JSC370 2025 - Final Report

Afra Azad

April 28, 2025

1 Introduction

Spam emails can be annoying and dangerous, often attempting to trick individuals into revealing sensitive information or falling victim to fraud. Studies show that 90% of successful cyberattacks originate from phishing attempts, making developing strong detection and prevention strategies critical. With the growing volume of spam emails circulating online, an automated approach to identifying such messages could significantly enhance cybersecurity and protect users. For my research project, I aim to answer the question: *Can we develop a model to detect spam emails and identify which keywords most strongly indicate spam content?*

I will explore this question using the CEAS, Nazario, Nigerian Fraud, and Spam-Assassin datasets.[1] These datasets provide a diverse range of email types collected across different periods: CEAS (2008) was built from spam collected through broadband network honeypots [2]; the Nazario dataset (2005) consists of hand-screened phishing emails gathered from public sources [3]; the Nigerian Fraud corpus (1998–2007) features classic "419" advance-fee scam emails [4]; and the SpamAssassin corpus (early 2000s) was curated to provide a benchmark set of spam and non-spam emails for spam filter development [5]. Each data point contains the body of the email, subject lines, sender, receiver, date sent, URL link, and a label indicating whether it is spam or not.

Using this dataset, I aim to train a machine-learning model to classify spam emails. Additionally, I will identify the most common keywords that contribute to scam classification, helping users recognize potential threats in their inboxes. Logistic regression, neural networks, and Bernoulli naive bayes will be tested as prediction models, as they can fit categorical and numerical data for binary classification.

2 Methods

I will conduct the analysis using built-in Python in RStudio. The spam email dataset was sourced from Kaggle, which combines the four different datasets: CEAS, Nazario, Nigerian Fraud, and Spam-Assassin. Since it was not possible to access the Kaggle API directly from RStudio, I instead concatenated the four datasets using Google Colab, ensuring that the columns were properly aligned throughout the process. Then, I will load the full dataset as a Pandas dataframe.

Firstly, I will examine the dataset's structure, such as its headers, tail, and dimensions. To clean the data, I checked for null values, data types and outliers. Variables that are not the correct data type, such as date-time, will be converted to the appropriate data type. If there are null values for the crucial variables email body, label, or date, then I will remove that data point. To utilize the timestamp of the email sent in our analysis, I will extract the hour in the date-time variable. Outlier data points will be examined by looking at the maximum and minimum quantities; however, I will only remove those data points if they are email bodies, labels, or dates. Then, I will generate summary statistics for the column, which includes count tables for distinct data, and descriptions for text and date. I will generate top 10 frequency tables for subject lines, senders, and receivers, and graphs for the date and time variables. Also, to consider the sender and receiver emails in the analysis, I will add the sender and receiver emails at the beginning of the email body with the '@' removed.

To clean the email text body, I will convert the text to lowercase and remove all non-alphabetic characters. Next, I will split the body into individual words based on spaces and filter out stop words (for example, “the”, “is”), hyperlinks, single-letter words, and other terms that lack informative value. Then, I normalize the tokens by applying stemming and lemmatization, where stemming reduces words to their root form (e.g., ”running” becomes ”run”), and lemmatization maps words to their dictionary base form, typically in the present simple tense (e.g., ”was” becomes ”be”). This careful selection process ensured that only meaningful words remained for analysis. The data will then be divided into a 10% train-test split.

The tokenized email body and date will be used for three interactive data visualizations to examine the word patterns for spam and spam emails. The first plot will show the relationship between word frequency and TF-IDF score for spam and non-spam emails, where the TF-IDF score measures how frequent a word is in a specific email relative to its importance across the entire dataset. The second plot will present the top 20 most frequent bigrams, labelled by whether they appear in spam or non-spam emails. The third plot is the top 10 most frequent email body tokens for each dataset, broken down by source, and labelled as spam or non-spam. The fourth plot will display the distribution of hours of the day when emails are sent for spam and non-spam emails. If I find a significant difference in timestamps for spam and non-spam emails, then I will consider the timestamp as a prediction variable alongside the email body tokens. There will also be a statistical summary table to discover the differences in lexical diversity between spam and non-spam emails. Lexical diversity measures the percentage of unique words within each group; a lower lexical diversity indicates a higher repetition of words.

Before modelling, I will pre-process the data by converting each email body into a bag-of-words representation, a vector where each index corresponds to a word in the overall vocabulary, and the value indicates the frequency of the word in the email. Since there is limited memory, I will consider only the top 6000 most frequent terms in our vocabulary. I will also include the hour of the day the email is sent as an additional feature, and standardize it to have a mean of zero and a variance of 1. In total, the vector will have 6001 features.

The three binary prediction models I will test are Bernoulli Naive Bayes, Logistic Regression, and Multi-Layer Perceptrons (neural networks). Naive Bayes is a probabilistic classifier that assumes features are conditionally independent given the class label. It is useful for text classification tasks due to its simplicity and ability to handle high-dimensional data. Logistic Regression models the probability of an email being spam using a sigmoid function applied to a linear combination of features. It is simple to interpret, but it is only effective when the data is linearly separable. Multi-Layer Perceptrons (MLPS) are feedforward neural networks that can learn complex patterns by combining multiple layers of nonlinear transformations. They can capture more abstract relationships in the data that may not be linearly separable.

