Project A: Classifying Sentiment from Text Reviews

Problem 1

1A: Bag-of-Words Feature Representation

We used an input preprocessor consisting of a custom made preprocessor and the default from CountVectorizer to clean the data. The default preprocessor removes punctuation, transforms all words into lowercase form, and only processes unigrams. Since the default preprocessor doesn't deal with numbers in the data, we combined it with a custom made preprocessor that excludes all the number inclusive tokens. After preprocessing our vocabulary based on minimum and maximum document frequency with CounterVectorizor, we found the final size of our vocabulary to be 703 for our best model's hyperparameters of C = 1.0, min_df = 5, max_df = 0.1 for 5-fold split. In CountVectorizer, we used counts instead of binary values, which means that we took into account the frequency of each word within each document rather than just checking their presence or absence. Our bag-of-words representation counts the amount of times a word (feature) appears in each document. We ignored any words that were not in the final vocabulary since our model is not capable of interpreting words that it is not trained on.

1B: Cross-Validation Design Description

Initially, we were planning on using accuracy as the performance metrics in our model because the data set is balanced. However, the project's specifications tell us that our model will be evaluated based on the area under the ROC curve. Therefore, we decided that optimizing our model based on AUROC would yield better results under these circumstances. For cross validation, we used the 'KFold' function within the 'GridSearchCV' of the Scikit-learn library. We decided to use 5 folds of size 480 each based on the fact that the standard is between 5 and 10, and we thought 5 would lead to less computation burden on the grid search in our program. We also ensured that the training data was shuffled before cross-validation to ensure that the training and validation data set were balanced. After using cross validation, we identified the selected parameter configuration to be C = 1.0, min_df = 5, max_df = 0.1 for a 5-fold split. Lastly we created a model with that parameter configuration and fitted it to all the training data, and this was our final model to apply on the test data.

