• ASSIGNMENT COVERSHEET

UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME	NAME OF STUDENT(s) (PRINT CLEARLY)	STUDENT ID(s)
41180 Data Analytics in Cybersecurity	*Suen Chun Hin Carven* *Vedant Surjan*	13266578 24986275

STUDENT EMAIL	STUDENT CONTACT NUMBER
Vedant.surjan@student.uts.edu.au	

NAME OF TUTOR	TUTORIAL GROUP	DUE DATE
Mingjian Tang	cmp1	31/03/2025

ASSESSMENT ITEM NUMBER & TITLE

☐ I acknowledge that if AI or another nonrecoverable source was used to generate materials for background research and self-study in producing this assignment, I have checked and verified the accuracy and integrity of the information used.

☐ I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.

☐ I confirm that I have read, understood and followed the advice in the Subject Outline about assessment requirements.

☐ I understand that if this assignment is submitted after the due date it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I have rewritten any material provided by AI or other nonrecoverable sources and where appropriate acknowledged their contribution. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

No content generated by AI technologies or other sources has been presented as my own work and I have rewritten any text provided by AI or other sources in my own words.

Statement of collaboration:

Signature of student(s): _____Chun and Vedant_____ Date:___31/03/2025_____

ASSIGNMENT RECEIPT

To be completed by the student if a receipt is required

SUBJECT NUMBER & NAME	NAME OF TUTOR

SIGNATURE OF TUTOR	RECEIVED DATE

STYLE GUIDE for ASSIGNMENT SUBMISSION

Before submitting an assignment, you should refer to the policies and guidelines set out in the following:

- FEIT Student Guide
- UTS Library - referencing
- HELPS - English and academic literacy support
- UTS GSU - coursework assessment policy and procedures

Unless your Subject Coordinator has indicated otherwise in the Subject Outline, you must follow the instructions below for submission of assignments in the Faculty of Engineering and Information Technology.

Writing style

It is usually best to write your initial draft in the default settings of your software without formatting. Use the following guides in your writing.

Purpose and audience: use the correct genre and language style expected for the particular task.

Language: use 'plain English' for all technical writing. More information about this language style can be found at www.plainenglish.co.uk/free-guides.html.

Use spelling and grammar software tools to check your writing. Edit your document.

Standards: always use:

- Australian spelling standards (Macquarie Dictionary)
- SI (International System of Units) units of measurement
- ISO (International Organisation for Standardisation) for writing dates and times for international documents. For example **yyyy-mm-dd** or **hh-mm-ss**. However, for most applications it is more helpful to present the date in full as **26 August 2016**.

Graphics and tables should:

- be numbered
- have an appropriate heading and/or caption
- be fully labelled
- be correctly referenced.

Presentation

Unless otherwise instructed, all assignment submissions should be **word processed** using spell-check and grammar-check software. Work should be well **edited** before submission. Use the following default settings:

Page setup: set margins at no less than 20mm all around.

Paper: print on A4 bond, double-spaced and preferably double-sided, left justified.

Font: use the software default style to provide consistency. The recommended style includes:

- 10-12 pt font
- consistent formatting with a limited number of fonts
- lines no more than 60 characters (use wider margins or columns if you need to make lines shorter)

Header should include:

- your name and student number
- the title of the paper or task.

Footer should include the page number and current date.

Cover sheet and statement of originality: all work submitted for assessment must be the original work of the student(s) submitting the work. A standard faculty cover sheet (see over) must be attached to the front of the submission. Any collaboration between the submitting student and others must be declared on the cover sheet.

Referencing

All sources of information used in the preparation of your submission must be acknowledged using the APA system of referencing. This includes all print, video, electronic sources.

Phrases, sentences or paragraphs taken verbatim from a source must be in quotation marks and the source(s) cited using both **in-text** referencing and a **reference list**.

Plagiarism is the failure to acknowledge sources of information. You should be fully aware of the meaning of plagiarism and its consequences both to your marks, position at the university and criminal liability. The plagiarism in your assignment submissions can be assessed both in hard copy and in soft copy through software such as Turnitin.

The UTS Library and UTS HELPS (web links above) provide extensive information for students on referencing correctly to support you in avoiding plagiarism.

Contributions Table

Name	Contribution
Vedant Surjan (24986275)	Introduction, Reasonable Solutions, Metrics, Results, Reflection, Conclusion and References
Chun Hin Carven Suen (13266578)	Introduction, Reasonable Solutions, Metrics, Results, Reflection, Conclusion and References

1. Introduction

In this modern age, spam detection has emerged as a critical challenge to maintain the integrity and usability of electronic communication systems. Email communication is a primary channel of communications, and the number unwanted emails (known as spam) have also greatly risen. Spam emails often contain unsolicited advertisements or malicious links which can overload a user's system and expose them to security risks. Effective spam detection systems are essential for filtering out unwanted messages to ensure users receive relevant information.

