Effectiveness of Machine Learning Models for Real-Time Spam Detection in Cellular Networks

SIT327 - Network Forensics

Date
Trimester 1 - 2024

Brianna Laird

Student ID s218175884

HD Report

Contents

1 List of Abbreviations

- **ML** - Machine Learning
- **DL** - Deep Learning
- **AI** - Artificial Intelligence
- **SMS** - Short Message Service
- **MMS** - Multimedia Messaging Service
- **ACSC** - Australian Cyber Security Centre
- **ASD** - Australian Signals Directorate
- **NLP** - Natural Language Processing
- **SVM** - Support Vector Machine
- **RF** - Random Forest
- **KNN** - K-Nearest Neighbours
- **LR** - Logistic Regression
- **NB** - Naive Bayes
- **NN** - Neural Network
- **LSTM** - Long Short-Term Memory
- **CNNs** - Convolutional Neural Networks
- **RAN** - Radio Access Network
- **C-RAN** - Cloud Radio Access Network
- **RRH** - Remote Radio Head
- **BBU** - Baseband Unit
- **FN** - False Negative
- **FP** - False Positive
- **TN** - True Negative
- **TP** - True Positive

2 Research Proposal

2.1 Introduction

With the growing number mobile devices, SMS remains a vital communication tool not only due to its reliability but also due to its accessibility without the need for internet connectivity. However, this widespread use has led to a significant increase in spam messages, which are unsolicited and often sent in bulk for commercial benefits or fraudulent purposes. The impact of spam SMS includes annoyance, potential financial loss, and potential identity theft. This proposal outlines a project to develop and test ML models for detecting spam SMS and introduces a novel approach of simulating these models within a SimPy-based network environment.

2.2 Background

The ease and low cost of sending bulk messages have exacerbated the issue of spam SMS, infringing on user privacy and security. Current measures are insufficient, highlighting the need for advanced spam detection systems leveraging ML. This project will utilise multiple datasets to predict and identify spam effectively, enhancing network forensics and security. A vast majority of current research focuses solely on creating a ML model to detect spam messages, but not actually testing within a discrete network simulation.

2.3 Objectives

Primary Objective

- To develop and evaluate multiple ML models for spam SMS detection.
- To simulate these models in a SimPy-based mobile network, testing their real-time effectiveness.
- To create an extensive report on the the accuracy of various models in detecting spam SMS.

Secondary Objective

- To document and analyse the prevalence and types of spam SMS.
- To provide insights into the history of spam SMS trends within Australia over the past five years.
- To explore the potential integration of successful models into live mobile networks.

2.4 Literature Review

Spam SMS detection has evolved significantly, with numerous ML techniques being deployed to tackle this persistent problem. Traditional methods such as LR, NB, and SVM are well-documented for their efficacy in binary classification tasks, including spam detection. Alzahrani and Rawat [1] have bench marked these techniques, highlighting their robustness in spam classification under varied datasets.

The advent of DL has introduced more sophisticated approaches like LSTM networks, which excel in text classification by learning long-term dependencies within data [2]. They demonstrated that LSTM models significantly outperform traditional methods, achieving an accuracy of 98.5% on a comprehensive UCI dataset.

Ensemble methods combining multiple ML algorithms have shown promise in enhancing classification accuracy and robustness. Hosseinpour and Shakibian [3] applied an ensemble of logistic regression and random forest to address data imbalance issues, yielding improved performance in spam detection tasks. This approach shows the potential of hybrid models in achieving superior predictive performance in real-world applications.

Dharani et al. [4], Jain and Mahadev [5], Jain et al. [6], Nagare et al. [7] all explore the use of ML models in spam SMS detection, with varying degrees of success. However, none of these studies have explored the use of a network simulation to test the effectiveness of these models in a real-world environment.

2.5 Methodology

Data Collection:

- The project will involve collecting multiple datasets containing labelled SMS messages, classified as either 'spam' or 'ham' (non-spam). This dataset will be compiled from publicly available sources. The project will also explore the prevalence and types of spam SMS in circulation to provide insights into the nature of spam messages.

Data Preprocessing:

- The data will undergo preprocessing to suit ML applications, including tokenisation, stop word removal, and vectorisation using TF-IDF.

Model Selection and Training:

- The project will evaluate various ML models, with selections made based on accuracy, precision, and recall metrics. In addition to how the models act within a simulated network environment.

Simulation and Testing:

- Using SimPy, the selected models will be tested within a simulated mobile network to assess performance and real-time applicability.

2.6 Expected Outcomes

The project aims to develop a high-accuracy ML model for spam SMS detection, with successful models tested in a simulated network environment. Insights into spam SMS prevalence and model effectiveness will be documented, advancing knowledge in both spam detection and simulation methodologies.

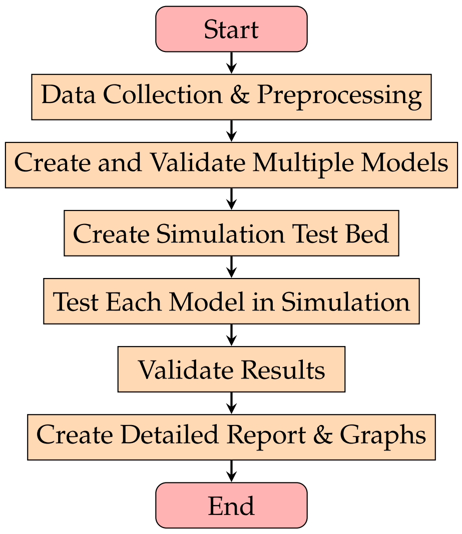
2.7 Future Work and opportunities

The project's outcomes will have implications for enhancing network security and forensics operations. Future work could involve expanding the model to detect other forms of spam, such as email spam, or developing a real-time spam detection system. It could also be extended to a small application where users can enter a message and the model will predict if it is spam or not.

2.8 Conclusion

This project will leverage advanced ML techniques and a novel simulation approach to enhance spam SMS detection, providing significant contributions to network security and ML applications in telecommunications.

Methodology Flowchart


Start
Data Collection & Preprocessing
Create and Validate Multiple Models
Create Simulation Test Bed
Test Each Model in Simulation
Validate Results
Create Detailed Report & Graphs
End

3 Introduction

In today's interconnected world, spam SMS continues to plague mobile users, posing not only a nuisance but also a serious cybersecurity threat. Traditional methods for combating this such as rule-based filters, often fall short against the evolving tactics of cyber criminals. Machine learning models present a promising solution due to their capacity to adapt and learn from continuous data streams, potentially outsmarting sophisticated spamming tactics. While the effectiveness of ML in spam detection is well-documented, its practical implementation, particularly within simulated network environments or alert systems, remains largely unexplored.

This research seeks to bridge this gap by conducting a comprehensive evaluation of various ML models within a simulated network setup, focusing on their real-time effectiveness in spam detection. By implementing these models directly within the BBU of the network, this study not only aims to enhance the efficiency of spam detection but also to significantly reduce the computational load on core network infrastructures.

Additionally, the integration of an alert system within this framework promises to change the utility of spam detection mechanisms. This system is designed to notify law enforcement agencies of unusual spikes in spam activity, potentially pinpointing the origins of such threats. By deploying ML models at the edge of the network, this research shows their dual role in both identifying spam and boosting wider forensic and law enforcement efforts.

Through this approach, the research aims to provide new insights into the capabilities of ML models in the context of modern telecommunications, offering a dual benefit of advanced spam detection and proactive cyber crime prevention.

3.1 Contributions

The contributions of this report are outlined as followed:

1. **Comprehensive Evaluation of Multiple ML Models:**
   - This report provides an extensive evaluation of various ml models including LR, KNN, NB, SVM and RF.
   - It details their effectiveness in detecting spam SMS, offering key insights prior to deployment in a simulated environment.
2. **Performance Analysis in a Simulated Environment:**
   - We analyse the performance of each ML model within a controlled simulated environment as messages transit across the network. The models are tested against both the training dataset and newly generated datasets to gauge robustness and adaptability.
3. **Integration of an Alert System for Spam Detection:**
   - An alert system is integrated within the simulation to enhance the functionality of the ML models. It triggers an alert when more than 20% of the last 100 messages are detected as spam, recording the location of the base-band unit, the exact time, and the spam ratio.
4. **Location-Based Detection for Real-World Application:**
   - The simulation explores the practical application of embedding ML models within base-band units for real-time spam detection in live networks. This approach aids in promptly identifying potential locations where cyber criminal activities might be occurring.
   - This also highlights the use of placing the ML model towards the edge of the network to ease computational demand on the core network.

3.2 Structure

The report structure is divided into 3 main sections:

Section 1

- The first section discusses the spam SMS problem within Australia as outlined in Scamwatch, ASD and ACSC reports as well as from Telstra. This section also discusses the current research around different approaches for using ML for spam sms detection. As well as the applications, benefits and challenges of the different approaches.

Section 2

- This section discusses the different ML models that will be evaluated within this report, being LR, NB, KNN, RF and SVM. This section also discusses the official data set used for the evaluation, and then goes over the results of each model after it was built.

Section 3

- Here the network simulation model is discussed, the structure and its purpose. The results of the simulation are also discussed including the alert and logging system, as well as the performance of each model within the simulated environment. This section also discusses the findings of the models and the open challenges.

3.3 The Spam SMS Problem

Spam SMS is a global issue, serving as both a nuisance and a way for scams and malware, with significant economic costs to both consumers and businesses. The ease of sending bulk messages due to technological advancements has intensified this issue. Statistics from the first 3 months of 2024 highlight the severity: approximately 25,000 spam SMS reports with $475k in losses[8]. This is an increase from 2023's 110,000 complaints and $27 million in damages[9]. The consistent financial toll from previous years[10, 11] shows the need for robust spam SMS detection systems. The statistics of the past 4 years are shown in figures: 1, 2, 3 and in 4.

A significant challenge in combating spam SMS is the under reporting of incidents. Many recipients overlook the importance of reporting, thereby obscuring the true scale of the problem and hindering targeted interventions. The need for Enhancing reporting mechanisms and increasing public awareness about the importance of reporting spam is critical to addressing this issue.

