



UNIVERSITY  
OF WOLLONGONG  
AUSTRALIA



SIM Global Education

# CSCI323: Modern AI Project Report

**Data-Driven Feature Engineering and  
Multi-Model Optimization  
for Enhanced Spam Detection**

**Group No. 18  
Spam Email Detection**

Name	UOW ID
Loh Chin Yee	7687278
Chua Cheng Yi	7897509
Yeonjae Lim	7914180
Jiwon Moon	7923569
Swam Pyae Aung	7687436
Oh Hwee Xin	6663175

**Assessor:** Mr. Sionggo Japit

# TABLE OF CONTENTS

<b>CHAPTER 1: INTRODUCTION.....</b>	<b>3</b>
1.1 Introduction.....	3
1.2 Background Theory.....	3
<b>CHAPTER 2: DATA HANDLING.....</b>	<b>4</b>
2.1 Data Collection and Initial exploration.....	4
2.2 Data Cleaning.....	4
2.3 Data Preprocessing.....	4
2.4 Data exploration.....	5
2.5 Feature Extraction.....	5
2.5.1 Importance of Metadata.....	5
2.5.2 Extracted Features.....	6
2.6 Feature Engineering.....	6
2.7 Data Transformation.....	6
2.7.1 Vectorization.....	6
2.7.2 One-hot Encoding.....	6
2.8 Feature Selection.....	7
2.8.1 Correlation Test.....	7
2.8.2 Dropped Features.....	7
<b>CHAPTER 3: MODELING.....</b>	<b>7</b>
3.1 Model Implementation.....	7
3.1.1 Logistic Regression.....	7
3.1.2 Random Forest.....	8
3.1.3 Dense Neural Network.....	8
3.2 Fine Tuning.....	8
3.2.1 Logistic Regression.....	8
3.2.2 Random Forest.....	8
3.2.3 Dense Neural Network.....	8
3.3 Evaluation.....	8
<b>CHAPTER 4: CONCLUSION.....</b>	<b>10</b>
<b>APPENDIX.....</b>	<b>11</b>
Peer Assessment.....	11
Source Code.....	11
<b>REFERENCES.....</b>	<b>12</b>

# CHAPTER 1: INTRODUCTION

---

## 1.1 Introduction

In today's digital communication landscape, the proliferation of spam emails poses a significant challenge, necessitating the development of sophisticated detection models. This project focuses on leveraging data-driven feature engineering to identify the most effective features from email headers and bodies, which are crucial for distinguishing between legitimate emails and spam. By extracting and analyzing features such as sender information, authentication header, subject lines, and message content, we aim to enhance the accuracy of spam detection.

To achieve this, we will employ three different machine learning models, each optimized to maximize detection performance. This approach not only allows us to compare the effectiveness of various algorithms but also to fine-tune them for optimal results. The models will be evaluated using metrics such as confusion matrix, roc-auc score, training-validation loss etc to ensure a comprehensive understanding of their capabilities. The integration of feature engineering with advanced machine learning techniques promises to advance the field of email spam detection, providing a robust framework for tackling the ever-evolving tactics used by spammers.

In addition to leveraging advanced machine learning techniques, this project incorporates regulatory frameworks like the CAN-SPAM Act to enhance spam detection. The act mandates features such as a 'List-Unsubscribe' header, enabling recipients to opt-out of unwanted emails and ensuring compliance with legal standards. By integrating this with technical features like authentication headers (SPF, DKIM, DMARC) and the 'X-Spam-Status' header, which verify sender identity and assess spam scores, our model can more accurately identify and filter spam.

The comprehensive approach not only improves detection accuracy but also contributes to a safer and more secure email environment by focusing on both the technical and practical aspects of spam detection.