1C : Hyperparameter Selection for Logistic Regression Classifier

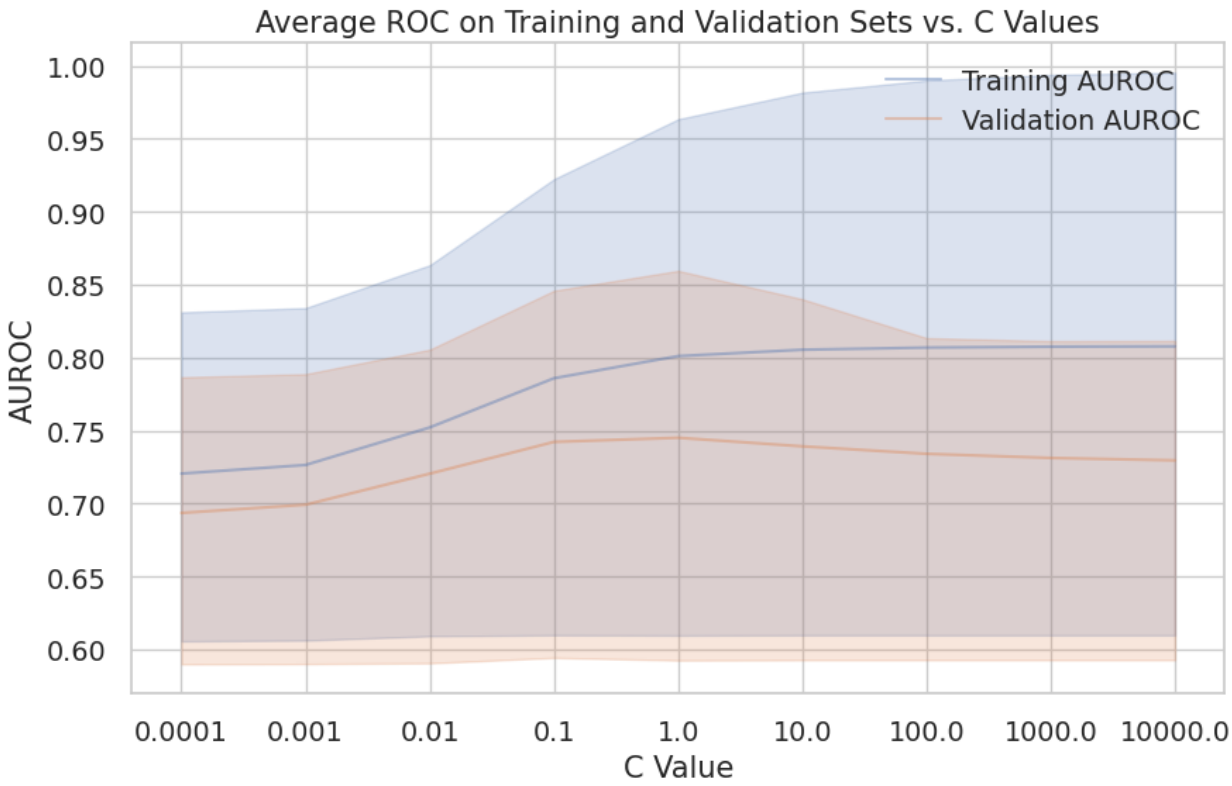

Figure 1. Average ROC on Training and Validation Sets vs. C Values

For our LogisticRegression classifier, we experimented on three parameters, namely, the minimum document frequency (min_df), maximum document frequency (max_df), and regularization penalty (C). Among these three parameters, min_df and max_df affect the content of our vocabulary list based on the frequency of each token in the documents, and C value influences the complexity of the logistic regression model. We tested the following sets of range of values for the three parameters:

1. Minimum Document Frequency: [5, 10, 15, 20, 80, 140, 160)
2. Maximum Document Frequency: [0.1, 0.2, 0.4, 0.8, 1.0]
3. Regularization Penalty (C): [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]

Since C value controls the model complexity, we focused our analysis on this parameter, and explored a wide range of values that includes very low and very high values in order to observe both underfitting and overfitting models. For all models with different parameter combinations, we plotted a graph to compare AUROC performance with the C values, and above is the graph that we obtained. The average AUROC for models across all C values are given by the blue and orange lines for the training and validation data respectively. As you can see, across all combinations of max_df and min_df values, models that have the C value of 1.0 produce the best validation result (where the AUROC peak flattens→decreases for validation and increases for training). As a conclusion, we reasoned that C = 1.0 produces a model complexity that generalizes data the best.

1D : Analysis of Predictions for the Best Classifier

For analyzing the predictions, we created a new data set with lines selected from the test set provided to us. This new data set included both positive and negative reviews from all three review kinds (amazon, imdb or yelp). Thus, our test set was quite comprehensive and balanced which makes our analysis of the best classifier's predictions more reliable. Below, you can find the confusion matrix for the heldout data. We selected 5 reviews from the 9 false positives and 10 reviews from the 16 false negatives which represented a pattern within our results.

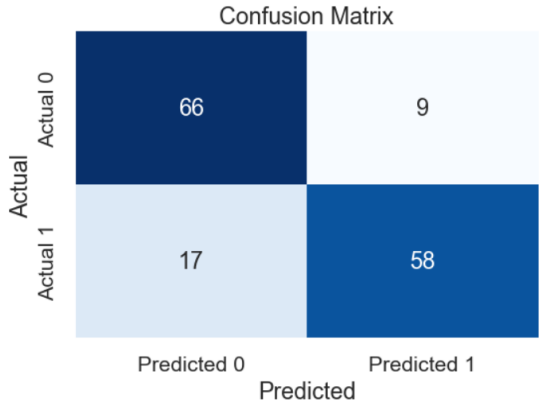

Figure 2. Classifier performance on held out data shown with Confusion matrix

False Positives:

1. This movie suffered because of the writing, it needed more suspense.
2. To be honest with you, this is unbelievable nonsense and very foolish.
3. The film is way too long.
4. Instead, we got a bore fest about a whiny, spoiled brat babysitting.
5. The food wasn't good.