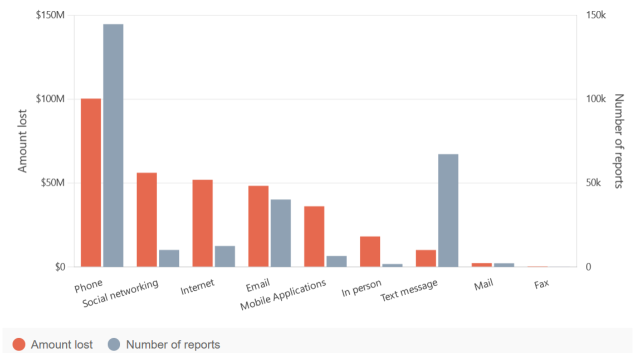


Figure 1: Reports vs. Loss - 2021 [11]

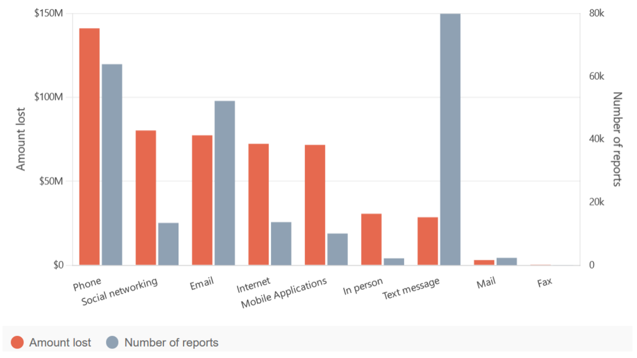


Figure 2: Reports vs. Loss - 2022 [10]

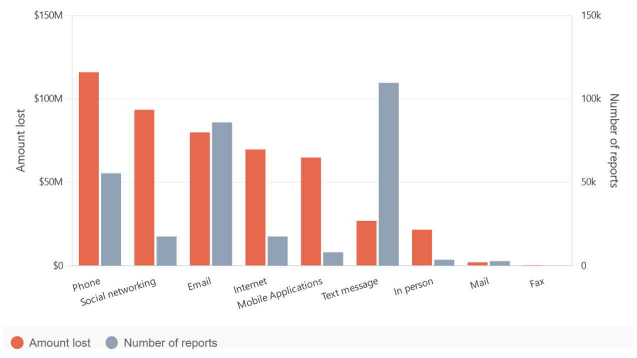


Figure 3: Reports vs. Loss - 2023 [9]

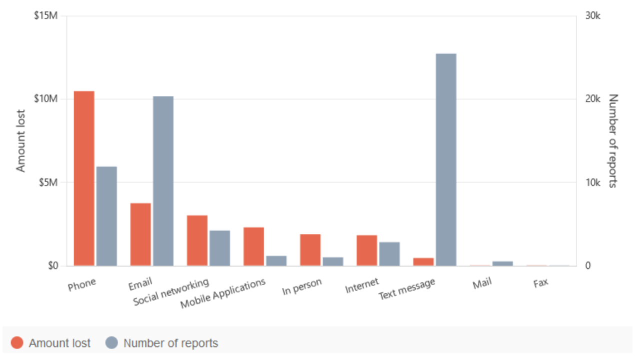


Figure 4: Reports vs. Loss - 2024 Jan/Feb [8]

In 2023, Telstra introduced a new reporting number, 7226, allowing customers to forward scam SMS and MMS messages directly for analysis. This initiative has been a significant step in combating the rise of sophisticated scams. Within a few months, over 250,000 messages were reported, providing crucial data to improve scam prevention measures. This was later integrated into the report spam button on apple messages, making an easier way for people to report. The service underscores the urgency of timely scam reporting, as the effectiveness of combating these scams depends heavily on the speed of reports receivedTelstra [12].

Statistical data from various telecommunications regulators illustrate the scale of the problem. For instance, reports indicate that billions of spam messages are sent monthly. Additionally, the adaptive strategies of spammers, who continually evolve their methods to bypass detection, complicate efforts to curb these messages effectively.

Government Cyber Threat Reports

The ACSC in conjunction with ASD has released several reports highlighting the growing threat of spam SMS and the broader landscape of cyber crime. The 2019-20 report[13], 2020-21 report[14], 2021-22 report[15], and the 2022-23 report[16] provide valuable insights into the evolving tactics of cyber criminals and the impact of spam SMS on various sectors. Key findings from these reports include:

Incident Reports: The number of reports recorded by the ACSC has shown a consistent increase each year:

- **2019-20:** Over 59,000 cyber crime reports.
- **2020-21:** Over 67,500 cyber crime reports, marking a 13% increase from the previous year.
- **2021-22:** Over 76,000 reports, another 13% increase from the previous year.
- **2022-23:** Nearly 94,000 reports, indicating continued growth in reported incidents.

This rise includes a notable increase in spam SMS activities, which are frequently integrated into broader phishing strategies. The growing trend reflects both an increase in cyber criminal activities and an improved reporting culture due to heightened awareness and reporting mechanisms. Across the reports, phishing, particularly via SMS, has been persistently utilised to exploit topical events. The COVID-19 pandemic was specifically exploited in multiple campaigns, with cyber criminals crafting messages that mimicked official communications to solicit personal data or distribute malicious links.

Evolution of Techniques and Sector-Specific Impacts

Sophistication in Spam SMS Techniques: There has been a clear evolution in the sophistication of spam SMS tactics. Cyber criminals have utilised emerging news and current events to craft messages that mimic authoritative sources, increasing the perceived legitimacy of their scams.

Sector Vulnerabilities: The reports highlight the effectiveness of these phishing campaigns across various sectors, particularly healthcare and critical infrastructure. The urgency and personal nature of SMS communications have made them a preferred vector for attacks aimed at extracting sensitive information or delivering ransomware.

Trends in Response and Preparedness

The trends over the past 4 years show the need for adaptive and responsive cybersecurity measures to counter the evolving threat landscape. The reports emphasise the importance of public awareness campaigns, cross-sector collaboration, and technological advancements to mitigate the risks posed by spam SMS and other cyber threats.

Trends in 2019-2020

- **Exploitation of Current Events:** Rapid adaptation to COVID-19 with pandemic-themed spam.
- **Sector Impact:** Major targeting of government and healthcare sectors via spam SMS and emails.

Trends in 2020-2021

- **Exploitation of Current Events:** Continued COVID-19 themed spam, with more sophisticated scams.
- **Sector Impact:** Expanding impact to finance and critical infrastructure.

Trends in 2021-2022

- **Exploitation of Current Events:** Diversified strategies targeting vaccine rollouts and aid programs.
- **Sector Impact:** Rise in SMS-based fraud in finance; persistent threats to healthcare and government.

Trends in 2022-2023

- **Exploitation of Current Events:** Use of geopolitical tensions and natural disasters in spam SMS.
- **Sector Impact:** Expanded attacks into education and retail sectors, showing pervasive threat nature.

Governmental and Industry Responses

Collaborative Efforts to Combat Phishing SMS: Initiatives involving major telecoms like Telstra and Optus have enhanced efforts to block phishing SMS impersonating official services, improving public safety.

Strategic Implications and Preventive Measures:

- **Adaptive Security Measures:** The dynamic nature of spam SMS threats requires ongoing updates to strategies and the implementation of advanced detection systems.
- **Future Preventive Strategies:** Emphasis is placed on public education about spam SMS and strengthening collaborations between governments and the private sector.
- **Policy and Strategic Recommendations:** Future policies should focus on enhancing public awareness and technical defences, with strategic partnerships essential for effectively combating spam SMS.

4 Review of current research

There has been an increase in spam SMS incidents, presenting significant security challenges and user disruptions. The limitations of conventional spam filters, which often fail to keep pace with the sophisticated tactics employed by spammers, show the need for more adaptable and robust mechanisms. ML offers a pathway to these advancements by dynamically distinguishing between legitimate and fraudulent messages.

Current research on ML in spam SMS detection has explored various models, each demonstrating distinct levels of accuracy and processing efficiency. This review goes into how different ML techniques have been applied, analysing their effectiveness in real-world applications. By examining these studies and performance metrics, this section aims to highlight the strengths and limitations of these models, setting the stage for their potential integration within simulated network environments.

4.1 Review of Recent Work

SVMs are consistently recognised for their robust performance in complex scenarios that involve high-dimensional data and non-linear decision boundaries. Jain et al. [6] and Salman et al. [17] showcase SVM's effectiveness in navigating the multifaceted spaces typical of SMS spam detection, characterised by a range of spam indicators and linguistic patterns. The ability of SVMs to maintain accuracy despite these complexities makes them a preferred choice for handling intricate data structures.

In contrast, NB classifiers, while excellent for processing multilingual text, require extensive feature engineering to perform robustly. Their adaptability is crucial in the global landscape of spam threats, dynamically adjusting their probabilistic models to the input texts from various languages, as noted by Sharma and Sharaff [18] and Aparna and Halder [19]. This flexibility makes NB classifiers particularly valuable in scenarios that demand rapid processing across diverse demographic settings.

RF are highlighted for their capacity to prevent over-fitting, a common challenge in spam detection. The ensemble approach of RF, integrating outputs from numerous decision trees, offers a strong defence against the over-fitting that can impair simpler models, especially in noisy data environments. This not only enhances the robustness of the model but also ensures consistency across the varied and complex feature sets often encountered in spam detection tasks [5].

However, methods like KNN face challenges when scaled to larger datasets, as detailed by Alzahrani and Rawat [1]. While KNN is straightforward and effective for smaller datasets, it struggles under the weight of increased computational demands and dimensionality, which significantly effects its performance as the size of the dataset increases. This contrast shows the diverse capabilities and limitations of different ML models in the context of spam detection.

The implementation of DL techniques, such as CNNs and LSTM networks, enhance spam detection. CNNs are good at capturing spatial hierarchies in text data, which proves invaluable in identifying subtly crafted spam messages that might elude less sophisticated models [20, 21]. LSTM networks, with their ability to process text sequences while preserving contextual information, achieve exceptional accuracy levels, managing an accuracy rate of 98.5% by Gadde et al. [2]. These models are great for their ability to handle sequence data where the temporal dynamics of words significantly influence message semantics.

A common trend is noticed in recent research, where the focus is on the models and how the perform on their own. Identifying a gap in the application of these models within a simulated or even live environment to assess their true capabilities or weaknesses when put to the test.

Emerging approaches in the integration of ML with other areas, such as smartphone forensics, have also shown promising results. Nagare et al. [7] and Peng et al. [22] explore how ML models can assist in filtering out spam messages during forensic analysis, thus reducing the data volume investigators need to manually process. This not only streamlines forensic workflows but also leverages the capabilities of ML to enhance the overall efficiency of investigations in digital environments.

Finally, the combined use of ensemble learning methods, which leverage the strengths of multiple algorithms, reflects a growing trend in the field. By integrating models like RF and LR, researchers have developed systems that surpass the performance of individual models, especially in unbalanced datasets where spam messages are outnumbered by legitimate communications [3].

In summary, the recent works in SMS spam detection highlights a clear trend towards more sophisticated, integrated, and adaptive ML strategies. These developments not only cater to the complex nature of spam messages but also enhance the ability of detection systems to scale and adapt in response to new challenges and emerging spamming techniques. In the table 1 the papers are compared.

Reference	Method Employed	Key Advantages and Application Scenarios
Jain et al. [6]	SVM	Excellent in handling high-dimensional data and non-linear decision boundaries; best for complex spam detection scenarios.
Jain and Mahadev [5]	RF	Prevents overfitting through ensemble approach; robust against noisy data, ideal for diverse and complex datasets.
Alzahrani and Rawat [1]	KNN	Simple and effective for small datasets but struggles with scalability and computational cost.
Sharma and Sharaff [18], Aparna and Halder [19]	NB	Adapts well to multilingual data, adjusting probability estimates based on linguistic features; suitable for global spam challenges.
Popovac et al. [20], Sethi et al. [21]	CNN	Captures spatial hierarchy in text, excellent for nuanced spam messages; computationally intensive but high performance.
Gadde et al. [2]	LSTM	Processes text sequences with high accuracy, preserving context; ideal for sequence data where context is crucial.
Hosseinpour and Shakibian [3]	Ensemble (RF & LR)	Combines strengths of multiple algorithms to handle unbalanced datasets effectively; enhances predictive performance.
Nagare et al. [7], Peng et al. [22]	Smartphone Forensics	Enhances forensic analysis by filtering spam, reducing manual data processing; applies ml to practical forensic challenges.
Sharma and Sharaff [18]	Genetic Programming	Evolves regex to anticipate and counteract new spamming techniques; proactive and adaptive to spam evolution.

Table 1: Comparative Analysis of ml Approaches for SMS Spam Detection

4.2 Conclusion

In conclusion, SMS spam detection has made significant strides through the application of various ML and DL techniques. This review demonstrates a robust response to the complexities inherent in spam messages, addressing issues such as high-dimensional data, over-fitting, computational efficiency, and the challenges of multilingual and evolving spam tactics. While this analysis shows that no single model is superior across all aspects, the choice of model heavily depends on specific factors such as dataset complexity and computational resources.

