CS 6375 Machine Learning Project 1 Report: Spam/Ham Email Classification

Prepared By:
Benjamin Walmer, bjw220002

Introduction:

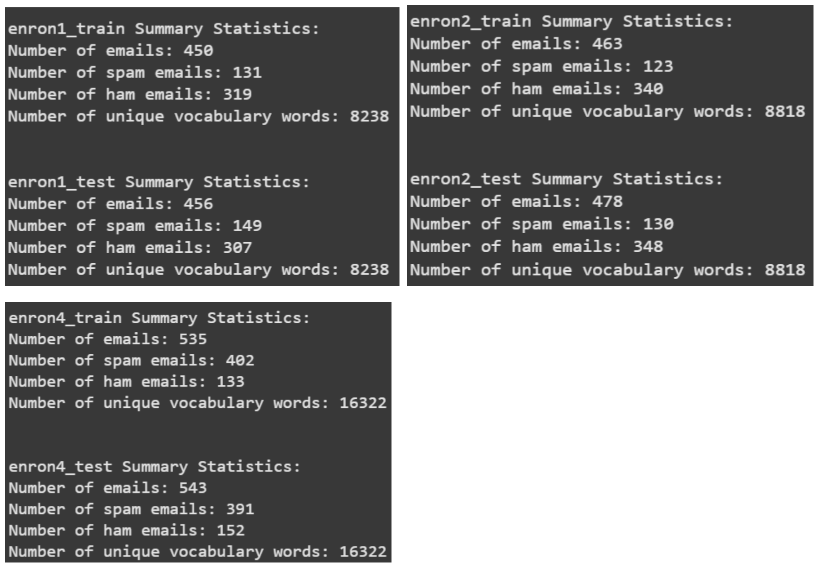
To begin this project, I first needed to load the provided datasets into my program. I used a public GitHub repository to load in the training and test datasets, and I used Pandas dataframes to store them.

Data Preparation:

Next, I needed to preprocess and prepare the data in order to store the emails as feature matrices. For this purpose, I imported the NLTK (Natural Language ToolKit) library, as well as the re and collections libraries. I used the NLTK library because it has useful methods such as word_tokenize, which I used to split the text into individual words (tokens) that my program could process. It also provided a list of stopwords that I used to filter out common words found across all emails. I used the re library to filter through the emails to remove punctuation, special characters, and common words found across all emails like "Subject:" "Forwarded", etc. I used the collections library, specifically an object called Counter, because it provided a dictionary-like structure to store all the words from the emails. It had useful methods like sorted, which sorts the words in alphabetical order, and most_common, which provides a list of the most common words in the counter, which I used when storing the vocabulary.

Bag of Words and Bernoulli Representations:

Using the libraries described above, I stored the emails from each dataset using a bag of words representation and a Bernoulli representation. The bag of words representation counts the frequency of each word, while the Bernoulli model simply records if a word is present or not (0 or 1). The output below describes some summary statistics from the 6 different datasets, which have the same number of emails and vocabulary words for both representations.



```
enron1_train Summary Statistics:
Number of emails: 450
Number of spam emails: 131
Number of ham emails: 319
Number of unique vocabulary words: 8238


enron1_test Summary Statistics:
Number of emails: 456
Number of spam emails: 149
Number of ham emails: 307
Number of unique vocabulary words: 8238
```

```
enron2_train Summary Statistics:
Number of emails: 463
Number of spam emails: 123
Number of ham emails: 340
Number of unique vocabulary words: 8818


enron2_test Summary Statistics:
Number of emails: 478
Number of spam emails: 130
Number of ham emails: 348
Number of unique vocabulary words: 8818
```

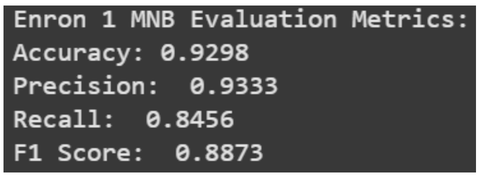
```
enron4_train Summary Statistics:
Number of emails: 535
Number of spam emails: 402
Number of ham emails: 133
Number of unique vocabulary words: 16322


enron4_test Summary Statistics:
Number of emails: 543
Number of spam emails: 391
Number of ham emails: 152
Number of unique vocabulary words: 16322
```

Figure 1: Summary Statistics across all 6 datasets.

