Project 2

Introduction

In this project we are supplied with several data sets, each of which contain input sentences which are online reviews. For the training data we are provided with output labels that indicate whether the corresponding review is positive or negative.

1. Preprocessing the data

To transform the data into more useful data, I read online about natural language processing and methods of transforming data for sentiment analysis.

I first removed capitalization from the data. Though I think capital words or phrases can convey more or less intensity with regards to positive/negative sentiment, I don't think it would be helpful for the purpose of our binary classification, so normalizing the capitalization to lower case was the choice I made.

Then I removed punctuation from the data. I couldn't think of any ways the inclusion of punctuation would help to convey positive or negative sentiment.

For the remaining data transformations, I referenced online sources to learn more about natural language processing and sentiment analysis.

Then I removed "stop words" which are words that do not add meaning to a phrase. I used the natural language toolkit stop word corpus to identify stop words. However, some of the NLTK stop words are words I wanted to actually include like "not," because a review like "this is not good" would be transformed into "this is good" which would not only result in anyone misclassifying the review let alone the classifier. The NLTK stop words set is not too large so it was pretty easy to read through and remove words that I wanted to include in the dataset.

Then I "lemmatized" the phrases, removing the inflections from words, which do not convey meaning. I thought this would be helpful because there could me multiple occurrences of the same word with different inflections across the dataset. Instead of treating these words as different words, I figured that the classifier would be better off treating those words as the same because in most cases the same word with different inflection conveys the same sentiment. The lemmatization method I am using is the Porter algorithm that is a part of the Natural Language Toolkit package.

Bellow is a sample of the transformation process of a single review.

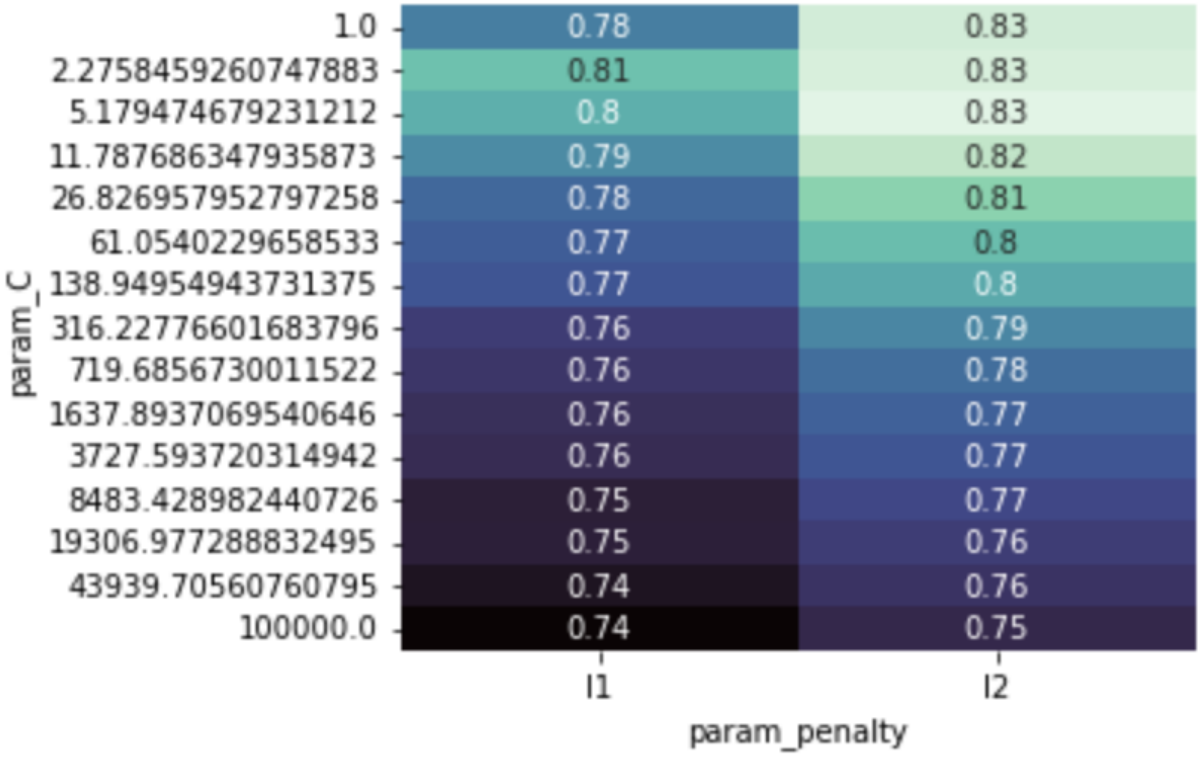
```
I'm very disappointed with my decision.
i'm very disappointed with my decision.
['i', "'m", 'very', 'disappointed', 'with', 'my', 'decision', '.']
["'m", 'disappointed', 'decision', '.']
["'m", 'disappoint', 'decis', '.']
['m', 'disappoint', 'decis']
m disappoint decis
```