False Negatives:

1. No shifting, no bubbling, no peeling, not even a scratch, NOTHING!I couldn't be more happier with my new one for the Droid.
2. I was looking for this headset for a long time and now that I've got it I couldn't be happier.
3. The eargels channel the sound directly into your ear and seem to increase the sound volume and clarity.
4. He was very impressed when going from the original battery to the extended battery.
5. Personally, I think it shows that people should learn to find a compromise them self without involving other people into issue.
6. BLACK WATER is a thriller that manages to completely transcend it's limitations (it's an indie flick) by continually subverting expectations to emerge as an intense experience.
7. I won't say any more - I don't like spoilers, so I don't want to be one, but I believe this film is worth your time.
8. Of course the footage from the 70s was grainy, but that only enhanced the film.
9. I'm a big fan of this series mostly due to Anne Rice's style, sensitivities and treatments.
10. Also there are combos like a burger, fries, and beer for 23 which is a decent deal.

Figure 3. Representative examples of false positives and false negatives for best classifier

In figure 3, you can see the examples of false positives and false negatives from the best classifier on held out examples. We observed negation words and the particular kind of words didn't have a consistent pattern on the test examples provided to the model. Within the false positives, we observed that the sentences mostly had a short length (~7 words), including words with negative connotations (e.g. hated, nonsense, foolish, whiny, spoiled, suffered) that had low frequency. There were a couple of words that may have been classified as positive (e.g., honest, fest, food, etc.)) . For the False Negatives, many of the sentences were longer (~16 words), including words with positive connotations (e.g. enhanced, decent, phenomenal, compromise, intrigued, worth) with low frequency. There were a couple of words with negative connotations (e.g., no, couldn't, spoilers, etc.). Conclusively, longer sentences with (low frequency) positive words failed to show up as positive and shorter sentences with (low frequency) negative words failed to show up as negative. This shows the model's difficulty in classifying sentences with a heightened or lack-of complexity (in length) and unseen words with already classified words.

1E : Report Performance on Test Set via Leaderboard

Our ultimate test set performance, which is measured in terms of the area under the ROC curve (AUROC), is 0.86592. This performance metric quantifies our model's ability to distinguish between positive and negative reviews, and it is a robust indicator of overall classification performance. After comparing this test set performance to our previous estimates that we got from cross-validation, we find a remarkable consistency between the two evaluation methods with an average AUROC of 0.8601. This consistency implies that our classifier generalizes well from the training set to the test set, which means that it demonstrates the model's effectiveness in classifying sentiment from text reviews. We hypothesize that one possible reason for the close agreement between cross-validation and test set performance may be the balanced nature of the dataset. Also, we believe that the other reason could be the robustness of the logistic regression classifier used.

Problem 2: Open-ended challenge

2A : Feature Representation description

For feature extraction, we used sklearn's TfidfVectorizer class, which is different from CountVectorizer because it takes into account both the word frequency and the importance of the words in the context of a document collection. Similar to 1A, we used a similar custom-made preprocessor that removes punctuation, transforms all words to lowercase, and excludes numbers in TfidfVectorizer. However, this time we decided to exclude the list of common English stop words such as (I, and, or, etc.) because we hypothesized that these words are not good indicators of the semantics of a sentence. Additionally, we increased the size of our tokes to include both unigrams, bigrams, and unigrams with bigrams. After preprocessing our vocabulary based on minimum and maximum document frequency with TfidVectorizer, we we found the final size of our vocabulary to be 559 for parameters for the following parameter combination in our model: ngram_range = (1, 2), min_df = 5, max_df = 0.1, and alpha = 0.5 for Multinomial Naive Bayes. For this model, we did 5-fold cross validation where each fold contained about 480 documents. For counting the frequency, TfidfVectorizer uses a weighted approach where words are represented based on their importance both within the document and across the whole corpus. Mathematically, the calculation is defined like this:

$$TFIDF(t, d) = TF(t, d) * IDF(t), \text{ where}$$

$TF(t, d) =$ frequency of a term t in a document d, and $IDF(t) = log(\frac{N}{df(t)})$, where $df(t) =$ number of documents in the corpus which contain the term t and $N =$ the total number of documents in the corpus. Lastly, ignoring words when making a prediction can be a limitation, and therefore we address this issue by applying a technique called smoothing in MNB. This involves adding a small constant to the counts of each word in the vocabulary during both training and classification to ensure no words have a probability of exactly zero. In the hyperparameter selection section, we discuss what values we tried for this parameter, alpha, called the Laplace smoothing.

2B : Cross Validation (or Equivalent) description

For this model, we also used AUROC as the performance metric because the project specifications say that the models are evaluated based on AUROC and therefore we wanted to optimize our model for it. For cross validation, we implemented the 'KFold' function within the 'RandomizedSearchCV' from the Scikit-learn library. We used 5 folds with each fold having about 480 reviews. For hyperparameter search, we pursued a customized technique that involves RandomizedSearch followed by Grid Search. First, we used a range of parameters for ngram_range, alpha, min_df, and max_df. Then, from the best model's parameters in RandomizedSearch, we narrowed down the range of parameters and performed a GridSearch on them. Our final model had the following parameter configuration: ngram_range = ((1, 2), alpha = 0.5, min_df = 5, and max_df = 0.1. After selecting hyperparameters, we fitted the model on all the training data and then applied it to the test data.

2C : Classifier description with Hyperparameter search

In our classifier, we used Multinomial Naive Bayes for several reasons. First, we found that MNB has a history of success in sentiment analysis, and has many successful real-world applications. Secondly, we have seen online that MNB performs well with limited data, which is the case for our model since we only have a data set with a size of 2400 reviews. For our model, we experimented with four different parameters including min_df, max_df, ngram_rang, and alpha values. Among all these parameters, alpha value directly affects the complexity of the MNB classifier and the others influence the feature selection process. For hyperparameter selection, we pursued a customized approach that both involved randomized search followed by grid search. First, we used randomized search on a very wide range of values show below:

RandomizedSearchCV Parameter Ranges:

1. Minimum Document Frequency: [1, 5, 10, 15, 80, 140, 160]
2. Maximum Document Frequency: [0.1, 0.2, 0.4, 0.8, 1.0]
3. ngram_range: [(1, 1), (1, 2), (2, 2)]
4. alpha: [0.05, 0.1, 0.5, 1.0, 2.0]

Then, based on the results from randomized search, we determined the promising hyperparameters, and fine-tuned the range of parameters and performed a GridSearch to find a better model. Below, you can see the range of values which we performed the Grid Search.

GridSearchCV Parameter Ranges:

1. Minimum Document Frequency: [1, 5, 10, 15]
2. Maximum Document Frequency: [0.1, 0.2]
3. ngram_range: [(1, 1), (1, 2)]
4. alpha: [0.05, 0.1, 0.5, 1.0, 2.0]

Because alpha value controls the complexity of the MNB, we focused our analysis on this parameter. For all models with different parameter combinations, we plotted a graph to compare AUROC performance with the alpha values, and below is the graph that we obtained. The average AUROC for models across all alpha values are given by the blue and orange lines for the training and validation data respectively.

