COMP 135: Project 2

Darcy Corson

April 28, 2025

1 Classifying Review Sentiment with Bag-of-Words Features

Bag-of-Words (BoW) is a fundamental concept in natural language processing and text analysis. It represents text data as a "bag", or collection, of words without considering grammar or word order, but while preserving the multiplicity of words in the text. To create a BoW, a vocabulary that includes every unique word found in the text is created. For each individual text entry, a corresponding feature vector is defined. The vector's length is equal to the number of unique words in the text's vocabulary. The value in each dimension usually reflects the presence (binary: 0 or 1) or frequency (count) of that word in a specific text entry. By focusing on word frequency and ignoring syntax and word order, BoW simplifies the text data in such a way that it is easier for computational models that attempt to perform tasks like sentiment analysis.

1.1 Question 1

In this project, a BoW model is employed for a specific purpose: sentiment analysis of single-sentence reviews from different websites (e.g. IMDb.com, Amazon.com, Yelp.com). These reviews are characterized by diverse language, and may include slang, spelling mistakes, and other colloquialisms. Given this, preprocessing of the data is performed to make it more uniform and manageable for the BoW model and subsequent classifiers. In fact, the BoW's feature generation pipeline involved several distinct steps. An explanation of each one of those steps, and the corresponding rationale behind them, is explained below.

Data Loading and Preparation
The first step in the pipeline involves loading the training data from CSV files into Pandas dataframes. This initial phase makes it possible to easily access and manipulate the text data. The decision to use Pandas for data handling was made because it makes data manipulation and analysis of tabular data (like CSV files) particularly efficient and easy.

Text Preprocessing

This phase prepares the text for feature extraction, and includes several sub-steps.

First, contractions in the text are expanded (e.g. converting “don’t” to “do not”). This standardization is important because it makes the text more uniform. By expanding contractions, the complexity of the language model is reduced. This is because the model is then able to treat different forms of the same expression consistently. Reducing the complexity of the model is advantageous because it improves the model’s understanding of text. That is, when the model encounters fewer variations in language, it is able to interpret and respond more accurately to inputs. As a result, prediction accuracy improves. Also, a model that is less complex is less likely to overfit. That is, it is better able to apply what it has learned from the training data more effectively to new, unseen data (generalization).

In addition, the text is pre-processed by the removal of non-letter characters. The removal of symbols and/or numbers focuses the analysis on words. By concentrating on just words, the model is more likely to capture relevant features that contribute to understanding the sentiment of the text. This is true because non-letter characters could likely add noise rather than meaningful information. Further, removing symbols and numbers standardizes the text, leading to a more consistent analysis. Also, by eliminating non-letter characters, the feature space is reduced. This makes the model less complex; a discussion of the benefit of making the model less complex is explained above. In some cases, numeric values could contribute to the sentiment content of the reviews. However, this is likely only true when the numeric values are standardized (e.g. all reviews offer a rating from 1 (poor) to 10 (good)). In the absence of such standardization, the decision was made to eliminate numbers assuming that their meaning would vary significantly from review to review.

Further, during pre-processing, all text was converted to lowercase. This was done to ensure uniformity, eliminating the variability introduced by capitalization (“Apple”, “apple”, and “APPLE” would each be treated as a distinct entity even though they are the same word). Thus, by converting all works to lowercase, the model recognizes and treats all such variations of a work as the same word. This uniformity reduces the complexity of the model by decreasing the size of the vocabulary, the importance of which is discussed above. Further, it enhances the accuracy of the model by preventing it from misunderstanding or mis-classifying text due to capitalization differences, and it ensures consistency in how the model processes and interprets text.

Finally, the text is pre-processed by merging the website name with the review text. By doing this, the context of the review source is incorporated. This decision was made based on the hypothesis that the website’s context might influence the language or sentiment of the review.

Removal of Stop Words

Common English stop words were removed from the text. Common stop words like ’the’, ’is’, ’at’, ’which’, and ’on’ are generally frequent across all texts regardless of their specific content or context. They are considered ’noise’ in the data. By removing these words, this noise is reduced, and the model is able to focus on more content-rich words which are more likely to be useful in distinguishing between different texts. Further, additional phone-related terms were also removed, assuming that they are overly common and unlikely to contribute to distinguishing between texts. By removing phone-related common terms, the model is less likely to focus on these generic terms and more likely to focus on terms that could signal differences in sentiment. With the noise reduced and the focus shifted to more meaningful words, the model is likely to be more accurate

Feature Extraction with TF-IDF Vectorization
TF-IDF vectorization was performed on the text, using scikit-learn's TfidfVectorizer, thereby transforming the pre-processed text into a numerical format (a matrix of TF-IDF features) that the model can understand and analyze. In the TF-IDF vectorization process, several parameters are selected to optimize the balance between computational efficiency and the richness of the feature set extracted from the text.

The 'max_features' parameter sets an upper limit on the number of features (words or n-grams) to be considered. This limit helps to manage the model's computational complexity. It also ensures that the most relevant features are retained. For my model, this parameter is set to 25000.

The 'min_df' (minimum document frequency) and 'max_df' (maximum document frequency) parameters play an important role in further filtering the vocabulary. 'Min_df' excludes words that appear too infrequently, as these may not provide significant insights and could contribute to over=fitting if the model learns to recognize rare terms. 'Max_df' removes words that are too common, as these are often less informative and do not contribute much to distinguishing between different texts. By setting these thresholds, the model focuses on words that are neither too rare nor too common, striking a balance that enhances the meaningfulness of the features. For my model, 'Min_df' is set to 0.00008, and 'Max_df' is set to 0.75.

The incorporation of n-grams, which are combinations of words up to a specified length (in this case, up to two words). This decision was made in an attempt to capture more context than what single words can offer. N-grams help in understanding phrases and the relationship between adjacent words, which is crucial in capturing contextual nuances and the overall semantics of the text. This approach is especially beneficial for understanding the meaning of phrases and expressions that rely on word order.

1.2 Question 2

For this project, a logistic regression model was constructed to serve as a binary classifier. The model operates by analyzing the feature-data, which, in this case, is derived from text that has been preprocessed and transformed into a numerical format using TF-IDF vectorization. This transformation converts the text into a structured format that the model can interpret (a set of numbers representing the significance of different words or phrases in the text). Then, the logistic regression algorithm uses this feature-data to learn the underlying patterns that distinguish the two classes in the training dataset. It does this by estimating probabilities — specifically, the probability that a given piece of text belongs to one class or the other. Through the training process, the model adjusts its internal parameters to minimize errors in these predictions. Once adequately trained, the logistic regression model is used to classify new, unseen text data, predicting which of the two categories each new example most likely belongs to based on the patterns it learned during training.