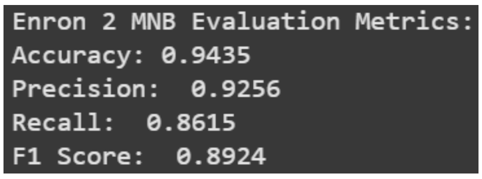
Notice that enron4_test has nearly double the amount of vocabulary words as the first two datasets. This is likely because it consists of majority spam emails, which may be longer and have more words than non-spam (ham) emails.

Multinomial Naive Bayes Models:

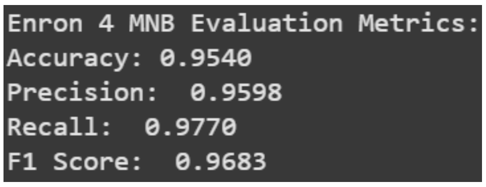
As described in the instructions for this project, I implemented the multinomial naive bayes model for the bag of words representation across all 3 datasets. Below are the outputs for all 3 test datasets using these models:



```
Enron 1 MNB Evaluation Metrics:
Accuracy: 0.9298
Precision:  0.9333
Recall:  0.8456
F1 Score:  0.8873
```



```
Enron 2 MNB Evaluation Metrics:
Accuracy: 0.9435
Precision:  0.9256
Recall:  0.8615
F1 Score:  0.8924
```



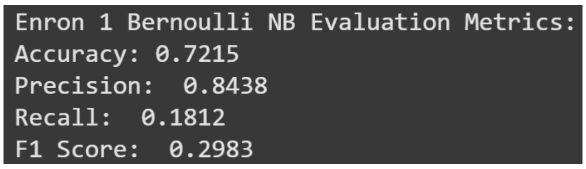
```
Enron 4 MNB Evaluation Metrics:
Accuracy: 0.9540
Precision:  0.9598
Recall:  0.9770
F1 Score:  0.9683
```

Figure 2: Multinomial Naive Bayes Model Evaluation across all 3 test datasets

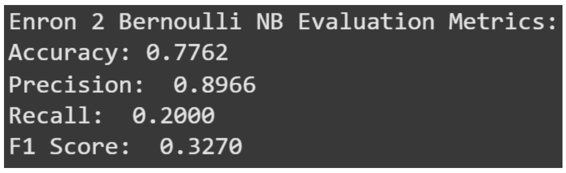
From these outputs, we see that the multinomial naive bayes model does quite well across all datasets, with accuracies of above 92% for all 3 models. Each model also has a high F1 score of between .88 and .97, meaning the models are good at both spam detection and at catching false alarms. The models also all have high precision values of over 90%, meaning that when each model predicts an email as "spam", it is correct more than 90% of the time. The recall is relatively high across all datasets, especially for the enron_4 dataset. For this model, the recall value of 0.9770 means that this model is correctly classifying over 97% of all spam emails in the enron_4 dataset, which is very impressive. Overall, the multinomial naive bayes model performed very well across all 3 datasets, especially on the enron_4 dataset which contained majority spam emails. It seems that the frequency of certain words is a very good predictor of whether an email is spam or not using the multinomial bayes model.

Bernoulli Naive Bayes Models:

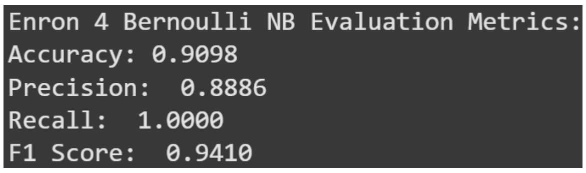
As described in the project instructions, I implemented a Bernoulli naive bayes model for the Bernoulli representations of the training data across all 3 datasets. Below are the outputs for all 3 test datasets using these models.



```
Enron 1 Bernoulli NB Evaluation Metrics:
Accuracy: 0.7215
Precision:  0.8438
Recall:  0.1812
F1 Score:  0.2983
```



```
Enron 2 Bernoulli NB Evaluation Metrics:
Accuracy: 0.7762
Precision:  0.8966
Recall:  0.2000
F1 Score:  0.3270
```