This project explores the application of machine learning techniques to develop an efficient spam detection mode. We focus on two distinct classification algorithms: Multinomial Naive Bayes (MNB) and AdaBoost (ABC). The MNB classifier is known for its effectiveness in text classification and ABC, which combines weak classifiers to form a strong classifier which enhances its effectiveness.

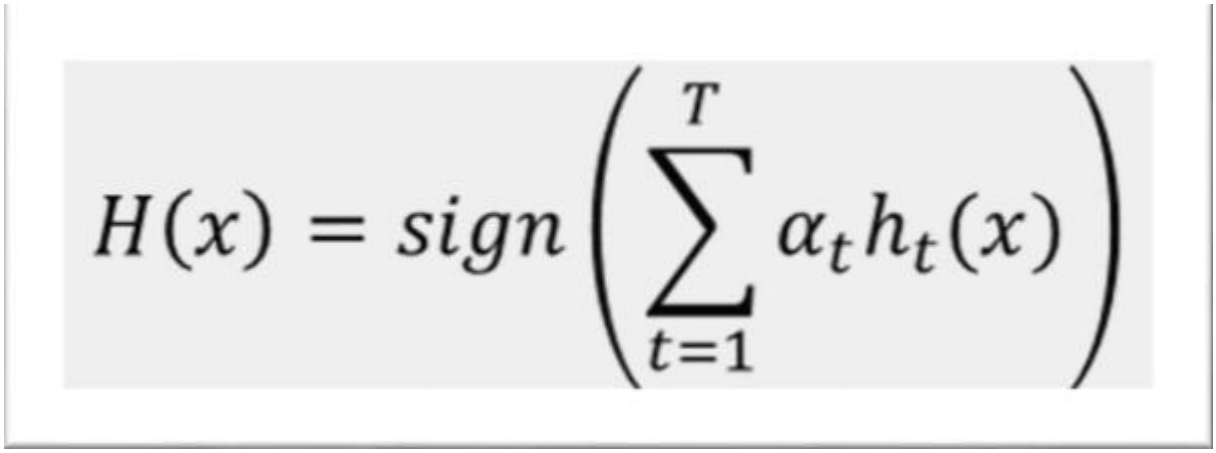
The report will dive into the comparative analysis of the spam filtering systems through various performance metrics to provide a comprehensive understanding of how each model performs. By enhancing spam filtering mechanisms through machine learning, this research and project offers practical insights into implementing effective spam detection systems in real-world scenarios.

2. Reasonable Solution on the Project

2.1 Solutions

2.1.1 Adaptive Boosting Classifier - Chun Hin Carven Suen

A method that could be potentially utilised to identify and filter for spam emails from legitimate ham messages is the adaptive boosting classifier (Adaboost), unlike traditional classifiers that rely on individual models to make predications, Adaboost combines a number of weaker models to form one strong classifier that refines performance through iteration(Freund & Schapire, 1997). In the case of spam filtering, this model learns from email data and iteratively adjusts itself to better classify difficult or misclassified content. Each weak learner in the model contributes to a stronger final decision with its weight being based upon its performance, allowing for the model to improve over time. This approach is effective because subtle variations in message structure and language can often lead to misclassification of data, with the model's ability to focus on these errors it enables it to construct a more accurate decision boundary between spam and ham messages (Zhang & Zhou, 2009).



$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Figure 1 : Adaboost final prediction formula

With the Adaboost model, it produces its final prediction by merging the outputs of its component models called weak learner, with each of these smaller models contributing to the final decision based on their performance. This is illustrated in figure 1, showing the final prediction formula. On the left side of the equation H(x) represents the final prediction made by the ensemble model. The right side of the formula consists of a weighted sum of all the individual weak learners $h_t(x)$ multiplied by the weight α_t that reflects the accuracy of that learner. The sign function is then applied to transform the result into a binary classification output such as spam or ham. The key takes of this equation being that it ensures that the more accurate learners have greater influence in the final prediction of the model.