In the development of the logistic regression model for this project, decisions were made to optimize its performance and ensure its appropriateness for the specific text classification task. The regularization strength, denoted by 'c', was selected using GridSearchCV, which is a method for finding the optimal balance between model complexity and generalization capability. Regularization is important in preventing overfitting, and the chosen value of 'c' reflects a balance struck between model accuracy and simplicity. Further, the choice of solver, important for the optimization process in logistic regression, was also determined through GridSearchCV. In logistic regression, the solver is the algorithm that is used for optimization, specifically for finding the parameter values that best fit the model to the training data. Options including 'newton-cg', 'lbfgs', 'liblinear', 'sag', and 'saga' were tested to find the most efficient algorithm for this dataset.

The decision to specifically tune the hyperparameters 'c' and 'solver' was aimed at optimizing the model's performance and efficiency. This fine-tuning of 'c' helps to achieve a balance between bias and variance, which is an important part of creating a model that generalizes well to new data. The decision to tune the choice of 'solver' was aimed at maximizing the efficiency and convergence of the optimization algorithm used in the logistic regression. Different solvers are designed to perform optimally under different conditions; for example, some solvers are more efficient for larger datasets, while others are better suited for smaller or high-dimensional datasets. Selecting the appropriate solver is important to ensuring that the model is robust, scalable, and performs efficiently across various data distributions and sizes. By tuning both 'c' and the solver, the goal was to develop a logistic regression model that is both accurate in its predictions and efficient and adaptable to different data scenarios, ensuring optimal performance and the ability to generalize.

The range for 'c' values tested, spanning from 10-4 to 104, were chosen because they cover a wide spectrum – from strong regularization to weak regularization. Such a broad range is important for comprehensively investigating how different degrees of regularization affect the model. Thus, by exploring such a wide range of 'c' values, it is possible to find the place where the model achieves the best balance between bias and variance. For some datasets, a higher 'c' might yield better results, allowing the model to fit complex patterns, while for other models, a lower c might be best in order to avoid overfitting.

Similarly, the range of solvers tested, 'newton-cg', 'lbfgs', 'liblinear', 'sag', and 'saga', cover a variety of optimization algorithms. Each solver has its strengths and is likely suited to different types of data and logistic regression problems. Thus, the goal of exploring different solvers was to understand how each one performs on the given datasets and to find the one that offered the best combination of speed, accuracy, and convergence stability.

The values of 'c' and 'solver' were selected based on their performance during the cross-validation process within GridSearchCV. In terms of 'c', at a regularization strength of 545.5594781168514, the model seems to have found a balance between bias and variance. This specific value was chosen because it provided the best generalization performance on the validation sets during cross-validation. That is, this value helped the model to learn well from the training data while also performing effectively on unseen data. The selection of 'saga' for 'solver' was made because it yielded the best performance in terms of convergence speed and model accuracy on the dataset. The 'saga' solver is a variant of Stochastic Average Gradient descent. It's known for being effective on large and/or sparse datasets.

The use of L2 regularization for the logistic regression model, also determined by the GridSearchCV's outcome, is attributable to the nature of L2 regularization and how it interacts with data and model structure. L2 regularization works by adding a penalty equal to the square of the magnitude of coefficients to the loss function. This approach is particularly effective for high-dimensional datasets, as it helps to reduce model complexity without eliminating a significant number of input features.

The training of this model involved splitting the dataset into training and validation sets in order to facilitate an accurate assessment of the model's effectiveness outside of its training environment. The use of GridSearchCV extended beyond hyperparameter tuning to the actual training process, employing cross-validation to enhance the robustness and reliability of the model. Specifically, k-fold cross-validation was performed. This method involves dividing the dataset into k subsets, or 'folds'. For each hyperparameter setting, the model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, each time with a different fold used as the validation set. Scikit-learn's 'GridSearchCV' was employed for this task, and it systematically works through multiple combinations of parameter tunes, cross-validating as it went to determine which tune gave the best performance. CV was set to 5 in 'GridSearchCV', indicating a 5-fold cross-validation. This means that the dataset was divided into 5 parts, with the model being trained on 4 parts and validated on the 5th part, iteratively.

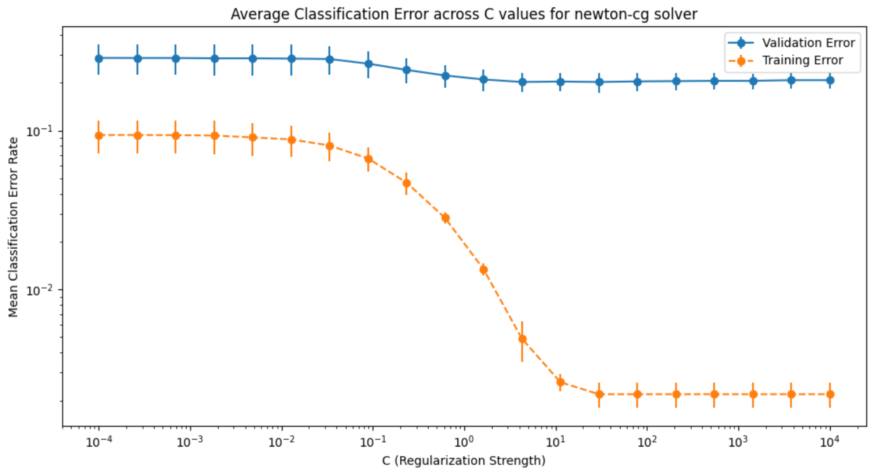


Figure 1: The plot depicts the mean classification error rate for the logistic regression model with the newton-cg solver across a logarithmic range of C values, showing a decrease in training error with increasing C while the validation error remains relatively constant.

Figure 1 shows a clear trend in the relationship between the regularization strength, ‘c’, and the classification error rate for the ‘newton-cg solver’. As ‘c’ increases, the training error decreases significantly, which suggests that reducing regularization allows the model to fit the training data more closely. However, the validation error remains relatively stable across the entire range of ‘c’ values, with a slight increase in error for very low values of ‘c’. This stability implies that the model generalizes well across different levels of regularization, but too much regularization leads to some degree of underfitting, where the model is too simple to capture the underlying patterns in the training data. The error bars, representing the variability of the error rate across cross-validation folds, are relatively small for validation error, indicating that the model’s performance is consistent across different subsets of the data.