I will perform a grid search with 5-fold cross-validation to systematically explore key hyperparameter values for each model type and identify the configuration that yields the best performance. In 5-fold cross-validation, the training data is split into five equal parts: in each iteration, four parts are used to train the model and the remaining part is used for validation. This process repeats five times, with each subset serving as the validation set once, and the model’s performance is averaged across all folds to utilize bagging and ensure a more reliable estimate. For Naive Bayes, tune the alpha parameter, which smooths probability estimates by incorporating prior beliefs. For logistic regression, tune the regularization strength, which penalizes large coefficient values to prevent overfitting. For MLPS, adjust the number of hidden layers and the number of neurons in each layer to optimize the model’s ability to learn complex patterns in the data. The activation function between each hidden layer is a rectifier, which outputs the maximum between its input and zero.

Using the testing data, I will evaluate each model based on its best hyperparameters by examining key performance metrics. A confusion matrix will be generated to display the number of true positives and true negatives (correct predictions) as well as false positives and false negatives (incorrect predictions). In this context, a positive refers to a spam email, and a negative refers to a non-spam email. Additionally, I will analyze the false positive rate, which measures the proportion of non-spam emails incorrectly classified as spam, and the false negative rate, which measures the proportion of spam emails incorrectly classified as non-spam. Since false positives, where legitimate emails are misclassified as spam, pose a more serious issue than false negatives, I will prioritize selecting a model that achieves high overall accuracy while minimizing the false positive rate.

Finally, I will determine which words significantly contribute to spam and non-spam predictions, and test out our best model with some new email examples that include email sender, receiver, and timestamp.

3 Results

The dataset consists of 49,860 rows and 7 columns, with 27064 spam and 20519 non-spam emails. Notable findings based on the summary statistics are that the majority of the emails were sent in 2001, the minimum date in the dataset was recorded as the year 102, and the maximum date was 2100. When examining these points individually, I found that the email body and label for these entries appeared normal, suggesting the date values might be typos. Since the date column is used in our analysis, I decided to remove these data points.

Subject Line for Spam Emails	**Frequency**
CNN.com Daily Top 10	2930
CNN Alerts: My Custom Alert	1406
Re:	585
123	255
Re:	123
How To Enlarge Penis Size	100
Qualitative replica watches for most exacting people	65
Penis Enlargment Reviews	62
Produce Stronger, Rock Hard Erections.	61
You our client!	61

Subject Line for Non-Spam Emails	**Frequency**
Re: [opensuse] openSUSE 11.0 and the Non-ready KDE4	103
Direct message from SpamExperts via web	84
Re: [opensuse] Defragging: possible? necessary?	76
Re: [opensuse] [OT] How much power does a PC really consume?	74
Re: [opensuse] ext3 check forced = frustration	69
Re: [Python-Dev] PEP 365 (Adding the pkg_resources module)	62
Re: [Python-3000] u'text' as an alias for 'text'?	61
Re: [Python-3000] range() issues	51
Re: [opensuse] [OT] How much power does a PC really consume?	50
Re: [opensuse] true news group?	49

Table 1: The table at the top lists the top 10 subject lines in spam emails across the four datasets. Most of these are "CNN Daily's Top 10," suggesting that many spam emails are disguised as daily corporate emails. The bottom table shows the top subject lines for non-spam emails, which are often follow-ups.

There are some notable findings in the variable charts. Based on Table 1, the most common subject lines for spam emails are "CNN Daily's Top 10," suggesting that many spam emails are disguised as daily corporate emails.

Sender	**Count**
qydlqcws-iacfym@issues.apache.org	462
Guido van Rossum <hoauf@python.org>	295
“Martin v. Löwis” <qpnysl@v.loewis.de>	276
“Carlos E. R.” <vyjwd.trpcau@telefonica.net>	208
Aaron Kulkis <cmiqlkx91@hotpop.com>	183
Rafael Garcia-Suarez <pvhuhqgncrxnu@gmail.com>	158
Christian Heimes <wluhe@cheimes.de>	152
Barry Warsaw <pjaxq@python.org>	131
iybz@pobox.com	124
Per Jessen <uee@computer.org>	113

Receiver	**Count**
user6@gvc.ceas-challenge.cc	1375
wkilxloc@opensuse.org	1230
user2.1@gvc.ceas-challenge.cc	1037
jose@monkey.org	938
user2.2@gvc.ceas-challenge.cc	922
user2.4@gvc.ceas-challenge.cc	738
user2.13@gvc.ceas-challenge.cc	686
user7@gvc.ceas-challenge.cc	674
user8.2-ext1@gvc.ceas-challenge.cc	655
user7-ext4@gvc.ceas-challenge.cc	655

Table 2: The top table shows the top 10 senders, while the bottom table shows the top 10 receivers in our database across the four datasets. Many of the most common senders and receivers are bot-generated email addresses that cannot be easily removed, which may introduce some limitations to our analysis.

Table 2 with the top 10 most common senders and receivers shows that the most common senders and receivers are bot-generated email addresses that cannot be easily removed, which may introduce some limitations in our modelling.