To reformat the sentences, I used the Bag of Words TF-IDF approach, which would be helpful for the classifiers as it provides some help distinguishing the important key words.

One issue with this representation is that the order of the words within the sentences are lost. This could be detrimental when approaches like neural network models could take advantage of the order of the words while trying to analyze the sentiment of a sentence.

2. Logistic Regression

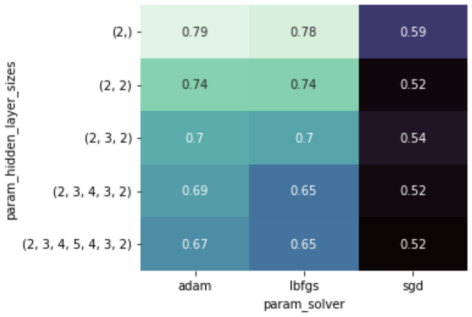
First, I tested multiple regularization penalty and regularization methods for the hyper parameters. The logistic regression model I build uses the liblinear solver. I tested regularization penalty values between 1 and 10000 in the logspace. Then I tried using both ridge (L1) and lasso (L2) regression. The results in the figure bellow show that the C=1.0 and L2 model seemed to perform the best. This makes sense as there are many features that seem to be subtle yet important. I would think that we do not really want to be feature selecting in this problem because of the subtleties and idiosyncrasies of language. Even the occurrence of some seemingly telling words like “good” or “bad” might not tell us the sentiment of the review (for example the case where these words are proceeded by negation words).

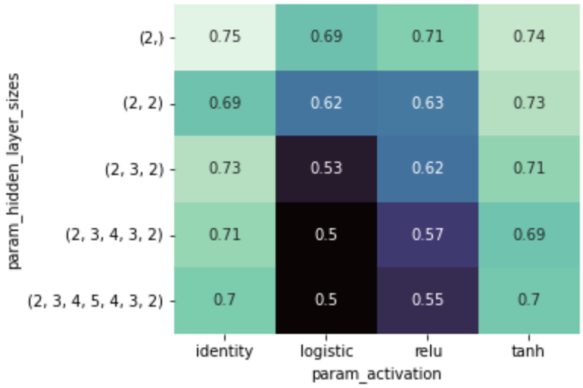


The best model in this case seems to be the L2 model with 1.0 as C. This model performed at 0.82875 AUROC with a standard deviation of 0.0165. To attain the results above, the sklearn model_selection library was used to perform 5 fold cross validation.

3. Neural Network Model (MLP)

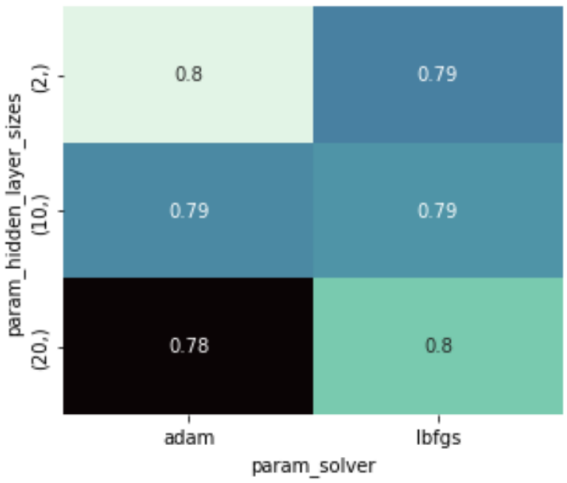
I first tried to test varying the hidden layer amount and size, but this proved to be too time consuming while also testing variation of activation and solver, so I decided to test with small hidden layer sizes and number of neurons to see if there would be anything telling before testing with larger networks.