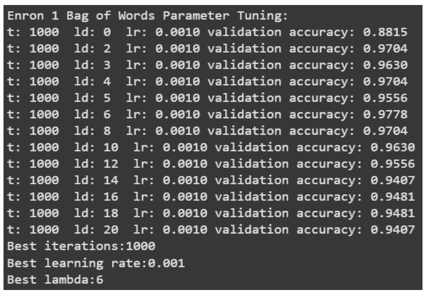
```
Enron 4 Bernoulli NB Evaluation Metrics:
Accuracy: 0.9098
Precision:  0.8886
Recall:  1.0000
F1 Score:  0.9410
```

Figure 3: Bernoulli Naive Bayes Model Evaluation across all 3 test datasets

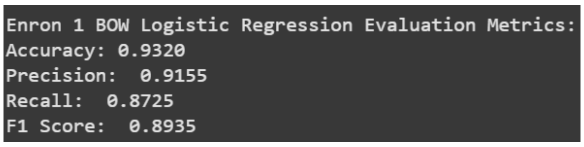
From these outputs, we see that for enron_1 and enron_2, the model accuracy has decreased significantly from the multinomial naive bayes models, with accuracies between 70 and 80%. While the precision is still relatively high, meaning when these models do predict spam they are 80-90% accurate, the recall is very low. These values explain that these models only found roughly 18-20% of the spam emails in their datasets, which also explains their low F1-scores. For the enron 4 dataset on the other hand, the accuracy and F1-scores decreased slightly from the multinomial bayes model (from roughly 95% to 91%, and from roughly 0.97 to 0.94). The recall of 1 means that every spam email in the dataset was correctly predicted as spam, however the precision of .886 means that roughly 12% of the emails that weren't spam were incorrectly predicted as spam. Overall, the Bernoulli naive bayes models performed worse than the multinomial naive bayes models, especially on the enron_1 and enron_2 datasets. Because there are so many vocabulary words, most of which are not contained in any single email, the way the Bernoulli naive bayes model makes predictions will be dominated by the priors. This is why for enron_1 and enron_2, the recall was so low, because it predicted majority ham emails since its original (prior) datasets had majority ham emails. And this is why for enron_4 we found a perfect recall of 1, since for this dataset the prior dataset had majority spam emails, so it predicted majority spam emails as well. We can conclude from this model that the presence of a word is not as influential as its frequency in these datasets when we are predicting whether an email is spam or not.

Logistic Regression Models:

I originally implemented the logistic regression model as described in the project handout. However, using for loops in all instances made the model have a very slow runtime, since O(nd) was very high for d = over 8,000 words in each dataset. I used AI to help vectorize some of the matrix multiplication so I could run logistic regression in a reasonable time on my personal computer. I also used the train_test_split method in sklearn to split the training dataset into validation and training data, with a random state of 11. When tuning hyper-parameters, I decided to choose a fixed number of iterations, T = 1000. I tried other values, T = 500, T = 750, T = 1250, for example. 1000 iterations provided a good balance of allowing my model to converge with a good accuracy while also not taking forever to run. For my learning rate, I also chose a fixed value of 0.001. I tried other learning rates, (0.0025, 0.005, 0.0075, 0.01), but this learning rate also provided a good balance of converging while also not dominating runtime. For each dataset, I tuned lambda, the regularization parameter. I tried to find the lambda value where accuracy on the validation set was the highest, and I used a range of lambdas to try to find the optimal value of lambda where both lambdas higher and lower had worse accuracies. Below are my outputs for tuning lambda for each of the 3 datasets and 2 representations, as well as their performance on the test datasets using the optimal lambda.



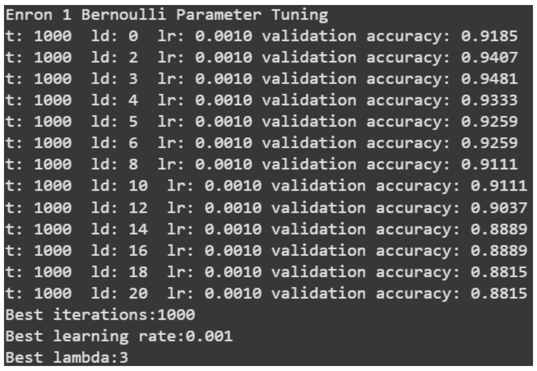
```
Enron 1 Bag of Words Parameter Tuning:
t: 1000  ld: 0  lr: 0.0010 validation accuracy: 0.8815
t: 1000  ld: 2  lr: 0.0010 validation accuracy: 0.9704
t: 1000  ld: 3  lr: 0.0010 validation accuracy: 0.9630
t: 1000  ld: 4  lr: 0.0010 validation accuracy: 0.9704
t: 1000  ld: 5  lr: 0.0010 validation accuracy: 0.9556
t: 1000  ld: 6  lr: 0.0010 validation accuracy: 0.9778
t: 1000  ld: 8  lr: 0.0010 validation accuracy: 0.9704
t: 1000  ld: 10  lr: 0.0010 validation accuracy: 0.9630
t: 1000  ld: 12  lr: 0.0010 validation accuracy: 0.9556
t: 1000  ld: 14  lr: 0.0010 validation accuracy: 0.9407
t: 1000  ld: 16  lr: 0.0010 validation accuracy: 0.9481
t: 1000  ld: 18  lr: 0.0010 validation accuracy: 0.9481
t: 1000  ld: 20  lr: 0.0010 validation accuracy: 0.9407
Best iterations:1000
Best learning rate:0.001
Best lambda:6
```