The proposed solution in this report not only aligns with the latest advancements but also introduces a novel integration of an alert system with ML models for spam detection. The unique aspect of this project lies in its focus on real-time threat identification and communication with law enforcement agencies. By developing an adaptive alert system that triggers notifications based on predefined spam activity thresholds, this research aims to enhance the proactive capabilities of spam detection systems.

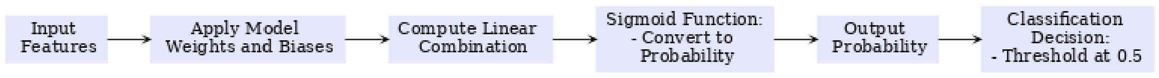
This proposed system is designed to be integrated within a simulated network environment, providing a realistic assessment of the model's performance and the system's operational efficacy. Such an approach is not commonly explored in existing literature.

4.3 The ML Models for Evaluation

The models used for spam detection leverage data sets to identify patterns and features that distinguish spam from legitimate messages. Commonly used ML techniques for spam SMS detection include LR, NB, KNN, RF, and SVMs. Each model has unique strengths and limitations, making them suitable for different use cases.

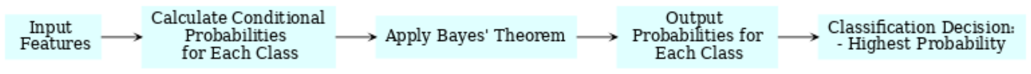
Logistic Regression

- LR is a popular ML technique for binary classification tasks, making it well-suited for spam SMS detection. By modelling the relationship between input features and the probability of a message being spam, LR can effectively distinguish between spam and legitimate messages.


Input Features
Apply Model Weights and Biases
Compute Linear Combination
Sigmoid Function: - Convert to Probability
Output Probability
Classification Decision: - Threshold at 0.5
Figure 5: Logistic Regression Model

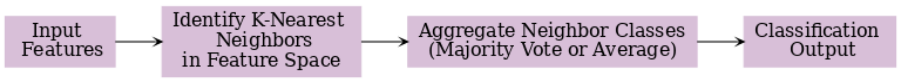
Naive Bayes

- NB is a probabilistic ML technique based on Bayes' Theorem, making it well-suited for spam SMS detection. By assuming that features are conditionally independent given the class label, NB can efficiently model the probability of a message being spam. While NB is computationally efficient and robust to noise, its assumption of feature independence may limit its performance on highly correlated features.


Input Features
Calculate Conditional Probabilities for Each Class
Apply Bayes' Theorem
Output Probabilities for Each Class
Classification Decision: - Highest Probability
Figure 6: Naive Bayes Model

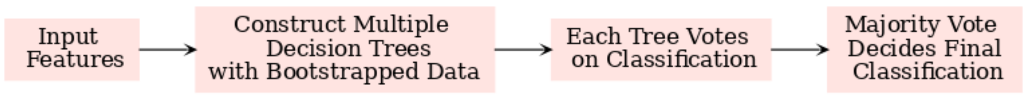
K-Nearest Neighbours

- KNN is a simple ML technique that classifies messages based on the majority class of their KNNs. KNN is non-parametric and does not make strong assumptions about the underlying data distribution, making it suitable for spam SMS detection. However, KNN may struggle with high-dimensional data and large datasets, as it requires computing distances between all data points.


Input Features
Identify K-Nearest Neighbors in Feature Space
Aggregate Neighbor Classes (Majority Vote or Average)
Classification Output
Figure 7: K-Nearest Neighbours Model

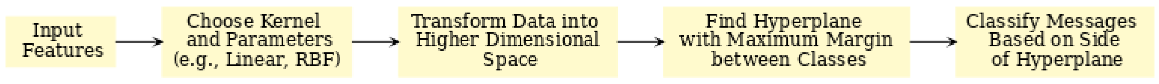
Random Forest

- RT is an ensemble ML technique that combines multiple DTs to improve classification accuracy. By aggregating the predictions of individual trees, RF can reduce over-fitting and improve generalisation performance. RF is robust to noise and outliers, making it well-suited for spam SMS detection.


Input Features
Construct Multiple Decision Trees with Bootstrapped Data
Each Tree Votes on Classification
Majority Vote Decides Final Classification
Figure 8: Random Forest Model

Support Vector Machines

- SVMs are ideal for binary classification tasks like spam SMS detection due to their ability to determine the optimal hyperplane for separating categories. These models excel at managing complex decision boundaries and are robust against overfitting, even in high-dimensional data. However, their performance heavily depends on the choice of kernel function and hyperparameter tuning.


Input Features
Choose Kernel and Parameters (e.g., Linear, RBF)
Transform Data into Higher Dimensional Space
Find Hyperplane with Maximum Margin between Classes
Classify Messages Based on Side of Hyperplane
Figure 9: Support Vector Machine Model

5 Evaluating the Different Machine Learning Models

The dataset was sourced from the UCI ML Repository [23]. This dataset has 5,574 messages from four sources, each message labelled as either ham or spam. To ensure consistency and relevance for the ML modelling, the dataset underwent significant pre-processing and cleaning, which involved the removal of irrelevant information and normalisation of message formats.

5.1 Examining The Dataset

A closer analysis of the dataset shows that it consists of 13.51% spam messages, totalling 747, and 86.49% ham messages, amounting to 4,827. The analysis reveals distinct patterns as shown in the figures 11 and 12.

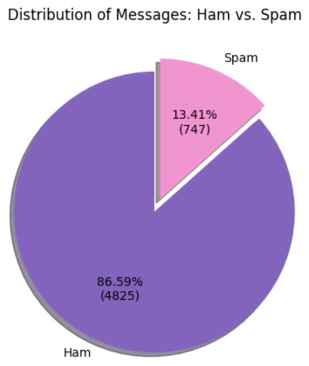


Figure 10: Dataset: Spam vs Ham

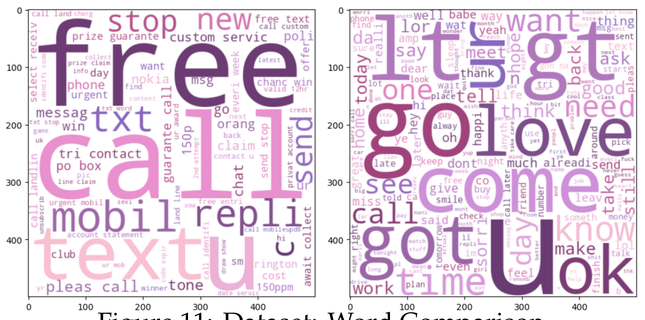


Figure 11: Dataset: Word Comparison

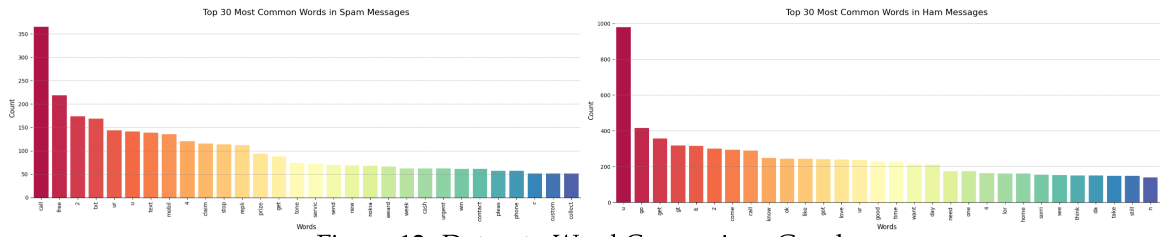


Figure 12: Dataset - Word Comparison Graph

5.2 Machine Learning Models Development

Data Preprocessing

- The code in the appendix A.1 was used to process the dataset for use in the ML models. It tokenised the messages, removed stop words, and applied lemmatisation to normalise the text. These steps were taken:

Algorithm 1 Text Pre-processing for ML Models

```
1: Clean Data: Rename columns and map 'ham' to 0 and 'spam' to 1 in the 'target' column.
2: Pre-processing Steps:
3: Replace email addresses, URLs, money symbols, phone numbers, and numbers
4: Tokenisation: Split text into words
5: Remove Stopwords: Filter out stopwords from the tokens
6: Stemming: Apply stemming to the words
7: Vectorisation: Convert the cleaned and processed text into a vector of token counts
8: Encode Target: Encode the target labels into numerical format
```

Logistic Regression

- The LR model A.2 demonstrated consistency with a 99% accuracy rate in both cross-validation and on the test set, across 2229 samples. With only 8 FN and 16 FP, the model's precision and recall across classes resulted in a weighted average F1-score of 0.99.

```
Best parameters: {'logreg__penalty': 'l2', 'logreg__C':
↪  0.0006551285568595509}
Best cross-validation score: 0.99
Test set accuracy: 0.99
Confusion Matrix:
 [[1927    8]
 [  16  278]]
Classification Report:
               precision    recall  f1-score   support

           0       0.99      1.00      0.99      1935
           1       0.97      0.95      0.96       294

    accuracy                           0.99      2229
   macro avg       0.98      0.97      0.98      2229
weighted avg       0.99      0.99      0.99      2229
```

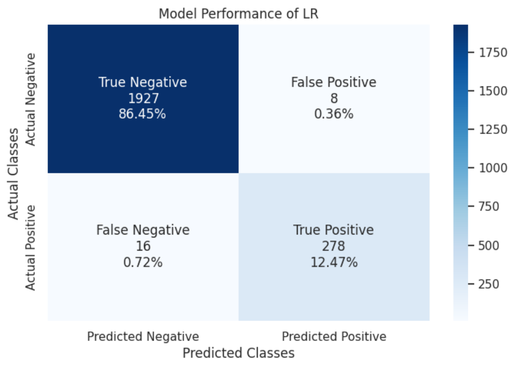


Figure 13: LR Model Evaluation Results

Algorithm 2 Logistic Regression with the Complexity: $O(d \cdot n \cdot \textbf{iterations})$

```
1: Set Up Pipeline: Create a pipeline with a standard scaler and logistic regression.
2: Define Parameters: Specify the hyper parameter space for LR regularisation strength and penalty.
3: Configure Cross-Validation: Set up stratified 5-fold cross-validation.
4: Initialise Random Search: Prepare search CV with 10 iterations, using accuracy as the scoring metric.
5: Fit Model: Fit the model using the training data.
6: Predict and Evaluate: Make predictions on the test set output the evaluation.
```

Naive Bayes

- The NB model A.3 showed good performance with an 89% accuracy on the test set. With 224 FP and 30 FN, it maintained decent precision, particularly for ham. However, it struggled more with identifying spam, as reflected in its weighted average F1-score of 0.90.

```
Test set accuracy: 0.89
Confusion Matrix:
 [[1711  224]
 [  30  264]]
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.88      0.93      1935
           1       0.54      0.90      0.68       294

    accuracy                           0.89      2229
   macro avg       0.76      0.89      0.80      2229
weighted avg       0.92      0.89      0.90      2229
```

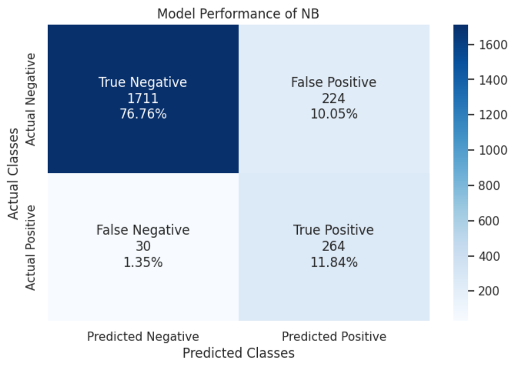


Figure 14: NB Model Evaluation Results

Algorithm 3 Gaussian Naive Bayes with the Complexity: $O(nd)$

```
1: Initialise Model: Create a Gaussian Naive Bayes model.
2: Fit Model: Fit the model using the training data.
3: Predict and Evaluate: Make predictions on the test set output the evaluation.
```