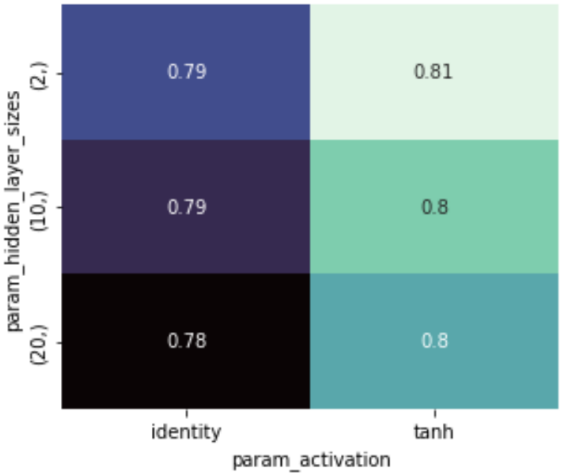




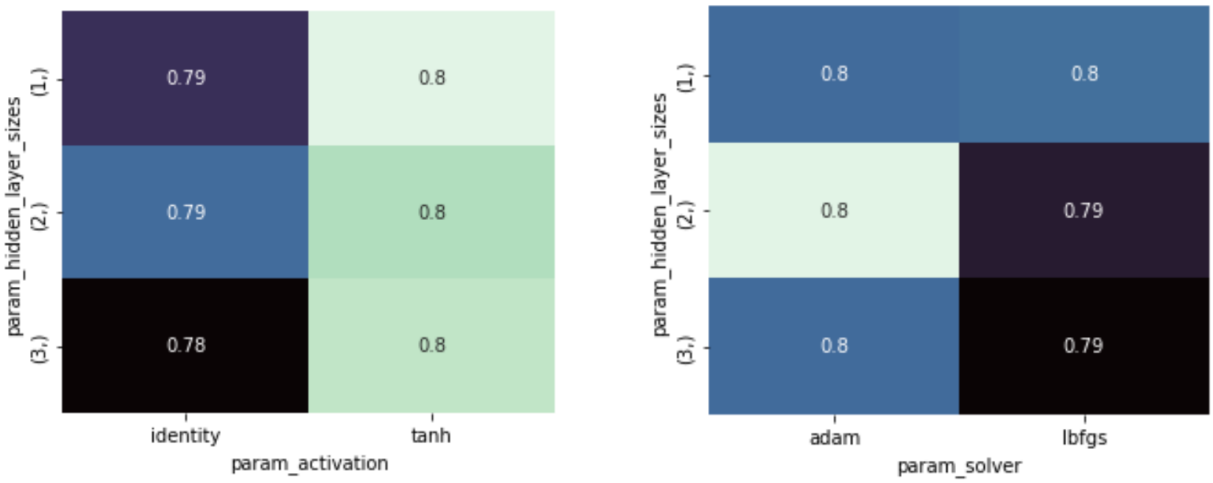
It became pretty clear after my first couple of tests that the smaller number of layers would be the best path forwards, due to the fact that while keeping the solver or activation function parameter constant, the performance improved with less layers. The sigmoid solver also seemed to perform poorly overall so I removed that from my following tests along with the logistic which noticeably performed poorly and was not the fastest.

After testing ranges of the layer size between 1 and 50, 1 remained the best performing, topping out at 0.7845 AUROC with a standard deviation of 0.014469. To attain the results above, the sklearn model_selection library was used to perform 5 fold cross validation.





With larger layer sizes the smallest layer still seemed to perform the best with 5 fold cross validation.



Ultimately the 1 layer 1 neuron MLP performed the best across multiple runs, kind of defeating the purpose of using a neural network. I think more than parameter exploration has to be done in this case for the neural networks to be more effective, but maybe that requires more sophisticated natural language processing work.