For this method of spam filtering, it was necessary to perform preprocessing of the data before the classifier could be applied. A preprocessing function was used on the dataset that cleaned and formatted the messages, with the use of python libraries such as string and nltk to filter out any stopwords and punctuation that can introduce noise and reduce classification accuracy. Afterwards the data then undergoes the process of tokenization where each message is split into tokens. Once tokenized the text is converted into numeral representations using the TF-IDF vectorization which assigns weights to each word based on its importance across the dataset. This is reflected in figure 2. Following this, additional features were added to support the classification process, as shown in figure 3, the dataset includes spam and characters columns, the spam column reflects a binary value that indicates if it is either spam or ham, while the characters conveys the size of the message by counting the total number of characters in it.

```
nltk.download('stopwords')
from nltk.corpus import stopwords

df = pd.read_csv('spam (1).csv', encoding='latin-1')[['v1', 'v2']]
df.columns = ['label', 'message']

df['label'] = df['label'].map({'ham': 0, 'spam': 1})
df['spam'] = df['label']
df['labels'] = df['spam'].map({0: 'ham', 1: 'spam'})
df['characters'] = df['message'].apply(len)

def preprocess(text):
    text = text.lower()
    text = ''.join([ch for ch in text if ch not in string.punctuation])
    tokens = text.split()
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    return ' '.join(tokens)

df['cleaned_message'] = df['message'].apply(preprocess)

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['cleaned_message'])
y = df['label']
```

Figure 2 : Preprocessing the data.

labels	message	spam	characters
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	0	111
ham	Ok lar... Joking wif u oni...	0	29
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	1	155
ham	U dun say so early hor... U c already then say...	0	49
ham	Nah I don't think he goes to usf, he lives around here though	0	61
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv	1	148
ham	Even my brother is not like to speak with me. They treat me like aids patent.	0	77
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune	0	160
spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.	1	158
spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030	1	154
ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.	0	109

Figure 3 : the dataset with assigned values

Following the preprocessing and vectorization stage, the dataset is divided into two parts for training and testing. Specifically, 80% of the data is allocated for training, while the remaining 20% is reserved for testing. This balance was selected to provide the model with sufficient data to learn from, while still retaining a representative subset for accurately evaluating its performance on unseen examples. During the testing phase the Adaboost model is evaluated on the testing data to assess its ability to correctly identify between spam and ham messages. Three different configurations were tested to observe the impact of hyperparameters on the model's performance. The first configuration used fewer estimators and a high learning rate, the second was a more balanced approach, and the third involved a higher number of estimators with a lower learning rate and deeper trees. These configurations were chosen to test how boosting strength and model complexity influence the classifier’s accuracy. The implementation of this

comparison process is shown in Figure 4, where each configuration is trained, tested, and evaluated using selected performance metrics.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)


configs = {
    'A': {'n_estimators': 50, 'learning_rate': 1.0, 'max_depth': 1},
    'B': {'n_estimators': 100, 'learning_rate': 0.5, 'max_depth': 2},
    'C': {'n_estimators': 200, 'learning_rate': 0.1, 'max_depth': 3}
}

results = {}


for name, params in configs.items():
    print(f"\n=== {name} ===")

    base = DecisionTreeClassifier(max_depth=params['max_depth'])
    model = AdaBoostClassifier(estimator=base,
                               n_estimators=params['n_estimators'],
                               learning_rate=params['learning_rate'],
                               random_state=42)

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:, 1]

    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
```

Figure 4: Python code that was used in the training of the model.

2.1.2 Multinomial Naive Bayes – Vedant Surjan

Note: ChatGPT was used to help with coding

Multinomial Naive Bayes (MNB) is a popular Natural Language Processing (NLP) algorithm which is ideal for tasks such as spam filtering and text classification. It comes from the basic Bayes algorithm which handles data that can be represented as the frequency of discrete data like word counts, which makes it ideal to work with text data (Sriram, 2024). The MNB model is effective at handling large vocabularies in documents and emails, allowing it to efficiently categorise text even with large text datasets (GeeksforGeeks, 2025).

The model assumes that all features are independent within the class labels, hence the naive terminology. For text classification, it implies that the occurrence of one word is assumed to be unrelated to the occurrence of any other (Sriram, 2024). On the other hand, multinomial refers to the frequency of a word or how often it appears in a specific category (e.g. the word 'free' appears more frequently in spam messages).

$$P(X) = \frac{n!}{n_1!\, n_2! \ldots n_m!} p_1^{n_1} p_2^{n_2} \ldots p_m^{n_m}$$

- n is the total number of trials or experiments conducted
- n_i is the number of times outcome i occurs in the total number of trials
- p_i is the probability of outcome i happening in a single trial (GeeksforGeeks, 2025)

The formula is the probability mass function of a multinomial distribution which is a generalisation of the binomial distribution. The multinomial distribution deals with scenarios where there are multiple possible outcomes for each trial. To visualise this, we can consider an experiment of rolling a dice multiple times and each face of the dice represents a different outcome. The multinomial distribution models a scenario like calculating the probability of rolling a '1' three times or a '4' five times or rolling a '6' twice in ten rolls.

Text Preprocessing

Before using the MNB model to filter between spam and non-spam messages, it is crucial to preprocess the data, converting the raw text into a format that the model can understand and process effectively.

- Lowercasing text helps reduce the complexity of the dataset by treating all words the same e.g. "Email" and "email".
- Removing common words like "and", "the", "is" that don't contribute much to the predictive power of the model due to their high frequency. This is done using CountVectrorizer with the stop_words='english' parameter.

- Vectorization helps transform text into a numeric format that the model can understand using the CountVectroizer and converting to token counts.