## 1.2 Background Theory

Natural Language Processing (NLP) plays a crucial role in this by converting unstructured email text into structured data that machine learning models can interpret. Techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) are employed to quantify the importance of words in an email, aiding models in identifying spammy language or phrases. [1] (Houlsby et al., 2019) Machine learning classifiers such as Logistic Regression, Random Forest, and Dense Model are commonly used in spam detection, with modern approaches also leveraging deep learning and neural networks for enhanced accuracy. These spam filters evolve by analyzing patterns in email content, sender behavior, and statistical features, and they are trained using labeled datasets to improve their effectiveness over time. [2] (Dada a et al, 2019)

Three distinct models were chosen for the spam email classifier:

- **Logistic Regression:** Chosen for its simplicity and effectiveness in binary classification tasks, Logistic Regression is particularly useful when dealing with linearly separable data. It provides a clear probabilistic interpretation of predictions and is less prone to overfitting, especially when regularized. [3] (G et al., 2022)
- **Random Forest:** This model was selected for its ability to capture complex, non-linear relationships between features. Unlike linear models, Random Forest can model interactions between multiple features, making it highly effective in handling the diverse and intricate patterns often present in spam emails. Additionally, its ensemble nature improves robustness and generalization, reducing the risk of overfitting while enhancing prediction accuracy. [4] (Belgiu & Drăguț, 2016)
- **Dense Neural Network (DNN):** DNNs were chosen for their powerful feature extraction capabilities, especially when working with large and complex datasets. Unlike traditional models, DNNs can

automatically learn and refine features from raw data, such as email content and metadata, through multiple layers of neurons. This makes them particularly adept at capturing subtle patterns that might be overlooked by other models, thereby improving overall detection accuracy. [5] (Ahmed et al., 2023)

Key features for spam detection include analyzing the content of the email, particularly the subject and body, for common spam indicators such as specific keywords, phrases, or patterns frequently associated with spam messages. Another critical aspect is scrutinizing the sender information, particularly the 'From' and 'Return-Path' fields, for inconsistencies or any association with known spam addresses. Additionally, emails with duplicate content—those containing identical or nearly identical subjects or bodies—are often flagged as spam. Finally, a thorough examination of the 'Received' headers is performed to trace the email's journey, identifying any anomalies or unusual patterns that might suggest the email is spam.

## CHAPTER 2: DATA HANDLING

---

### 2.1 Data Collection and Initial exploration

This study employs TREC 2007 Public Spam Corpus(TREC 07p) to develop spam detection models. Initially, we considered several datasets, but ultimately chose TREC 07p because it retains comprehensive email metadata. This rich metadata allows sophisticated feature selection and engineering. The realistic patterns and accurate labeling make it ideal for building robust supervised learning models with high precision. The TREC07 dataset includes 75,419 emails, with 50,199 labeled as spam and 25,220 as ham, highlighting a class imbalance that requires adjustment during modeling. The dataset features two columns: the label (spam(1) and ham(0)) and the origin, containing all email data.

From the first 3 emails of spam and ham emails read through and analyzed, we noticed distinct trends:

- Ham emails: Feature continuous conversations or subscribed content
- Spam emails: Often discuss weight loss pills, Viagra, and safety assurance

### 2.2 Data Cleaning

- **Multipart Email Handling and Separation of Headers and Bodies:** We parsed each email to separate headers from the body. Headers provide valuable metadata like sender information and routing paths, which can reveal technical indicators of spam. The body contains the actual message, analyzed for text patterns typical of phishing or spam. For multipart emails, each section is individually decoded and processed.
- **Handling Missing Data:** To ensure the integrity and consistency of the dataset. Empty email bodies and headers lines are filled with an empty string ("").

### 2.3 Data Preprocessing

- **HTML and XML Processing:** To extract meaningful text from HTML or XML emails, we utilize BeautifulSoup to parse the HTML or XML content, converting it into plain text for analysis without interference. To identify potential spam, we also check for the presence of 'mailto' anchor tag in the email HTML body and extract it, as HTML content often embeds links to phishing sites or tracks user behavior.
- **Removal base64 encoded content:** Ensure that the model is trained solely on textual data, which is crucial for focusing on linguistic patterns and avoiding noise

- **Email body and subject text processing:** To process the text of email bodies and subjects, the text is converted to lowercase, punctuation is removed. The text is tokenized into individual words using the NLTK library. After tokenization, common stop words, which are frequently used words that add little meaning, are removed using the NLTK stopwords corpus.
- **Email header processing:** Following the initial exploration and research, we process headers that could be potentially useful ('List-Unsubscribe', 'X-Spam-Status', etc) for spam email analysis. Empty row is replaced with ('') or ([]) or None depending on condition.

