Leigh Tanji, Avtar Rekhi

Question 1: Bag-of-Words Representation

1A. Bag of Words Design Decision Description

The data was pre-processed using the CountVectorizer function in the SciKit Learn library. **All punctuation in the text was replaced with a space**, and **all letters were set to lower-case**. Then, each review was split into tokens, using spaces as delimiters. The default tokenization function was used, which discarded words that were one character long, such as "a" and "I". Numbers, for example "two" and "2", were left as they were and not converted to their word or numerical forms.

A dictionary was assembled from the tokens, and a count of how frequently each token occurred across reviews was created. The dictionary was then reduced according to frequency limits set. These limits were initially chosen by considering which words would be filtered out at different threshold frequencies was altered, which suggested a max_df of 0.08 would remove many words with while ensuring that no words with semantically positive or negative meanings were removed. However, upon evaluating the AUROC of the heldout folds, **the optimum frequency values of 2 occurrences for min_df, and 1.0 (off) for max_df were selected**.

The final vocabulary of the bag of words representation contained 1912 unique tokens. This vocabulary was used in later steps to generate feature vectors for input text. **Any words not in this vocabulary list are ignored** and do not contribute to either fitting a model or using one to classify reviews.

1B. Cross Validation Design Description

Cross validation was done across **5 folds, where each fold had 480 reviews**, as 5 or 10 folds is the standard for machine learning.

We used Scikit Learn's RandomizedSearchCV function to perform cross validation, hyperparameter selection and classifier training. **This shuffled and divided the training data into 5 folds**. It then created **100 random combinations** of the penalization, regularization strength and maximum iteration hyperparameters that are used for logistic regression, as described in section 1C.

For each of the 100 sets of hyperparameters, the function uses **cross-validation on a logistic regression model and assesses the mean AUROC across the 5 folds** for said model. Hyperparameter selection was conducted by selecting the combination of hyperparameter that resulted in the highest mean AUROC across the 5 folds.

The function ultimately outputs the logistic regression model with the selected hyperparameters. **This resulting model is trained using the entire training set**.

1C. Hyperparameter Selection for Logistic Regression Classifier

The scikit-learn implementation of logistic regression model using the liblinear solver was selected due to the limited size of the training data set available, as it is recommended for small data sets. The following hyperparameters were considered for this model - the penalization type, regularization strength (C), and max iterations (max_iter).

In order to perform hyperparameter selection and model fitting within 1 minute, RandomizedSearchCV was used to explore the the hyperparameter space. This function randomly selected 100 combinations of the hyperparameters across the following ranges: the penalties considered were **L1 and L2**, while C and max_iter were logarithmically distributed - **300 log-spaced values across** 10^{-9} **to** 10^{6} and **200 log-spaced values across 1 to 50000**, respectively. Section 1B describes the details of how cross validation is performed and the combination of hyperparameter that results in the greatest mean AUROC on held-out data across the 5 folds are output. The best combination of hyperparameters from this step was subsequently used as a baseline for further refinement.