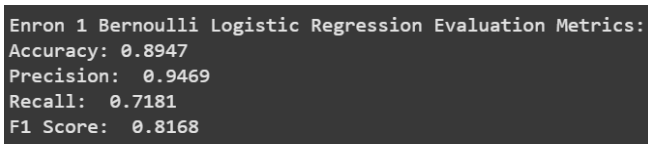
```
Enron 1 BOW Logistic Regression Evaluation Metrics:
Accuracy: 0.9320
Precision:  0.9155
Recall:  0.8725
F1 Score:  0.8935
```

Figure 4: Logistic Regression Tuning and Performance for Enron_1 BOW Representation

For the Enron_1 bag of words representation, we see that the logistic regression model had the highest accuracy on the validation set with a lambda value of 6. This seems to be the optimal lambda value, as lower values (0-5) and higher values (8-20) have lower accuracies. In terms of performance, the logistic regression model performs quite well, even slightly out-performing the multinomial naive bayes equivalent in terms of its accuracy and f1-score.



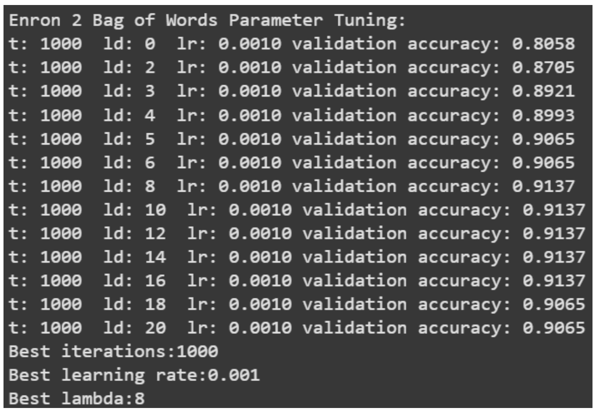
```
Enron 1 Bernoulli Parameter Tuning
t: 1000  ld: 0  lr: 0.0010 validation accuracy: 0.9185
t: 1000  ld: 2  lr: 0.0010 validation accuracy: 0.9407
t: 1000  ld: 3  lr: 0.0010 validation accuracy: 0.9481
t: 1000  ld: 4  lr: 0.0010 validation accuracy: 0.9333
t: 1000  ld: 5  lr: 0.0010 validation accuracy: 0.9259
t: 1000  ld: 6  lr: 0.0010 validation accuracy: 0.9259
t: 1000  ld: 8  lr: 0.0010 validation accuracy: 0.9111
t: 1000  ld: 10  lr: 0.0010 validation accuracy: 0.9111
t: 1000  ld: 12  lr: 0.0010 validation accuracy: 0.9037
t: 1000  ld: 14  lr: 0.0010 validation accuracy: 0.8889
t: 1000  ld: 16  lr: 0.0010 validation accuracy: 0.8889
t: 1000  ld: 18  lr: 0.0010 validation accuracy: 0.8815
t: 1000  ld: 20  lr: 0.0010 validation accuracy: 0.8815
Best iterations:1000
Best learning rate:0.001
Best lambda:3
```



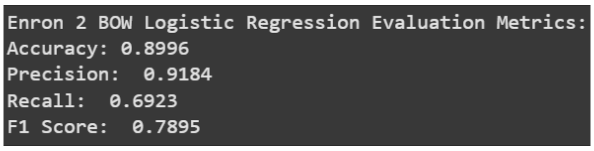
```
Enron 1 Bernoulli Logistic Regression Evaluation Metrics:
Accuracy: 0.8947
Precision:  0.9469
Recall:  0.7181
F1 Score:  0.8168
```