## 2.4 Data exploration

- **Common word analysis on email subjects:** This analysis identifies frequently used words in spam and legitimate email subjects. Spam subjects often include terms like 'viagra', 'cheap' and 'free' indicating promotional content, while legitimate subjects feature words like 'meeting' and 'update,' reflecting work-related communication.
- **Bigram analysis on email body:** Examining word pairs in email bodies reveals that spam emails commonly feature bigrams like 'pills x' and 'save up,' indicating promotional themes. In contrast, legitimate emails frequently contain bigrams such as 'mailing list' and 'please read,' reflecting structured communication.
- **Return path and From header analysis:** Comparing domains in the 'return\_path' and 'from' headers shows that spam emails often have exact domain matches, while legitimate emails commonly exhibit domain mismatches for valid reasons.
- **Textual analysis on Mailto:** Unigrams from the 'mailto\_header' column reveal that 'List-Unsubscribe' is the most common term in both spam and legitimate emails, with spam emails featuring it more frequently. This indicates that 'List-Unsubscribe' is a significant term for further analysis.
- **Mailto analysis**
  - **Distribution analysis:** The presence of a 'list-unsubscribe' header strongly correlates with non-spam emails, supporting the hypothesis that this header is a key indicator of legitimate emails.
  - **Correlation analysis:** A heatmap shows a strong negative correlation (-0.79) between the 'list\_unsubscribe\_populated' feature and emails being labeled as spam, highlighting this header as an important indicator of legitimacy.

## 2.5 Feature Extraction

### 2.5.1 Importance of Metadata

To enrich the feature set with information that is often less manipulated and more structured than body text. Metadata fields such as Return-Path, From, Received, To, Subject, Authentication and X-Spam-Status are extracted and analyzed individually. These fields are then converted into features that can be used to train the spam detection model. While the body text can be highly variable and manipulated by spammers, metadata is typically added by email servers and systems, making it more reliable and less prone to obfuscation. Features extracted from metadata can reveal patterns or anomalies indicative of spam, such as unusual sending paths, suspicious sender addresses, or abnormal email structures.

## 2.5.2 Extracted Features

- **Header Extraction** : The 'return\_path' tracked email origins to detect spoofing, while the 'from' header was checked for subtle variations suggesting phishing. The 'received' header analyzed the routing path for unusual hops, indicating potential unauthorized routing. The 'to' header revealed bulk mailing through inconsistencies, and the 'subject' header was scanned for phishing-related keywords. The 'x\_spam\_status' provided an initial spam assessment, and 'authentication' headers like 'DKIM' and 'SPF' verified the email's authenticity, flagging potential spam when these checks failed.
- **List-Related Headers** : The 'list\_unsubscribe' header allows recipients to opt out of future emails, signaling compliance with regulations and associating with non-spam. The 'list\_subscribe' header indicates a legitimate subscription service. The 'list\_post' header points to a structured, legitimate communication channel, while the 'list\_help' header provides transparency and support, further legitimizing the email.
- **Body Content** : "mailto" links are crucial indicators in spam detection. 'mailto\_anchor', found within the email body as clickable text (e.g., "Contact Us"), are often used in phishing to direct recipients to malicious addresses. 'mailto\_header', appearing in headers like 'reply\_to' or 'return\_path', can be manipulated by spammers to mask the email's origin or redirect replies, making them vital for identifying potential spam.

## 2.6 Feature Engineering

- **mailto\_populated** : It was created to represent the presence or absence of "mailto" links for each email.
- **list\_unsubscribe\_populated** : It was created to represent the presence or absence of 'list\_unsubscribe' information for each email,
- **body\_duplicates** : The value indicates the frequency of a particular email body appearing across multiple emails.
- **is\_authenticated** : The column indicates whether an email has been authenticated using methods such as SPF, DKIM, or DMARC, with a binary value.
- **relay\_count** : The column derived from the number of Received headers, is considered an indicator of spam mail when excessively high.

