Project A, CS 135
Deekshith Komira
Fall - 2023
Tufts University

Problem 1: Bag-of-Words Feature Representation

1A: Bag-of-Words Design Decision Description

To determine the best way to clean the data, we tried different techniques sequentially, performed multiple cross-validation iterations with different values of k on the training data, and recorded the average AUC on the validation set. The table below illustrates our experiments.

No.	Method	Validation AUC
1	No Preprocessing	0.8804
2	Lowercase + Correct Spelling + Remove Punctuations	0.8814
3	Lowercase + Correct Spelling + Remove Punctuations + Remove 15 least freq words	0.8814
4	Lowercase + Correct Spelling + Remove Punctuations + Remove 500 least freq words	0.8810
5	Lowercase + Correct Spelling + Remove Punctuations + Remove stop words	0.8744
6	Lowercase + Correct Spelling + Remove Punctuations + Remove 15 most freq words	0.8658
7	Lowercase + Correct Spelling + Remove Punctuations + Remove Numbers	0.8803
8	Lowercase + Correct Spelling + Remove Punctuations + Remove Words of < 3 Characters	0.8817
9	Lowercase + Correct Spelling + Remove Punctuations + Remove Words of < 4 Characters	0.8581
10	Lowercase + Correct Spelling + Remove Punctuations + Binary	0.8813
11	**Lowercase + Correct Spelling + Remove Punctuations + remove less than 3 length words + Binary**	**0.8821**

Table 1: Results of different experiments conducted to determine the best preprocessing technique[1]

To get a **baseline** to compare any preprocessing methods we wanted to test with, we calculated the AUC without conducting any preprocessing on the data and calculated a cross-validation score of **0.8804**.
Next, we implemented some basic preprocessing steps - **converting the text to lowercase, correcting spelling, and removing punctuation (2 in Table 1)**. This improved our cross-validation score to **0.8814**. We expected this since uppercase, wrongly spelled words, and punctuation do not contribute to the sentence's sentiment.
Next, we tried multiple experiments (**3 to 7 in Table 1**) and observed that the more information we removed from the texts, the worse the cross-validation scores were. Next, we tried to remove information from the text minimally. We determined that words with less than three characters (like I, a, an) do not contribute to the sentence's sentiment, so we removed them. We did the following - **converted the text to lower case, corrected spelling, removed punctuation, and removed words having less than three characters (8 in Table 1)** and got a slight increase in the cross-validation score, **0.8817**. To further take this idea, we tried removing the words with less than four characters (**9 in Table 1**) (like I, a, an, the, not), but we saw a significant drop in the cross-validation score. This again points to our reasoning that removing more information leads to bad performance.
Finally, we tried the idea of having binary vectors for the texts in the feature representation matrix instead of having the count of occurrences. By converting to binary vectors, less common words are treated the same as

[1] For the first experiment, we tried the C value across 71 log-spaced values between 10^-6 and 10^6 and determined the optimal value to be close to 1. Hence, for the next set of experiments, we use a smaller range (0.01, 0.1, 1, 1.99, 3, 5) to speed up our experimenting.

frequently used words, which lowers the impact of outliers or other irrelevant words. We applied this technique to our candidate ideas from our previous experiments (**2 and 8 in Table 1**) and got better performance (**10 and 11 in Table 1**).
Individually testing different preprocessing methods allowed us to test their functionalities and understand how each change affected the cross-validation score. We learned that, for the most part, applying minimal modifications to the dataset allowed us to achieve a better cross-validation score. Ultimately, we decided to clean the data using the following: **converted the text to lowercase, corrected spelling, removed punctuation, and removed words having less than three characters**. After obtaining the cleaned texts, we used the **binary feature representation** for our modeling process (**11 in Table 1**). We got our vocabulary set after cleaning the data, and our vocabulary set had **4391 words**. We used the same vocabulary set with 4391 words for the test data to get the binary feature representation; hence, the out-of-vocabulary words in the test data were ignored.