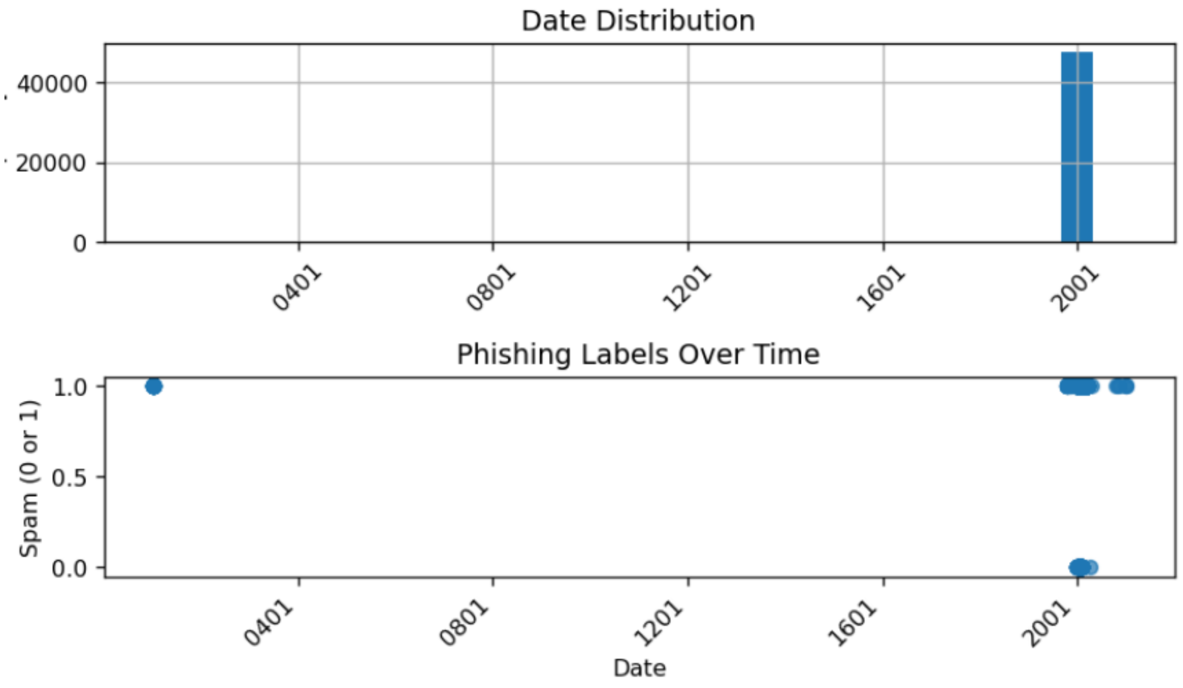

Figure 1: Top plot is the distribution of the date an email is sent, and bottom plot is the number of spam and non-spam emails sent over time. It appears that nearly all the emails were sent in 2001, and spam emails were sent within a wider period than non-spam emails.

I examined the distribution of emails over time and found that nearly all the emails were sent in 2001, as shown in Figure 1. Additionally, analyzing spam labels over time revealed that non-spam emails sent were concentrated near the beginning of 2001, whereas spam emails were sent throughout 2001.

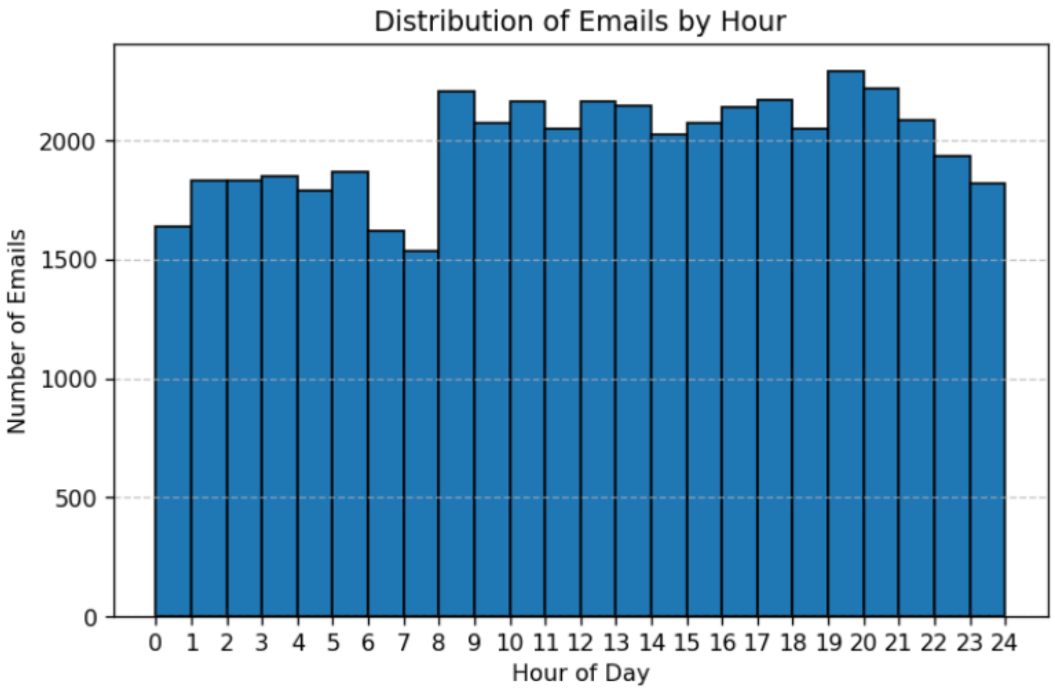

Figure 2: Histogram of the hour of the day an email is sent. It appears to be uniformly distributed, with more emails sent during the day between 8 am to 10 pm.