## 2.7 Data Transformation

### 2.7.1 Vectorization

We use the TfidfVectorizer from scikit-learn for text vectorization, which effectively captures word frequency and importance across the corpus. TF-IDF vectorization is applied to text-heavy columns like body content, subject, and sender addresses, converting text into numerical vectors for machine learning models. This step follows data extraction and precedes modeling. By adjusting the max\_features parameter, we aim to control the model's subsequent dependence on text columns. This approach allows us to regulate the influence of textual data on the model's decision-making process, potentially mitigating overfitting and enhancing generalization capabilities.

### 2.7.2 One-hot Encoding

One-hot encoding was applied to the 'x\_spam\_status' and 'domain\_comparison' columns to convert categorical data into a binary format. Using 'pd.get\_dummies()', each category within these columns was

transformed into separate binary columns (e.g., 'x\_spam\_status\_no', 'x\_spam\_status\_yes'). These new columns were then converted from boolean to binary (0 or 1) using a custom function, and added to the list of binary categorical columns.

## 2.8 Feature Selection

### 2.8.1 Correlation Test

- **mailto\_populated** : A Phi test was conducted and the analysis yielded a Phi coefficient of -0.807, suggesting a significant inverse relationship wherein the absence of 'mailto' information correlates with an increased likelihood of the email being classified as spam.
- **list\_unsubscribe\_populated** : The Phi test was conducted and the analysis produced a Phi coefficient of -0.795, revealing a notable inverse correlation, where the absence of list-unsubscribe information corresponds with a higher probability of the email being classified as spam.
- **body\_duplicates** : A Point-Biserial test was conducted, and the result showed a correlation of 0.1359, indicating a low correlation.
- **is\_authenticated** : Before checking the correlation through validation, we examined the distribution of spam (label) based on the 'is\_authenticated' status. The results showed that 99.2% of emails were labeled as non-spam when authenticated, and 28.8% were non-spam when not authenticated, leading to the expectation of a high correlation with the label. However, a subsequent Phi test yielded a Phi coefficient of -0.372, indicating a relatively low correlation. This result is likely attributable to the lack of authentication-related information in a significant portion of the data.
- **x\_spam\_status** : Cramér's V test was conducted, and the value of 0.757 indicated a high correlation.
- **domain\_comparison** : Cramér's V test was conducted, and the value of 0.644 indicated a moderately high correlation.

### 2.8.2 Dropped Features

Certain columns were excluded from the model due to their low or negative correlation with the target variable (label). Specifically:

- 'body\_duplicates' : This feature exhibited a very low correlation with the target label, making it unlikely to contribute significantly to the model's predictive power. Consequently, it was removed from the final set of features.
- 'relay\_count' : Although this feature had a relatively higher correlation, the correlation was negative, meaning that as the number of relays increased, the probability of an email being spam decreased. This counterintuitive result could potentially confuse the model during training, so it was also excluded.
- 'to' : The recipient's information was not considered relevant for spam detection. Since spam is typically sent to large volumes of recipients, analyzing sender-related features is likely more informative than recipient details.

## CHAPTER 3: MODELING

---

### 3.1 Model Implementation

#### 3.1.1 Logistic Regression

The max\_features parameter was set to 10, 100, 1, and 1('subject', 'body', 'from\_email' and 'return\_email') after experiments to prevent overfitting. The C value was set to 0.01 to reduce complexity and

avoid overfitting. The model exhibited strong performance, achieving AUC scores of 0.9808 on the validation set and 0.9826 on the test set, demonstrating robust predictive capability. However, as a linear model, Logistic Regression might not capture complex patterns in the data.

