CS135: Project A Report

Will Cusato and Constantine Moseley

August 13, 2026

Problem 1

1A: Bag-of-Words Design Description

Preprocessing of the data was achieved using `scikit-learn`'s `CountVectorizer`. The only preprocessing applied was making all characters lowercase via `CountVectorizer`'s built-in `lowercase` option; no punctuation stripping or other cleaning was performed, as we found simpler preprocessing to be sufficient. The vocabulary was determined using the training set within each fold of cross-validation via `CountVectorizer`'s `min_df` and `max_df` parameters, meaning the vocabulary size varied across folds. Hyperparameter selection revealed that unrestrictive values of `min_df` = 0.0 and `max_df` = 1.0 yielded the best results, producing a vocabulary that ranged from 25,400 to 25,700 words, depending on the training fold. This large vocabulary ensures that a representative portion of words in the test set are captured. Out-of-vocabulary words encountered at test time are ignored, as `CountVectorizer` only encodes words present in the training-fold vocabulary. Word counts rather than binary indicators of occurrence are used, motivated by the desire to retain as much information as possible: how frequently a word appears may carry semantic signal beyond mere presence or absence.

1B: Cross Validation Design Description

All cross-validation was performed using `scikit-learn`'s `KFold` with 10 folds, a shuffled split, and a random seed of 12345 to ensure reproducibility. Stratified K-fold was considered but deemed unnecessary, as upper-level texts comprise approximately 55% of the training set, making the class balance sufficiently even. With 10 folds, each validation fold contains about 550 texts, roughly 10% of the training data. The performance metric used throughout is AUCROC, evaluated on both training and validation folds; validation AUCROC guides hyperparameter selection for generalization, while comparison to training AUCROC helps diagnose over- and underfitting.

Two sequential cross-validation stages were conducted. The first performed a grid search over pairings of `min_df` and `max_df`. Because we anticipated a codependent relationship between these hyperparameters, their interaction was visualized via a heat map of validation AUCROC. During this stage, C, and the number of iterations, n, were fixed at their defaults ($C = 1$, $n = 1000$) to avoid confounding the search. The second cross-validation stage performed a grid search over values of C and the maximum number of solver iterations, using the best `min_df` and `max_df` from the first stage. Because the L-BFGS solver is incompatible with `scikit-learn`'s `warm_start` feature, we implemented a separate pipeline for each candidate value of n rather than iteratively training one model.

After both cross-validation stages, we used the best-performing combination of all four hyperparameters—`min_df`, `max_df`, C, and n—to train one final pipeline on the full training set, which is then applied to produce binary classifications on the held-out test set.

1C: Hyperparameter Selection

Logistic regression is well-suited to the task of classifying texts using a bag-of-words feature representation because it provides an interpretable model with intuitive hyperparameters. We first performed a grid search for `min_df` and `max_df`, taking $C = 1$ and $n = 1000$, as described above. We explored a small range of `min_df` values, under the belief that the most substantive words for classifying text only appear in a small number of documents. We tested the each of the values in the set $\{0.0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. For `max_df`, we tested a broader set of values: $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Importantly, the greatest value of `min_df` tested cannot exceed the minimum value of `max_df` tested, as such a configuration would yield an empty vocabulary. For each pair of values, we computed the training and validation AUCROC. The AUCROC results are visualized as heatmaps in Figure 1.