In Figure 2, we can see that most emails are sent during the day from 8 am to 10 pm, which makes sense as people will not be sending emails during the night.

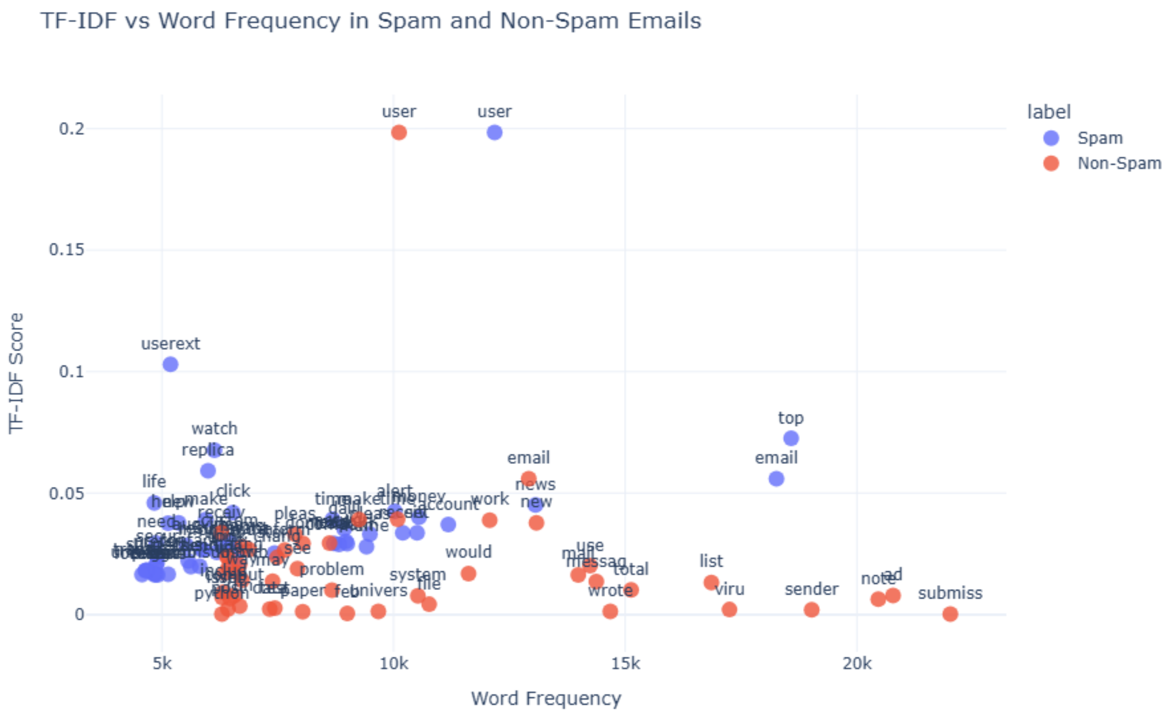

Figure 3: Scatterplot depicts the relationship between word frequency and TF-IDF score for the top 50 most frequent spam and non-spam words. There is a general linear relationship between TF-IDF score for frequency of spam email words, and a weak relationship between TF-IDF and frequency of non-spam email words. This could indicate that words in spam emails that are generally quite frequent are specifically frequent within spam emails.

The three visualizations showed interesting trends in the spam and non-spam words. I have provided a screenshot of all the interactive figures for ease of reference, but they are also in the visuals tab on the website. In the website Figure 3, there is a linear relationship between word frequency and TF-IDF. However, non-spam word frequency has little to no relationship with TF-IDF and generally has a lower TF-IDF score. This indicates that frequent words in spam emails tend to be highly distinctive to spam (having high TF-IDF scores), while frequent words in non-spam emails are more common across a broader range of email topics, leading to weaker TF-IDF scores.