### **3.1.2 Random Forest**

In the Random Forest model, the number of trees, 'n\_estimators', was set to 100, the maximum depth of each tree, 'max\_depth', was limited to 3 to prevent overfitting, and the minimum number of samples required at each leaf node, 'min\_samples\_leaf', was set to 5. It achieved a validation AUC score of 0.9763 and matched Logistic Regression on the test set with an AUC of 0.9826. The model demonstrated strong precision, recall, and F1-scores across both sets. This suggests that while Random Forest offers the benefit of capturing non-linear relationships, it might be more sensitive to hyperparameter tuning and data characteristics.

### **3.1.3 Dense Neural Network**

In the Dense Neural Network model, dense layers with 64 units and ReLU activation were used, followed by a 0.5 dropout to prevent overfitting. The final layer used a sigmoid activation for binary classification. The model was compiled with the Adam optimizer (learning rate 0.001) and binary cross entropy loss. The model demonstrated outstanding performance, achieving a validation AUC of 0.9963 and a test AUC of 0.9965. High precision, recall, and F1-scores confirmed its strong classification ability. Compared to the other models, the Dense Model excelled, likely due to its capacity to capture complex, non-linear patterns, making it the top performer in this analysis.

## **3.2 Fine Tuning**

### **3.2.1 Logistic Regression**

To improve the Logistic Regression model, hyperparameter tuning was performed using GridSearchCV. Unlike the basic model, this process optimized parameters like C=1 for TFIDF vectorization. The tuning led to a slightly advanced performance, with the validation AUC score rising to 0.9925 and the test AUC to 0.9934, indicating a more refined model.

### **3.2.2 Random Forest**

Hyperparameter tuning using GridSearchCV was applied to the Random Forest model to optimize parameters like max\_depth, min\_samples\_leaf and n\_estimators. This led to an enhancement in performance, with validation AUC rising to 0.9812 and test AUC to 0.9832. The improved AUC, precision, recall, and F1-scores indicate that the model now captures data patterns more effectively.

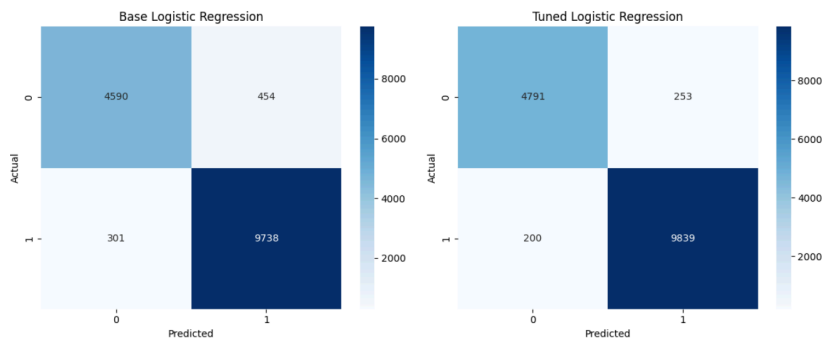
### **3.2.3 Dense Neural Network**

Hyperparameter tuning optimized the DNN by systematically searching for the best dense\_units, dropout\_rate, and batch\_size combinations. This resulted in an improvement in performance, with validation AUC at 0.9963 and test AUC at 0.9965, improving precision, recall, and F1-scores.