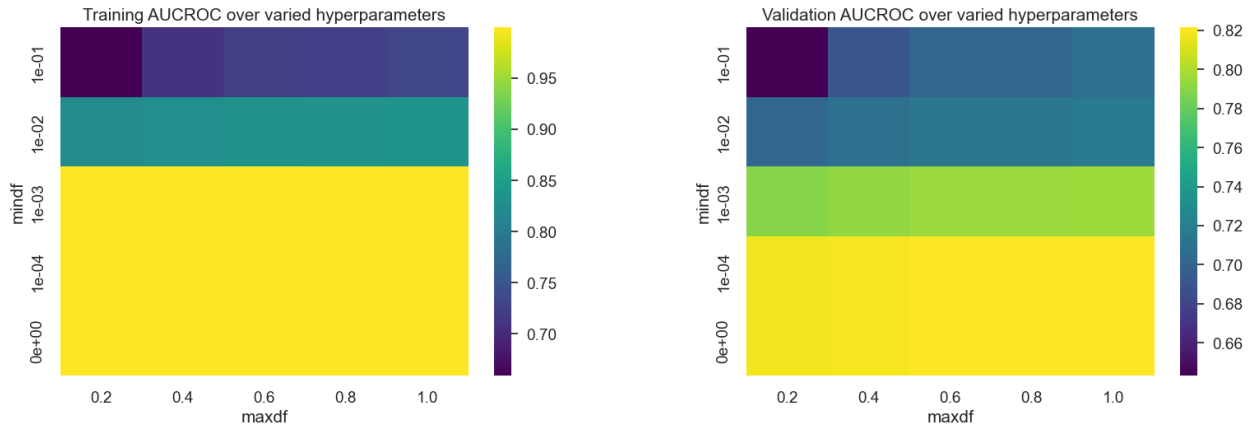

Figure 1: Training (left) and validation (right) AUCROC heatmaps over the `min_df` and `max_df` hyperparameter grid

For both training and validation predictions, as `min_df` decreases, AUCROC increases. Meanwhile, increasing `max_df` consistently increases model performance on the validation dataset. We found that the best-performing `min_df`–`max_df` pairing on the validation set was (0.0, 1.0), corresponding to a complete vocabulary.

Using our optimal `max_df` and `min_df`, we then performed another grid search over the penalization parameter C and the number of solver iterations n. We tested 9 C values that were logarithmically spaced between 10^{-2} and 10^{2} and 9 values of n, ranging from 40 to 120 in steps of 10. Our range of C values was chosen as a characteristic region of good performance. The chosen range for n spans values where the model is both far from converging and close to fully converged. The AUCROC results are visualized as 2D heatmaps in Figure 2, and as a 1D plot over C in Figure 3.

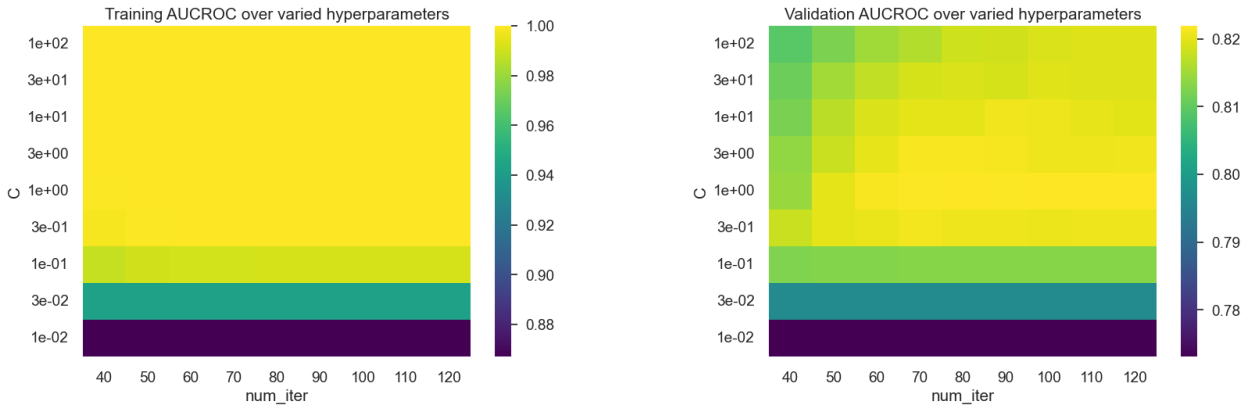

Figure 2: Training (left) and validation (right) AUCROC heatmaps over the penalization parameter C and number of solver iterations n