```
# Text Preprocessing
spam_df['text'] = spam_df['text'].str.lower()  # Convert text to lower case
vectorizer = CountVectorizer(stop_words='english')  # Initialize the CountVectorizer
X = vectorizer.fit_transform(spam_df['text'])  # Vectorize text data
y = np.where(spam_df['label'] == 'spam', 1, 0)  # Convert labels to binary format
```

Figure 5: Python code for text preprocessing

Splitting Dataset

Splitting the dataset allows to evaluate performance of model more accurately.

- 80% of the data is used as the training set.
- Using the model's prediction against the known output, 20% of the data was allocated as the test set.

```
# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Figure 6: Code was splitting the dataset

Message Length Visualisation

Provides insightful info into the distribution of the lengths of messages in spam and non-spam messages.

- Message length is calculated using .apply(len) method on text column
- A histogram is plotted using sns.histplot with colour coded labels to distinguish spam or ham messages

```
# Message Length Visualization
spam_df['message_length'] = spam_df['text'].apply(len)
sns.histplot(data=spam_df, x='message_length', hue='label', element='step', stat='density', common_norm=False, ax=axes[2, 0])
axes[2, 0].set_title('Message Length Distribution')
axes[2, 1].axis('off')  # Turn off the last subplot (not used)
```

Figure 7: Python code for message length visualization

Testing Model on 10 Random Samples

This process involved evaluating the model's accuracy on a small, randomly selected subset of test data.

- Random.sample function to randomly select 10 messages from dataset
- Used the trained model to predict labels of the selected messages
- Comparing the predicted labels against actual labels to calculate accuracy and performance of model.

```
# Testing on 10 random samples from the test set
sample_indices = random.sample(range(X_test.shape[0]), 10)
X_sample = X_test[sample_indices]
y_sample = y_test[sample_indices]
y_pred_sample = mnb.predict(X_sample)
sample_accuracy = accuracy_score(y_sample, y_pred_sample)
print("Accuracy on 10 random samples:", sample_accuracy)
```

Figure 8: Testing the model on 10 random samples from the dataset

Finding Most Common Words and Bi-grams

Identifying most common words and bi-grams reveals patterns in the spam and non-spam messages.

- CountVectorizer counts the occurrence of each word in the dataset after filtering out the stop words.
- CountVectorizer with ngram_range=(2,2) to count consecutive pairs of words in text which provides additional context that can potentially be missed when looking at just a single word.

It is important to note that different values of alpha were tested to see the parameter which had the best accuracy. These processes help prepare the data to better allow the model to get trained and tested correctly. It also helps the model to evaluate and understand the extrapolated data to gain insights into the performance and the dataset's characteristics and features.

Once the processes are completed, the code outputs useful graphs and information about accuracy and effectiveness of the MNB model but also showcases key features of the dataset.

The figure below shows the top 15 most common words found in spam and ham messages in the entire dataset. As seen, the most common word in spam messages is "free" and the most common word in ham (non-spam) is "gt". In the most common words for spam messages, a common trend can be noticed where the spam emails are using words and numbers to grasp the receiver's attention (words like "free", "cash", "claim" etc). Whereas in the ham messages, it primarily comprises of informal words used in casual conversations (words like "ok", "love", "good" etc). The key difference is that spam messages contain words to create a sense of urgency to engage the receiver and conversely the ham messages contain simple and informal words that don't create urgency and are words used in casual conversations.