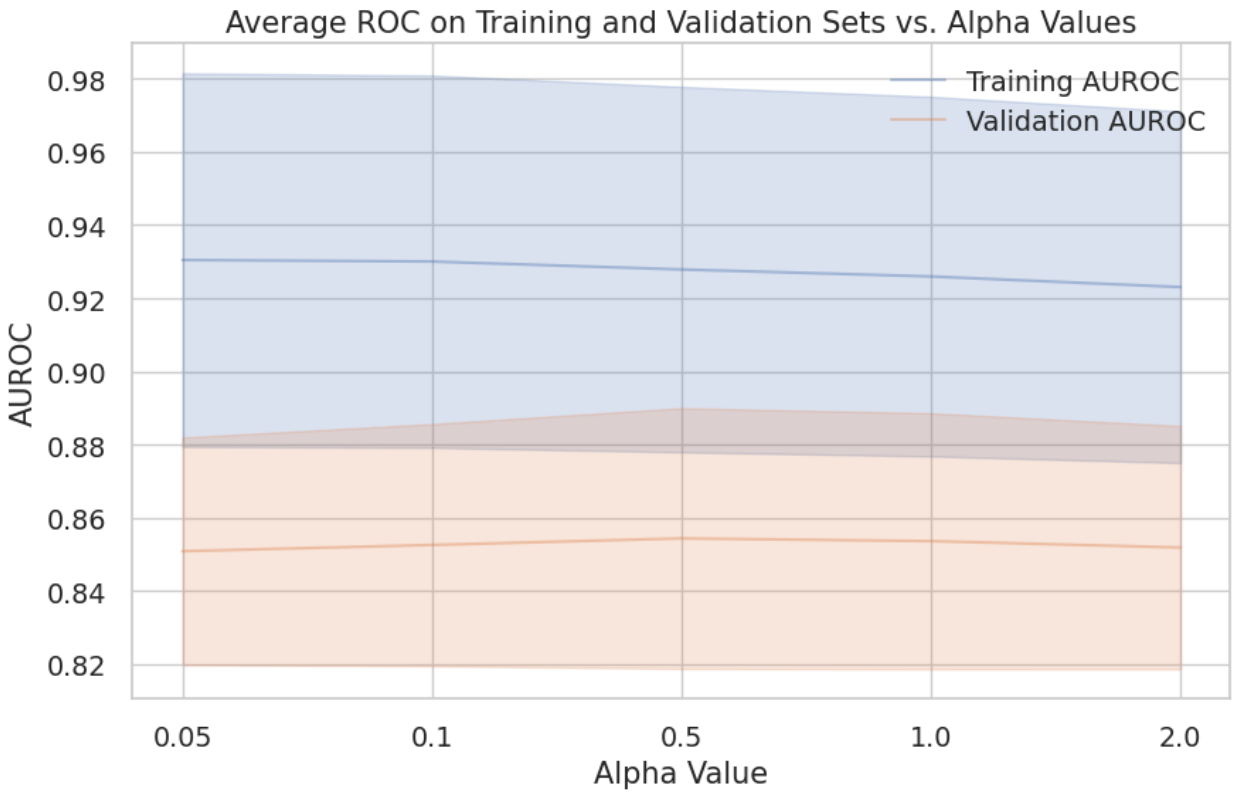

Figure 4. Average ROC on Training and Validation Sets vs. Alpha Value

As you can see, across all combinations of max_df, min_df values, and ngram_range values, the models that have the alpha value of 0.5 produce the best validation result (where the AUROC peak flattens→decreases for validation and increases for training). This value is in the median of the range. Based on this graph, we concluded that an alpha value of 0.5 produces a model that generalized data the best.

2D : Error analysis

For analyzing the predictions, we used the same test data set from above. Below, you can find the confusion matrix on the held out data. We selected 9 reviews from the 12 false positives and 10 reviews from the 16 false negatives which represented a pattern within our results.