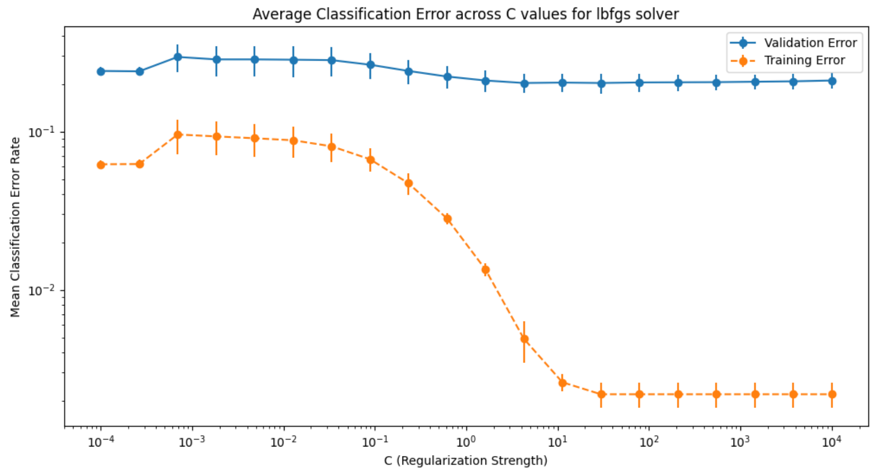


Figure 2: The graph illustrates the mean classification error rate for the logistic regression model using the ‘lbfgs’ solver, where the training error decreases with a higher ‘c’ value, while the validation error exhibits minimal fluctuation, suggesting an optimal regularization range for model generalization.

Figure 2 summarizes the error rates of a logistic regression model using the ’lbfgs’ solver as we adjust the regularization parameter ‘c’. When ‘c’ is small, the model has a high error on the training data, suggesting underfitting. As ’c’ increases, the training error decreases sharply, indicating the model fits the training data better. However, the validation error remains relatively flat across the range of ’c’ values, with a slight increase at lower ’c’ values. This stability in validation error suggests that the model’s ability to generalize to unseen data does not substantially improve with less regularization. The uniform performance across a broad spectrum of ’c’ values suggests that the model’s predictive reliability is not sensitive to variations in regularization strength, particularly at higher ’c’ values where the error rate levels off.

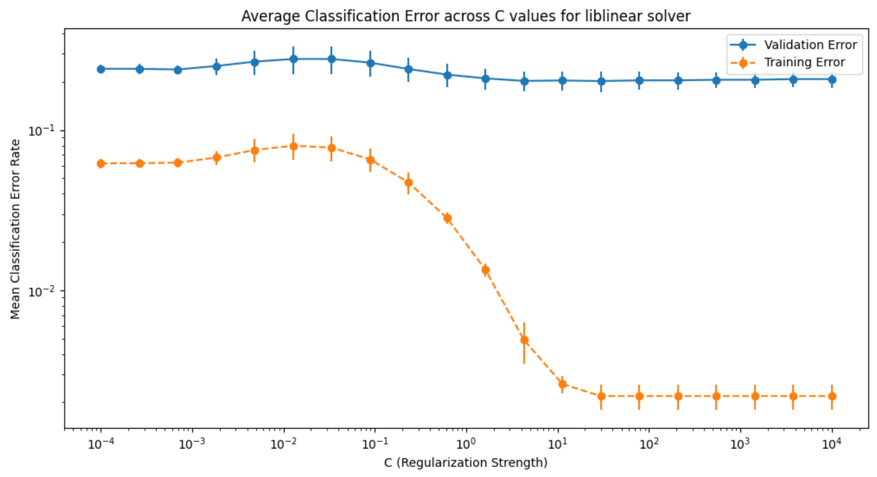


Figure 3: The graph shows the average classification error rate for the logistic regression model using the 'liblinear' solver, revealing a decrease in training error with increasing 'c' values, while the validation error remains consistent, indicating the model's insensitivity to higher regularization strength.

Figure 3 shows that the training error exhibits a downward trend as the regularization strength decreases (which corresponds to increasing 'c' values), suggesting that the model fits the training data more closely when less constrained by regularization. Instead, the validation error maintains a relatively flat line across the range of 'c' values, implying that loosening regularization does not lead to overfitting and the model's performance on unseen data is stable. The fact that the validation error does not rise with lower levels of regularization may indicate that the model has reached its capacity in learning from the data, and additional complexity does not introduce overfitting. This uniformity in validation suggests that the chosen features and model complexity are well-suited to the underlying structure of the dataset.

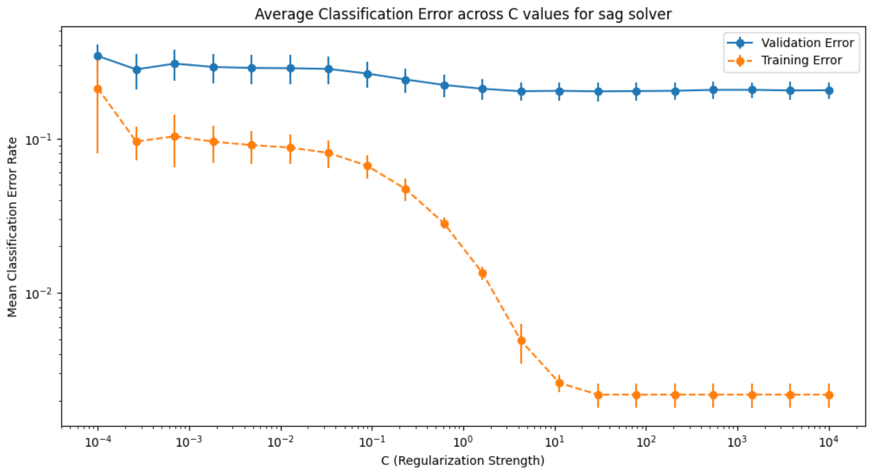


Figure 4: The graph presents the average classification error rate for the logistic regression model employing the 'sag' solver, illustrating a declining trend in training error as regularization strength decreases (higher 'c' values), while the validation error remains relatively constant.

Figure 4 shows that the model is becoming more tailored to the training data as the regularization decreases, which is indicated by the reduction in training error. However, the validation error does not show a corresponding increase, which typically would indicate overfitting; instead, it levels off. This could imply that within the tested range of 'c', the model doesn't overfit the data, even with less regularization. It suggests that the model with the 'sag' solver has a good capacity for learning from the training data without becoming too specific to it, maintaining a generalization that holds across the unseen validation data. The error rates for the validation set are low and stable, which is a sign of a strongly performing model within this regularization range.