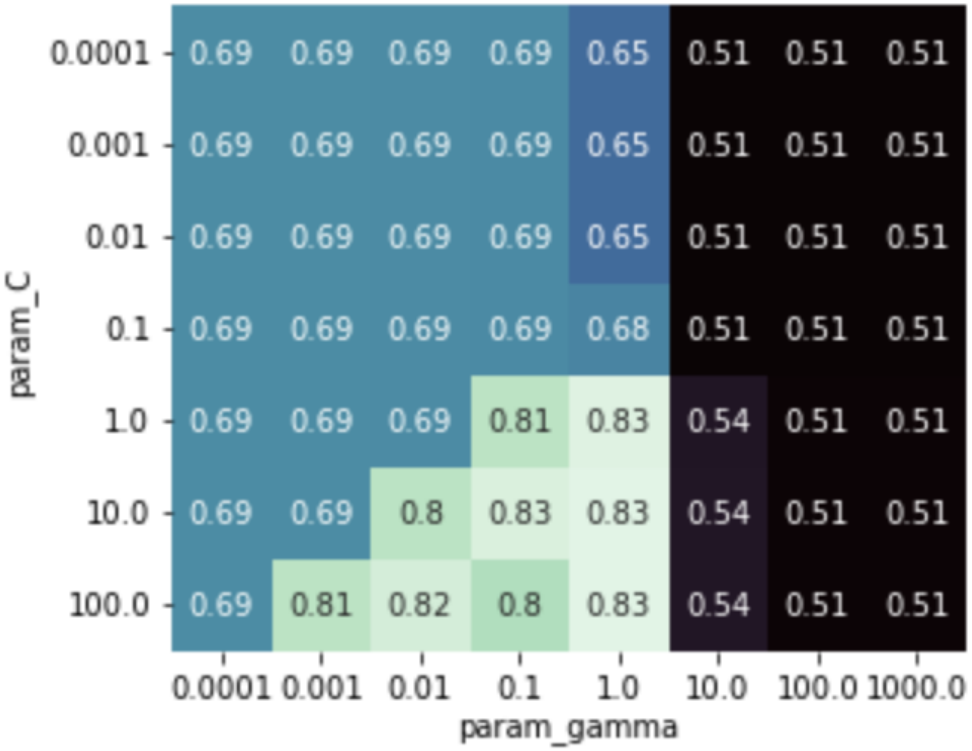
I would speculate the the neural networks would allow for a lot better performance, but without enough time I was unable to learn about to topic enough to push this model to its potential.

4. SVM

I decided to try a support vector machine classifier as the assignment spec suggests because intuitively the clustering ability of the SVM's could correspond with the clustering of sentiment between negative, neutral, and positive.

The parameters I tuned were the gamma and C parameters. I created models using a range for C between 0.0001 and 100 and for gamma 0.0001 to 1000.

The best performing model ended up being the 1.0 gamma 100 C model. I continued testing with 1.0 gamma and higher values for C but the models did not seem to perform better than the best model that had 0.833749 AUROC with a standard deviation of 0.0177.



To attain the results above, the sklearn model_selection library was used to perform 5 fold cross validation.

5. Leaderboard Submission

Out of the three classifiers, the SVM seems to work the best, slightly edging out the logistic regression model. I think this in a way makes sense because the SVM is a slightly more subtle and nuanced version of the classic logistic regression.

However it is likely the case that the model is overfit to the training data because when used on the testing data, we see AUROC drop slightly to 0.8266. The model seems to fail where the language of the review is unique with its vocabulary or on the fence about how they feel.

Sources

1. Sources

How to transform data:

https://www.watermelonblock.io/2018/11/21/sentiment-analysis-data-transformation/

Natural Language Toolkit:

https://www.nltk.org/

Getting started with NLTK:

https://realpython.com/python-nltk-sentiment-analysis/

Preprocessing:

https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0

Porter Algorithm:

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

2. Sources

Sklearn model selection:

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

3. Sources

Sklearn MLP documentation:

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

4. Sources

Sklearn SVM svc documentation:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html