K-Nearest Neighbours

- The KNN model A.4 achieved a 92% accuracy in cross-validation, which slightly increased to 93% on the test set. It had 161 FN, while it still maintained a high precision for ham and a reasonable overall weighted average F1-score of 0.91.

```
Best parameters: {'knn__n_neighbors': 1, 'knn__weights':
↪  'uniform'}
Best cross-validation score: 0.92
Test set accuracy: 0.93
Confusion Matrix:
 [[1932    3]
 [ 161  133]]
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      1935
           1       0.98      0.45      0.62       294

    accuracy                           0.93      2229
   macro avg       0.95      0.73      0.79      2229
weighted avg       0.93      0.93      0.91      2229
```

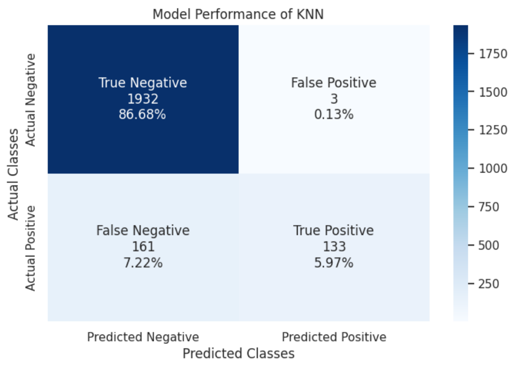


Figure 15: KNN Model Evaluation Results

```
Algorithm 4 K-Nearest Neighbours with the Complexity: $O(n^2dk)$
1: Set Up Pipeline: Create a pipeline with a standard scaler and KNN classifier.
2: Define Parameter Grid: Set up the parameter grid for KNN.
3: Initialise Grid Search: Configure Grid Search CV with 5-fold cross-validation.
4: Fit Model: Fit the model using the training data.
5: Predict and Evaluate: Make predictions on the test set output the evaluation.
```

Random Forest

- The RF model A.5 achieved a cross-validation accuracy of 95% and maintained 94% on the test set. With 67 FN and 19 FP. The model's ability was reflected in a weighted average F1-score of 0.95.

```
Best parameters: {'max_depth': None, 'min_samples_leaf':
↪  1, 'min_samples_split': 5, 'n_estimators': 100}
Best cross-validation score: 0.97
Test set accuracy: 0.98
Confusion Matrix:
 [[1932    3]
 [  48  246]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1935
           1       0.99      0.84      0.91       294

    accuracy                           0.98      2229
   macro avg       0.98      0.92      0.95      2229
weighted avg       0.98      0.98      0.98      2229
```

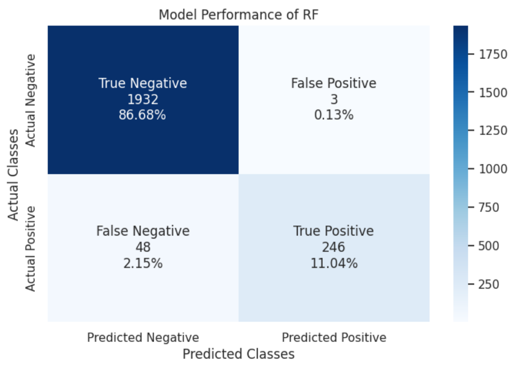


Figure 16: RF Model Evaluation Results

```
Algorithm 5 Random Forest with the Complexity: $O(knd \log n)$
1: Initialise Model: Create a Random Forest Classifier.
2: Define Parameter Grid: Set up the parameter grid for Random Forest.
3: Initialise Grid Search: Configure Grid Search CV with 3-fold cross-validation.
4: Fit Model: Fit the model using the training data.
5: Predict and Evaluate: Make predictions on the test set output the evaluation.
```

Support Vector Machine

- The SVM model A.6 demonstrated robustness with a cross-validation accuracy of 96% and 95% accuracy on the test set. It had 53 FN and 12 FP, high precision and slightly lower recall for spam resulted in a weighted average F1-score of 0.96.

```
Best parameters: {'svm__C': 0.1, 'svm__gamma': 'scale',
↪  'svm__kernel': 'linear'}
Best cross-validation score: 0.97
Test set accuracy: 0.96
Confusion Matrix:
 [[1930    5]
 [  79  215]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      1.00      0.98      1935
           1       0.98      0.73      0.84       294

    accuracy                           0.96      2229
   macro avg       0.97      0.86      0.91      2229
weighted avg       0.96      0.96      0.96      2229
```

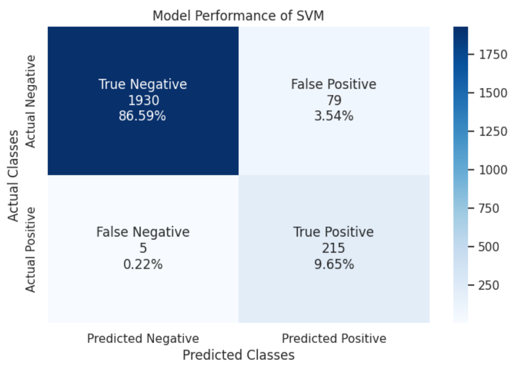


Figure 17: SVM Model Evaluation Results

```
Algorithm 6 Support Vector Machine with the Complexity: $O(n^2 d)$ to $O(n^3 d)$
1: Set Up Pipeline: Create a pipeline with a standard scaler and SVM.
2: Define Parameter Grid: Set up the parameter grid for SVM.
3: Initialise Grid Search: Configure Grid Search CV with 3-fold cross-validation.
4: Predict and Evaluate: Make predictions on the test set output the evaluation.
```

5.3 Findings

The LR model was the standout, demonstrating exceptional accuracy and robustness in classifying between classes with a perfect F1-score. The SVM model also displayed strong performance, with high precision and effective handling of both classes, resulting in a good F1-score. The RF model provided a balanced classification capability, though it encountered some challenges in spam detection, which slightly impacted its overall effectiveness. The data in 2 shows that the models generally perform well for spam detection, but reveals lower recall rates for spam SMS.

These findings show the importance of choosing the right model based on the specific needs of the data and the task. While LR showed the highest accuracy and least bias, models like SVM and RF provided strong alternatives with good generalisability. The NB and KNN models, despite their lower performance metrics, still do well in scenarios where the trade-offs between precision and recall are acceptable. This analysis shows the important role of model selection and hyperparameter tuning in achieving optimal classification results.

Algorithm	Ham Detection			Spam Detection			Overall Performance		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Accuracy	Precision	Recall
LR	0.98	0.61	0.94	0.88	1.00	0.94	0.90	0.91	0.90
NB	0.87	0.88	0.87	0.63	0.61	0.62	0.81	0.81	0.81
KNN	0.88	1.00	0.94	0.98	0.61	0.75	0.90	0.91	0.90
RF	0.83	0.41	0.58	1.00	0.41	0.58	0.85	0.87	0.85
SVM	0.81	1.00	0.89	0.96	0.31	0.47	0.82	0.85	0.82

Table 2: Comprehensive Result Comparison of ML Models for Spam Detection

6 Network Simulation Model

The network simulation model is implemented using Python and the simpy library, designed to emulate the operational dynamics of a C-RAN 5G architecture. The model simulates various network components including RRHs, BBUs, and Core Networks. The purpose of this simulation is to evaluate the effectiveness of ML models in a simulated environment, and their ability to work in an alert detection system. In addition, the ability to monitor locations for different spam levels to aid in law enforcement investigations is also evaluated. The simulation code can be viewed in the appendix B.

6.1 Model Structure

The structure of the simulation model consists of several interconnected components that emulate the data flow and processing within a 5G network:

- **Logging:** The simulation has a logging feature, the messages are logged as they go over the network by each device. Logging the location, classification, and alert logs for spam warning.
- **MobileDevice:** Represents a mobile device that sends SMS messages. It will send the messages from the dataset over the network.
- **RadioUnit:** Acts as the RRH, which receives messages from the MobileDevice and forwards them to the BBU. It introduces a random delay.
- **BasebandUnit:** The BBU processes messages with 10 BBUs in the simulation, each with its own dataset containing varied spam levels and a ML model for message classification.
- **CoreNetwork:** The Core Network acts as the main core the messages go through on their way.
- **NetworkController:** This component controls the flow of messages between the different network components. It assigns the path, in addition holds the evaluation metrics.

6.2 Simulation Procedure

The simulation procedure is outlined as follows:

1. Initialise the simulation model with necessary components and parameters including the number of BBUs and the additional datasets C.
2. Mobile Devices send messages at random intervals to the RRH, which forwards them to the BBU for processing. Here, the ML model classifies the messages as spam or non-spam.
3. Throughout the simulation, each network component logs processing times and classification outcomes. The BBU monitors spam levels and logs alerts when thresholds are exceeded.
4. The simulation runs until all messages are processed. Subsequently, each model's performance and the overall simulation results are evaluated and documented.

The simulation assesses the model's classification accuracy and its applicability in a real-world 5G network, focusing on factors such as processing delays and message transmission dynamics. The objective is to explore the benefits of deploying ML models within a 5G environment for enhanced spam detection and network security. This study also examines the advantages of situating these models at the network's edge, which potentially reduces the burden on the Core Network and supports real-time reporting to law enforcement when detecting spikes in spam activity from specific regions.

6.3 Alert and Logging System

An alert system was implemented, and two models notably succeeded in sending alerts when the spam threshold was exceeded. The NB model was the most active, sending a total of 797 alerts throughout the simulation. Following it, the LR model sent 183 alerts, demonstrating effective monitoring. In contrast, the other models such as KNN, RF, and SVM sent significantly fewer alerts, with counts of 5, 8, and 2 respectively.

It's important to note the LR model had a 90% accuracy 24, while the NB model had an 81% accuracy 25. This suggests that the LR model was more effective in detecting spam messages, meaning there could be false positive alerts within the NB model. However, the promising alert results show further re-training of the models could improving the alert detection system.

The figures 18 and the 19 show the results of the alert system for the NB and LR models respectively. They show the amount of alerts that occurred at each BBU throughout the simulation. While the two figures 20, and 21 show the timeline of the alert detection system for the NB and LR models. The timeline shows the frequency of alerts sent by the models over the course of the simulation. The NB model was more active in sending alerts, with a higher frequency compared to the LR model.

