UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME	NAME OF STUDENT(s) (PRINT CLEARLY) *SURNAME FIRSTNAME*	STUDENT ID(s)
41180 Data Analytics in Cybersecurity	**Shah OM**	24831101

STUDENT EMAIL	STUDENT CONTACT NUMBER
Om.S.Shah@student.uts.edu.au	**N/A**

NAME OF TUTOR	TUTORIAL GROUP	DUE DATE
Dr Bo Liu	**Computer Lab 07**	**26/03/2026**

ASSESSMENT ITEM NUMBER & TITLE

Assessment task 1: Project 1: Email spam filtering

☒ 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) _____ **Date** 24/03/2026

Contributions by members

NAME	CONTRIBUTION
ARHAM SHERWANI	33.3%
OM SHAH	33.3%
PHYO HTET AUNG	33.3%

An even effort was made by all group members to complete this assessment.

Methods presented by:

Naïve Bayes by Arham Sherwani

Deep Learning (Conv1D CNN) by Om Shah

SVM by Phyo

CONTENTS

1. INTRODUCTION

This project focuses on classifying emails as either spam or legitimate using different machine learning approaches. Spam detection is a common problem, and building an effective model requires more than just achieving high accuracy. Since datasets often contain more legitimate emails than spam, it is important to use evaluation metrics that reflect how well spam is actually detected.

Three models were explored in this study: Naive Bayes, Support Vector Machine (SVM), and a deep learning Convolutional Neural Network (CNN). Each model approaches the task differently, which makes it possible to compare their strengths and weaknesses. The aim was to determine which model performs best overall, with a focus on how effectively spam emails are identified while keeping incorrect classifications low.

To support this, the models were evaluated using multiple metrics, including precision, recall, F1-score, and ROC-AUC. This allowed for a more complete understanding of performance and helped identify which model was most suitable for real-world spam filtering.

2. PROPOSED SOLUTION - CNN

```
Model: "sequential"
+---------------------------------------------------------------------------+
| Layer (type)                    | Output Shape           |       Param # |
|---------------------------------+------------------------+---------------|
| embedding (Embedding)           | (None, 80, 100)        |       792,000 |
|---------------------------------+------------------------+---------------|
| conv1d (Conv1D)                 | (None, 76, 128)        |        64,128 |
|---------------------------------+------------------------+---------------|
| global_max_pooling1d            | (None, 128)            |             0 |
| (GlobalMaxPooling1D)            |                        |               |
|---------------------------------+------------------------+---------------|
| dense (Dense)                   | (None, 64)             |         8,256 |
|---------------------------------+------------------------+---------------|
| dropout (Dropout)               | (None, 64)             |             0 |
|---------------------------------+------------------------+---------------|
| dense_1 (Dense)                 | (None, 1)              |            65 |
+---------------------------------------------------------------------------+
 Total params: 2,593,349 (9.89 MB)
 Trainable params: 864,449 (3.30 MB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 1,728,900 (6.60 MB)
```

Figure 2.1 – Model Architecture

As shown in ***Figure 2.1***, a 1D Convolutional Neural Network (CNN) was used to classify emails as either spam or legitimate. This model was chosen because spam detection depends heavily on identifying local patterns in text, such as suspicious words, repeated phrases, and short combinations of terms that commonly appear in spam messages. CNNs are well suited to this type of text classification because they can learn these local features directly from sequences of words ***(Kim, 2014)***.

Before the emails could be processed by the model, the text had to be converted into a numerical form. This was achieved through tokenisation, in which each email was split into individual words and each word was assigned a numerical index.

```
max_words = 20000
max_len = 80
embedding_dim = 100

tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)

train_seq = tokenizer.texts_to_sequences(X_train)
test_seq = tokenizer.texts_to_sequences(X_test)

X_train_pad = pad_sequences(train_seq, maxlen=max_len, padding="post", truncating="post")
X_test_pad = pad_sequences(test_seq, maxlen=max_len, padding="post", truncating="post")

word_index = tokenizer.word_index
num_tokens = min(max_words, len(word_index) + 1)
```

Figure 2.2 - Text Tokenisation and Sequence Padding for CNN Input

As shown in ***Figure 2.2***, the tokenizer was restricted to the 20,000 most frequent words, while unseen words were mapped to an out-of-vocabulary token to ensure they could still be processed consistently. The tokenised emails were then converted into numerical sequences and padded to a fixed length of 80 tokens. Following ***Hakami (2025),*** shorter emails were padded, and longer emails were truncated so that all input sequences had a consistent length. This helps support efficient batch processing by ensuring a uniform input shape during model training.