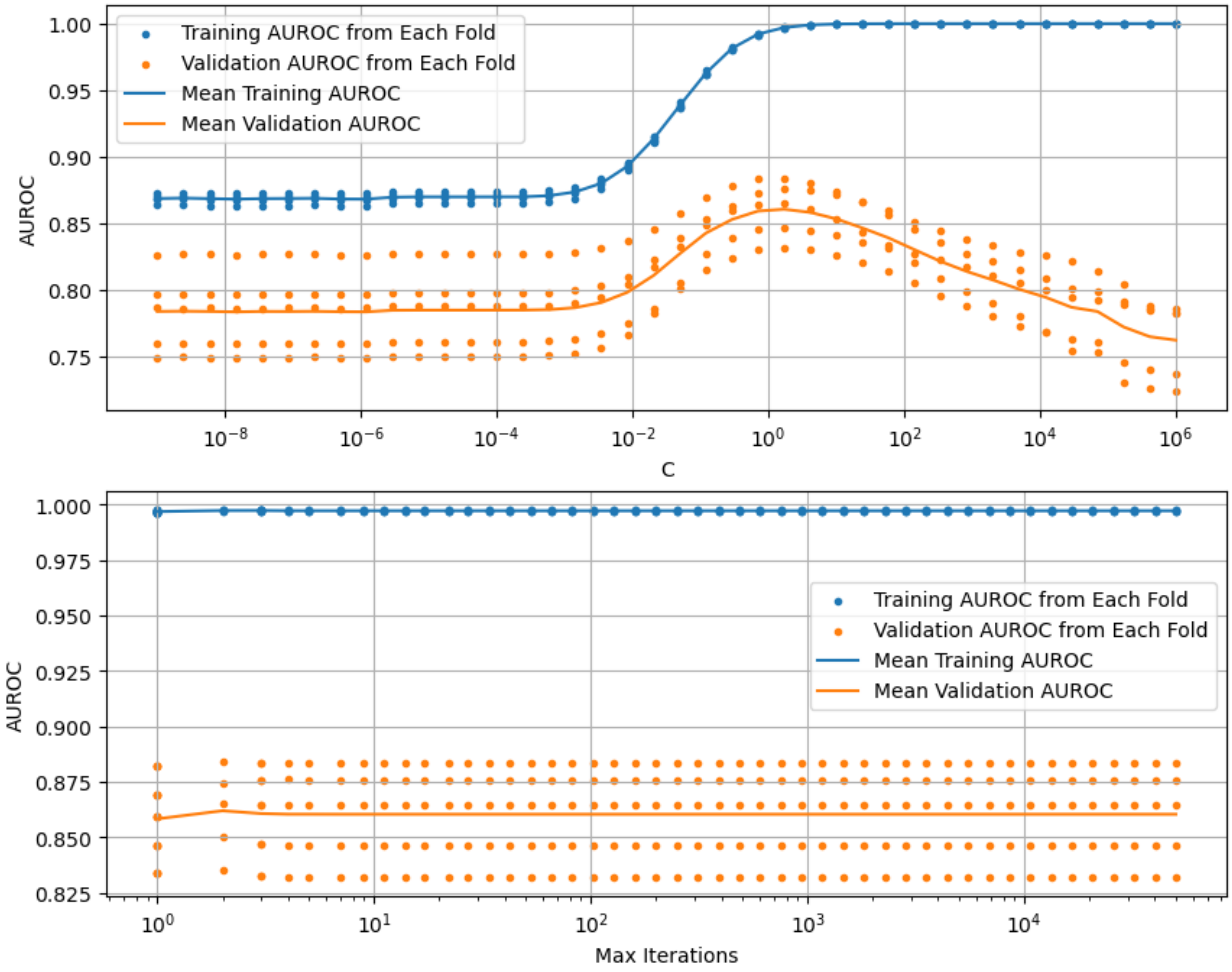

Figure 1: AUROC against hyperparameter values for model one

As the RandomisedSearchCV had covered the majority of the hyperparameter space, **an exhaustive search on a smaller range was able to hone in on the optimum hyperparamaters**. GridSearchCV was used for this to vary only one hyperparameter at a time, and using the best performing values from the RandomisedSearchCV for the others. The parameter were distributions for this varied C as **200 log-spaced values between 10^{-2} and 10^{2}**. The penalty term was allowed to vary between **L1 and L2**. The max_iter was constrained to **50 log-spaced values between 80 and 200** as a plot of AUROC as max_iter varies showed that it did not contribute to the complexity of the model beyond 3 iterations. **A floor of 80 was selected as this allowed our models to converge within the iteration limit**. The combination that led to the highest mean heldout AUROC score was selected, specifically a **L2 penalisation with C of 1.703 and max_iter of 98**. While max_iter could be increased with little impact, the C selected was decisively the optimum value from the ranges evaluated.