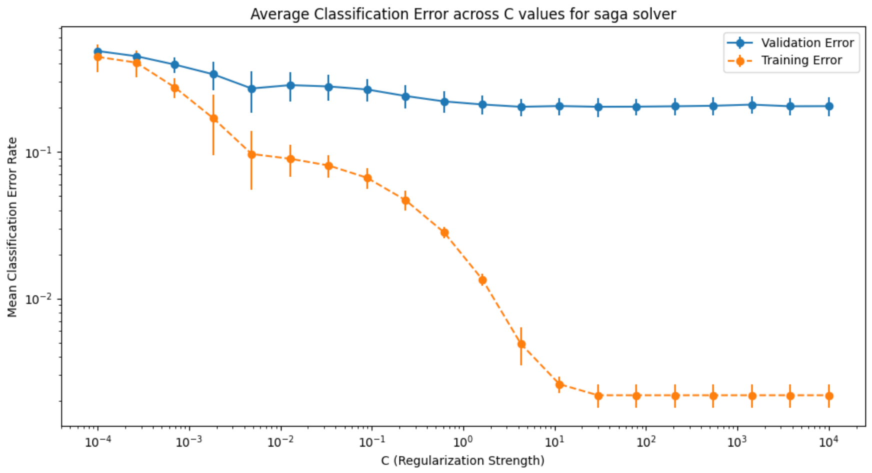


Figure 5: The graph displays the average classification error rates for the logistic regression model using the 'saga' solver, with the training error declining as 'c' increases, indicating less regularization, while the validation error plateaus, suggesting a limit to beneficial model complexity.

In Figure 5, the logistic regression model's performance is evaluated with the 'saga' solver across a range of 'c' values affecting regularization strength. The training error decreases steadily as 'c' increases, reflecting the model's improved fit to the training data due to a reduction in regularization strength. This trend continues until the error rate begins to level off, indicating diminishing returns on model complexity. Further, the validation error decreases initially and then stabilizes, forming a plateau. This pattern suggests that while the model benefits from less regularization initially, there is a point beyond which further reducing regularization does not yield significant improvements in generalization to new data. The error rates for validation remain low and fairly consistent across a broad span of 'c' values, highlighting that the model's performance is not overly sensitive to the exact value of 'c' within this plateau range. It is important to note that the error bars on the validation scores are relatively small, underscoring the stability of the model's generalization.

Analysis of the series of graphs for different solvers for the logistic regression model across varying regularization strengths reveals a clear pattern: while training error consistently decreases with reduced regularization (higher 'c' values), validation error typically lowers to a point before stabilizing, indicating an optimal regularization range for generalization. This trend suggests that the models are sufficiently complex to capture the training data's patterns without overfitting, as evidenced by the stable error rates on unseen data. The performance of different solvers is largely consistent, with none showing a significant propensity for overfitting across the tested 'C' values, pointing to the robustness of the logistic regression models with respect to these hyperparameters for this specific dataset.

The error bars in the graphs represent the standard deviation of the model's error rate across the folds of cross-validation, serving as a measure of the results' uncertainty. Smaller error bars on the training data suggest that the model's performance is consistent when learning from different segments of the data. In contrast, the larger error bars on the validation data at lower 'c' values indicate greater variability in the model's generalization ability. As the regularization decreases (higher 'c' values), the error bars on the validation data get smaller, implying more consistent performance on unseen data and increased confidence in the model's predictive stability.

1.3 Question 3

The MLP model developed for this project is a type of neural network used for text classification. It was implemented using scikit-learn's 'MLPClassifier'. An important part of this model's development was the hyperparameter tuning phase. Using 'GridSearchCV', the model's alpha (regularization parameter) and learning_rate_init (initial learning rate) were systematically varied and evaluated. This process involved training multiple versions of the model with different hyperparameter combinations and selecting the combination that maximized cross-validation accuracy. This rigorous tuning was performed in order to optimize the model's performance, ensuring that it strikes a balance between learning the training data patterns and generalizing well to unseen data. The selection of hyperparameters to tune was based on their impact on the model's performance. The alpha parameter controls the extent of regularization, which helps prevent overfitting by penalizing larger weights. The learning_rate_init parameter affects the speed and quality of the model's convergence during training. By adjusting these parameters, the model's ability to learn from the training data effectively and generalize to new, unseen data is enhanced.

Incorporated into the hyperparameter tuning process was cross-validation. Cross-validation entailed assessing the model's performance on different subsets of the training data to gain a holistic understanding of its generalization capabilities. Specifically, a 5-fold cross-validation strategy was used. In this approach, the training data was divided into five equal subsets or folds. The model was then trained and evaluated five times, each time using a different fold as the validation set while the remaining folds constituted the training data. The hyperparameter values were selected based on their performance across these 5-fold cross-validation iterations. This ensured that the chosen values enhanced the model's capacity to extract meaningful patterns from the training data and empowered it to generalize effectively when presented with entirely new and previously unseen data. The chosen alpha value was 1, and the chosen initial learning rate was 0.001.

After identifying the optimal alpha and learning_rate_init hyperparameters, the MLP model was trained on the entire training dataset. This step allowed for the model to develop its internal weights, which define its decision-making process. The trained model was then employed to generate probability predictions on test data.

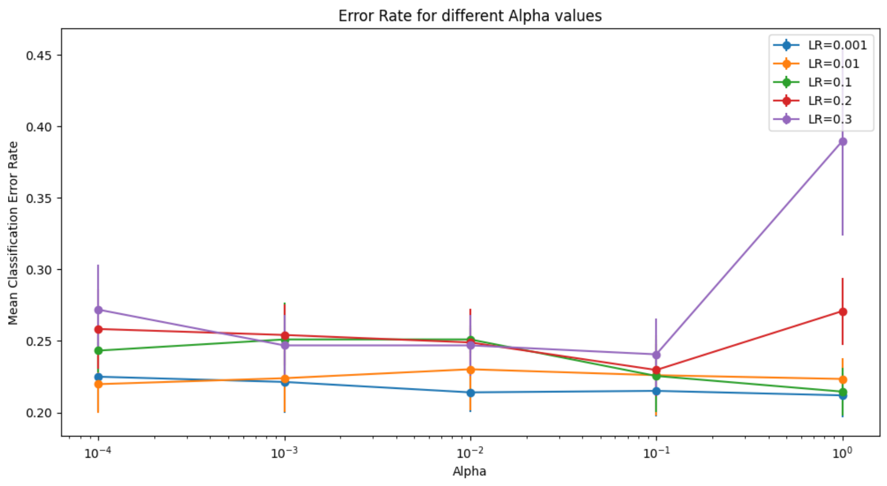


Figure 6: The graph shows the mean classification error rate for different alpha values, comparing the performance of various learning rates (LR) for the MLP model, including error bars for the validation error.