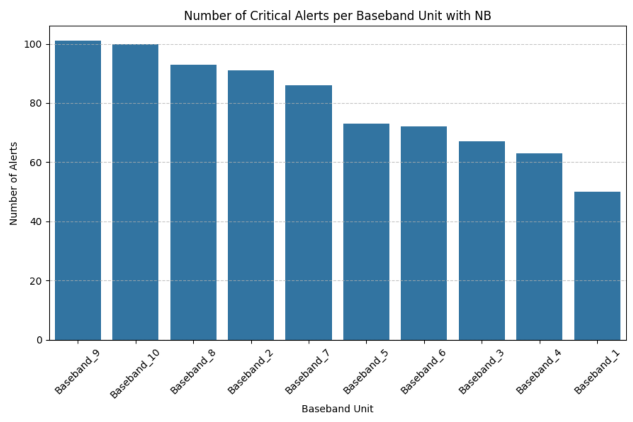


Figure 18: NB Model Alert System Results

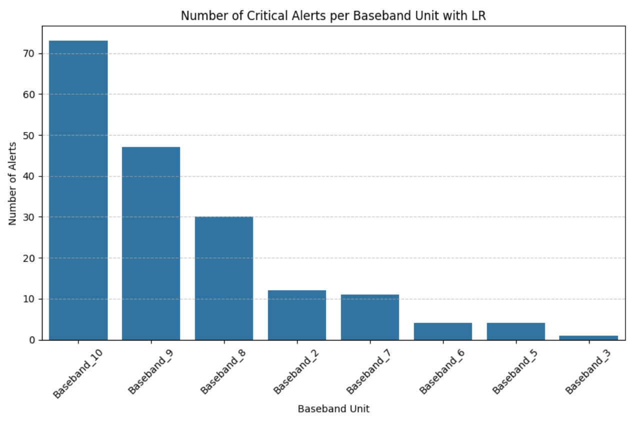


Figure 19: LR Model Alert System Results

The deployment of the alert system within the network infrastructure not only enhances spam detection but also holds significant potential for forensic applications and aiding law enforcement. By systematically logging alerts and their corresponding data and location, the system provides a detailed and actionable record that can be leveraged for investigative purposes. For instance, the frequent alerts from the NB model, detailed in Figures 18 and 20, illustrate not only the model's sensitivity to potential threats but also its utility in providing precise timestamps and locations where suspicious activities are concentrated. This can be crucial for identifying patterns of behaviour associated with cyber criminal activities, enabling authorities to respond more effectively. The integration of this alert system can facilitate real-time response capabilities, allowing law enforcement to intercept malicious activities at an early stage, thereby enhancing the overall security framework and reducing the impact of spam-related crimes.

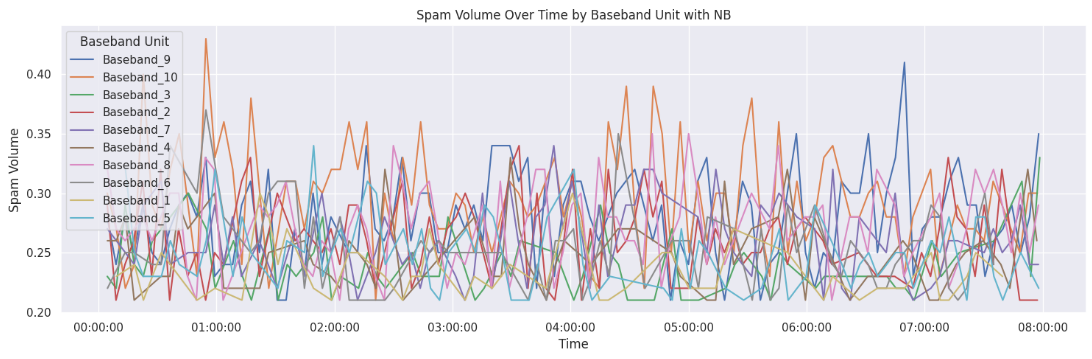


Figure 20: NB Simulation Timeline

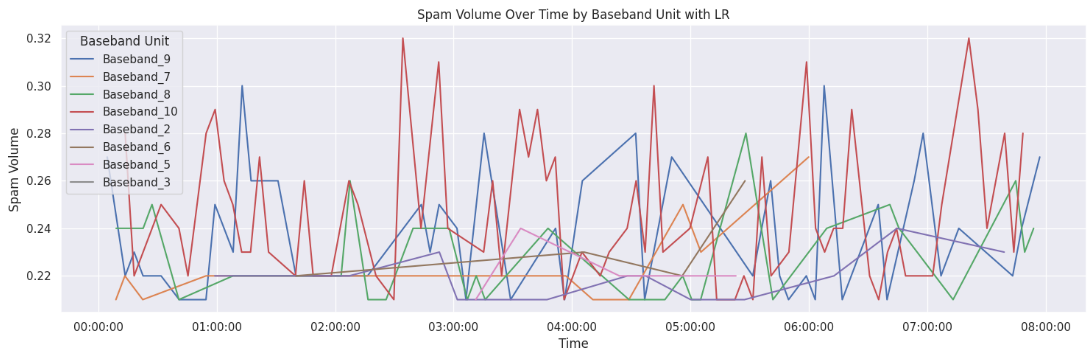


Figure 21: LR Simulation Timeline

The simulation was repeated multiple times using both the LR and NB models, altering the frequency of SMS messages sent in each instance. The results demonstrated consistent performance across all runs, with a similar volume of logs recorded for each BBU. This consistency highlights the effectiveness of the ML models in reliably sending alerts whenever spam SMS messages are detected.

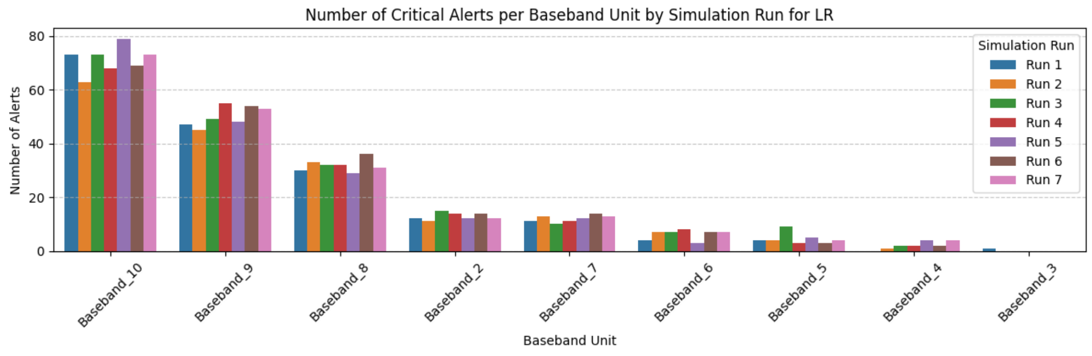


Figure 22: LR Simulation Run Times

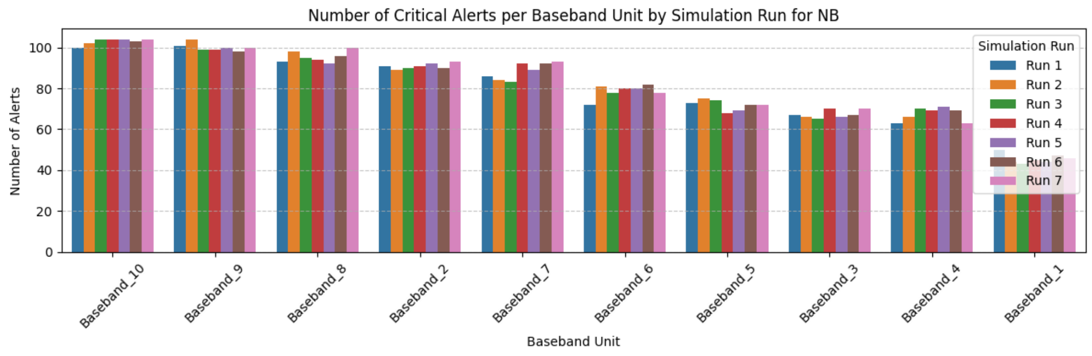


Figure 23: NB Simulation Run Time

6.4 Machine Learning Models Evaluation

Logistic Regression

The LR model demonstrated high accuracy with an accuracy of 90%, it effectively balanced precision and recall across the datasets. It sent 183 alerts when the spam threshold was exceeded.

```
Total Evaluation - Accuracy: 0.90
Confusion Matrix:
[[78701   299]
 [10514 16206]]
Classification Report:
              precision    recall  f1-score   support

         Ham       0.88      1.00      0.94     79000
        Spam       0.98      0.61      0.75     26720

    accuracy                           0.90    105720
   macro avg       0.93      0.80      0.84    105720
weighted avg       0.91      0.90      0.89    105720
```

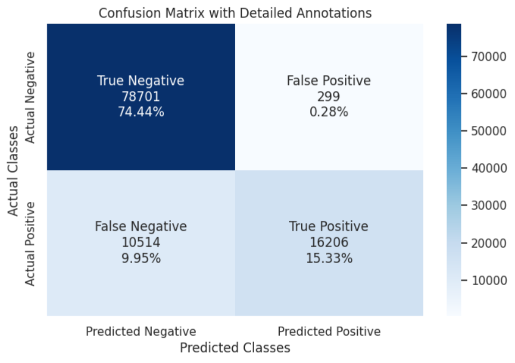


Figure 24: LR Model Evaluation Results

Naive Bayes

The NB model achieved an accuracy of 81%, displaying considerable activity in alert generation during the simulation. It sent 797 alerts when the spam threshold was exceeded.

```
Total Evaluation - Accuracy: 0.81
Confusion Matrix:
[[69588  9412]
 [10519 16201]]
Classification Report:
              precision    recall  f1-score   support

         Ham       0.87      0.88      0.87     79000
        Spam       0.63      0.61      0.62     26720

    accuracy                           0.81    105720
   macro avg       0.75      0.74      0.75    105720
weighted avg       0.81      0.81      0.81    105720
```

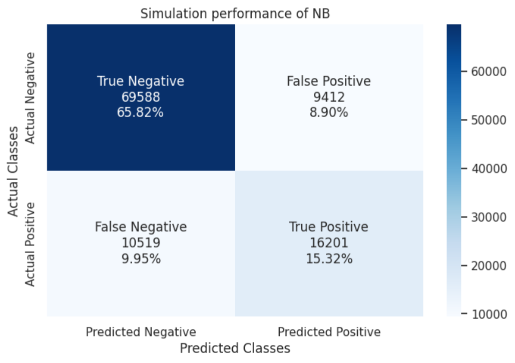


Figure 25: NB Model Evaluation Results

K-Nearest Neighbours

The KNN model showed its efficacy with a 90% accuracy rate in the simulation. The KNN model was less active in sending alerts, with only 5 alerts sent when the spam threshold was exceeded.

```
Total Evaluation - Accuracy: 0.90
Confusion Matrix:
[[78701   299]
 [10514 16206]]
Classification Report:
              precision    recall  f1-score   support

         Ham       0.88      1.00      0.94     79000
        Spam       0.98      0.61      0.75     26720

    accuracy                           0.90    105720
   macro avg       0.93      0.80      0.84    105720
weighted avg       0.91      0.90      0.89    105720
```

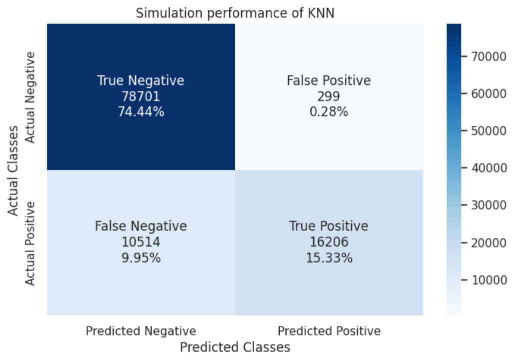


Figure 26: KNN Model Evaluation Results

Random Forest

The RF model achieved an 85% accuracy rate, excelling in precision with a perfect score in spam detection. The RF model was less active in sending alerts, with only 8 alerts sent.

```
Total Evaluation - Accuracy: 0.85
Confusion Matrix:
[[78954    46]
 [15855 10865]]
Classification Report:
              precision    recall  f1-score   support

         Ham       0.83      1.00      0.91     79000
        Spam       1.00      0.41      0.58     26720

    accuracy                           0.85    105720
   macro avg       0.91      0.70      0.74    105720
weighted avg       0.87      0.85      0.82    105720
```