Figure 1 is a simplified representation of the grid search, conducted over the wider range to show the effect of each hyperparameter on AUROC. On the top plot, it is clear that a **low C value leads to underfitting**, where large weights are penalized too much, preventing the performance on both the training and heldout data sets from improving. In contrast, **a high C is shown to lead to overfitting**, where the training data performance is high but the heldout data performance decreases. **There is an optimum zone in the range of $C = O(1)$ where the AUROC on the heldout set is maximised**.

The bottom plot repeats this for max_iter. In this case, while there is a small variation when increasing from a max_iter of 1, this plot shows that further increases do not have an appreciable impact on the AUROC score, and hence we can infer that max_iter does not significantly impact model complexity.

1D. Analysis of Predictions for the Best Classifier

		Examples			
Category	Freq.	y	$\hat{y}$	$\hat{p}$	Review
Clear	25.81%	1	0	0.421	"Just what I wanted."
Negated	22.58%	0	1	0.505	"It is not good"
Double Neg	22.58%	1	0	0.274	"You won't be disappointed"
Confusing	9.68%	0	1	0.669	"Excellent starter wireless headset."
Witty	9.68%	1	0	0.230	"Waste your money on this game"
Sarcasm	3.22%	0	1	0.500	"The loudspeaker option is great, the bumpers with the lights is very ... appealing."

Table 1: Examples of errors made by the model by category. Freq indicates the fraction of mistakenly predicted reviews that belonged to the corresponding category. For each review, its corresponding true label (y), predicted label ($\hat{y}$), and prediction probability ($\hat{p}$) is listed.

The chosen model had more difficulty predicting positive reviews compared to negative review overall. **Out of the 31 prediction errors, 70.97% were false negatives and the remaining were false positive**.

All the reviews that were mislabeled were categorized into six error types in Table 1:

- Clear - Reviews that were evidently positive or negative.
- Negated - Reviews that included a word or phrase that negated the positive sentiment afterwards.
- Double Negative - Reviews that had double negative phrases.
- Confusing - Reviews with unclear sentiments from a human's perspective.
- Witty - Reviews that had had jokes about how the product was a "good waste of money" or "belongs in the garbage".
- Sarcasm - Reviews that clearly meant the opposite of its literal meaning.

A quarter of the errors came from reviews that had a clear negative or positive sentiment. Considering that our model had higher rates of false positives and all the $\hat{p}$ associated with this category were in the 0.4 range, this may be because there were one or two words with very strong negative weights that pushed the label from being positive to negative.

Although the rest of the categories grouped different types of errors made by the model, the mistakes can be explained as shortcomings of the "bag-of-words" model. Because individual words were given weights regardless of the context, reviews with obviously negative words (e.g. "terrible", "bad") were likely to be given negative labels even when it was negated. This hypothesis is supported by the lower $\hat{p}$ values associated with the Double Negative vs. Negated categories. Additionally, the Confusing category, which contained reviews that could have been categorized either way by a person, had $\hat{p}$ that were seemingly random. Finally, the Witty group had labels that matched the mood of the sentence that could only be identified as the opposite sentiment due to subtle phrasing. For example, "waste your money" would be interpreter as negative if the word "waste" was not a command in this case.

1E. Report Performance on Test Set via Leaderboard

The **leader board AUROC score on the test set was 0.892, which was higher than the heldout AUROC performance at 0.873.** This could be because the five-fold cross validation tunes the parameters based on only a part of the training data, while the test set score is based on an estimator created using the whole training set.

Question 2: Open-Ended Challenge

2A. Feature Representation Description

The final dictionary contained 2,565 lemmatized, unigrams and bigrams that appeared at least once in the training set.

All words in the reviews were **made into lower case and lemmatized** using the NLTK's WordNetLemmatizer to allow different versions of the same words to be grouped together. These updated reviews were fed into CountVectorizer that **tokenized by delimiting on white space**. This avoided the issue from problem one, where contractions were separated by punctuation. For example, "didn't" became "didn" and "t". Finally, any words that appeared at least once (min_df) were kept since the AUROC score decreased with increasing min_df.