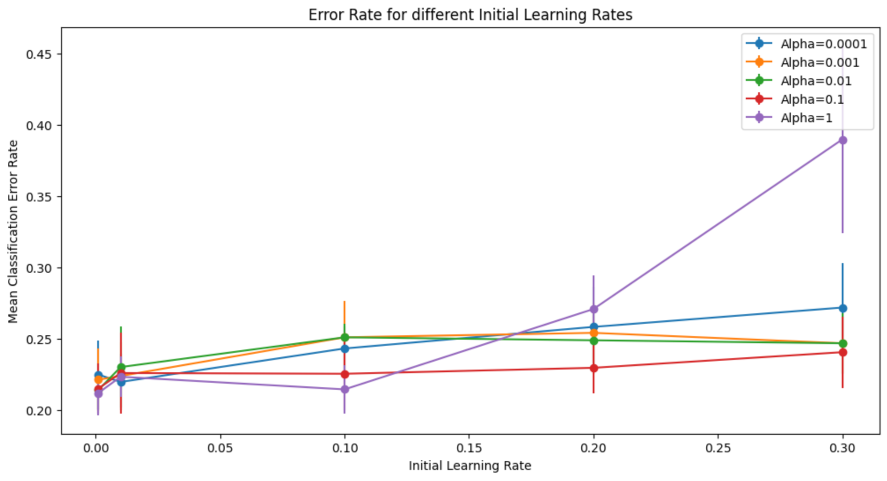


Figure 7: The graph shows the mean classification error rate across various initial learning rates, indicating how different alpha parameters affect the accuracy of of the MLP model, including error bars for the validation error.

The graphs presented in Figures 6 and 7 offer a comprehensive overview of the mean classification error rate as a function of different alpha values and learning rates (LR) for the MLP model. Figure 6 demonstrates consistent error rates at lower learning rates (e.g. 0.001, 0.01), indicating a reduced sensitivity to fluctuations in the regularization parameter alpha. This consistent performance is disrupted at higher learning rates (e.g. 0.2, 0.3), where a pronounced increase in error rate is observed as alpha increases, culminating in a significant spike at an alpha of 1. This spike may signal performance deterioration due to over-regularization, which could lead to underfitting. That is, here, the model may become inadequately flexible to capture the underlying data pattern.

A steep increase in error rate at LR = 0.3 and alpha = 1 implies a potential overfitting situation. However, it is important to consider both the model's training and validation error rates to validate this hypothesis fully. Commonly, overfitting is characterized by a discrepancy where the training error is significantly lower than the validation error, indicating that the model's complexity excessively matches the training data's idiosyncrasies at the cost of its ability to generalize.

The uncertainty associated with the model's performance is represented by the error bars in the figures, reflecting the standard deviation across the 5-fold cross-validation tests. Smaller error bars at lower learning rates suggest that the average performance is not only more stable here but also reliably reproducible across different cross-validation tests. The larger error bars seen at higher learning rates, particularly at LR = 0.3 and alpha = 1, demonstrate a greater variability in performance here, suggesting that the model may be unreliable in these configurations. It would be interesting to assess the significance of the performance discrepancies by examining the overlap of these error bars, as non-overlapping error bars may indicate statistically significant differences.

1.4 Question 4

The SVM model created for this project was developed using the 'SVC' (Support Vector Classification) class from scikit-learn. This model is particularly skilled at finding the optimal hyperplane that separates different classes in a high-dimensional space, making it a good choice for tasks like classifying text data. One of the key steps in developing the SVM model was the hyperparameter tuning process. This process involved using 'GridSearchCV' to experiment with various combinations of hyperparameters, specifically focusing on 'c' (regularization parameter), 'gamma' (kernel coefficient for the 'rbf' kernel), and the type of kernel used. By systematically varying these parameters and evaluating their impact on model performance, the hyperparameter tuning process aimed to find the optimal balance between underfitting and overfitting, ensuring the model's robustness and accuracy.

The choice of hyperparameters directly influences the model's ability to learn from data. As discussed, the 'c' parameter plays a crucial role in controlling the trade-off between achieving a low training error and a low testing error. A higher value of 'c' could lead to a more complex decision boundary, possibly increasing the risk of overfitting. The range of 'c' values tested was chosen to cover a broad spectrum from strong regularization (0.1) to weak regularization

(1000).

The 'gamma' parameter in the 'rbf' kernel determines how far the influence of a single training example reaches, with lower values implying 'far' and higher values meaning 'close.' A low gamma value indicates a large similarity radius, which results in more points being grouped together. A high gamma value leads to points being considered similar only if they are very close to each other, which can lead to overfitting on the training data. The range of 'gamma' values tested spans from high influence of individual data points (1) to low influence (0.0001).

For the SVM model, the hyperparameter values that were selected and employed after the tuning process were a 'c' value of 10, a 'gamma' value of 1, and the Radial Basis Function (rbf) kernel. These selections reflect specific modeling choices. The choice of 10 as the 'c' value implies moderate regularization, encouraging the model to both correctly fit the training data and avoid overfitting. The choice to set the 'gamma' parameter to 1 was made in an attempt to balance the influence of close and more distant data points. That is, the model is encouraged to fit the data's intricacies without being overly sensitive to minor variations. The decision to employ the RBF kernel type was made in order to use a non-linear approach to find the decision boundary. This choice was made because the data may not be linearly separable in its original feature space.

The use of 'GridSearchCV' extended beyond hyperparameter tuning to the actual training process, employing cross-validation to enhance the robustness and reliability of the model. 5-fold cross-validation was used. This involved splitting the training dataset into five parts, using four parts for training and one part for validation, iteratively. This method ensures that every data point gets to be in the validation set exactly once and in the training set four times, providing a robust estimate of the model's performance on unseen data.

The SVM model was trained using the entire training dataset, which allowed it to learn and adapt to the intricacies of the text data. The trained model was then used to make predictions on the test data.