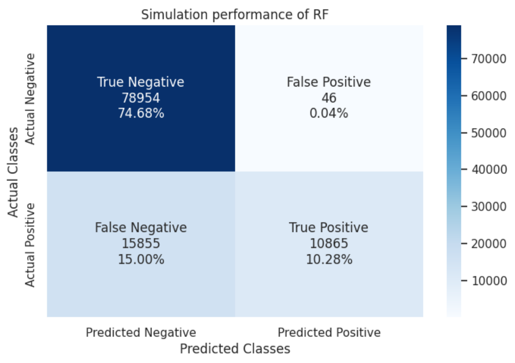


Figure 27: RF Model Evaluation Results

Support Vector Machine

The SVM had an accuracy rate of 82%. While it offers high precision, its low recall for spam at 31% highlights its conservative nature. The SVM model was least active, with only 2 alerts sent.

```
Total Evaluation - Accuracy: 0.82
Confusion Matrix:
[[78672   328]
 [18515  8205]]
Classification Report:
              precision    recall  f1-score   support

         Ham       0.81      1.00      0.89     79000
        Spam       0.96      0.31      0.47     26720

    accuracy                           0.82    105720
   macro avg       0.89      0.65      0.68    105720
weighted avg       0.85      0.82      0.78    105720
```

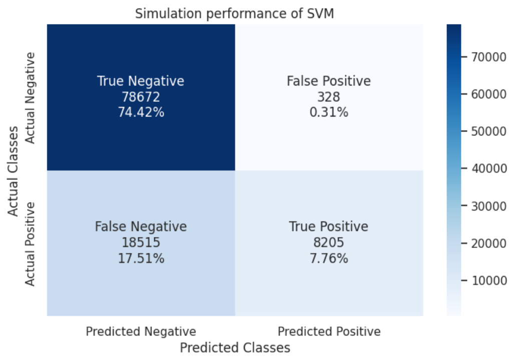


Figure 28: SVM Model Evaluation Results

6.5 Findings

The evaluation of multiple ML models within the simulated 5G network found significant insights into the operational effectiveness and suitability of these models for spam detection and network security enhancement. The following are key findings from the simulation:

- **Effectiveness in Spam Detection:** The LR and KNN models demonstrated high accuracy (90%) and were effective in distinguishing between ham and spam messages. Their robust performance suggests that these models are well-suited for environments requiring high reliability and minimal FP.
- **Alert Responsiveness:** The NB model was the most active in terms of alert generation, sending a total of 797 alerts. This indicates a high sensitivity to spam detection, although it also suggests a potential for higher FP rates, as reflected in its lower accuracy of 81%.
- **Precision and Recall Balance:** The RF model showed precision in spam detection (100%), but its lower recall rate (41%) indicates that while it is very accurate when it identifies spam, it frequently misses other spam messages. This could limit its applicability in scenarios where failing to detect spam could lead to significant security breaches.

- **Computational Efficiency:** Positioning the ML models at the edge of the network, specifically within the BBUs, effectively reduced the computational load on the Core Network. This setup not only enhanced the efficiency of message processing but also ensured quicker responsiveness in alerting and spam detection.
- **Forensic and Law Enforcement Utility:** The alert system, particularly with the high activity of the NB model, provided detailed logs that could be invaluable for forensic analysis. The data collected can help in identifying spam trends and potentially pinpointing sources of spam, aiding law enforcement in cyber crime investigations.
- **Model Adaptability:** The varying performance across different metrics for each model shows the importance of model selection based on specific network requirements and spam detection goals.

When comparing the table 3, the recall scores show a great improvement with only the NB staying the same. When comparing the results to the original table 2, they are also similar, but with a slight decline in LR. The biggest disparity was within the alert detection system with the amount of alerts varying heavily.

Algorithm	Ham Detection			Spam Detection			Overall Performance			
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Accuracy	Precision	Recall	Alerts
LR	0.88	1.00	0.94	0.98	0.61	0.75	0.90	0.91	0.90	183
NB	0.87	0.88	0.87	0.63	0.61	0.62	0.81	0.81	0.81	797
KNN	0.88	1.00	0.94	0.98	0.61	0.75	0.90	0.91	0.90	5
RF	0.83	1.00	0.91	1.00	0.41	0.58	0.85	0.87	0.85	8
SVM	0.81	1.00	0.89	0.96	0.31	0.47	0.82	0.85	0.82	2

Table 3: Comprehensive Result Comparison of ML Models for Spam Detection

The insights from this simulation show the dual advantages and obstacles associated with deploying sophisticated ML techniques in 5G network environments. Continuous refinement and calibration of these models are crucial to keep pace with dynamic network requirements and emerging security threats. Future work should aim at enhancing these models to boost their sensitivity without adversely affecting their accuracy.

6.6 Open Challenges

While the integration of ML models into 5G and beyond networks holds promise, several open challenges remain that must be addressed to fully realise their potential:

- **Model Scalability:** As network traffic volume grows, ensuring that ML models can scale effectively without a loss in performance is critical.
- **Dynamic Adaptation:** Networks constantly evolve with new devices and user behaviours. ML models must dynamically adapt to these changes to maintain effectiveness.
- **Interoperability:** Ensuring that ML models can operate seamlessly with different network technologies and standards is crucial for broad deployment.
- **Privacy Concerns:** Implementing ML solutions must not compromise user privacy. Models need to be designed with robust mechanisms to protect sensitive data.
- **Regulatory Compliance:** With varying global regulations on data and communications, ensuring that ML models comply with these legal frameworks is essential, they must remain ethical.
- **Threat Evolution:** Cyber threats are continuously evolving; ML models must be perpetually updated and tested against the latest threat vectors to remain effective.

Addressing these challenges will require coordinated efforts to ensure that the deployment of ML models enhances network security and efficiency without introducing new vulnerabilities or inefficiencies.

7 Conclusion

This research has explored the integration of ML techniques within a simulated network environment. The results demonstrate the potential of ML models to enhance network efficiency and spam detection capabilities for alert systems, while also shedding light on the inherent challenges such as scalability, adaptability, and privacy concerns.

Throughout the simulation, certain models like LR and NB proved highly effective in real-time spam alert detection, offering robust performance metrics that suggest their viability for real-world applications. However, the research also highlighted the critical need for ongoing optimisation to ensure these models can adapt to dynamic network conditions and evolving cyber threats.

The open challenges discussed show the complexities involved in deploying ML solutions across diverse and ever-changing network architectures. Addressing these challenges will not only require innovative technological advancements but also a collaborative approach involving network operators, cybersecurity experts, and policymakers to ensure that the deployment of these technologies aligns with global standards and regulations.

The potential for using ensemble methods and developing more adaptive ML algorithms presents a promising avenue for research. These developments could lead to more balanced performance across various conditions and further reduce the computational load on core network. Enhancing privacy-preserving features within these models will be essential to maintaining user trust and compliance with stringent data protection laws.

In conclusion, while the path forward is full of challenges, the benefits of integrating ML into 5G networks that range from improved spam detection to enhanced network management make it worthwhile. Continued research and will be essential in overcoming the obstacles and realising the full potential of ML in this exciting and rapidly evolving field.

A Machine Learning Models Code

A.1 Data Preprocessing

```
df = pd.read_csv('spam_data.csv', encoding='latin-1')
df.rename(columns=lambda x: x.strip(), inplace=True)

df = df[['target', 'text']]
df['target'] = df['target'].replace({'ham': 0, 'spam': 1})
df['Count'] = df['text'].apply(len)
ps = PorterStemmer()
corpus = []
nltk.download('stopwords')
for i in range(0, 5572):
    msg = df['text'][i]
    msg = re.sub('\b[\w\-.]+?@\w+?\.\w{2,4}\b', 'emailaddr', df['text'][i])
    msg = re.sub('(http[s]?\S+)|(\w+\.[A-Za-z]{2,4}\S*)', 'httpaddr', df['text'][i])
    msg = re.sub('£|\$', 'moneysymb', df['text'][i])
    msg = re.sub('\b(\+\d{1,2}\s)?\d?[\-(.]?\d{3}\)?[\s.-]?\d{3}[\s.-]?\d{4}\b', 'phonenumbr', df['text'][i])
    msg = re.sub('\d+(\.\d+)?', 'numbr', df['text'][i])
    msg = re.sub('[^\w\d\s]', ' ', df['text'][i])

    if i<2:
        print("\t\t\t\t MESSAGE ", i)

    if i<2:
        print("\n After Regular Expression - Message ", i, " : ", msg)

    # Each word to lower case
    msg = msg.lower()
    if i<2:
        print("\n Lower case Message ", i, " : ", msg)

    # Splitting words to Tokenise
    msg = msg.split()
    if i<2:
        print("\n After Splitting - Message ", i, " : ", msg)

    # Stemming with PorterStemmer handling Stop Words
    msg = [ps.stem(word) for word in msg if not word in set(stopwords.words('english'))]
    if i<2:
        print("\n After Stemming - Message ", i, " : ", msg)

    # preparing Messages with Remaining Tokens
    msg = ' '.join(msg)
    if i<2:
        print("\n Final Prepared - Message ", i, " : ", msg, "\n\n")

    # Preparing WordVector Corpus
    corpus.append(msg)

cv = CountVectorizer()
x = cv.fit_transform(corpus).toarray()
y = df['target']
print (y.value_counts())
le = LabelEncoder()
y = le.fit_transform(y)
```

A.2 Logistic Regression

```
pipeline = Pipeline([('scaler', StandardScaler()), ('logreg', LogisticRegression(solver='liblinear',
↪  random_state=33, tol=0.01, max_iter=1000))])
param_dist = {'logreg__C': np.logspace(-2, 2), 'logreg__penalty': ['l2']}
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=33)
random_search = RandomizedSearchCV(pipeline, param_dist, n_iter=10, cv=cv, verbose=1, scoring='accuracy', n_jobs=1,
↪  random_state=33)
random_search.fit(xtrain, ytrain)
best_model = random_search.best_estimator_
```

A.3 Naives Bayes

```
gnb = GaussianNB()
gnb.fit(xtrain, ytrain)
ypred = gnb.predict(xtest)
```

A.4 K-Nearest Neighbours

```
knn_pipeline = Pipeline([('scaler', StandardScaler()), ('knn', KNeighborsClassifier())])
param_grid = {'knn__n_neighbors': range(1, 10), 'knn__weights': ['uniform', 'distance']}
grid = GridSearchCV(knn_pipeline, param_grid, cv=5, verbose=1, scoring='accuracy', n_jobs=-1)
grid.fit(xtrain, ytrain)
best_model = grid.best_estimator_
ypred = best_model.predict(xtest)
```

A.5 Random Forest

```
rf = RandomForestClassifier(random_state=33)
param_grid = {'n_estimators': [10, 50, 100], 'min_samples_split': [2, 5, 10],  'min_samples_leaf': [1, 2]}
grid = GridSearchCV(rf, param_grid, cv=3, verbose=1, scoring='accuracy', n_jobs=-1)
grid.fit(xtrain, ytrain)
best_model = grid.best_estimator_
```

A.6 Support Vector Machine

```
svm_pipeline = Pipeline([('scaler', StandardScaler()), ('svm', SVC(random_state=33))])
param_grid = {'svm__C': [0.1, 1, 10], 'svm__kernel': ['linear', 'rbf'],  'svm__gamma': ['scale', 'auto']}
grid = GridSearchCV(svm_pipeline, param_grid, cv=3, verbose=1, scoring='accuracy', n_jobs=-1)
grid.fit(xtrain, ytrain)
best_model = grid.best_estimator_
```