The first layer of the model was an embedding layer. It turned each word into numbers the model could work with. Pre-trained GloVe embeddings were used so the model did not have to learn every word relationship from scratch. Instead, it started with some built-in understanding of how words are connected. This was helpful in text classification, because words with similar meanings can influence the final prediction in similar ways ***(Pennington et al., 2014).***

Next, the Conv1D layer moved across the email text to detect short patterns that could help separate spam from legitimate emails. Some of these patterns might be suspicious words, repeated phrases, or common word combinations. Global max pooling was then used to keep the strongest features and remove less important information. This made the model simpler and reduced the amount of data passed forward. Finally, the remaining features were sent to a dense layer for classification, while dropout was used during training to help reduce overfitting.

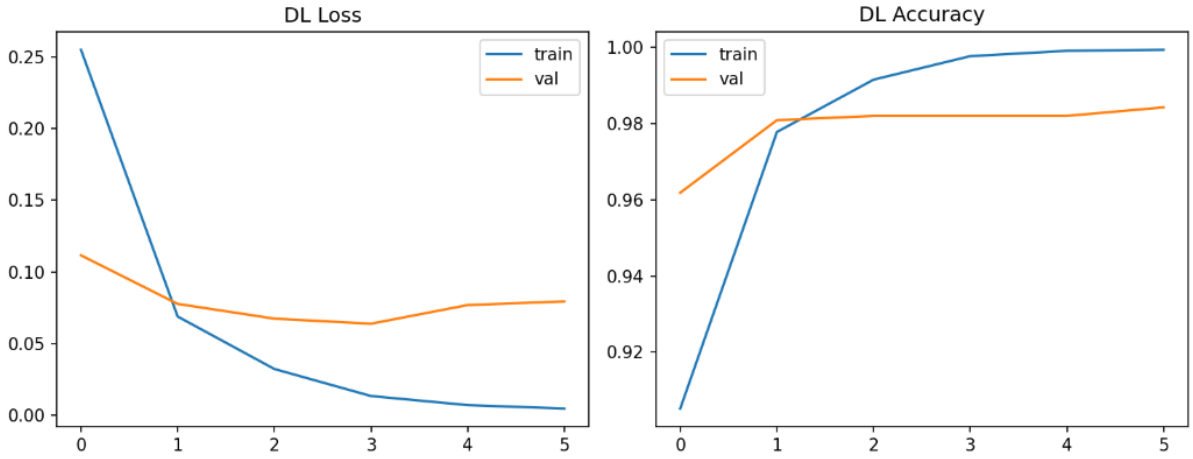


Figure 2.3 - Deep Learning Training and Validation Loss and Accuracy Across Epochs

Figure 2.3 depicts that the training loss decreased steadily across the epochs. This suggests that the model was learning the training data well. The validation loss also improved at the beginning, then stayed mostly stable, with only a small increase near the end. Training accuracy increased quickly and came close to perfect, while validation accuracy also improved fast before settling at a high level.

Overall, these results suggest that the model generalised well. However, the growing gap between training and validation performance in the later epochs points to mild overfitting.

This model was chosen because spam email classification is an important machine learning problem, and CNNs are well suited to learning local text features ***(Dada et al., 2019; Kim, 2014).*** With the addition of pre-trained GloVe embeddings, the model was able to represent words more effectively and capture useful semantic relationships within the emails. Overall, deep learning methods that use NLP and word embeddings have also been applied effectively to spam email classification ***(AbdulNabi & Yaseen, 2021).***

3. METRICS AND RESULTS

The results showed that all three models performed well overall, but their ability to detect spam was not the same. This is important because the test set contained many more ham emails than spam emails, with 966 ham and 149 spam. Because of this class imbalance, accuracy on its own does not give the full picture. Spam precision, recall, and F1-score are more useful for comparing model performance.

Model 1 - CNN

```
--- Model Evaluation (Deep Learning CNN) ---
              precision    recall  f1-score   support

         ham       0.99      1.00      0.99       966
        spam       0.97      0.93      0.95       149

    accuracy                           0.99      1115
   macro avg       0.98      0.97      0.97      1115
weighted avg       0.99      0.99      0.99      1115
```

Among the three models, the deep learning CNN achieved the strongest overall performance. As shown in ***Figure 3.1***, it recorded the highest accuracy, spam recall, spam F1-score, and ROC-AUC, indicating that it was the most effective model for distinguishing spam from legitimate emails.