Figure 5: Logistic Regression Tuning and Performance for Enron_1 Bernoulli Representation

For the Bernoulli representation of enron_1, a lambda value of 3 was chosen, as higher and lower lambdas tended to have lower accuracies. In terms of model performance, the logistic regression model greatly outperformed its naive bayes counterpart, raising the accuracy from 72% to 89% and the F1-score from around 0.3 to 0.82. The logistic regression model did a much better job of identifying the spam emails correctly, as its recall of 0.72 is much higher than the naive bayes recall of 0.18.



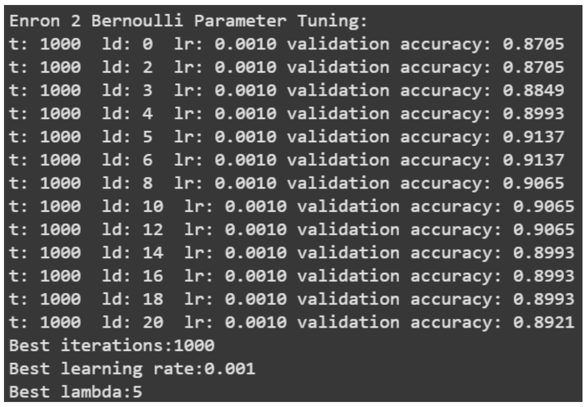
```
Enron 2 Bag of Words Parameter Tuning:
t: 1000  ld: 0  lr: 0.0010 validation accuracy: 0.8058
t: 1000  ld: 2  lr: 0.0010 validation accuracy: 0.8705
t: 1000  ld: 3  lr: 0.0010 validation accuracy: 0.8921
t: 1000  ld: 4  lr: 0.0010 validation accuracy: 0.8993
t: 1000  ld: 5  lr: 0.0010 validation accuracy: 0.9065
t: 1000  ld: 6  lr: 0.0010 validation accuracy: 0.9065
t: 1000  ld: 8  lr: 0.0010 validation accuracy: 0.9137
t: 1000  ld: 10  lr: 0.0010 validation accuracy: 0.9137
t: 1000  ld: 12  lr: 0.0010 validation accuracy: 0.9137
t: 1000  ld: 14  lr: 0.0010 validation accuracy: 0.9137
t: 1000  ld: 16  lr: 0.0010 validation accuracy: 0.9137
t: 1000  ld: 18  lr: 0.0010 validation accuracy: 0.9065
t: 1000  ld: 20  lr: 0.0010 validation accuracy: 0.9065
Best iterations:1000
Best learning rate:0.001
Best lambda:8
```



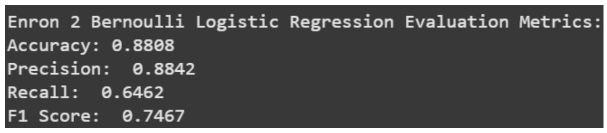
```
Enron 2 BOW Logistic Regression Evaluation Metrics:
Accuracy: 0.8996
Precision:  0.9184
Recall:  0.6923
F1 Score:  0.7895
```

Figure 6: Logistic Regression Tuning and Performance for Enron_2 BOW Representation

For the bag of words representation of enron_2, a lambda value of 8 was chosen, as higher and lower lambda values had either the same accuracy or a lower accuracy on the validation set. In terms of model performance, the logistic regression model underperformed compared to its naive bayes counterpart, as its accuracy of 0.9 was lower than the naive bayes accuracy of 0.94. This model did a much worse job of identifying the spam emails correctly, as its recall went from 0.86 to 0.6923, which is why its f1 score is also lower.