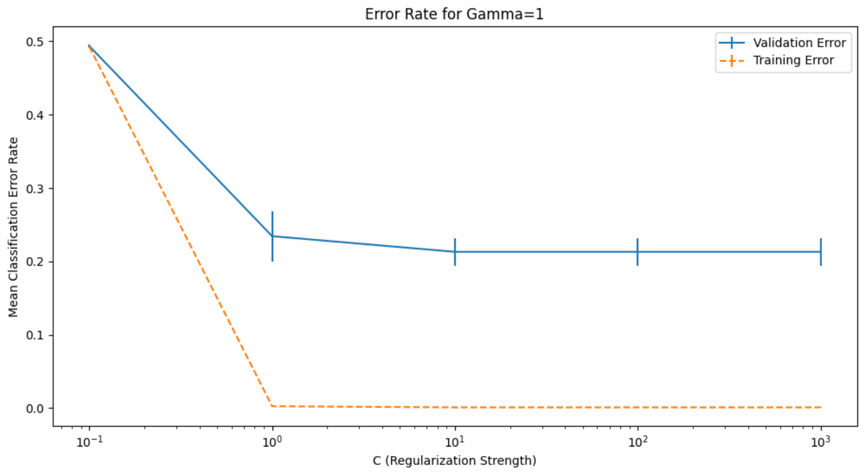


Figure 8: The graph displays the mean classification error rate for the SVM model with a gamma value of 1, plotting the error rates for both the training and validation data sets across different regularization strength (c) values on a logarithmic scale, including error bars for the validation error.

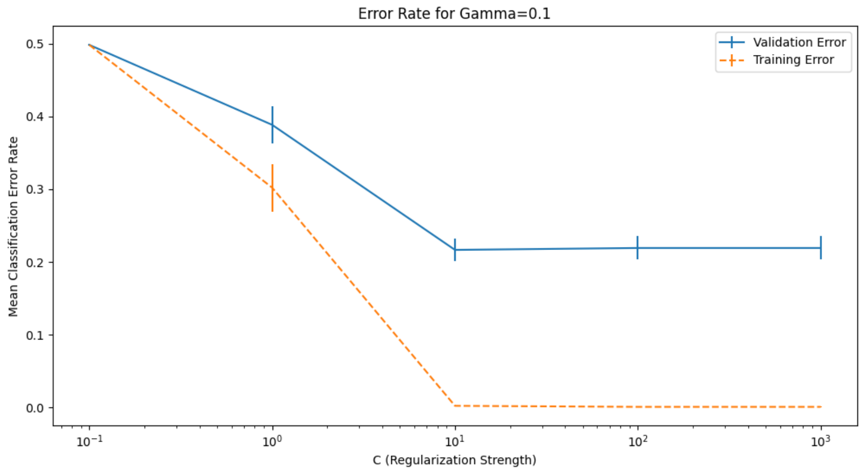


Figure 9: The graph displays the mean classification error rate for the SVM model with a gamma value of 0.1, plotting the error rates for both the training and validation data sets across different regularization strength (c) values on a logarithmic scale, including error bars for the validation error.

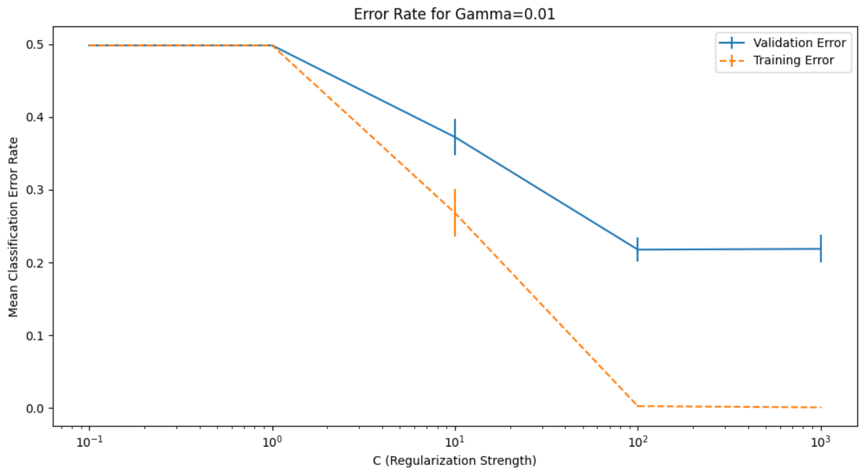


Figure 10: The graph displays the mean classification error rate for the SVM model with a gamma value of 0.01, plotting the error rates for both the training and validation data sets across different regularization strength (c) values on a logarithmic scale, including error bars for the validation error.

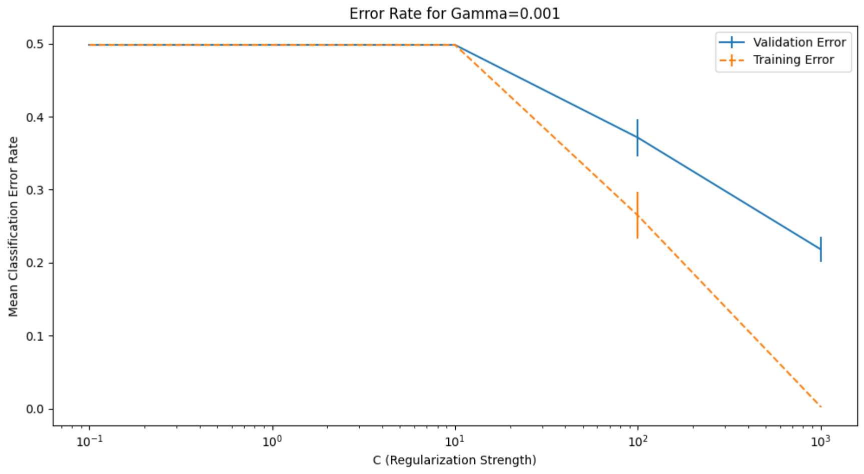


Figure 11: The graph displays the mean classification error rate for the SVM model with a gamma value of 0.001, plotting the error rates for both the training and validation data sets across different regularization strength (c) values on a logarithmic scale, including error bars for the validation error.