Other preprocessing methods that were abandoned, included parsing words by parts-of-speech and excluding stop words. NLTK's universal POS tags were used to label each word as nouns, verbs, adjectives, etc. in hopes of removing proper nouns, misspellings, and translating numbers into words (e.g. 2 $\rightarrow$ two) However, there were too many tokens that were mislabeled. Excluding NLTK's stop words was also tried. Unfortunately, attempting this change led to the AUROC score dropping by 0.7. Therefore, stop words were ultimately included in the dictionary.

2B. Cross Validation or Equivalent Description

This process was mostly the same as problem one. Again, RandomizedSearchCV was used to **conduct cross validation across 5 shuffled folds** (480 reviews each), hyperparameter selection, and classifier training. 100 random sets of hyperparameters were assessed with this method and the mean AUROC was produced.

2C. Classifier Description with Hyperparameter Search

The process for choosing a classifier and performing hyperparameter selection is largely unchanged from problem one. Hyperparameters for C and max_iter were selected for a logistic regression model via the use of the RandomizedSearchCV function. It evaluated 5 fold cross validation on 100 hyperparameter combinations randomly distributed across **200 log-spaced values between** 10^{-6} **and** 10^{6} for C and **100 log-spaced values between 1 and 5000** for max_iter. Subsequently a grid search refined the ranges, assessing **200 log-spaced values between** 10^{-1} **and 200** for C and **50 log-spaced values between 80 and 300** for max_iter. The combination that led to the highest mean heldout AUROC score was selected, specifically a **L2 penalisation with C of 80 and max_iter of 200.**

The primary difference in this implementation is the generation of the bag of words, the cleaning and tokenization, and feature vectorization applied to inputs, as discussed in section 2A.