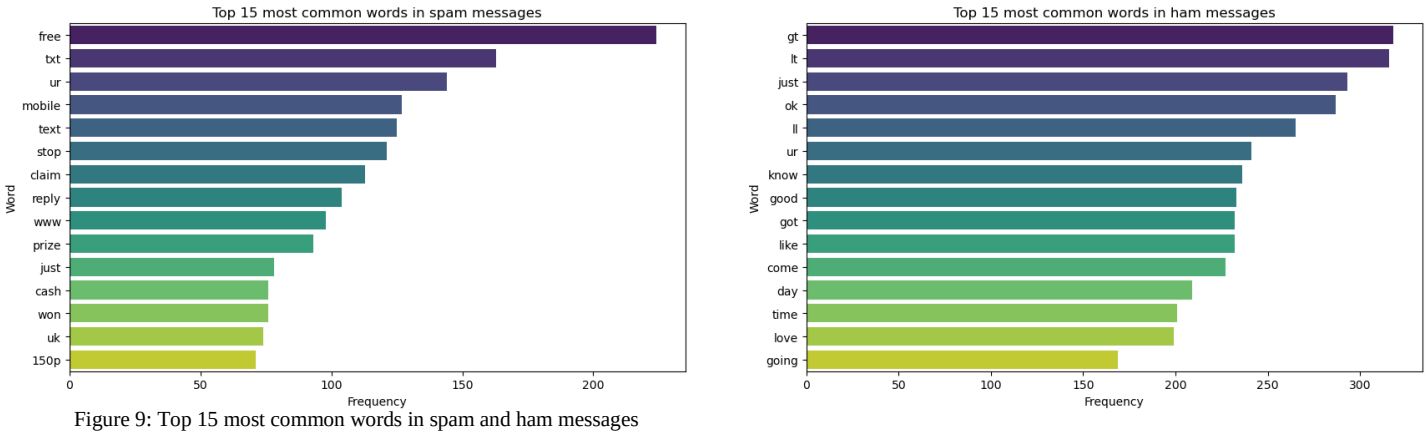


Figure 9: Top 15 most common words in spam and ham messages

The figure below shows the top 15 most common bi-grams in both spam and ham messages. Bi-grams are pairs of consecutive words that help provide context which is often lost in single-word analysis. In the spam messages, the most common bi-grams include phrases like “1000 cash” and “prize guaranteed” which instantly grabs the receiver’s attention. The most common phrases in the ham messages include “let know” and “It gt” which are more informal words used in casual conversation. A common trend with the spam messages is that captures the readers attention while the ham messages are informal words and don’t create urgency.

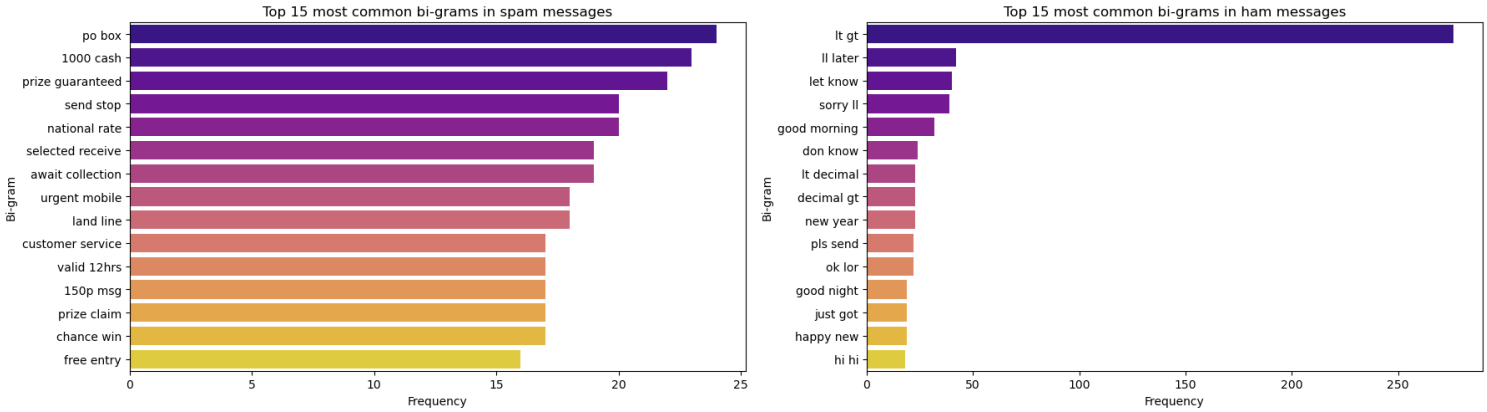


Figure 10: Top 15 most common bi-grams in spam and ham messages

In addition to these graphs, below is a figure showing the length of messages found in the spam or ham messages. The message length distribution shows the frequency of messages at different lengths to help identify patterns. The graph shows that in general, spam messages are longer than ham messages, which follows the trend from previous graphs. Ham messages contain informal words and phrases, so contents are much shorter compared to the spam messages where it trying to grasp the receiver’s attention and making the message seem legitimate.