## **3.3 Evaluation**

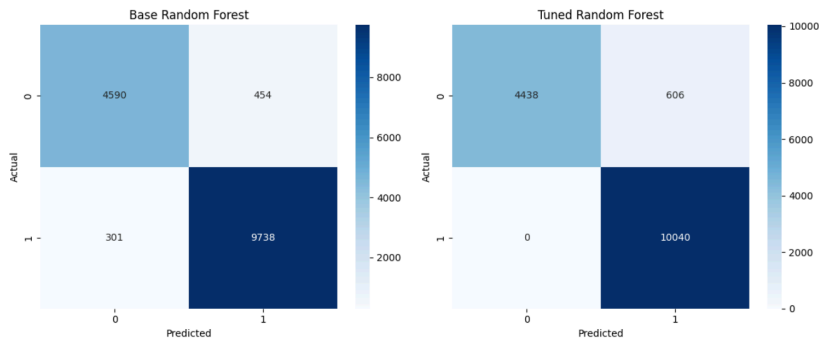
### **3.3.1 Logistic Regression**





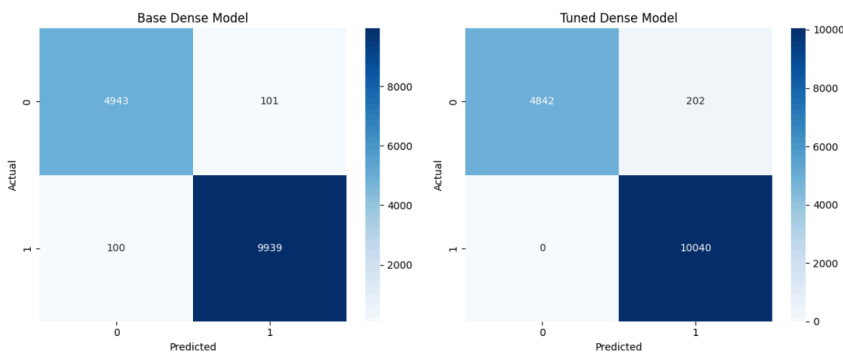
As a result, the number of True Positive and True Negative increased, while False Positive and False Negative decreased, allowing the tuned model to classify the target values more effectively. However, as the optimal C-value parameter is set to 1, the risk of overfitting increases compared to the original base model. While the tuned model demonstrates superior numerical performance, the base model is expected to exhibit greater stability in terms of model robustness.

### 3.3.2 Random Forest



This approach applied restricted tree depth and minimum sample size, and achieved an improved AUC score. By increasing the maximum depth of the trees, each tree could learn more deeply, and by reducing the minimum leaf node sample size, more splits were made. After tuning, there was a positive change with an increase in the number of True Positives and a decrease in False Negatives. Although the number of True Negatives decreased and False Positives increased, the AUC score still improved compared to the base model. Thus, while the base model may be more stable, the tuned model's performance suggests that it relies less on specific thresholds and makes better classification decisions across the entire range of thresholds.

### 3.3.3 Dense Model



Given that the base model was already configured with parameters to prevent overfitting, it demonstrated strong performance with high True Positive and True Negative counts. However, in pursuit of

further model enhancement, the batch size was increased. A larger batch size allowed each batch to include more samples, leading to more stable and averaged weight updates. This adjustment reduced the impact of noise during the training process, improved the AUC score, and still effectively prevented overfitting. Similar to the Random Forest model, there is room for improvement as the number of True Negatives decreases and False Positives increase. However, the increase in AUC indicates more effective classification decisions across the entire range of thresholds.

In a nutshell, Various parameters were applied from the base model stage for each model, and high accuracy was confirmed from the outset. Nevertheless, to further improve model performance, GridSearchCV was conducted to test various parameters, ultimately identifying the most suitable hyperparameters for the models. This approach optimized model performance. However, during the tuning process, some parameter values directly related to overfitting were modified. Therefore, before practical deployment, it is deemed necessary to consider a balance between performance and generalization capabilities.

Among the three models, the Dense Neural Network model achieved the highest performance with an AUC score of 0.9965 due to its ability to automatically learn and refine complex feature representations from raw data, followed by Logistic Regression and Random Forest.

## CHAPTER 4: CONCLUSION

---

This project highlighted the critical role of feature engineering in significantly improving model performance. Initially, our model achieved a 90% accuracy rate using just two text columns, but after implementing feature engineering techniques, this accuracy improved to around 98% and 99%. These enhancements were achieved through capturing intricate patterns and relationships within the data, which allowed for better interpretability and reduced data complexity.

Furthermore, the project underscored the necessity of model optimization to manage overfitting, a common challenge in machine learning. Although our base models initially showed high accuracy, particularly due to the strong predictive power of features like email subject and body, fine-tuning and balancing the dataset were essential in refining our models. The Dense Neural Network model emerged as the top performer after addressing overfitting concerns through techniques like adjusting hyperparameters and employing balanced class weights. Furthermore, to enhance the model's generalization performance beyond the level achieved through grid search tuning, specific parameters can be selectively adjusted.