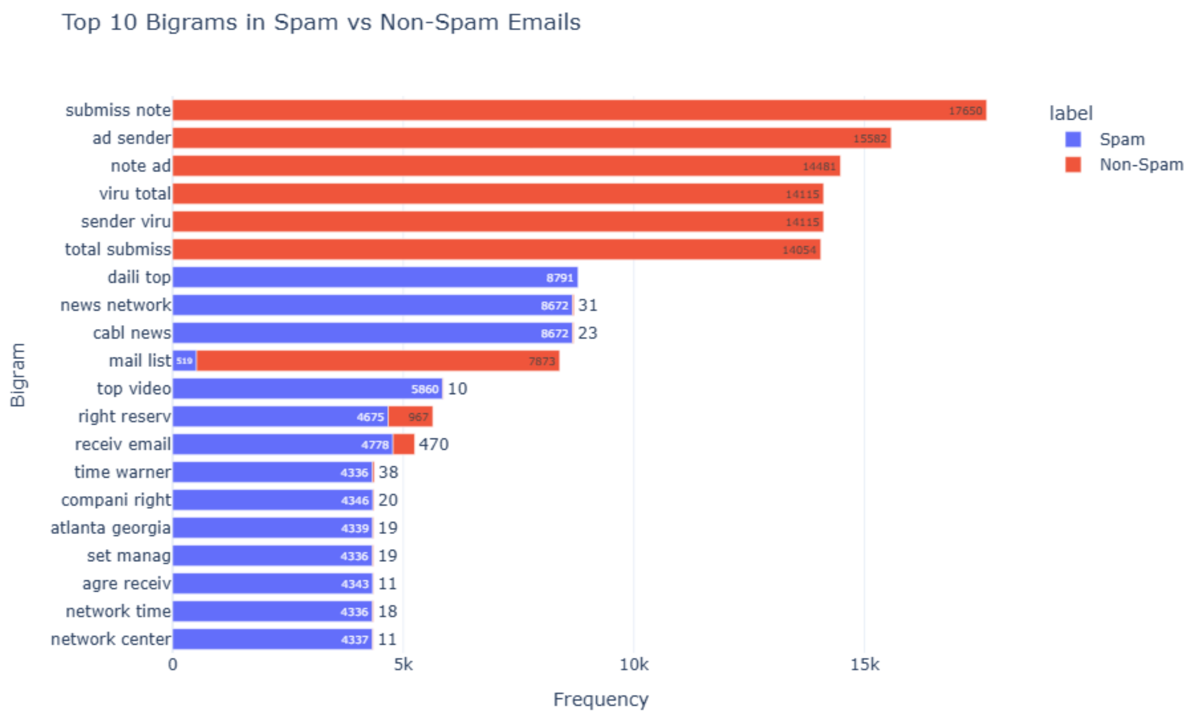

Figure 4: This chart shows the top 20 word pairs (bi-grams) for both spam and non-spam emails. All of the top 8 bi-grams are from non-spam emails, which indicates spam emails might use a wider vocabulary of words compared to non-spam emails.

In the figure on Figure 4, the top 8 bi-grams are all non-spam emails, probably due to repetitive work-related words such as ”submission” and ”sender”. The lower words are spam bi-grams. They are more diverse, likely due to the varied content and tactics used in phishing or promotional messages.

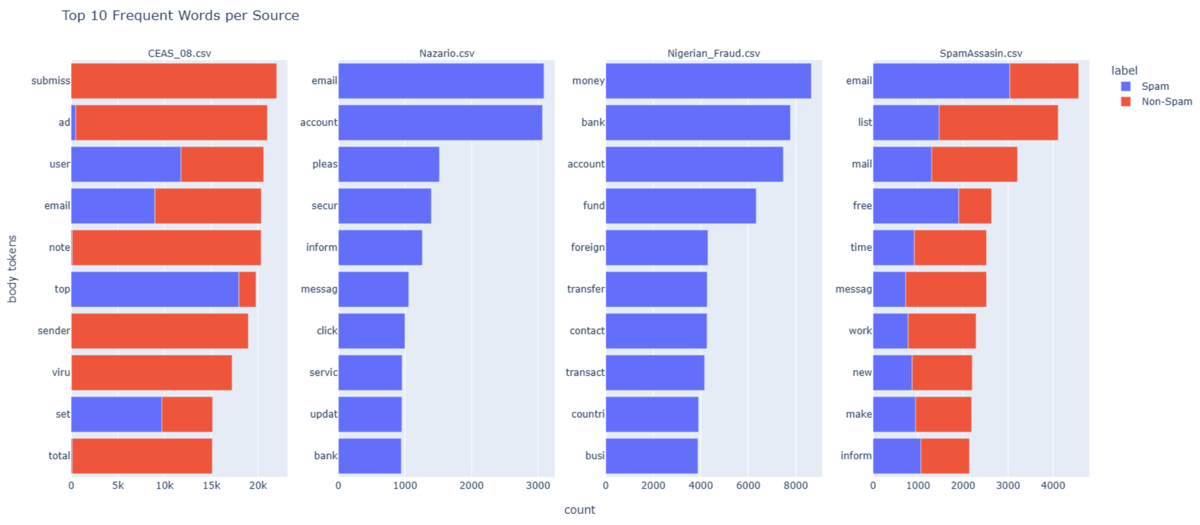

Figure 5: This chart displays the top 10 most frequent email body tokens for each dataset, broken down by source and labelled as spam or non-spam. The CEAS_08 source has the largest frequency for its top 10 words, and most of the terms are non-spam. This makes sense as the words with the highest frequency tend to be non-spam.

In the website Figure 5, the most common words in the CEAS_08 dataset primarily come from non-spam emails, while the top words from the Nazario and Nigerian Fraud datasets are exclusively from spam. In contrast, the SpamAssassin dataset shows a more even split between spam and non-spam top words. Although CEAS_08 contains mostly spam emails, its most frequent token appears over 20,000 times, suggesting that CEAS_08 may heavily influence the word distribution in the overall combined dataset. CEAS_08 contains a wider range of spam emails rather than specific emails, such as Nigerian Fraud, so this can help us generate a more general model.