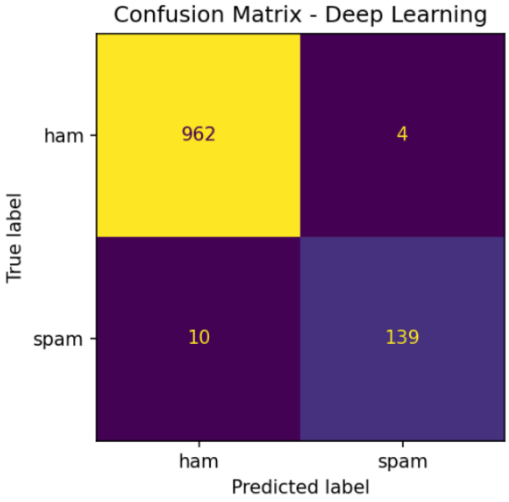
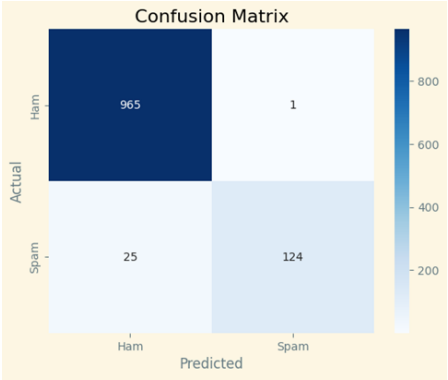


Figure 3.2, confusion matrix, supports this. The model correctly classified 962 ham emails and 139 spam emails, while only 4 ham emails were incorrectly labelled as spam and 10 spam emails were incorrectly labelled as ham. This suggests that the CNN achieved a strong balance between catching spam and avoiding false spam warnings.

Model 2 – Naïve Bayes

	precision	recall	f1-score	support
ham	0.97	1.00	0.99	966
spam	0.99	0.83	0.91	149
accuracy			0.98	1115
macro avg	0.98	0.92	0.95	1115
weighted avg	0.98	0.98	0.98	1115

The Naïve Bayes model also produced high overall accuracy as well as the highest spam precision at 99.2% ***(Figure 3.3)***. This means that when it predicted an email as spam, it was almost always correct. However, its lower spam recall and F1-score show that it missed more spam emails than the other models.

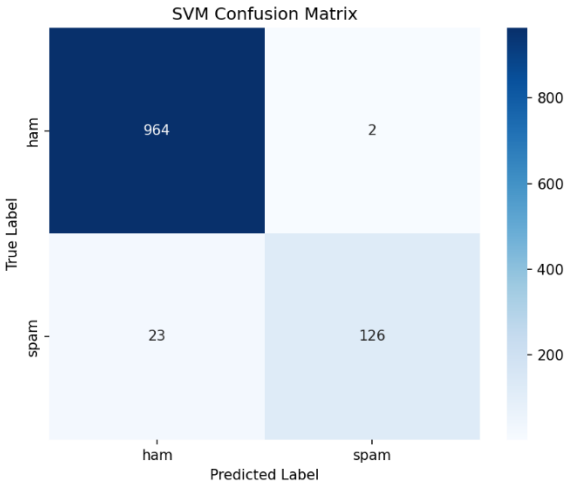


The confusion matrix in ***Figure 3.4*** supports this, showing very few false positives but a larger number of spam emails incorrectly classified as ham. This suggests that Naive Bayes was a conservative model that reduced false alarms at the expense of allowing more spam to pass through.