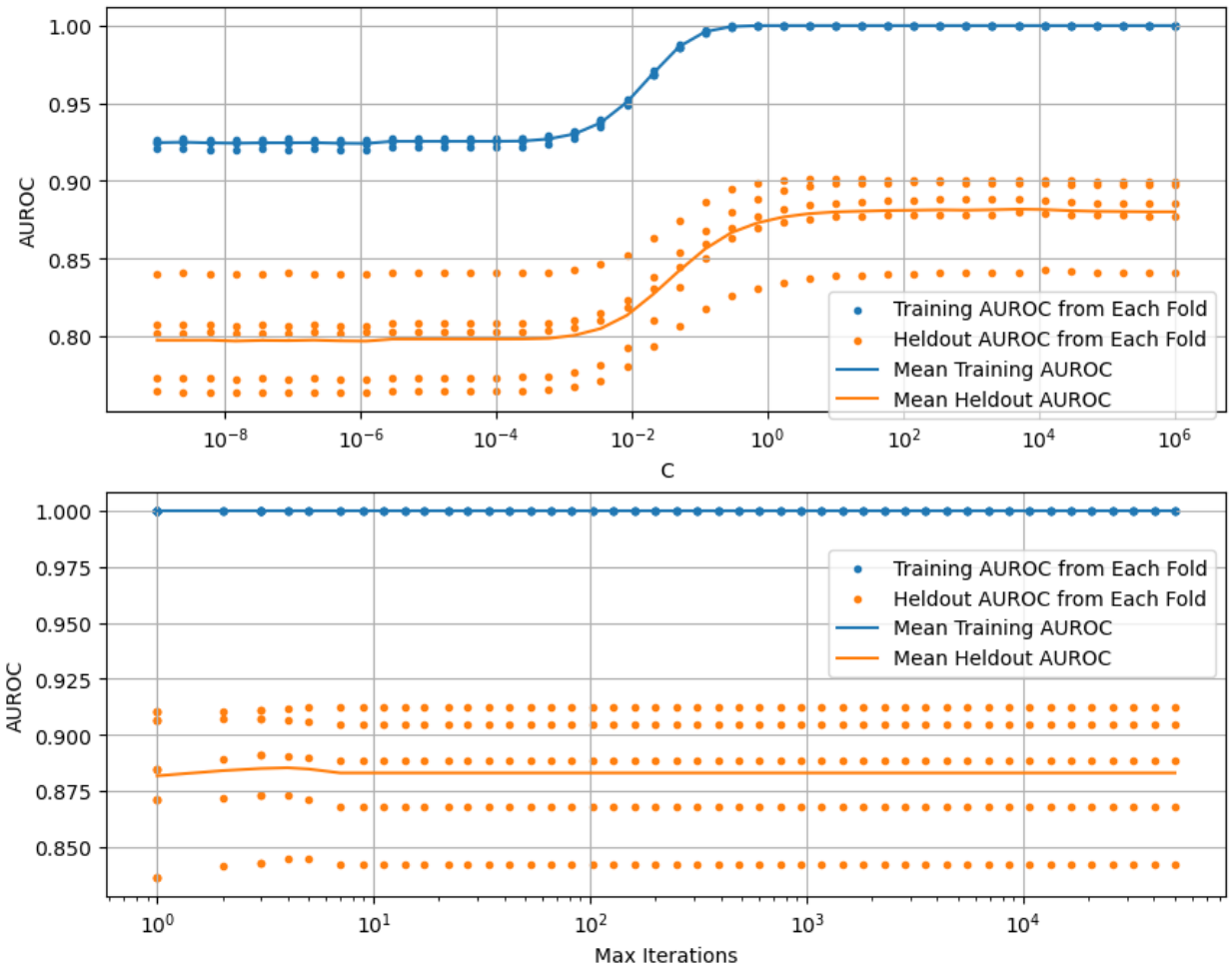

Figure 2: AUROC against hyperparameter values for model two

Figure 2 shows a representation of how the AUROC scores vary with respect to changes in hyperparameters. As with problem one, max_iter was found to not alter the model complexity significantly, however a value of 250 was selected to ensure convergence was achieved.

The plot of AUROC against C shows the underfitting expected at low C values, however it does not show a reduction in heldout AUROC as C tends to large values. This is an indication that when the C value is high enough that the model weights are equivalent to when they are not being penalised.

2D. Error Analysis

The resulting model correctly classified all reviews from the heldout data set, with no false positives or negatives.

However, running predictions on the test data shows that this model does misclassify reviews for many of the same reasons as the original model, as shown in Table 2. However, notably it appears to perform better for double negatives, with very few misclassifications of this category. This can be explained by the inclusion of bigrams, which would allow many double negatives to be correctly weighted.

Based on the performance of the earlier model, it is likely that the training data set is a good representation of the the test data set. As such, the excellent performance of this model at classifying the training data, and the poorer performance at classifying the test data is a strong indicator that this model is overfit.

	Examples			
Category	y	$\hat{y}$	$\hat{p}$	Review
Clear	1	0	0.484	"Does everything it should and more."
Negated	0	1	0.984	"Not a good bargain."
Double Neg	1	0	0.000	"I went to Bachi Burger on a friend's recommendation and was not disappointed."
Confusing	0	1	0.669	"When a song could explain the emotions of the subjects better, such as when Jay Adams' unfortunate life was a subject of talk, the song Old Man by Neil Young was played, which evokes many emotions."
Witty	0	1	0.553	"They brought a fresh batch of fries and I was thinking yay something warm but no!"
Sarcasm	0	1	0.500	"The loudspeaker option is great, the bumpers with the lights is very ... appealing."

Table 2: Examples of errors made by the model by category. For each review, the true label (y), predicted label ($\hat{y}$), and prediction probability ($\hat{p}$) is listed. Note that the true label (y) is the assumed sentiment as labels are not provided

2E. Test Set Performance

Our problem two model performed slightly worse than that of problem one's. The final AUROC score for problem one's model and problem two's model was 0.8915 and 0.8466, respectively. Since the second model had 0 training error, which is most likely over fitting to the test set and does not generalize well. Although both models were produced using pipelines that should have prevented data leaks and handled the uniform creation of the models, there may have been bugs during the preprocessing that encouraged overfitting.