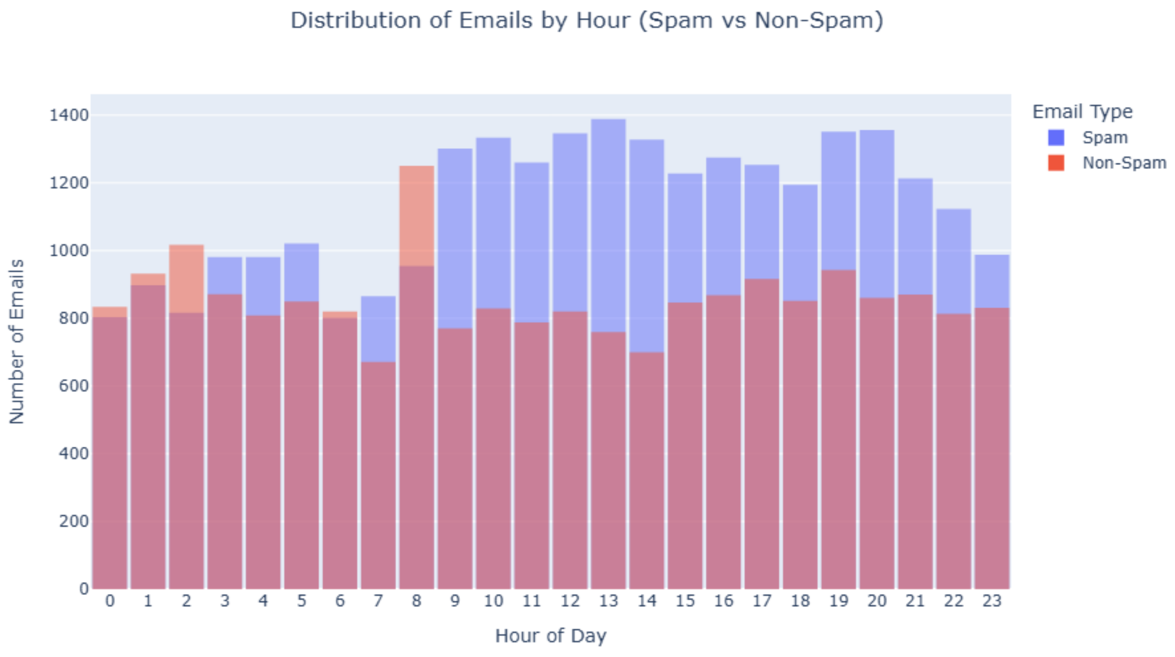

Figure 6: This histogram compares the distribution of spam and non-spam emails by hour of the day. It appears spam emails are more frequently sent during the day from 9 am to 10 pm, and less so at night. On the other hand, non-spam emails are sent consistently throughout the day, with a peak at 8 am.

In Figure 6 on the website, we can see spam emails tend to be sent during the day from 9 am to 10 am; whereas, non-spam emails are consistently sent throughout the day, especially at 8 am. This makes sense, as many work emails are scheduled to be sent at the beginning of the work day at 8. This means the hour timestamp can be a significant variable for our modelling.

Metric	**Spam Emails**	**Non-Spam Emails**
Unique Words	150,547	131,351
Total Words	2,054,541	3,646,719
Average Words per Email	75.10	177.72
Lexical Diversity	0.0733	0.0360

Table 3: The following are summary statistics comparing spam and non-spam emails. Spam emails tend to contain fewer words but have more unique words and a larger lexicon. This suggests that spam emails may use more uncommon words than non-spam emails.

Finally, as shown in the vocabulary summary Table 3, spam emails tend to contain fewer total words but a greater number of unique words, indicating a larger and more varied lexicon. This suggests that spam emails often use words that are less common than non-spam emails. This observation is consistent with the results in Figure 4, where words in non-spam emails have the highest frequency, and in Figure 3, where words in spam emails display higher TF-IDF scores for their more distinctive vocabulary.

In our bag-of-words representation, the initial vocabulary size exceeded 250,000 words, which exceeded the available memory capacity for storing the full count vector. Therefore, I reduced the vocabulary to the 6,000 most frequent words in the dataset, with the standardized hour variable for use in prediction.