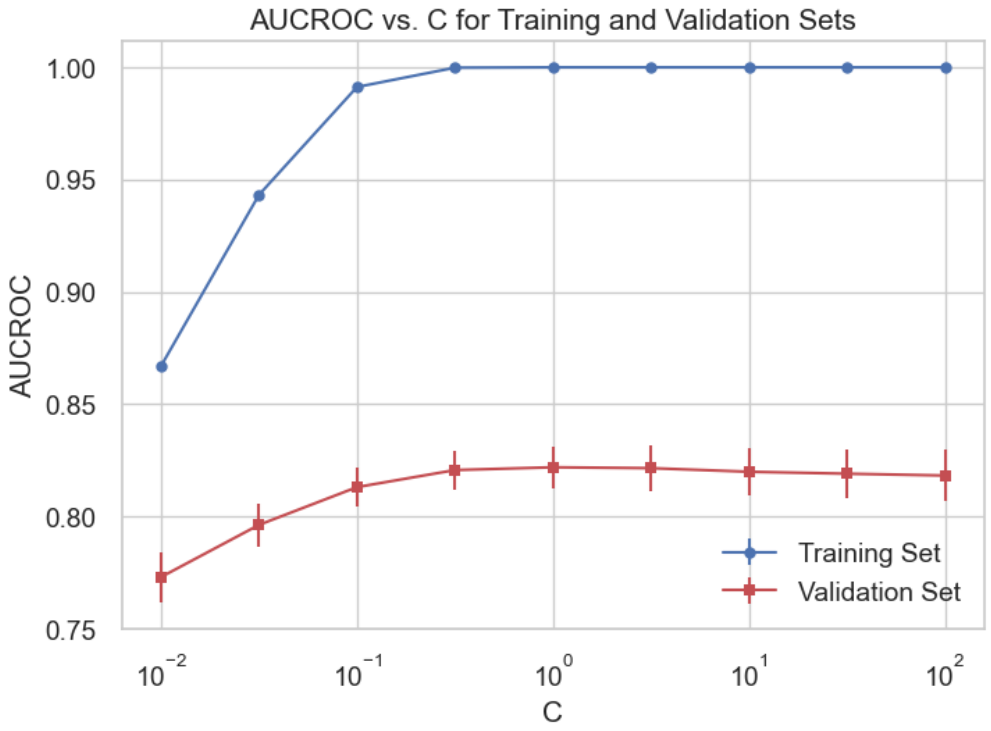

Figure 3: Training and validation AUCROC over penalization parameter C, with error bars showing standard deviation across 10-fold cross-validation splits

We found that the values $C = 1$ and $n = 80$ achieve the best possible AUCROC of 0.8219. Performance is similar for nearby values of C and n, so the optimal values may vary across different K-fold cross-validation splits. As C increases, the model becomes more complex, underfitting for lower values such as $C = 0.01$ and overfitting for values such as $C = 100$. In the overfit regime, training AUCROC approaches 1.0 but validation AUCROC degrades considerably, whereas in the underfit regime, both training and validation AUCROC underperform relative to intermediate values. Large penalization terms, corresponding to $C = 10^{-2}$, 3×10^{-2}, 10^{-1}, produce little dependence

on iteration count, while models with weaker regularization exhibit substantial sensitivity to n (we observe poor performance for large C and small n).

1D: Reflection on Test Set Performance

We trained our final model with `min_df` $= 0.0$, `max_df` $= 1.0$, $C = 1$, and $n = 80$. Our AUROC on test data, as provided on Gradescope, was 0.6627, far lower than the k-fold average validation AUROC of 0.8219. We hypothesized that the discrepancy between our test and validation AUROC was partially due to the model being overly complex. Our k-fold average validation AUROC for $C = 0.1$ and $n = 80$ was 0.8131, lower than our best score but still substantially better than our test performance. We expected that lowering our value of C by a factor of 10 might result in a model that generalizes better to the test set. With this model, we achieved an improved test AUROC of 0.690, confirming our hypothesis.

Even with this improvement, there remains a large discrepancy between our validation and test set performance. We believe this discrepancy arises from the way the cross-validation splits were created. Specifically, we used the standard `KFold` cross-validation function, which blindly splits the data. We expect that using a different fold creation method would yield more consistent results. In particular, the `GroupKFold` function, which groups similar data together, should improve performance consistency.

Looking at the raw training dataset shows that many passages come from the same author and text. With baseline `KFold` cross-validation, these passages are split across the training and validation folds. As a result, validation performance is likely artificially inflated because the model is trained on texts that are very similar to those in the validation set. Creating folds in which all texts by the same author are placed within a single fold addresses this issue, which we believe will improve the consistency between the validation and test sets. Importantly, when using `GroupKFold`, texts of the same reading level may also be grouped together, potentially disturbing class stratification. Maintaining stratification across folds is essential, which motivates our decision to use `StratifiedGroupKFold` in the subsequent analysis.

Problem 2

2A: Feature Representation Description

Initial feature representation efforts were focused in using the numeric fields provided in `x_test.csv`. In particular, a subset of the numeric fields was hand-picked to be included in the feature representation. When selecting numeric fields, particular emphasis was placed on using features that are independent of each other. In this way, redundancies or confounding elements can be reduced in our feature representation. One particular example of this came in selecting a readability metric. It was decided that `avg_sentence_length` would be included in the feature representation, which dissuaded the use of the Kincaid and other readability metrics that use similar justification. In the end it was decided that `avg_word_length, type_token_ratio, avg_sentence_length, sentiment_polarity, readability_DaleChallIndex` would be included in the feature representation.