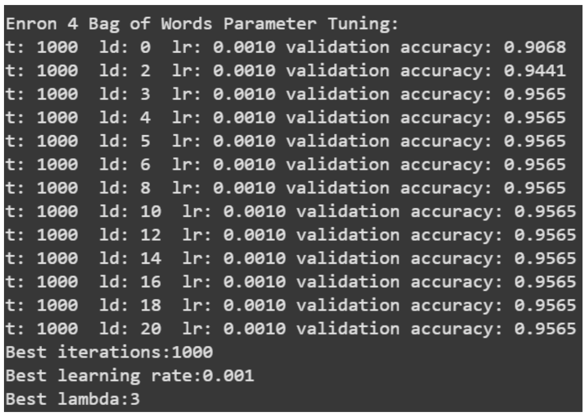
```
Enron 2 Bernoulli Parameter Tuning:
t: 1000  ld: 0  lr: 0.0010 validation accuracy: 0.8705
t: 1000  ld: 2  lr: 0.0010 validation accuracy: 0.8705
t: 1000  ld: 3  lr: 0.0010 validation accuracy: 0.8849
t: 1000  ld: 4  lr: 0.0010 validation accuracy: 0.8993
t: 1000  ld: 5  lr: 0.0010 validation accuracy: 0.9137
t: 1000  ld: 6  lr: 0.0010 validation accuracy: 0.9137
t: 1000  ld: 8  lr: 0.0010 validation accuracy: 0.9065
t: 1000  ld: 10  lr: 0.0010 validation accuracy: 0.9065
t: 1000  ld: 12  lr: 0.0010 validation accuracy: 0.9065
t: 1000  ld: 14  lr: 0.0010 validation accuracy: 0.8993
t: 1000  ld: 16  lr: 0.0010 validation accuracy: 0.8993
t: 1000  ld: 18  lr: 0.0010 validation accuracy: 0.8993
t: 1000  ld: 20  lr: 0.0010 validation accuracy: 0.8921
Best iterations:1000
Best learning rate:0.001
Best lambda:5
```



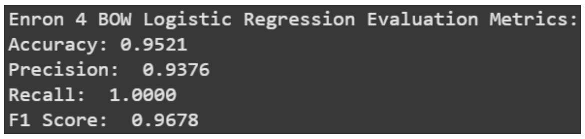
```
Enron 2 Bernoulli Logistic Regression Evaluation Metrics:
Accuracy: 0.8808
Precision:  0.8842
Recall:  0.6462
F1 Score:  0.7467
```

Figure 7: Logistic Regression Tuning and Performance for Enron_2 Bernoulli Representation

For the Bernoulli representation, we see that a lambda value of 5 was chosen since it had the highest accuracy on the validation set (ties go to the lower lambda value, which is why lambda = 6 was not chosen). The logistic regression model performed much better than its naive Bayes counterpart, as its accuracy increased from 77 to 88%, and its f1-score increased from 0.32 to 0.74. As with the Bernoulli representation of enron_1, the logistic regression model did a much better job of identifying the spam emails correctly in the dataset, as its recall went from 0.2 to 0.64, which is why its f1-score is also significantly higher.



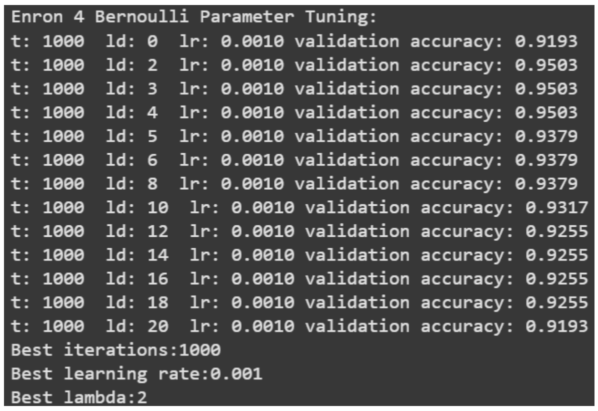
```
Enron 4 Bag of Words Parameter Tuning:
t: 1000  ld: 0  lr: 0.0010 validation accuracy: 0.9068
t: 1000  ld: 2  lr: 0.0010 validation accuracy: 0.9441
t: 1000  ld: 3  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 4  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 5  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 6  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 8  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 10  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 12  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 14  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 16  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 18  lr: 0.0010 validation accuracy: 0.9565
t: 1000  ld: 20  lr: 0.0010 validation accuracy: 0.9565
Best iterations:1000
Best learning rate:0.001
Best lambda:3
```



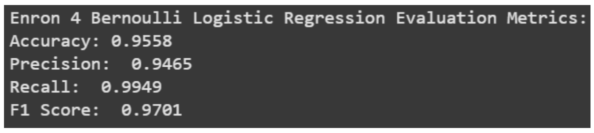
```
Enron 4 BOW Logistic Regression Evaluation Metrics:
Accuracy: 0.9521
Precision:  0.9376
Recall:  1.0000
F1 Score:  0.9678
```