Model 3 – SVM

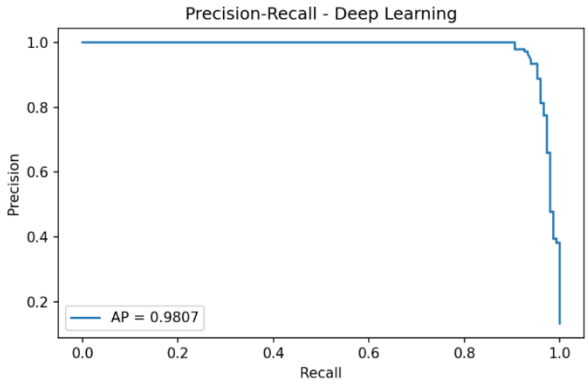
```
========================================
CLASSIFICATION REPORT
========================================
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       966
        spam       0.98      0.85      0.91       149

    accuracy                           0.98      1115
   macro avg       0.98      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```

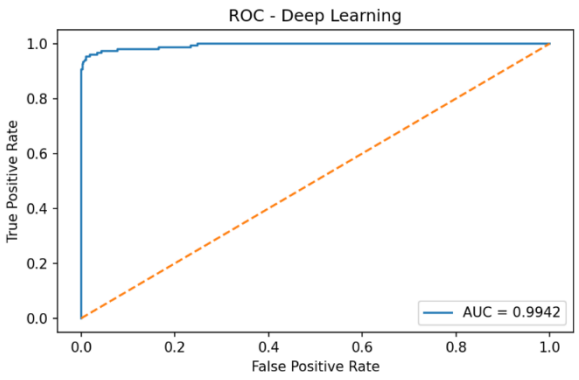
As shown in ***Figure 3.5***, the SVM model performed slightly better than Naive Bayes overall, although it still did not match the CNN. It achieved high accuracy and strong spam precision, but its lower spam recall and F1-score show that it still missed a noticeable number of spam emails.



The confusion matrix in ***Figure 3.6*** supports this, showing very few false positives but 23 spam emails incorrectly classified as ham. This suggests that the SVM was effective overall, but less reliable than the CNN at identifying spam.

When the three models are compared directly, the CNN was the most effective model for this task. Although Naive Bayes achieved the highest precision, precision alone is not enough in spam filtering. A model that misses too many spam emails is less useful in practice, even if its spam predictions are highly accurate. The SVM improved slightly on Naive Bayes in recall and F1-score, but the CNN clearly performed best across the most important measures.





It had the highest accuracy, the highest spam recall, the highest spam F1-score, and the strongest overall ranking performance based on ROC-AUC and average precision. The CNN training curves in ***Figure 2.3*** also help explain why it performed better. Training loss decreased steadily across the epochs, while training accuracy rose quickly and came close to perfect performance. Validation loss improved early and then remained fairly

stable, with only a small increase in later epochs. Validation accuracy also increased quickly and then stabilised at a high level of around 98.4%. This pattern suggests that the model learned the training data effectively and generalised well to unseen data. There was some sign of mild overfitting in the later epochs because the gap between training and validation performance became slightly wider, but this did not appear to seriously reduce test performance.

Overall, the results indicate that all three models were capable of classifying emails as spam or ham with high accuracy. However, the CNN was the strongest choice for the final model. It offered the best balance between precision and recall, detected more spam emails than the other two approaches, and still kept false positives very low. For a spam filtering task, this makes it the most practical and reliable model out of the three.

4. REFLECTION

This project showed that high accuracy does not automatically mean a model is doing the job well. At first, all three models looked strong because their accuracy scores were high, but a closer look showed clear differences in how well they detected spam. Since there were more ham emails than spam emails in the dataset, accuracy alone gave only a limited view of performance. Precision, recall, and F1-score were much more useful for showing how well each model handled spam. The comparison also showed that each model had a different trade-off. Naive Bayes and SVM were both careful when predicting spam, which kept false positives low, but that caution also meant that more spam emails were missed. The CNN handled this balance better by catching more spam while still keeping false positives low.

Tokenisation, padding, and word embeddings were not just setup steps, but important parts of how the text was represented to the models showing the value of pre-processing and model evaluation. The results suggested that the stronger text representation used in the CNN helped it capture more meaningful patterns in the emails. The project also showed the value of using more than one evaluation method. Confusion matrices made the errors easy to identify, while ROC and precision-recall curves gave a clearer picture of overall model quality. The training graphs also helped show that even strong models still need careful interpretation, especially when there are signs of mild overfitting. Overall, the project showed that choosing a model is not just about finding the highest score, but about understanding which model best suits the real goal of the task.

5. CONCLUSION

This study compared three models for spam email classification: Naive Bayes, SVM, and a deep learning CNN. All three performed well overall, but they did not perform equally when it came to detecting spam. Because the dataset was imbalanced, the most useful comparison came from spam-focused metrics rather than accuracy alone.

The CNN delivered the strongest overall results. It achieved the highest accuracy, the highest spam recall, and the highest spam F1-score. It also showed very strong ROC-AUC and average precision, confirming that it separated spam and ham effectively. In simple terms, it caught more spam emails than the other models while still keeping false positives low.

Naive Bayes and SVM both gave solid results, but they were more conservative. Naive Bayes had the highest spam precision, which meant that when it predicted spam, it was usually correct. The downside was that it missed more spam emails. SVM performed slightly better than Naive Bayes overall, but it still did not match the CNN, especially in recall.

In the end, the CNN was the best model for this study. It gave the strongest balance between detecting spam and avoiding unnecessary false alarms. The results suggest that deep learning, combined with effective text preprocessing and pre-trained word embeddings, can be a strong approach for spam email classification.

6. REFERENCES

Kim, Y. (2014). Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746-1751). Association for Computational Linguistics. doi:10.3115/v1/D14-1181

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532-1543). Association for Computational Linguistics. doi:10.3115/v1/D14-1162

Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: Review, approaches and open research problems. *Heliyon, 5*(6), e01802. doi:10.1016/j.heliyon.2019.e01802

AbdulNabi, I., & Yaseen, Q. (2021). Spam email detection using deep learning techniques. *Procedia Computer Science, 184*, 853-858. doi:10.1016/j.procs.2021.03.107

Hakami, H. (2025). Automatic classification of mobile apps to ensure safe usage for adolescents. *PLOS ONE, 20*(1), e0313953. https://doi.org/10.1371/journal.pone.0313953