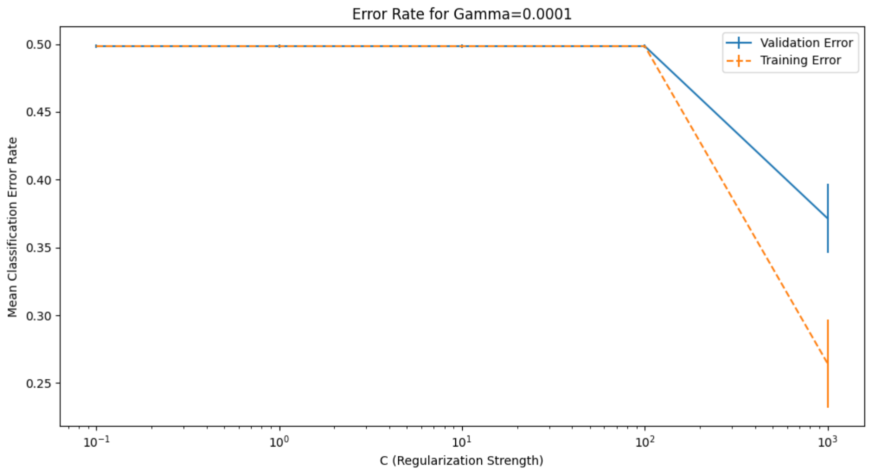


Figure 12: The graph displays the mean classification error rate for the SVM model with a gamma value of 0.0001, plotting the error rates for both the training and validation data sets across different regularization strength (c) values on a logarithmic scale, including error bars for the validation error.

The graphs shown in Figure 8, Figure 9, Figure 10, Figure 11, and Figure 12 depict the error rates for an SVM classifier with varying regularization strengths (c) across different gamma values during a hyperparameter tuning process. In these graphs, the error rates for both the training and validation sets are plotted against the logarithmic scale of ‘c’ values. From these graphs, it can be observed that as the value of ‘c’ increases, the validation error tends to decrease up to a certain point before stabilizing or increasing. This indicates that the optimal range for ‘c’ lies within this turning point. This trend is consistent across different gamma settings. The error bars, representing the standard deviation of the cross-validation scores, suggest that the average performance is relatively stable, particularly for higher ‘c’ values where the error bars are smaller.

For gamma values of 1 and 0.1 (Figure 8 and Figure 9), the validation error decreases sharply as ‘c’ increases from a low value before leveling out. This suggests that a small amount of regularization (larger ‘c’) may benefit the model. For gamma = 0.01 (Figure 10), the error rate shows a continuous decrease, but with a flatter slope. At gamma = 0.001 and 0.0001 (Figure 11 and Figure 12), the model starts to show signs of overfitting as indicated by the diverging paths of training and validation error rates. In these cases, the training error continues to decrease with higher ‘c’ values while the validation error begins to rise. This suggests that the model is fitting too closely to the training data and may not generalize well to unseen data.

The optimal hyperparameter settings exist where the validation error is lowest and the error bars are smallest, indicating both high performance and stability across different folds of the cross-validation. As discussed above, this occurs at a ‘c’ value of 10, a ‘gamma’ value of 1.

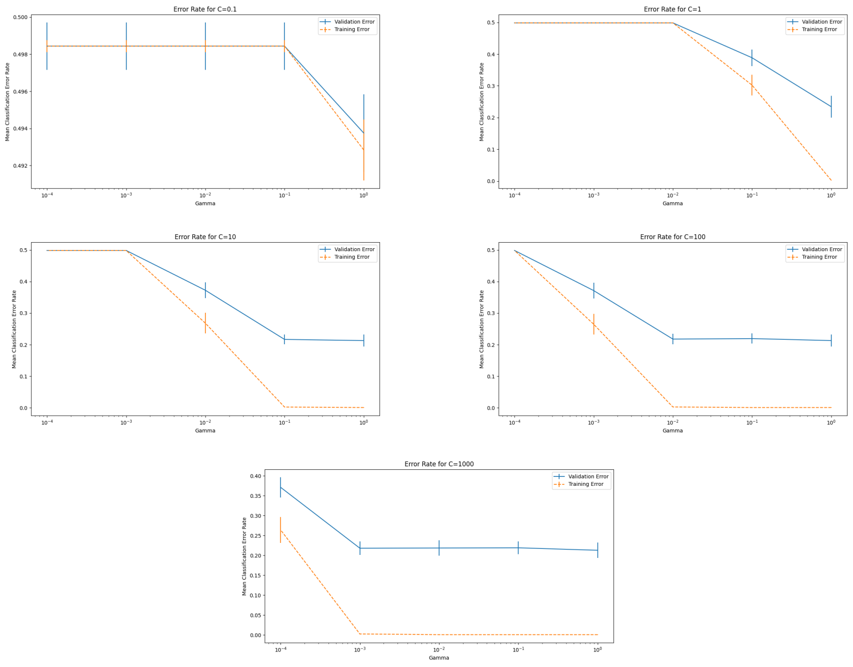
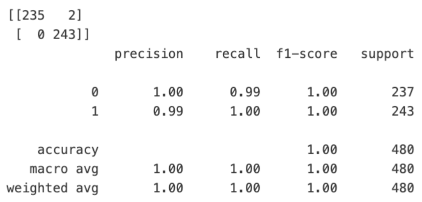


Figure 13: Error rates for different values of the regularization parameter C in the SVM model, demonstrating the impact of varying C on classifier performance.

The additional graphs provided in Figure 13 give more detail about how the SVM classifier’s performance varies with changes in the ‘gamma’ parameter for specific values of the regularization parameter ‘c’. While the first set of graphs presented the performance across different ‘c’ values for a fixed ‘gamma’, these graphs show the performance across different ‘gamma’ values for fixed ‘c’ values. Analysis of these graphs reveals several additional insights.

First, the graphs in Figure 13 demonstrate that, for each fixed value of ‘c’, varying gamma can have a significant impact on both the training and validation error rates. This underscores the importance of tuning gamma, which controls the influence of a single training example. Further, for smaller values of ‘c’ (e.g. 0.1 and 1), the model does not appear to overfit, as indicated by the close training and validation errors. However, as ‘c’ increases (e.g. 100 and 1000), the training error decreases while the validation error either increases or levels off. This signals overfitting. Also, the trend of the error rates as gamma varies is consistent across different values of ‘c’. There is a general pattern where increasing gamma leads to higher error rates, suggesting that a high gamma value may be detrimental to the model's ability to generalize, regardless of the regularization strength. Finally, unlike the first set of graphs, the graphs in Figure 13 show the impact of varying gamma on the model's uncertainty, as reflected by the error bars. The varying error bars suggests that, as gamma changes, the level of uncertainty in the model's predictions also fluctuates. This highlights the sensitivity of the SVM model to this hyperparameter.