Looking ahead, there is room for further improvement and exploration. Future work could include conducting a more detailed feature importance analysis to better understand the contribution of each feature. Additionally, incorporating interpretability tools such as SHAP and LIME would provide deeper insights into model decisions. Applying our model to different datasets could also help in addressing the data shift problem and testing the model's robustness against adversarial attacks.

In summary, this project not only achieved its goal of developing a highly accurate spam detection model but also laid the groundwork for further advancements. The lessons learned from feature engineering, model optimization, and future prospects will aid us in future development.

## APPENDIX

---

### Peer Assessment

Student Number	Student Name	Signature	Proposed Index Value	Individual Content Claimed
7687278	Loh Chin Yee		100%	<ul style="list-style-type: none"><li>- Data Preprocessing</li><li>- Document</li><li>- Presentation Slides</li></ul>
7897509	Chua Cheng Yi		100%	<ul style="list-style-type: none"><li>- Data Analysis</li><li>- Document</li><li>- Presentation Slides</li></ul>
7914180	Yeonjae Lim		100%	<ul style="list-style-type: none"><li>- Model Implementation</li><li>- Document</li></ul>
7923569	Jiwon Moon		100%	<ul style="list-style-type: none"><li>- Model Implementation</li><li>- Document</li><li>- Presentation Slides</li></ul>
7687436	Swam Pyae Aung		100%	<ul style="list-style-type: none"><li>- Model Tuning</li><li>- Document</li><li>- Presentation Slides</li></ul>
6663175	Oh Hwee Xin		100%	<ul style="list-style-type: none"><li>- Data Analysis</li><li>- Document</li><li>- Presentation Slides</li></ul>

### Source Code

[https://colab.research.google.com/drive/1vE-vgN992LJbrw\\_jQD\\_IC4MCdjmah-ph?usp=drive\\_link](https://colab.research.google.com/drive/1vE-vgN992LJbrw_jQD_IC4MCdjmah-ph?usp=drive_link)

## REFERENCES

---

- [1] Houlisby, N., Giurciu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S., 2019. Parameter-Efficient Transfer Learning for NLP. ArXiv. Available at: <https://api.semanticscholar.org/CorpusID:59599816> [Accessed 16 Aug. 2024].
- [2] Dada, E.G., Bassi, J.S., Chiroma, H., Abdulhamid, S.M., Adetunmbi, A.O. and Ajibuwa, O.E., 2019. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6), p.e01802. Available at: <https://doi.org/10.1016/j.heliyon.2019.e01802> [Accessed 16 August 2024].
- [3] G et al. (2022). Logistic Regression Technique for Prediction of Cardiovascular Disease. *Global Transitions Proceedings*. Available at: <https://doi.org/10.1016/j.gltp.2022.04.008> [Accessed 16 August 2024].
- [4] Belgiu, M., & Drăguț, L. (2016). Random forest in remote sensing: A review of applications and future directions. *Isprs Journal of Photogrammetry and Remote Sensing*, 114, 24-31. Available at: <https://doi.org/10.1016/j.isprsjprs.2016.01.011> [Accessed 16 August 2024].
- [5] Ahmed, M., Akter, M., Rahman, M., Rahman, M., Paul, P., Parvin, M. & Antar, A., 2023. Deep Neural Network Based Spam Email Classification Using Attention Mechanisms. *Journal of Intelligent Learning Systems and Applications*, 15, pp.144-164. Available at: <https://doi.org/10.4236/jilsa.2023.154010> [Accessed 16 August 2024].
- [6] Grootendorst, M.R., 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. ArXiv. Available at: <https://api.semanticscholar.org/CorpusID:247411231> [Accessed 16 Aug. 2024].
- [7] Bird, S., Klein, E., & Loper, E., 2009. *Natural Language Processing with Python*. O'Reilly Media. Available at: <https://www.nltk.org/book/> [Accessed 16 Aug. 2024].
- [8] Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2nd ed.). Pearson Prentice Hall. Available at: <https://web.stanford.edu/~jurafsky/slp3/> [Accessed 19 Aug. 2024].