It was determined that this feature representation alone was not sufficient in representing the data. Therefore, BERT embeddings were used as well. Using the provided `x_train_BERT_embeddings.npz`,

feature representations of the text were created and compounded with numerical metrics to determine performance. Numerical metrics were determined using the same criteria as described above. In the end, it was found that the numerical metrics impacted performance negatively on the test set. Consequently, only the BERT embeddings were used in our final submission.

2B: Cross Validation Design Description

As discussed in section 1D, `scikit-learn`'s `StratifiedGroupKFold` is employed in cross-validation efforts to group like-texts together. This is motivated by the artificial performance benefit seen in the validation folds when like texts are separated between the training and validation folds. In total, 10 folds are created with a shuffled split. We group the passages by author. Initial cross-validation efforts used the same random seed, 12345, as problem 1, which led to large amounts of variance within the performance of each fold. A second random seed, 1, was tested to see if the variance in our original seed was particularly bad. There was no formal analysis done to pick a seed that reduced variance because it is not a hyperparameter that dictates performance. All subsequent analysis uses a random seed of 1 in the creation of the cross-validation split.

Similarly to 1B, a cross-validation grid search was run for the hyperparameters C and n. Because only a single hyperparameter grid search is being run, the scale of the hyperparameter grid search was expanded. In total nine C values and nine n values were testing representing 81 total $C-n$ pairings.. All other characteristics, like performance metric (AUCROC) and size of fold (550 texts) of the $C-n$ grid search, were left the same from 1B. One final classifier model will be trained using the entire training set and the best $C-n$ pairing as found by the cross-validation.

2C: Classifier Description and Hyperparameter Search

Similar to 1C, we train a Logistic Regressor model to classify our passages, and perform a hyperparameter search over C, the regularization parameter, and number of L-BFGS iterations n. Logistic Regression is well-suited to the task of classifying texts using their BERT embeddings because it provides a sufficiently complex model that is still interpretable (assuming we understand the BERT embedding vectors). It is preferable over a K-nearest neighbors approach because it assigns a numerical weight to each feature in the BERT embedding vector, creating a continuous model: K-nearest neighbors is piecewise constant and may break down in regions of the feature space with fewer data points.

We performed a two-dimensional grid search over 9 logarithmically spaced values of C in the range $[10^{-4},\ 10^{4}]$, and 9 linearly spaced values of n in the range $[50,\ 450]$. C, the regularization parameter, affects model complexity by controlling the penalty weight: smaller values of C correspond to a larger penalty, resulting in greater regularization, while larger values of C, corresponding to a smaller penalty, leads to a more complex model. Seeing as $C = 10^0$ is often used as a standard value (it is the default value in `scikit-learn`'s `LogisticRegressor`), we decided to explore a range of 9 values centered upon 10^0. n, the maximum number of L-BFGS iterations, determines model complexity by controlling early stopping. Large values of n, such as 1000, tend to result in overfit models, while small values of n, like 50, tend to produce underfit models, as the algorithm has not yet had the opportunity to begin reaching a minimum. Taking this into consideration, we search 9 linearly spaced values in the range $[50, 450]$ to find an optimal n.

We trained our `LogisticRegressor` for each pair of hyperparameter values (C, n) given by the

ranges above. For each of the 10 splits (generated by as described in section 2B), we trained the model on the training fold and evaluated it by using it to make predictions on the validation fold and computing the AUROC. Finally, we compared the average AUROCs over the 10 splits to find the best-performing values of C and n. We found that the optimal values were $C = 0.01$ and $n = 100$.