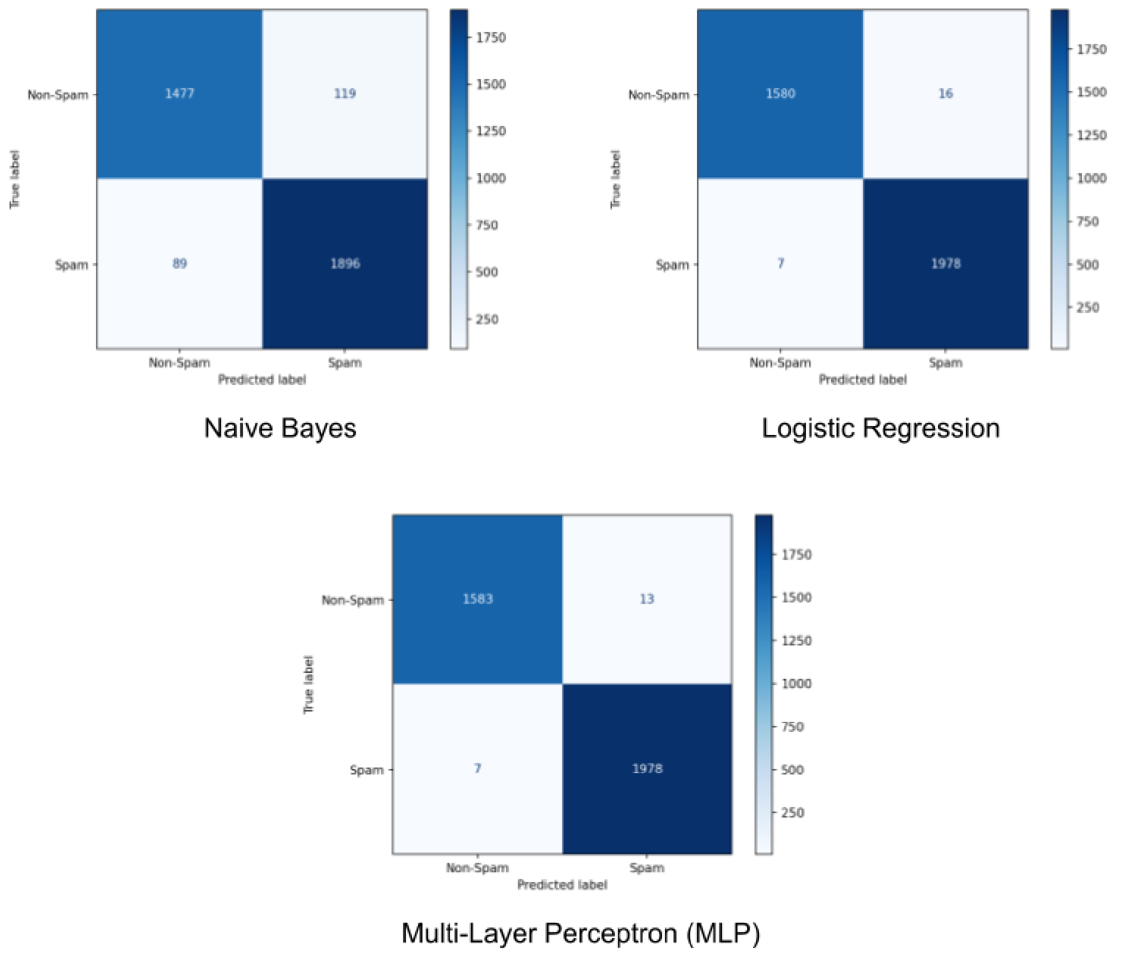

Figure 7: Final confusion matrices for our three best models for Naive Bayes, logistic regression, and MLP on our test dataset. It appears MLP has the lowest number of false positives, which is our most important metric, as I don't want important non-spam emails labelled as spam.

Model	Hyperparameters	Best Value	CV Accuracy	Test Accuracy	FPR	FNR
Naive Bayes	0.0, 0.1, 0.5, 1.0, 2.0	0.1	94.93%	94.19%	7.46%	4.48%
Logistic Regression	0.01, 0.1, 1, 10	10	99.23%	99.36%	1%	0.35%
MLP Classifier	(50, 50), (100), (128, 64, 32, 16), (200, 100, 50)	(200, 100, 50)	99.42%	99.44%	0.81%	0.35%

Table 4: Summary of best hyperparameters, cross-validation accuracy, test accuracy, and false positive (FPR) and false negative rate (FNR) on test data for each model. Naive Bayes assumes a Bernoulli distribution, where the hyperparameter α controls the amount of smoothing applied to the probability. For logistic regression, the hyperparameter is the regularization strength $\frac{1}{C}$. For the MLP classifier, each value i in the tuple represents the number of neurons in hidden layer i.

Based on the final results shown in Table 4, the multi-layer perceptron (MLP) model with 200 neurons in the first layer, 100 in the second, and 50 in the third achieved the highest test accuracy of 99.44% and the lowest false positive rate of 0.81%, making it the model selected for prediction examples.

As illustrated in Figure 7, the MLP model also produced the fewest false positives among all models. However, due to the complexity and lack of interpretability of neural networks, identifying specific words associated with spam or non-spam predictions is challenging. Therefore, the second-best model, logistic regression, is used for interpretability purposes. Logistic regression allows for straightforward interpretation of model coefficients, helping to identify words most strongly associated with spam and non-spam emails. The best logistic regression model was found with a regularization strength of $\frac{1}{10}$, achieving a test accuracy of approximately 99.36% and a false positive rate of 0.1%. Notably, before incorporating the hour the email was sent as a feature, the MLP and logistic regression models achieved test accuracies of 98.82% and 98.63%, respectively, highlighting that the sending hour is a meaningful variable in improving model performance.