B Simulation Program Code

```
simulation_dir = '../'
model_dir = './models'
data_dir = './data'
results_dir = './results'
log_dir = './logs'
abs_model_dir = os.path.abspath(os.path.join(simulation_dir, model_dir))
model_path = os.path.join(abs_model_dir, 'logistic_regression_model.pkl')
model = load(model_path)
cv_path = os.path.abspath(os.path.join(simulation_dir, model_dir, 'count_vectorizer.pkl'))
with open(cv_path, 'rb') as file:
    cv = pickle.load(file)

class SimulationTimeFilter(logging.Filter):
    def __init__(self, env):
        super().__init__()
        self.env = env

    def filter(self, record):
        hours, remainder = divmod(self.env.now, 3600)
        minutes, seconds = divmod(remainder, 60)
        record.simulation_time = "{:02}:{:02}:{:05.2f}".format(int(hours), int(minutes), seconds)
        return True

# A function to set up logging for the simulation, one log is for regular messages and the other is for alerts
# Alerts are critical messages that require immediate attention
def setup_logging(env):
    normal_log_filename = 'simulation_logs.csv'
    alert_log_filename = 'alert_logs.csv'

    # Set up file handlers for both normal and alert logs
    # Open in append mode to avoid overwriting previous logs
    normal_handler = logging.FileHandler(os.path.join(log_dir, normal_log_filename), mode='a')
    alert_handler = logging.FileHandler(os.path.join(log_dir, alert_log_filename), mode='a')

    # Define CSV formatters with appropriate fields
    normal_fields = ['Simulation Time', 'Log Level', 'Message', 'SMS Content']
    normal_formatter = logging.Formatter('%(simulation_time)s,%(levelname)s,%(message)s,%(sms_content)s')

    alert_fields = ['Simulation Time', 'Log Level', 'Message', 'Baseband Unit', 'Spam Volume']
    alert_formatter =
    ↪  logging.Formatter('%(simulation_time)s,%(levelname)s,%(message)s,%(baseband_unit)s,%(spam_volume)s')

    # Write headers to the log files if they are newly created
    if os.stat(os.path.join(log_dir, normal_log_filename)).st_size == 0:
        with open(os.path.join(log_dir, normal_log_filename), 'w') as f:
            csv.writer(f).writerow(normal_fields)
    if os.stat(os.path.join(log_dir, alert_log_filename)).st_size == 0:
        with open(os.path.join(log_dir, alert_log_filename), 'w') as f:
            csv.writer(f).writerow(alert_fields)

    normal_logger = logging.getLogger('normal_logger')
    normal_logger.setLevel(logging.INFO)
    normal_logger.addHandler(normal_handler)
    normal_logger.propagate = False
    normal_handler.setFormatter(normal_formatter)
    normal_logger.addFilter(SimulationTimeFilter(env))
    alert_logger = logging.getLogger('alert_logger')
    alert_logger.setLevel(logging.CRITICAL)
    alert_logger.addHandler(alert_handler)
    alert_logger.propagate = False
    alert_handler.setFormatter(alert_formatter)
    alert_logger.addFilter(SimulationTimeFilter(env))

    return normal_logger, alert_logger

# A function to load the SMS data and preprocess it
def load_sms_data(file_path):
    df = pd.read_csv(file_path, encoding='latin-1')
    df.rename(columns=lambda x: x.strip(), inplace=True)
    df = df[['target', 'text']]
    df['target'] = df['target'].replace({'ham': 0, 'spam': 1})

    # Preprocess text data
    def preprocess_text(text):
        text = re.sub(r'\b[\w\-.]+?@\w+?\.\w{2,4}\b', 'emailaddr', text)
        text = re.sub(r'(http[s]?\S+)|(\w+\.[A-Za-z]{2,4}\S*)', 'httpaddr', text)
        text = re.sub(r'£|\$', 'moneysymb', text)
        text = re.sub(r'\b(\+\d{1,2}\s)?\d?[\-(.]?\d{3}\)?[\s.-]?\d{3}[\s.-]?\d{4}\b', 'phonenumbr', text)
        text = re.sub(r'\d+(\.\d+)?', 'numbr', text)
        text = re.sub(r'[^\w\d\s]', ' ', text)
        text = text.lower()
        words = text.split()
        ps = PorterStemmer()
        words = [ps.stem(word) for word in words if word not in set(stopwords.words('english'))]
        return ' '.join(words)

    # Apply preprocessing to text column
    df['processed_text'] = df['text'].apply(preprocess_text)
    features = cv.transform(df['processed_text']).toarray()
    return list(zip(df['target'], features, df['text']))

# The MobileDevice class is responsible for sending SMS messages
class MobileDevice:
    def __init__(self, env, messages, controller, message_count, logger):
        self.env = env
        self.messages = [(target, features, message_content, env.now) for target, features, message_content in
        ↪  random.sample(messages, message_count)]
        self.controller = controller
        self.logger = logger

    # The messages are sent at different intervals based on the time of day
    def send_sms(self):
        for target, features, message_content, _ in self.messages:
            sent_time = self.env.now
            interval = random.uniform(0.01, 1.0)
            yield self.env.timeout(interval)
            self.logger.info(f"SENDING FROM DEVICE: A message has been sent from the device", extra={'sms_content':
            ↪  message_content})
            selected_radio_unit = random.choice(self.controller.radio_units)
            yield self.env.process(selected_radio_unit.process(target, features, message_content, sent_time))

# The radiounit is responsible for processing messages its received from the mobile device
# It then sends them to its specific baseband unit
class RadioUnit:
    def __init__(self, env, next_unit, logger):
        self.env = env
        self.next_unit = next_unit
        self.logger = logger

    # The process method simulates the time taken to process a message
    def process(self, target, features, message_content, sent_time):
        base_delay = 0.02
        random_delay = random.uniform(0.01, 0.1)
        total_delay = base_delay + random_delay
        yield self.env.timeout(total_delay)
        self.logger.info(f"RECEIVED AT RADIO UNIT: The message has been received at the radio unit, with a delay
        ↪  of: {total_delay:.2f}s:", extra={'sms_content': message_content[:30]})
        yield self.env.process(self.next_unit.process(target, features, message_content, sent_time))

# The BasebandUnit acts as a logical operator, it detects the spam SMS messages
# It alerts the network controller when the spam ratio exceeds a certain threshold
# It also then processes the messages to continue being sent over the network
class BasebandUnit:
    def __init__(self, env, next_unit, model, identifier, normal_logger, alert_logger):
        self.env = env
        self.next_unit = next_unit
        self.model = model
        self.identifier = identifier
        self.resource = simpy.Resource(env, capacity=self.dynamic_capacity())
        self.radio_units = [RadioUnit(env, self, normal_logger) for _ in range(2)]
        self.normal_logger = normal_logger
        self.alert_logger = alert_logger
        self.predictions = []
        self.actuals = []
        self.message_count = 0
        self.spam_count = 0

    # The dynamic_capacity method is used to vary the capacity over each minute
    def dynamic_capacity(self):
        current_second = int(self.env.now % 60)
        if 0 <= current_second < 30:
            return 200
        else:
            return 100

    # The process method simulates the time taken to process a message
    # It also predicts whether the message is spam or ham using the machine learning model
    # It then logs the message processing and sends it to the next unit
    def process(self, target, features, message_content, sent_time):
        with self.resource.request() as req:
            yield req
            predicted = self.model.predict(features.reshape(1, -1))[0]
            self.predictions.append(predicted)
            self.actuals.append(target)
            is_spam = predicted == 1
            delay = random.uniform(0.001, 0.3)
            yield self.env.timeout(delay)
            self.message_count += 1
            if is_spam:
                self.spam_count += 1
            # Use the appropriate logger based on message classification
            if is_spam:
                self.normal_logger.warning(f"PROCESSED AT BASEBAND UNIT: SPAM SMS message was processed at the
                ↪  baseband unit {self.identifier}", extra={'sms_content': message_content[:30]})
            else:
                self.normal_logger.info(f"PROCESSED AT BASEBAND UNIT: HAM SMS message was processed at the baseband
                ↪  unit {self.identifier}", extra={'sms_content': message_content[:30]})
            yield self.env.process(self.next_unit.process(target, features, message_content, sent_time))
            if self.message_count % 100 == 0:
                self.check_spam_ratio()

    # This method checks the spam ratio and alerts the network controller if it exceeds a certain threshold
    def check_spam_ratio(self):
        if self.message_count > 0:
            spam_ratio = self.spam_count / self.message_count
            if spam_ratio > 0.40:
                self.alert_logger.critical(f"CRITICAL ALERT: High spam volume detected at {self.identifier} and
                ↪  critical action is needed", extra={'baseband_unit': self.identifier, 'spam_volume':
                ↪  f"{spam_ratio:.2f}"})
            self.message_count = 0
            self.spam_count = 0

    # The evaluate_model method evaluates the performance of the model
    def evaluate_model(self):
        accuracy = accuracy_score(self.actuals, self.predictions)
        conf_matrix = confusion_matrix(self.actuals, self.predictions)
        class_report = classification_report(self.actuals, self.predictions, target_names=['Ham', 'Spam'])
        evaluation_text = f"{self.identifier} - Accuracy: {accuracy:.2f}\nConfusion
        ↪  Matrix:\n{conf_matrix}\nClassification Report:\n{class_report}\n"
        print(evaluation_text)
        return evaluation_text

# The core network acts as the base for the network, it receives the messages from the baseband unit
# It then processes the messages to be sent to the mobile device
class CoreNetwork:
    def __init__(self, env, next_unit):
        self.env = env
        self.next_unit = next_unit
        self.model = model

    # The process method simulates the time taken to process a message
    def process(self, target, processed_text, message_content, sent_time):
        yield self.env.timeout(1)
        classification = 'Spam' if self.model.predict([processed_text])[0] == 1 else 'Ham'
        yield self.env.process(self.next_unit.process(target, message_content, classification, sent_time))

# The ToMobileDevice is the device that receives the messages from the core network
# It then processes the messages and logs the results
class ToMobileDevice:
    def __init__(self, env, logger=None):
        self.env = env
        self.results = []
        self.predictions = []
        self.actuals = []
        self.logger = logger  # Accept and store the logger

    # The process method simulates the time taken to process a message
    def process(self, target, message_content, classification, sent_time):
        yield self.env.timeout(1)  # Simulate processing time at the mobile device
        latency = self.env.now - sent_time
        predicted = 1 if classification == 'Spam' else 0
        correct = (target == predicted)
        correctness = 'correctly' if correct else 'incorrectly'
        spam_label = predicted
        if self.logger:
            self.logger.info(f'MESSAGE RECEIVED ON MOBILE: A message has been received at the device with the
            ↪  latency of {latency:.2f}s', extra={'sms_content': message_content[:30]})
        self.predictions.append(predicted)
        self.actuals.append(target)

# The NetworkController class is responsible for setting up the network components and running the simulation
# It creates the baseband units, radio units, and mobile devices as well as links them together
# It sets up the different baseband units and gives each a unique identifier
class NetworkController:
    def __init__(self, env, datasets, normal_logger, alert_logger):
        self.env = env
        self.datasets = datasets
        self.basebands = []
        self.evaluation_results = []
        self.to_mobile_device = ToMobileDevice(env, logger=normal_logger)  # Pass the logger

        # Create the baseband units and radio units
        for i, data in enumerate(self.datasets):
            identifier = f"Baseband_{i+1}"
            core_network = CoreNetwork(env, self.to_mobile_device)
            baseband = BasebandUnit(env, core_network, model, identifier, normal_logger, alert_logger)
            radio_unit1 = RadioUnit(env, baseband, normal_logger)
            radio_unit2 = RadioUnit(env, baseband, normal_logger)
            baseband.radio_units = [radio_unit1, radio_unit2]
            self.basebands.append((baseband, data))
    # The evaluate_all_basebands method evaluates the performance of all baseband units
    def evaluate_all_basebands(self):
        total_predictions = []
        total_actuals = []
        for baseband, data in self.basebands:
            evaluation_result = baseband.evaluate_model()
            self.evaluation_results.append(evaluation_result)
            total_predictions.extend(baseband.predictions)
            total_actuals.extend(baseband.actuals)

        # Evaluate overall performance
        total_accuracy = accuracy_score(total_actuals, total_predictions)
        total_conf_matrix = confusion_matrix(total_actuals, total_predictions)
        total_class_report = classification_report(total_actuals, total_predictions, target_names=['Ham', 'Spam'])
        total_evaluation_text = f"Total Evaluation - Accuracy: {total_accuracy:.2f}\nConfusion
        ↪  Matrix:\n{total_conf_matrix}\nClassification Report:\n{total_class_report}\n"
        print(total_evaluation_text)
        self.evaluation_results.append(total_evaluation_text)
        return self.evaluation_results

    # The save_evaluation_results method saves the evaluation results to a file
    def save_evaluation_results(self, filename):
        with open(filename, 'w') as file:
            for result in self.evaluation_results:
                file.write(result + '\n')

    # The run method starts the simulation
    def run(self):
        for baseband, data in self.basebands:
            mobile_device = MobileDevice(self.env, data, baseband, message_count=len(data),
            ↪  logger=baseband.normal_logger)
            self.env.process(mobile_device.send_sms())
        self.env.run()

def main():
    env = simpy.Environment()
    normal_logger, alert_logger = setup_logging(env)
    common_data_path = os.path.join(data_dir, 'spam_data.csv')
    common_sms_data = load_sms_data(common_data_path)

    # Prepare the datasets for the ML Models
    datasets = []
    num_basebands = 5  # Sets the number of baseband units in the network
    for i in range(num_basebands):
        data_path = os.path.join(data_dir, f'generated_spam{i}.csv')
        sms_data = load_sms_data(data_path)
        # Combine the common dataset with the individual dataset for each baseband
        combined_data = sms_data + common_sms_data  # Combining lists of tuples
        datasets.append(combined_data)

    print("Data loaded successfully.")
    controller = NetworkController(env, datasets, normal_logger, alert_logger)
    print("Simulation starting...")
    controller.run()
    print("Simulation complete.")

    print("Evaluating model performance...")
    evaluation_results = controller.evaluate_all_basebands()
    controller.save_evaluation_results(os.path.join(results_dir, 'evaluation_results.txt'))
    print("Results and evaluations saved successfully.")

if __name__ == "__main__":
    main()
```