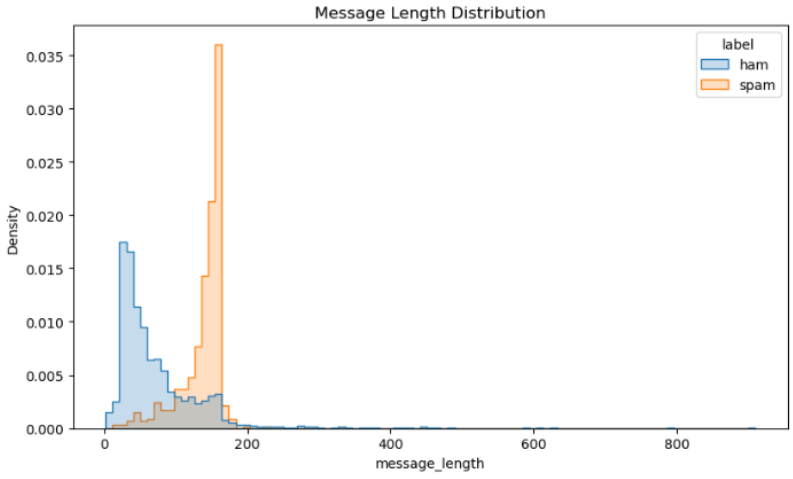


Figure 11: Message Length Distribution for spam and ham messages

3. Metrics and Results

3.1 Adaptive Boosting Classifier – Chun Hin Carven Suen

The AdaBoost model was tested with three configurations, each adjusting the number of estimators, learning rate, and tree depth. In Figure 12, the confusion matrices for all three setups are shown side by side, illustrating how often each configuration correctly identified spam and ham messages. Among them, configuration c which used 200 estimators, a 0.1 learning rate, and a tree depth of 3 offered the best balance, with about 96% accuracy, 92% precision, and 93% recall. This means it was better at catching spam without generating too many false positives. Figure 13 illustrates this by visualizing the metrics in a bar chart, underscoring Config C's more balanced performance relative to the other two. Finally, Figure 14 summarizes these results in a single table, confirming that Config C consistently outperformed the other configurations, highlighting AdaBoost's effectiveness in this spam detection task.

	Config A		Config B		Config C	
	Predicted Ham	Predicted Spam	Predicted Ham	Predicted Spam	Predicted Ham	Predicted Spam
Actual Ham	957	8	959	6	962	3
Actual Spam	58	92	52	98	50	100

Figure 12: Confusion matrix for all 3 Adaboost configurations.