1B: Cross Validation Design Description
We did a K-Fold cross-validation technique using the training data to determine the best hyperparameter value (the value of C in logistic regression). We tried the values of **K ranging from 5 to 10**. We reasoned that this is the optimal range of K values that we can use since when K = 5, 80% of the data is used as training and the rest 20% for testing in each iteration, and when K = 10, it is 90% for training and 10% for testing. This range of splits of the whole training data is a common practice in the machine learning modeling process.
Since our final model will be evaluated based on AUC, we use **AUC on the validation set as the performance metric** for searching the optimal hyperparameter using the cross-validation process. We took the average value of AUC obtained for the different values of K we tried (5 to 10) and chose the C value corresponding to the highest average value of AUC on the validation set (**Figure 1 in 1C section**).
We used **sklearn's cross_validate function** to do the whole cross-validation process. This function returns the train and validation scores for the metric chosen and the K values we selected.
After using the abovementioned cross-validation process to determine the optimal value of the hyperparameter (the value of C in Logistic Regression), we use the **whole training data** with the optimal value of C **to train the final model**.

1C: Hyperparameter Selection for Logistic Regression Classifier
Since in this problem, we were asked to use logistic regression, we implemented the same. We think that starting off with logistic regression is a good choice since it is a **simple model** to implement and will serve as a **good baseline** to compare our future explorations. For the logistic regression model, we searched the **C hyperparameter**, an L2-regularization penalty on the weights, across **71 log-spaced values between 10^-6 and 10^6**. While doing this process, we used the **lbfgs** as the **solver** and **maximum iterations** to be **10000**. We use the default option for the solver since it is a standard option. The default value for maximum iterations is 100, but for certain values of C, we were facing convergence issues, so we increased this to 10000, a value at which we didn't face any convergence issues for the C values that we tried.
The hyperparameter C is the inverse of the regularization strength. It helps to keep the model's coefficients small, making sure that we are not overfitting the training data. Higher values of C correspond to less regularization, while lower values result in more regularization. **This translates to very high values of C will lead to overfitting, and very low values of C will lead to underfitting**. We go with the range mentioned above since log-spaced allows us to cover a wide range of values and hence go from a very low value of C (10^-6) to a very high value of C (10^6). Given the wide range of values, we anticipate that at C = 10^-6, there will be underfitting, and at C = 10^6, there will be overfitting, and we should be able to find the sweet spot in between. Below are the figures from our comprehensive analysis using the K-Fold cross-validation technique to find the optimal value of C for the logistic regression model.

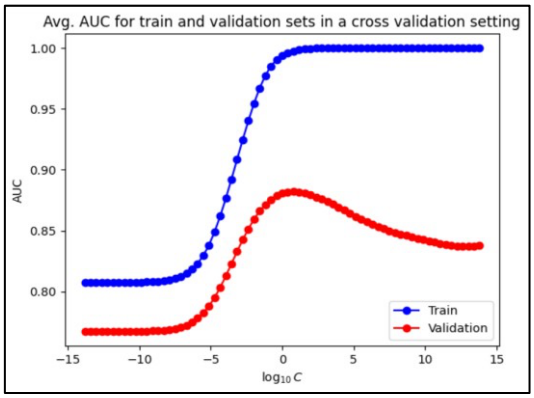

Figure 1: Avg. AUC for training and validation sets in the K-Fold cross-validation process. We use this figure to determine the optimal value of C.

From **Figure 1**, we get the maximum average AUC on the validation set at **C = 2.2022 ($\log_{10}$C = 0.343**), with the **AUC = 0.8820**. There are two uncertainties associated with this value – the first one being that this is the value determined by taking the average AUC across all the K values (5 to 10) we tried and the second one being that for each value of K, the AUC may differ across the different folds. To understand these uncertainties, we plot the two figures below.

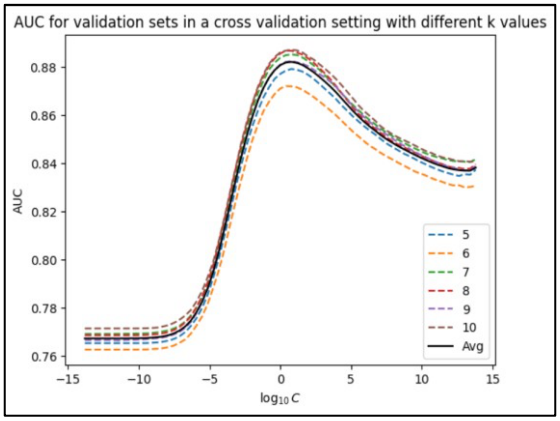

Figure 2: AUC on the validation sets for different values of K (5 to 10). Includes the average line.