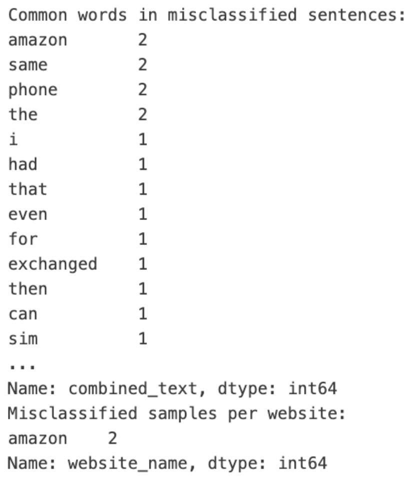
1.5 Question 5



```
[[235   2]
 [  0 243]]
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       237
           1       0.99      1.00      1.00       243

    accuracy                           1.00       480
   macro avg       1.00      1.00      1.00       480
weighted avg       1.00      1.00      1.00       480
```



```
Common words in misclassified sentences:
amazon       2
same         2
phone        2
the          2
i            1
had          1
that         1
even         1
for          1
exchanged    1
then         1
can          1
sim          1
...
Name: combined_text, dtype: int64
Misclassified samples per website:
amazon    2
Name: website_name, dtype: int64
```

Figure 14: These two outputs are produced by the logistic model's classification report. They show near-perfect performance metrics and a summary of the most common words in misclassified sentences, with all errors coming from samples associated with the Amazon website.

The logistic regression model may be outperforming the SVM and MLP models on the labeled dataset for several reasons. This could be due to several factors. First, logistic regression excels when the relationship between variables is linear. Thus, the logistic regression model may be working very well because, given the data and features at hand, it is able to find an approximate straight-line boundary that does a good job of separating the classes in the dataset. If this is true, the difference between classes may be appropriately captured by the weighted sum of input features, which is what logistic regression is designed to do. Further, the logistic regression model may be outperforming the others because it is less complex and has fewer parameters to estimate. Due to its relative simplicity, it may be less prone to overfitting than the more complex SVM and MLP models. The logistic regression's built-in handling of sparse data, such as that produced by TF-IDF vectorization, allows it to perform well without the need for extensive hyperparameter tuning. Also, logistic regression may train faster because it does not require complex operations such as matrix inversions or iterative training over multiple layers. This could lead to more efficient hyperparameter tuning and quicker convergence.

Though the SVM model and the MLP model both offer greater flexibility to model non-linear relationships, this flexibility may sometimes lead to overfitting if the model is not managed correctly with regularization, feature selection, and the right choice of hyperparameters. Thus, the logistic regression model's balance of simplicity, efficiency, and effectiveness in managing linear separability is likely making it perform better than the other models evaluated in this project.

The logistic model's confusion matrix, shown in Figure 14, indicates that the logistic regression classifier made 2 false positive errors (where the actual class was 0, but the classifier predicted 1), and made no false negative errors (where the actual class was 1, but the classifier predicted 0). The remaining 478 predictions were correct, with 235 true negatives (correctly predicted as class 0) and 243 true positives (correctly predicted as class 1). With an accuracy of 99.58% overall, this model is highly accurate on the labeled dataset. The two misclassifications are Type I errors (false positives), which may suggest that the classifier is slightly biased towards predicting class 1. This could be due to the class distribution in the training data, the features being more predictive of class 1, or the regularization parameter being tuned in such a way that it favors one class over the other.

Given the very common words in the misclassified sentences (Figure 14), the false positive errors might not be due to highly specialized or technical language in reviews but, instead, to the use of common terms that are not as informative. That is, the model may be struggling with sentences that have less distinctive language or are more generic in nature. If this is the case, the logistic model may be less effective in cases where it must rely on more subtle linguistic cues to make correct classifications. As currently constructed, the logistic classifier may require additional context or features to correctly distinguish these cases.

Both misclassified examples come from the "Amazon" website (see Figure 14), which suggests that there may be a specific characteristic of the reviews from this website that isn't being captured by the model. Thus, if "amazon" reviews are underrepresented in the training data, the model may not have learned enough about the language used in those reviews, leading to higher error rates for that source. Subsequently, the model may benefit from more representative training data from this source or feature engineering that better captures the nuances of the text from the Amazon website.

The absence of false negatives in the model's performance on the labeled data suggests that it might be confidently predicting class 1, which in an imbalanced dataset could be a sign of overfitting.

Overall, while the logistic regression model is performing very well, the limited errors that it makes seem to be associated with generic language and potentially unique features of reviews from the "Amazon" website. In order to further improve the model, it may be wise to collect more diverse training samples from Amazon, implementing different text preprocessing techniques, or exploring more complex models that can capture very subtle nuances in the text.

1.6 Question 6

The confusion matrix and the classification report from the training phase (Figure 14) show that the logistic regression model achieved near-perfect performance on the labeled dataset. This suggests that the model was able to classify almost all of the training examples correctly. However, when the logistic regression model was applied to unseen data, the performance yielded an error rate of 0.18 and an AUROC of approximately 0.89247. This indicates that 18% of predictions made on the unseen data were incorrect, which is much higher than the error rate during training/cross-validation. The disappointing performance of the logistic regression model may be the result of overfitting, where the model might have learned the idiosyncrasies of the training data—including noise and outliers—to a degree that impaired its ability to generalize to new, unseen data.

Despite the increased error rate, the logistic regression model was able to achieve an AUROC score of 0.89247 on the unseen data. This suggests that, though the model doesn't predict perfectly, it is still able to rank the positive class higher than the negative class fairly consistently. This characteristic of the model might be useful for applications where ranking is more important than absolute predictions.

The disparity between the model's training performance and its performance on unseen data underscores the importance of diverse evaluation metrics. Taken alone, accuracy can be deceptive. This is especially true for imbalanced datasets. In contrast, the AUROC score gives a more nuanced perspective of the model's performance across various classification thresholds. This makes AUROC a valuable metric for understanding the model's ability to discriminate between positive and negative reviews.

The logistic regression model's underperformance on unseen data could indicate a shift in the data distribution. Further, it may also indicate that the cross-validation process did not capture the full complexity of the data. However, overall, the model's high performance on the training set, combined with the respectable AUROC and lower accuracy on unseen data, shows that, while the model has learned certain patterns well, it may be too tailored to the training set and needs adjustments to improve its generalizability.

2 Prediction Submissions

Submitted to gradescope as per instructions.