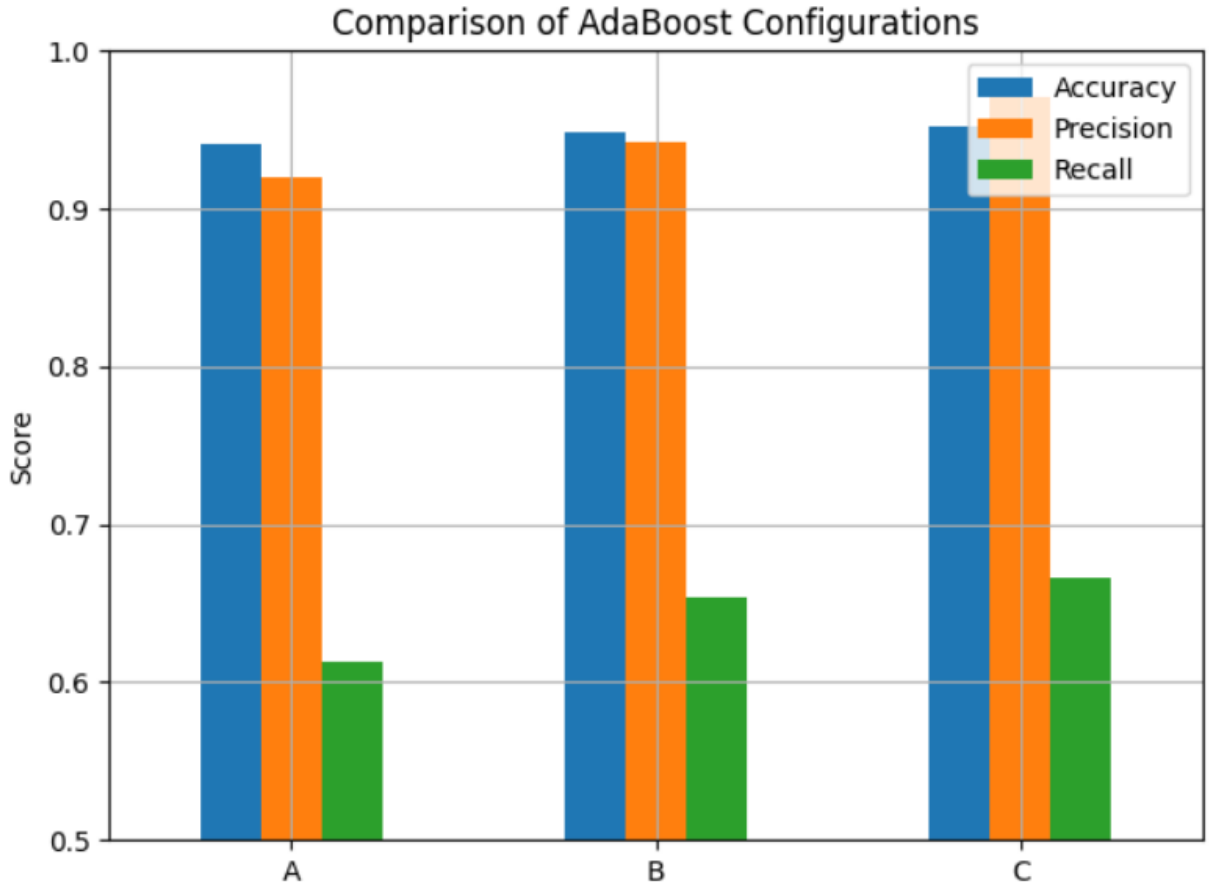


Figure 13: graph that compares each configuations metrics

Results for all Adaboost Configurations

	Accuracy	Precision	Recall
A	0.940807	0.920000	0.613333
B	0.947982	0.942308	0.653333
C	0.952466	0.970874	0.666667

Figure 14: Results from the Adaboost model.

3.2 Multinomial Naive Bayes – Vedant Surjan

The Multinomial Naive Bayes was tested with three different parameters to determine which parameter performed the best in terms of accuracy. The MNB model preprocesses and vectorize the model using CountVectorizer which simply counts the number of times each word appears in the document. The counts are used as features for training the classifier as the assumption is that the frequency of words influences the decision about whether a message is spam or ham. Figures 15, 16 and 17 show the alpha values, the confusion matrix, the classification report and accuracy of the model at different alpha values. The parameters are based on the alpha values tested which are 0.01, 0.1 and 1.0. Alpha is a smoothing parameter applied to the MNB model to handle cases where the word may not have appeared in the training data but appears in the testing data. A lower alpha value means less smoothing. The highest accuracy is 98.4% at $alpha = 0.01, 0.1$and the lowest accuracy is 97.8% at $alpha = 1.0$.

Overall, alpha values at 0.01 and 0.1 provide strong filtering performance however 0.1 offers a better balance by reducing the false negatives which is better for spam detection as missing spam could be more dangerous. At 0.1, the model correctly predicted non-spam messages 938 times, it incorrectly classified 11 non-spam messages as spam which although acceptable, can lead to inconveniences. The model missed 7 spam messages and classified them as non-spam which can lead to security risks, and lastly it accurately identified 159 spam messages. With the results being high across the board for precision, recall, f1-score and support values, $alpha = 0.1$ provides impressive results and high effectiveness in filtering spam and non-spam correctly.

Alpha: 0.01
Accuracy: 0.9838565022421525
Confusion Matrix:

	Predicted Ham	Predicted Spam
Actual Ham	940	9
Actual Spam	9	157

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	949
1	0.95	0.95	0.95	166
accuracy			0.98	1115
macro avg	0.97	0.97	0.97	1115
weighted avg	0.98	0.98	0.98	1115

Figure 15: Alpha = 0.01

Alpha: 0.1
Accuracy: 0.9838565022421525
Confusion Matrix:

	Predicted Ham	Predicted Spam
Actual Ham	938	11
Actual Spam	7	159

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	949
1	0.94	0.96	0.95	166
accuracy			0.98	1115
macro avg	0.96	0.97	0.97	1115
weighted avg	0.98	0.98	0.98	1115