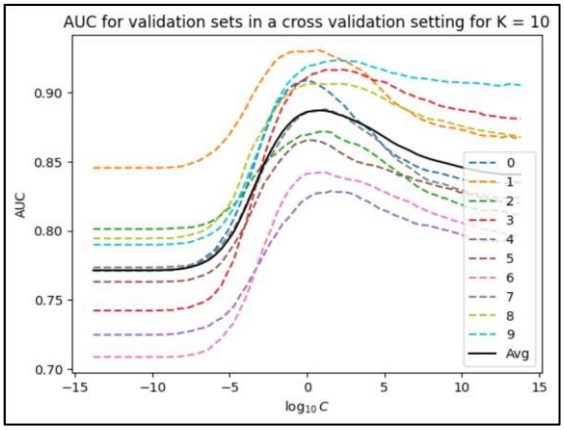

Figure 3: AUC on the validation sets of different folds in a 10-fold cross-validation. Includes the average line.

Figure 2 is for the **first uncertainty**. We observe that for all the different values of K that we tried, the AUC on the validation set peaks at the same C value. From this plot, we determined that the AUC is highest for K = 10. Using this information, we plot **Figure 3** to understand the **second uncertainty**. This figure shows the validation AUC for the ten folds created as part of the 10-fold (K = 10) cross-validation. Again, here, we observe that, for each fold, the validation AUC peaks at approximately the same C value. Hence, based on these plots, **we conclude that the value of C = 2.2022 is decisive with very little uncertainty**.

1D: Analysis of Predictions for the Best Classifier

True value: \ Predicted Value:	0	1
0	94	23
1	25	98

Table 2: The number of FN, FP, TN, and TN from a sample of 240 predictions

After analyzing the test data by looking at different elements of the reviews and their predictions, we concluded that the classifier is generally **not affected by the length** of the review. Instead, it is **affected more strongly by the type of review and existence of negation words** such as "not" or "shouldn't". We compared the distribution of the length of reviews for the whole corpus with the distribution of the length of reviews for those that were incorrectly predicted. Because both distributions reflect the same shape.

From our calculations, **17.1%** of the amazon reviews, **15.9%** of the Yelp reviews, and **26.1%** of the imdb reviews were incorrectly classified. Therefore, the imdb reviews are more susceptible to an incorrect prediction than amazon and yelp reviews. We hypothesize that the reason for this is that imdb reviews are more unique compared to amazon and yelp reviews, as they contain more specific information about movies, characters, and plot compared to the Amazon and Yelp reviews. An example of a **false positive** review from imdb is "**And, FINALLY, after all that, we get to an ending that would've been great had it been handled by competent people and not Jerry Falwell.**" An example of a **false negative** is "**It is rare when a film-maker takes the time to tell a worthy moral tale with care and love that doesn't fall into the trap of being overly syrupy or over indulgent.**" These examples showcase that niche information is leading the statement to be falsely classified. Furthermore, we looked into the effect of negative words on whether the classifier falsely predicts the outcome. From our analysis, we concluded that in longer sentences, the classifier falsely predicted reviews with negative words. However, in shorter sentences, the negative words appear to not incorrectly predict the outcome of the review. This is because in longer sentences, a string of "positive" words can be prefaced with "not", meaning that they are all supposed to be translated as negative. However, in this case, the classifier only recognizes one negative word and several positive words since we just consider unigrams in our feature representation.

1E: Report Performance on Test Set via Leaderboard

The AUC on the test set obtained is **0.8829**. This value is very close to the validation's AUC (0.8820). Since the test data is unseen, very similar values establish that our model did not overfit in the cross-validation process.

Problem 2: Open-ended challenge

2A: Feature Representation description

Because of the meticulous testing that we conducted to determine the effectiveness of specific modifications to the dataset and the results that we obtained from this testing, we concluded to **clean the dataset** in Problem 2 using the following: **converted the text to lowercase, corrected spelling, removed punctuation, and removed words having less than three characters**. However, where our feature representation differs significantly is when we implement the **use of TF-IDF** in replacement of the B.o.W representation. TF-IDF assigns each word in a document a score representing its significance and is calculated by taking the term frequency in a particular document and its rarity across the entire corpus into consideration. This differs from the bag of words approach, which simply showcases whether a word occurs in a sentence or not. We saw an increase in the cross-validation AUC after implementing TF-IDF for the logistic regression model due to its ability to weigh words based on their importance and reduce the impact of frequent and unimportant words without changing the text itself. We got our

vocabulary set after cleaning the data, and our vocabulary set had **4391 words**. We used the same vocabulary set with 4391 words for the test data to get the binary feature representation; hence, the out-of-vocabulary words in the test data were ignored.