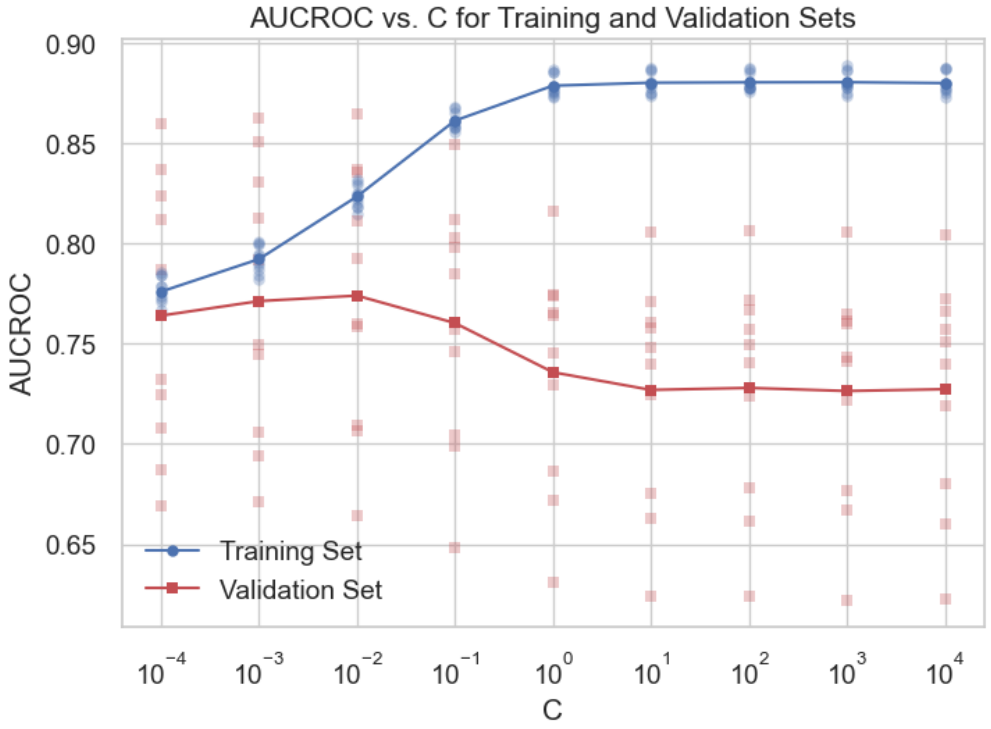

Figure 4: Average training and validation AUROC over C at $n = 100$

Figure 4 shows the training and validation AUROC over the hyperparameter C. The blue and red lines represent the average fold-averaged AUROC for the training and validation datasets, respectively. The transparent points represent the raw AUROC for each of the 10 splits. We see clear evidence of underfitting at the lowest point, $C = 10^{-4}$: the training and validation AUROC is similar, with a worse validation AUROC than the at the maximum of $C = 10^{-2}$. Around $C = 10^{-1}$ or $C = 10^{0}$ we begin to see evidence of overfitting: training AUROC plateaus around 0.88, while validation AUROC is substantially lower than at $C = 10^{-2}$. We hypothesize that the training plateau around an AUROC of 0.88 is due to early stopping, caused by the fixed $n = 100$. Without early stopping, we would expect the training AUROC to converge to 1 as we increase C, relaxing the penalty.

While this chart confirms that the optimal value of $C = 10^{-2}$ for $n = 100$, the AUROC across cross-validation folds at each regularization level varies largely. While in theory, this should decrease our confidence that our chosen $C = 10^{-2}$ performs better than neighboring C values with similar averages, we suspect that the variability in validation performance is due to how the data was split into training and validation folds. To investigate, we examine our raw AUROC matrix, reproduced in Table 1. We find that within each fold, the best-performing AUROC is always either $C = 10^{-2}$ or $C = 10^{-3}$, with $C = 10^{-2}$ being more common. This observation suggests that the choice of $C = 10^{-2}$ is not random, that the high variance is due to the similarity between the passages in

each validation fold-training fold pairing. This claim is supported by the fact that the same folds consistently perform best and worst across all values of C: Fold 4 has the highest AUROC across all C values, while Fold 6 consistently has the lowest.

C	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
10^{-4}	0.7869	0.8238	0.7081	0.8600	0.8118	0.6691	0.7323	0.8368	0.6870	0.7249
10^{-3}	0.7918	0.8304	0.7057	0.8624	0.8124	0.6713	0.7493	0.8509	0.6940	0.7450
10^{-2}	0.7927	0.8356	0.7092	0.8649	0.8111	0.6644	0.7599	0.8373	0.7068	0.7586
10^{-1}	0.7853	0.8122	0.7047	0.8495	0.7982	0.6481	0.7459	0.8031	0.6993	0.7574
1	0.7645	0.7747	0.6865	0.8160	0.7737	0.6307	0.7292	0.7653	0.6721	0.7454
10	0.7577	0.7609	0.6754	0.8055	0.7712	0.6239	0.7244	0.7482	0.6628	0.7400
10^{2}	0.7571	0.7670	0.6779	0.8065	0.7720	0.6238	0.7243	0.7495	0.6616	0.7404
10^{3}	0.7613	0.7603	0.6766	0.8060	0.7652	0.6219	0.7218	0.7436	0.6667	0.7412
10^{4}	0.7572	0.7660	0.6803	0.8045	0.7724	0.6224	0.7193	0.7513	0.6604	0.7399