Top 10 Spam Words	**Top 10 Non-Spam Words**
room	poster
frederick	apc
confess	qualif
poor	arg
bless	aff
benin	blue
initi	internet
löwi	ici
marin	muslim
sound	snip

Table 5: The table shows the top 10 words most strongly associated with predicting spam and non-spam emails. Many of these words are exclusive to either spam or non-spam emails, often representing obscure or infrequent terms.

Using the trained logistic regression model, I can interpret the importance of specific words by examining their coefficients. Words with the highest positive coefficients increase the likelihood of an email being classified as spam, while those with the most negative coefficients increase the likelihood of being classified as non-spam, as shown in Table 5. Because the text was normalized through stemming and lemmatization, some words appear in unfamiliar forms, making interpretation more challenging. As a result, many of these terms offer limited practical insight into identifying spam based on recognizable keywords. The majority of the top words seem highly specific to particular companies, products, or topics in the emails, rather than broadly indicative of spam. This suggests that uncommon or highly specialized words tend to dominate the largest coefficient values, as they are more likely to appear primarily in either spam or non-spam emails.

Field	**Example 1 (Spam)**
Email Text	Apply to our part-time, flexible, contract and be paid bi-weekly $2000.
Hour Sent	12
Sender	hiring@quickcashnow.com
Receiver	john.doe@hotmail.com
Predicted Label	Spam

Field	**Example 2 (Non-Spam)**
Email Text	Hi team, Just a reminder that we have our weekly sync tomorrow at 10 AM in the usual meeting room. Please be on time and bring any updates you'd like to share. Thanks, Alice.
Hour Sent	15
Sender	alice.smith@company.com
Receiver	team@company.com
Predicted Label	Not Spam

Table 6: Example predictions in showing email text, hour of the day email is sent, sender, receiver, and the prediction. It appears our model accurately predicts spam and not spam for these simple examples, which aligns with the 99.44% accuracy of the best neural network model.

Based on Table 6, our model demonstrates accurate prediction of spam and non-spam emails.

4 Conclusion

Based on our current datasets, the multi-layer perceptron (MLP) model was identified as the best-performing approach, achieving an overall accuracy of 99.4% for predicting spam and non-spam emails using the email body content, sender, receiver, and the hour of the day as predictors. Other models, such as logistic regression, also performed well, suggesting that the email data is largely linearly separable. Additionally, the consistently high accuracies indicate that the language and terminology used in spam and non-spam emails are distinct enough to allow a bag-of-words model to be highly effective for spam detection. While spam emails are often easily recognizable to individuals familiar with common spam patterns, they can still pose a significant threat to elderly users or those less experienced with email communication. Therefore, models like these remain valuable tools for protecting vulnerable users from fraudulent emails.

Despite the high accuracy, several limitations should be considered. First, the emails in the dataset were all sent around 2001, making the content somewhat outdated for predicting patterns in more modern email communication. Additionally, some datasets, such as the Nigerian Fraud dataset, represent a very narrow subset of spam emails, limiting their ability to generalize across all types of spam. Second, as shown in Table 2, many of the sender and receiver email addresses appear to be bot-generated, likely created for automated spam detection research. As a result, these synthetic addresses may not accurately reflect realistic email communication, which could affect the generalizability of our model when applied to more authentic datasets. Finally, our model used a bag-of-words vectorization approach on the email content. While effective for identifying word frequencies, the bag-of-words representation ignores grammar, word order, and contextual meaning. Furthermore, removing stop-words and standardizing words to their present simple tense strips away nuances such as sentence structure and verb tense, which could be informative. More sophisticated models that capture semantic meaning, such as those based on embeddings or contextual language models, may offer improved performance than my model.

Despite the limitation, our models are shown to perform well in distinguishing between spam and non-spam emails, suggesting their potential for use in real-world spam detection tools and applications. The model can be expanded to also consider spam messages and calls by utilizing similar tools, such a word frequency.

References

[1] Al-Subaiey, A., Al-Thani, M., Alam, N. A., Antora, K. F., Khandakar, A., & Zaman, S. A. U. (2024, May 19). *Novel Interpretable and Robust Web-based AI Platform for Phishing Email Detection.* ArXiv.org. Retrieved from `https://arxiv.org/abs/2405.11619`.

[2] Dredze, M., Gevaryahu, R., & Elias-Bachrach, A. (2007). *Learning Fast Classifiers for Image Spam.* Conference on Email and Anti-Spam (CEAS). Retrieved from `https://www.cs.jhu.edu/~mdredze/datasets/image_spam/`.

[3] Champa, A. I., Rabbi, M. F., & Zibran, M. F. (2024). *Curated Datasets and Feature Analysis for Phishing Email Detection with Machine Learning.* ResearchGate. Retrieved from `https://www.researchgate.net/publication/382212910_Curated_Datasets_and_Feature_Analysis_for_Phishing_Email_Detection_with_Machine_Learning`.

[4] Tatman, R. (n.d.). *Fraudulent E-mail Corpus.* Kaggle. Retrieved from `https://www.kaggle.com/datasets/rtatman/fraudulent-email-corpus`.

[5] The Apache Software Foundation. (n.d.). *Apache SpamAssassin Public Corpus.* Retrieved from `https://spamassassin.apache.org/`.