2B: Cross Validation (or Equivalent) description

We use the same design as described in the 1B section for our cross-validation process with two changes. The first one is that we only try values of **K ranging from 7 to 10** based on the observation in the previous problem that for K = 7 to 10, we see very similar validation AUC values, but for 5 and 6, it is quite different (Figure 2). The second change here is that since we decided to use **Multilayer Perceptron (MLP)**, the **hyperparameter** that we search using the cross-validation process is the **number of neurons in the hidden layer**. We took the average value of AUC obtained for the different values of K we tried (7 to 10) and chose the number of hidden units corresponding to the highest average value of AUC on the validation set (**Figure 5 in 2C section**).

2C: Classifier description with Hyperparameter search

For this problem, we use the **Multilayer Perceptron (MLP)** model. For tabular data, tree-based models like random forest and XGBoost are the proven choices. **But when it comes to textual data and images, neural networks are the preferred algorithms, with many proven examples published**. Hence, for this reason, we go with the most simple form of neural networks – MLP for this problem. We tried simple implementations of tree-based methods and did not see any improvement over the logistic regression model implemented as part of problem 1. The below figure shows the architecture of our MLP model.

```
model = Sequential()
model.add(Dense(units, input_dim=n_features, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['AUC'])
```

Figure 4: Multilayer Perceptron (MLP) model architecture

The first layer is the input layer, followed by a fully connected **hidden layer with a variable number of neurons, which we optimize by the process of cross-validation**. The hidden layer uses **ReLU** as the activation function and also includes an **L2-regularization** penalty on the weights of this layer, with **lambda = 0.01**. The hidden layer is followed by an **output layer**, with one unit using the **sigmoid activation function**, giving out values between 0 and 1. Standard values for a binary classification problem are used for the loss and optimizer parameters. Since our final model will be evaluated based on AUC, we use **AUC as the metric.**

The hyperparameter that we search for is the **number of neurons in the hidden layer (hidden units)**. Low values of neurons translate to a simple model, and hence, we expect the model to underfit. High values of neurons equate to a complex model, and hence, we expect the model to overfit. **The number of neurons in the hidden layer that we try are 8, 16, 32, 64, 128, 256, 512**. Usually, in practice, less than eight neurons are not used since, with a very small number of neurons, the model will not be able to learn patterns in the data. To start with, 512 seemed a good upper limit, and based on a few rounds of training, we observed that the train AUC reached close to 1; hence, increasing this number further did not make sense. Empirical observations and heuristics have shown that increasing the number of hidden units by a multiple of 2 is a good strategy. Hence, we follow the same. The below figures are from our comprehensive analysis using the K-Fold cross-validation technique to find the optimal value of hidden units for the MLP model.

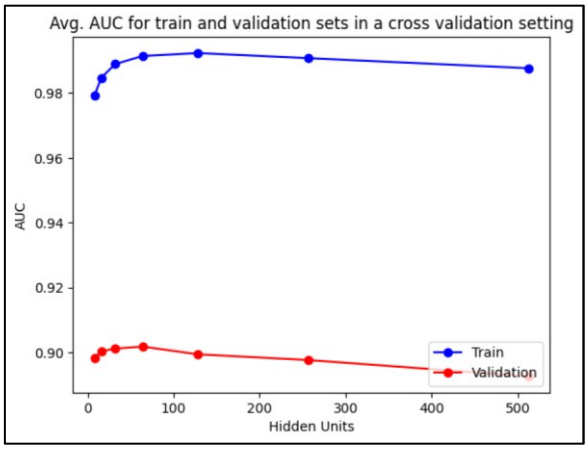

Figure 5: Avg. AUC for training and validation sets in the K-Fold cross-validation process. We use this figure to determine the optimal value of hidden units.