Table 1: AUROC for all values of C values across validation folds, used to generate Figure 4

2D: Reflection on Model Performance

Our Problem 1 test AUROC is **0.6898**, while our Problem 2 test AUROC is **0.7280**.
For Problem 1, our most confidently correct and most confidently incorrect predictions are reported in Tables 2 and 3, and for Problem 2, our most confidently correct and most confidently incorrect predictions are reported in Tables 4 and 5.

Author	Book Title	Probability	True Label
Aristotle	The Ethics of Aristotle	0.9992	1
Sax Rohmer	The Insidious Dr. Fu Manchu	0.9977	1
Marcus Aurelius	Meditations	0.9970	1

Table 2: Most confidently correct predictions on held-out data for problem 1

Author	Book Title	Probability	True Label
Stendhal	The Red and the Black: A Chronicle of 1830	0.0054	1
James Joyce	Ulysses	0.0222	1
William Shakespeare	The Comedy of Errors	0.9764	0

Table 3: Most confidently incorrect predictions on held-out data for problem 1

Author	Book Title	Probability	True Label
François Rabelais	Gargantua and Pantagruel	0.9743	1
Saint Thomas More	Utopia	0.9740	1
Marcus Aurelius	Meditations	0.9735	1

Table 4: Most confidently correct predictions on held-out data for problem 2

Author	Book Title	Probability	True Label
Jules Verne	From the Earth to the Moon; and, Round the Moon	0.9574	0
Jules Verne	From the Earth to the Moon; and, Round the Moon	0.9438	0
Howard Pyle	Men of Iron	0.9284	0

Table 5: Most confidently incorrect predictions on held-out data for problem 2

Our Problem 2 model performed better than our problem 1 model, with a test set AUROC of 0.73 compared to 0.69. While using BERT embeddings for our feature representation helped, the biggest improvement came as a result of grouping our validation and test data by author using `StratifiedGroupKFold`, so that there was no overlap of author between the two sets. This decreased our validation AUROC, but brought it much closer to the test AUROC, which allowed us to perform a more accurate hyperparameter search for C and n.

The model in problem 1 has both confident false positives and false negatives. When the search is expanded to the six most inaccurate predictions, this trend holds. Examining the false negatives, *The Red and the Black: A Chronicle of 1830* stands out as an outlier: the predicted passage is the novel's table of contents, which contains common words and repeatedly uses the word "chapter." A passage from *Ulysses* also appears among the false negatives. While *Ulysses* is notoriously difficult, the predicted passage contains common words. Without considering meaning, as is the case with a BoW representation, excerpts from hard texts may be represented with simple vocabularies. Other false negatives contain plain English dialogue from novels with adult-themed subject matter, such as *The Beautiful and Damned*, whose complex themes of war and wealth are invisible to a BoW representation. The only other false positive in the six most inaccurate predictions is a Shakespeare excerpt. Shakespeare is known to fabricate and conjoin words that appear nowhere else in the training data and will therefore be excluded from the model vocabulary, making it unsurprising that his passages are misclassified.

For the model in problem 2, four of the six most inaccurate predictions come from Jules Verne across two texts. Verne's writing contains obscure vocabulary: *From the Earth to the Moon* uses niche astronomical terminology, while *Twenty Thousand Leagues under the Sea* contains obscure adjectives such as "calcareous" and "luxuriant" and rare nouns like "verdure" and "sward" that pertain to vegetation. We believe that BERT tokenizes these obscure words in a way that describes the text as difficult. BERT contextualizes text well, but when passages are truncated, as they are in this dataset, full context is lost, pushing predictions toward one. This also explains why our Problem 2 model consistently produces false positives rather than false negatives. Pairing BERT with a BoW representation could address this, as rare vocabulary would then be encoded as direct features rather than relying solely on contextual embeddings.

With these observations in mind, and considering that both models' test AUROC is significantly lower than their validation AUROC, we hypothesize that both models are slightly overfit. If we were to continue working on this project, we would address this weakness by fine-tuning hyperparameters that control complexity, such as C and n, and by regularizing slightly more than we find is optimal in our hyperparameter searches, in an attempt to generalize better to the test set.