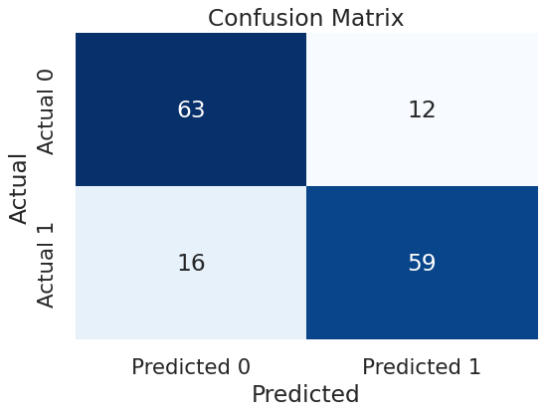

Figure 5. Classifier performance on held out data shown with Confusion matrix

False Positives:

1. It only recognizes the Phone as its storage device.
2. Trying to make a call on these is an exercise in frustration.
3. It also had a new problem.
4. Not as good as I had hoped.
5. Doesn't do the job.
6. This short film certainly pulls no punches.
7. Same evening, him and I are both drastically sick.
8. I got food poisoning here at the buffet.
9. The food wasn't good.

False Negatives:

1. No shifting, no bubbling, no peeling, not even a scratch, NOTHING!I couldn't be more happier with my new one for the Droid.
2. The eargels channel the sound directly into your ear and seem to increase the sound volume and clarity.
3. BLACK WATER is a thriller that manages to completely transcend it's limitations (it's an indie flick) by continually subverting expectations to emerge as an intense experience.
4. Initially the local sites in the film, which was filmed here in Buffalo, intrigued me.
5. I won't say any more - I don't like spoilers, so I don't want to be one, but I believe this film is worth your time.
6. Of course the footage from the 70s was grainy, but that only enhanced the film.
7. I'm a big fan of this series mostly due to Anne Rice's style, sensitivities and treatments.
8. Nicest Chinese restaurant I've been in a while.
9. The owner used to work at Nobu, so this place is really similar for half the price.
10. The menu had so much good stuff on it i could not decide!

Figure 6. Representative examples of false positives and false negatives for best classifier

In figure 6, you can see the examples of false positives and false negatives from the best classifier on held out examples. We observed negation words and the particular kind of words didn’t have a consistent pattern on the test examples provided to the model. Similarly with the classifier in problem 1 we observed that False Positive sentences mostly had a short length (~6 words), including words with negative connotations (e.g. frustration, drastically sick, poisoning) that had low importance (by its Tfidf calculation). In addition, there were some words of positive connotation (e.g., good, new) which

had high importance. For the False Negatives, many of the sentences were longer (~15 words), including words with positive connotations (e.g. happier, transcend, sensitivities) with low importance. Thus, similar to problem 1, shorter sentences with (low importance) negative words failed to show up as negative and longer sentences with (low importance) positive words failed to show up as negative. Despite having an alpha value to help give unrecognized words probabilistic importance, these examples show that the model fails to identify positivity/negativity in contexts where unrecognized words change the meaning of sentences with words labeled opposite to its intended probability or where the sentence is too simple or complex.

2E : Report Performance on Test Set via Leaderboard

Our ultimate test set performance (measured by AUROC) on the leaderboard test set is 0.92542, surpassing our performance on the ultimate test set in Problem 1. Comparing this test set performance to our previous estimates in cross-validation, we found that our model did slightly better on the test set which had 0.91302 average AUROC value. The overall improvement (compared to Problem 1) shows that our second model is more successful than our first model in terms of distinguishing between positive and negative reviews. The reason for this increased performance might be due to several factors. First, for the second model, we use a different feature extraction method built in the TfidfVectorizer. This method assigned importance weights to words based on their frequency in a document and also their rarity across the entire corpus, which downweighted common words that didn't carry much sentiment information. This might have produced more reliable weights assignments for semantics analysis. Secondly, the classifier that we used a much simpler model than Logistic Regression named Multinomial Naive Bayes, which makes it less prone to overfitting, and is able to address sparsity in the text data using the alpha value to ensure that all terms seen contribute to the probability estimates. Lastly, we used a custom approach for hyperparameter selection. Narrowing down the parameters with random search and then applying grid search to explore the best model benefited us in getting even closer to finding the best parameter combination. We believe that these three major changes in our model contributed to our model's performance by around 6 percent.