From **Figure 5**, we get the maximum average AUC on the validation set at **Number of Hidden Units = 64**, with the **AUC = 0.9018.** As explained in 1C, there are two uncertainties associated with this value.

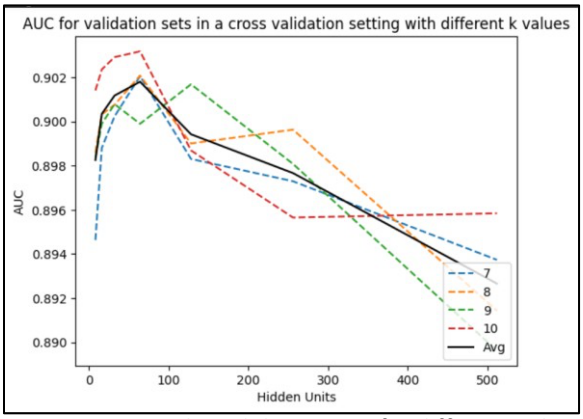

Figure 6: AUC on the validation sets for different values of K (7 to 10). Includes the average line.

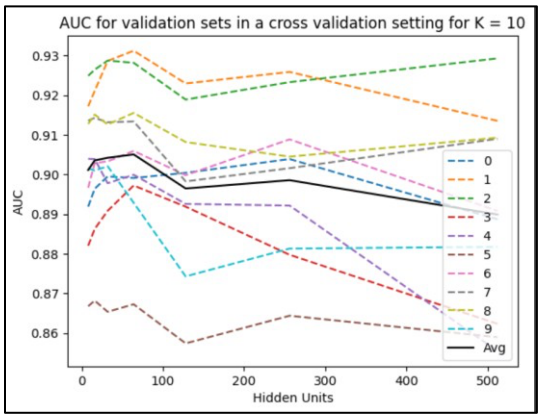

Figure 7: AUC on the validation sets of different folds in a 10-fold cross-validation. Includes the average line.

Figures 6 and 7 are for the **first and second uncertainties** respectively. We observe that for all the different values of K that we tried, the AUC on the validation set peaks at the same number of hidden units except for one. From figure 7, again, here, we observe that, for each fold, the validation AUC peaks at the same number of hidden units except for one. Hence, based on these plots, **we conclude that the value of 64 is decisive with very little uncertainty**.

2D: Error analysis

Predicted Value: / True value:	**0**	**1**
0	106	11
1	36	87

Table 3: The number of FN, FP, TN, and TN from a sample of 240 predictions

After analyzing the test data by looking at different elements of the reviews and their predictions, we maintained the conclusion that the classifier is **generally not affected by the length** of the review and is **instead affected by the type of review and existence of negation words** such as “not” or “shouldn’t”. Similarly, to the method we used to check the effectiveness of length in problem 1, we compared the distribution of the length of reviews for the whole corpus with the distribution of the length of reviews for those that were incorrectly predicted. Through this, we saw that both distributions are alike, therefore highlighting that the length of reviews does not affect the classifier’s accuracy. In this test set, **15.1%** of the Amazon reviews, **17.1%** of the Yelp reviews, and **25.0%** of the imdb reviews were incorrectly classified. From these statistics, it is clear that the imdb reviews are significantly more prone to being incorrectly classified compared to the Amazon and Yelp reviews. However, unlike the results from problem 1, we saw a significant difference in the confidence at which the reviews are classified to be incorrect. In problem 2, they were assessed slightly more accurately than in problem 1. For example, in problem 1, the **FN** review “**Highly recommended for all ages, although the younger set will probably not appreciate some of the more subtle references, they will certainly appreciate one galley scene in particular!**” had a predicted probability of **0.039**. However, in problem 2, although it was **still a FN value**, it had a predicted probability of **0.343**, which is significantly higher in the right direction. **This shows that the whole pipeline in problem 2 became more accurate, even though it couldn’t cross the line and make the right prediction**. This difference in confidence is more strongly evident in longer sentences than in short. We feel that this is due to the TF-IDF feature representation, which reduces the noise from unnecessary values in the longer sentences and helps more significant keywords be weighed more heavily and the use of an enhanced classifier.

2E: Report Performance on Test Set via Leaderboard

The AUC on the test set obtained is **0.8930**. This value is very close to the validation's AUC (0.9018). Since the test data is unseen, very similar values establish that our model did not overfit in the cross-validation process.