Figure 8: Logistic Regression Tuning and Performance for Enron_4 BOW Representation

For the bag of words representation, a lambda of 3 was chosen, as it had the highest accuracy at the lowest lambda value. The logistic regression model performed similarly to its naive bayes counterpart, as they both had roughly 95% accuracy. While the logistic regression model has higher recall (a perfect recall actually), it has a lower precision, and results in nearly the same f1-score as the naive bayes model of the same dataset.



```
Enron 4 Bernoulli Parameter Tuning:
t: 1000  ld: 0  lr: 0.0010 validation accuracy: 0.9193
t: 1000  ld: 2  lr: 0.0010 validation accuracy: 0.9503
t: 1000  ld: 3  lr: 0.0010 validation accuracy: 0.9503
t: 1000  ld: 4  lr: 0.0010 validation accuracy: 0.9503
t: 1000  ld: 5  lr: 0.0010 validation accuracy: 0.9379
t: 1000  ld: 6  lr: 0.0010 validation accuracy: 0.9379
t: 1000  ld: 8  lr: 0.0010 validation accuracy: 0.9379
t: 1000  ld: 10  lr: 0.0010 validation accuracy: 0.9317
t: 1000  ld: 12  lr: 0.0010 validation accuracy: 0.9255
t: 1000  ld: 14  lr: 0.0010 validation accuracy: 0.9255
t: 1000  ld: 16  lr: 0.0010 validation accuracy: 0.9255
t: 1000  ld: 18  lr: 0.0010 validation accuracy: 0.9255
t: 1000  ld: 20  lr: 0.0010 validation accuracy: 0.9193
Best iterations:1000
Best learning rate:0.001
Best lambda:2
```



```
Enron 4 Bernoulli Logistic Regression Evaluation Metrics:
Accuracy: 0.9558
Precision:  0.9465
Recall:  0.9949
F1 Score:  0.9701
```

Figure 9: Logistic Regression Tuning and Performance for Enron_4 Bernoulli Representation

For the Bernoulli representation of the enron_4 dataset, tuning the model produced a lambda value of 2 which had the highest accuracy compared with other lambda values on the validation set. The model outperformed its naive bayes counterpart in terms of accuracy (95% > 90%) and in terms of its f1-score (.97 > .94). It seems to do a better job than the naive bayes model of not predicting ham emails as spam, as its precision increased from around 0.89 to 0.94.

Conclusions:

Overall, I believe the logistic regression performed better, on average across all datasets, than the naive bayes model. The logistic regression model performed similarly to the multinomial naive bayes model and much better than the Bernoulli naive bayes model in terms of accuracy and f1 score. Intuitively, it seems that a more complex model would be a better predictor for complex data like email text, and tuning the hyperparameters allows for more control in the model than in the naive bayes model. The logistic regression model captures more of the complex relationships that may be present in the emails. Additionally, the naive bayes model assumes feature independence, which is likely violated as the likelihood of some words being in an email may increase depending on the presence of other words being in the same email. While this assumption being violated may not greatly impact the naive bayes model's performance, these are a few of the reasons for why the logistic regression may have performed better on this dataset than the naive bayes model.

Across all datasets, the multinomial naive bayes model on the bag of words representation had the highest accuracy and performance. This makes sense as this model and representation accounts for the frequencies of words in each email. Spam emails may be more likely to have repeated words or phrases, which would not be accounted for in the Bernoulli representation. The multinomial naive bayes model factors in these frequencies which may have been a reason why it slightly outperformed the logistic regression model, across all datasets, in terms of accuracy and f1 score for the bag of words representation.

For the Bernoulli representations of the data, the logistic regression model greatly outperformed the Bernoulli naive bayes model. As previously described, the naive bayes model accounts for the absence of words as well as their appearance in an email. These absence terms lead to a smaller weight for the likelihood, which means the prior likelihoods dominate and therefore don't lead to great predictions accuracies For example in enron_1 and enron_2, the priors were majority ham and therefore the recall (percentage of spam emails being correctly predicted) was very low. The logistic regression model on the other hand, does a better job of learning the decision boundary between the ham and spam emails. This model was more flexible and much more accurate on the Bernoulli representations of the data than the naive bayes model.