Figure 16: Alpha = 0.1

```
Alpha: 1.0
Accuracy: 0.97847533632287
Confusion Matrix:
             Predicted Ham  Predicted Spam
Actual Ham             933              16
Actual Spam              8             158
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       949
           1       0.91      0.95      0.93       166

    accuracy                           0.98      1115
   macro avg       0.95      0.97      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

Figure 17: Alpha = 1.0

3.3 Discussion

When comparing the results between adaboost and Multinomial Naive Bayes, though they both achieved high performances in spam detection there were notable differences in their outcomes. The best Adaboost configuration achieved around 96% accuracy while the MNB model had scored a higher accuracy of 98.4%. This suggests that while Adaboosts has benefits of iterative learning and handlings of edge cases, the simplier approach of MNB proved to be extremely efficient. Ultimately, the choice between these models depends on whether one values the simplicity and speed of the MNB model or the refined, iterative improvement provided by AdaBoost model.

4. Reflection

Vedant

The completion of this project came with its challenges due to issues with understanding the scope of the assignment and the distribution of workload. However, the lectures, labs and tutorials made it easier to understand the scope of the assignment and successfully complete the project. The assignment gave me a deeper understanding of Artificial Intelligence and machine learning concepts and how important data is to train AI. Before the project, I had a basic understanding of AI and machine learning but experimenting with the codes provided in the labs along with the codes for this project, allowed me to comprehend the procedures of training these algorithms for various purposes and preprocessing data. Diving into the machine learning algorithms and its functionality not only helped improve my knowledge on the topic but also helped improve my coding skills. On the other hand, my knowledge and experience with Python had been very limited which made it difficult to understand the contents of the labs and project. I utilised the tutorials to understand code and used ChatGPT to understand certain aspects of the project like the process

of preprocessing the data and training the algorithms to complete the tasks more accurately. Although I found it challenging, it was also interesting to learn the various filtering algorithms and how they work. The project results were satisfactory but to further improve upon my work, I can practice my Python coding skills to better understand the various functions and tools along with further researching into how the different parameters of MNB can affect the results and ways to improve speed and accuracy.

Chun Hin Carven Suen

Intially, this project felt very challenging due my schedule being very intense and not leaving as much time as I would like to collaborate with my team members. Though with the completion of this assignment i was able to gain valuable insights into the inner workings of machine learning and obtain a deeper understanding of the training required for these models to work. I learned how seemingly small changes in the preprocessing process, vectorization and parameters can influence the outcome of the model and its results. This experience has highlighted to me the importance of clear communication and team interaction. With a disagreement between members hindering our projects process and leading to delays and issues further on.

Additionally, exploring the ensemble methods such as Adaboost proved interesting as I discovered how the algorithm focuses on misclassified instances to refine its predictions. Observing how the parameters could significantly shift the result metrics was also very interesting. I found the tutorials and their explanation on the models and python to be quite useful, though at times it was very difficult for me to understand the resources provided helped me significantly. This hands-on experience not only deepened my technical skills but also strengthened my ability to troubleshoot complex issues under pressure. I now feel better equipped to manage similar challenges in future projects, appreciating machine learning and its processes.

5. Conclusion

Ultimately, this project offers the investigation and implementation of two email spam filtering soulutions of the Adaboost model and the Multinomial naive bayes model. Both have provieded the oppitunity to gather deeper insights into the effectiveness and accuracy of these machine learning models, and their ability to distinguish between spam and legitmate emails. Although both methods were hold their own merits in this project, Multinomial naive bayes was shown to perform better in the metrics and results out of all configurations and models.

References

1. GeeksforGeeks (2025) *Multinomial naive Bayes*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/multinomial-naive-bayes/ (Accessed: 31 March 2025).
2. Ibm (2024) *What are naïve Bayes classifiers?*, *IBM*. Available at: https://www.ibm.com/think/topics/naive-bayes (Accessed: 31 March 2025).
3. Sriram (2024) *Multinomial naive Bayes explained: Function, Advantages & disadvantages, applications*, *upGrad blog*. Available at: https://www.upgrad.com/blog/multinomial-naive-bayes-explained/ (Accessed: 31 March 2025).
4. Freund, Y., & Schapire, R. E. (1997). *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of Computer and System Sciences, 55(1), 119–139. https://doi.org/10.1006/jcss.1997.1504
5. Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). *A Bayesian approach to filtering junk e-mail*. Learning for Text Categorization: Papers from the 1998 Workshop. AAAI Technical Report WS-98-05. https://www.microsoft.com/en-us/research/publication/a-bayesian-approach-to-filtering-junk-e-mail/
6. Zhang, Y., & Zhou, Z.-H. (2009). *Cost-sensitive face recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(10), 1758–1764. https://doi.org/10.1109/TPAMI.2009.51