C Dataset Generation

The additional data sets were generated with the following code.

```
import pandas as pd
import random
from conversations import generate_message
```

```python
def generate_dataset(total_entries, spam_ratio):
    num_spam = int(total_entries * spam_ratio)
    num_ham = total_entries - num_spam

    data = []
    for _ in range(num_spam):
        data.append(["spam", generate_message(True)])
    for _ in range(num_ham):
        data.append(["ham", generate_message(False)])

    # Shuffle the dataset to mix spam and ham messages
    random.shuffle(data)

    df = pd.DataFrame(data, columns=['target', 'text'])
    df.to_csv('/mnt/j/sit327/spam-generator/generated_spam2.csv', index=False)
    print("Dataset generated with {} entries ({}% spam).".format(total_entries, int(spam_ratio * 100)))

# Parameters: Total entries and the spam ratio
generate_dataset(12000, 0.34)
```

The generator pulled from the following file, this is a snippet as the file is much more extensive for generating a deep variety of messages.

```python
from random import choice, randint
from faker import Faker
fake = Faker()

def generate_message(is_spam):
    spam_keys = ["free", "call", "text", "stop", "mobile", "reply", "win", "prize", "cash", "urgent", "claim",
    ↪  "guaranteed", "congratulations", "visit", "click", "credit", "offer", "unsubscribe"]
    fines = ["tolls", "toll", "fine", "penalty", "fee", "charge", "ticket", "violation", "citation", "summons",
    ↪  "invoice", "bill", "payment", "due", "past due", "overdue", "late", "delinquent", "unpaid", "outstanding"]

    spam_templates = [
        f"Urgent! Please call {fake.phone_number()} from your landline. Your ABTA complimentary 4* Tenerife Holiday
        ↪  or £5000 cash await collection SAE T&Cs Box {fake.url()}",
        f"TOLL NOTICE: act now or you'll be fined ${randint(50, 200)} {fake.url()} or face jail time",
        f"FINAL NOTICE: act now {fake.url()} to avoid jail time",
        f"Your apple ID has been locked. Click {fake.url()} to unlock",]

    ham_templates = [
        f"Hey, I'm at {choice(location)} right now. I'll be home in 30 minutes. See you soon!",
        f"Mom called. She wants us to visit this weekend for {choice(['her birthday', 'a family reunion', 'a
        ↪  barbecue'])}. Should I tell her we'll go?",
        f"hey bestie, how's it going? I'm thinking of going to {choice(['the beach', 'the park', 'the mall'])} this
        ↪  weekend. Wanna join?",
        f"Quick question: Do you have the recipe for grandma's famous {choice(['pie', 'lasagna', 'roast
        ↪  chicken'])}? Want to try making it this weekend.",
        f"Saw a great deal on flights to {choice(['Japan', 'Italy', 'Iceland'])} for next spring. Should we
        ↪  consider it? Always wanted to see the {choice(['cherry blossoms', 'Colosseum', 'Northern Lights'])}."]

    if is_spam:
        return choice(spam_templates)
    else:
        return choice(ham_templates)
```

References

[1] A. Alzahrani and D. B. Rawat, "Comparative Study of Machine Learning Algorithms for SMS Spam Detection," in *2019 SoutheastCon*. IEEE, 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9020530

[2] S. Gadde, A. Lakshmanarao, and S. Satyanarayana, "SMS Spam Detection using Machine Learning and Deep Learning Techniques," in *2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021*. Institute of Electrical and Electronics Engineers Inc., 3 2021, pp. 358–362. [Online]. Available: https://ieeexplore.ieee.org/document/9441783

[3] S. Hosseinpour and H. Shakibian, "An Ensemble Learning Approach for SMS Spam Detection," in *2023 9th International Conference on Web Research, ICWR 2023*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 125–128. [Online]. Available: https://ieeexplore.ieee.org/document/10139070/

[4] V. Dharani, D. Hegde, and Mohana, "Spam SMS (or) Email Detection and Classification using Machine Learning," in *Proceedings - 5th International Conference on Smart Systems and Inventive Technology, ICSSIT 2023*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 1104–1108. [Online]. Available: https://ieeexplore.ieee.org/document/10060908/

[5] H. Jain and M. Mahadev, "An Analysis of SMS Spam Detection using Machine Learning Model," in *Proceedings - 2022 5th International Conference on Computational Intelligence and Communication Technologies, CCICT 2022*. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 151–156. [Online]. Available: https://ieeexplore.ieee.org/document/9913532/

[6] T. Jain, P. Garg, N. Chalil, A. Sinha, V. K. Verma, and R. Gupta, "SMS Spam Classification Using Machine Learning Techniques," in *Proceedings of the Confluence 2022 - 12th International Conference on Cloud Computing, Data Science and Engineering*. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 273–279. [Online]. Available: https://ieeexplore.ieee.org/document/9734128/

[7] S. M. Nagare, P. P. Dapke, S. A. Quadri, S. B. Bandal, and M. R. Baheti, "Short Message Service (SMS) Mobile Spam Detection using Naïve Bayes," in *2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*. Institute of Electrical and Electronics Engineers, 1 2024, pp. 67–70. [Online]. Available: https://ieeexplore.ieee.org/document/10493926/

[8] Australian Competition and Consumer Commission (ACCC), "Scam Statistics 2024," 2024. [Online]. Available: https://www.scamwatch.gov.au/research-and-resources/scam-statistics?scamid=all&date=2024

[9] ——, "Scam Statistics 2023," 2023. [Online]. Available: https://www.scamwatch.gov.au/research-and-resources/scam-statistics?scamid=all&date=2023

[10] ——, "Scam Statistics 2022," 2022. [Online]. Available: https://www.scamwatch.gov.au/research-and-resources/scam-statistics?scamid=all&date=2022

[11] ——, "Scam Statistics 2021," 2021. [Online]. Available: https://www.scamwatch.gov.au/research-and-resources/scam-statistics?scamid=all&date=2021

[12] Telstra, "Keep Snitching on Scammers: How Our New 7226 Reporting Number is Fighting Off SMS and MMS Scams," 2023. [Online]. Available: https://www.telstra.com.au/exchange/keep-snitching-on-scammers--how-our-new-7226-reporting-number-is

[13] Australian Signals Directorate and Australian Cyber Security Centre, "ACSC Annual Cyber Threat Report - 2019 to 2020," ASD, ASCS, Tech. Rep., 2020. [Online]. Available: https://www.cyber.gov.au/about-us/view-all-content/reports-and-statistics/asdacsc-annual-cyber-threat-report-july-2019-june-2020

[14] Australian Cyber Security Centre and Australian Signals Directorate, "ACSC Annual Cyber Threat Report - 2020 to 2021," ASD, ACSC, Tech. Rep., 2021. [Online]. Available: https://www.cyber.gov.au/about-us/view-all-content/reports-and-statistics/asdacsc-annual-cyber-threat-report-july-2020-june-2021

[15] Australian Signals Directorate and Australian Cyber Security Centre, "ASCS Annual Cyber Threat Report - 2021 to 2022," ASD, ACSC, Tech. Rep., 2022. [Online]. Available: https://www.cyber.gov.au/about-us/view-all-content/reports-and-statistics/acsc-annual-cyber-threat-report-july-2021-june-2022

[16] Australian Cyber Security Centre and Australian Signals Directorate, "ACSC Annual Cyber Threat Report - 2022 to 2023," ASD, ACSC, Tech. Rep., 2023. [Online]. Available: https://www.cyber.gov.au/about-us/view-all-content/reports-and-statistics/asd-cyber-threat-report-july-2022-june-2023

[17] M. Salman, M. Ikram, and M. A. Kaafar, "Investigating Evasive Techniques in SMS Spam Filtering: A Comparative Analysis of Machine Learning Models," *IEEE Access*, vol. 12, pp. 24 306–24 324, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10431737

[18] D. Sharma and A. Sharaff, "Identifying Spam Patterns in SMS using Genetic Programming Approach," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. IEEE, 2019, pp. 396–400. [Online]. Available: https://ieeexplore.ieee.org/document/9065686/

[19] K. Aparna and S. Halder, "Detection of Multilingual Spam SMS Using NaïveBayes Classifier," in *5th IEEE International Conference on Cybernetics, Cognition and Machine Learning Applications, ICCCMLA 2023*. Institute of Electrical and Electronics Engineers Inc., 2023, pp. 89–94. [Online]. Available: https://ieeexplore.ieee.org/document/10346960/

[20] M. Popovac, M. Karanovic, S. Sladojevic, M. Arsenovic, and A. Anderla, "Convolutional Neural Network Based SMS Spam Detection," in *2018 26th Telecommunications Forum (TELFOR)*. Institute of Electrical and Electronics Engineers Inc., 2018, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/8611916

[21] P. Sethi, V. Bhandari, and B. Kohli, "SMS spam detection and comparison of various machine learning algorithms," in *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*. Institute of Electrical and Electronics Engineers, 2017, pp. 28–31. [Online]. Available: https://ieeexplore.ieee.org/document/8284445

[22] L. Peng, X. Zhu, and P. Zhang, "An Efficient Model for Smartphone Forensics Using SMS Spam Filtering," in *2020 3rd International Conference on Hot Information-Centric Networking, HotICN 2020*. Institute of Electrical and Electronics Engineers Inc., 12 2020, pp. 166–169. [Online]. Available: https://ieeexplore.ieee.org/document/9350843

[23] K. Almeida, Hidalgo, and A. Gómez, "SMS Spam Collection," 2011. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection