwords performs best. Even though the best performing models of logistic regression, svm and MLP have similar testing accuracy scores, with svm even having a slightly better and more consistent score, MLP is the model that achieves both good accuracy and log loss, as well as a lower training accuracy (SVM: 99%, MLP: 93%), which suggests that it overfits the data less than svm. The training accuracy and log loss of the best svm model are almost perfect, as shown in the figure, suggesting that it greatly overfits the data, which means the model can perform worse on unseen data points, compared to MLP model. Furthermore, MLP has equally good training accuracy as Logistic Regression, yet the true accuracy of MLP appears to have more values in the range 82% than logistic regression. Another reason why I think the neural network model is better than other models is that the MLP classifier is more flexible. We can imitate the behavior of logistic regression using MLP (using solver 'sigmoid'), so the performance of MLP is at least as good as that of Logistic Regression. MLP has more parameters available to tune, hence if there are some changes to the nature of the data in the future, we are likely to find another set of parameters that works (can be changing the numbers of hidden layers, etc).

On the test set (792 data points, roughly one-third of the original data), we misclassified 159 instances, 55 of which are from Amazon, 53 from IMDB and 51 from Yelp, so our model doesn't appear to be biased towards any one website. Examining the texts, I can classify some of the common mistakes the model made:

1. Double negatives: this is the most common mistake. Some sentences such as "*Because both ears are occupied, background **is not distracting** at all*", "*You **won't be disappointed***", "*You **can't go wrong** with any of the food here*" can make the model confused because we only use unigrams as the basis for classification. Phrases of the form "not + …" won't get considered and hence such sentences are likely to get misclassified.
2. Nouns that have negative/positive connotation: these reviews don't use specific adjectives to describe the negative or positive experience, but they use various analogies that imply so, such as "*I give it **2 thumbs down***", "*He's a national*

treasure", "No **complaints**!", "*It has been a **winner** for us*". These nouns/images may not be common enough to be considered or to have any impact on the classification, plus the occasional problem of "phrases" (instead of individual words to express feelings) are the reasons why the model may confuse these reviews.

3. Expression through special characters: this includes reviews that express their feelings via emoticons and exclamation marks, for example, which are stripped off at the time of training and evaluating, thus these factors are not taken into account at all. Some cases in point are: "*GO AND SEE IT!*", "*10/10*" (for this case, the string is empty after preprocessing)

6. Predict the unseen data

The autograder error rate is 19.8%, so my accuracy is around 80%. I think this is expected. The discrepancy between the cross validated accuracy scores and this test accuracy is around 2%. My model does overfit the data, which explains why the true test accuracy is a little lower than the accuracy we see while fitting the model. However, this is not a huge gap, and other models could have performed worse since their cross validated accuracy is not as good and their overfitting is more pronounced.