AI for Social Good Project Report

A group of people standing around a computer

AI-generated content may be incorrect.

**Project Title**: AI SMS Scam Detection

[1. Introduction 3](#_Toc196758548)

[Project Objective: 3](#_Toc196758549)

[Overview of Solution: 3](#_Toc196758550)

[2. Dataset Selection and Preprocessing 3](#_Toc196758551)

[Challenges and Resolutions 5](#_Toc196758552)

[3. EDA 6](#_Toc196758553)

[E-commerce scam dataset details 7](#_Toc196758554)

[Financial scam dataset details 10](#_Toc196758555)

[Cryptocurrency Scam Dataset Details 13](#_Toc196758556)

[Phishing Scam Dataset Details 15](#_Toc196758557)

[Ham Dataset: 18](#_Toc196758558)

[4. Algorithm Design 20](#_Toc196758559)

[Design Overview 20](#_Toc196758560)

[Rationale for Algorithm Choice 21](#_Toc196758561)

[Innovation and References 22](#_Toc196758562)

[5. Training Process and Performance 22](#_Toc196758563)

[Data Splitting 22](#_Toc196758564)

[Tools and Libraries 24](#_Toc196758565)

[Performance Metrics: 25](#_Toc196758566)

[Logistic Regression 26](#_Toc196758567)

[Multinomial Naive Bayes 26](#_Toc196758568)

[Random Forest Classifier 26](#_Toc196758569)

[LSTM 27](#_Toc196758570)

[CNN 27](#_Toc196758571)

[GRU 27](#_Toc196758572)

[LDA 28](#_Toc196758573)

[8. Human-Centred AI Design 29](#_Toc196758574)

[User experience design elements involved in the website: 29](#_Toc196758575)

[8. Conclusion 31](#_Toc196758576)

[9. Future Development Plans 32](#_Toc196758577)

[10. Project Prototype 33](#_Toc196758578)

# 1. Introduction

## Project Objective:

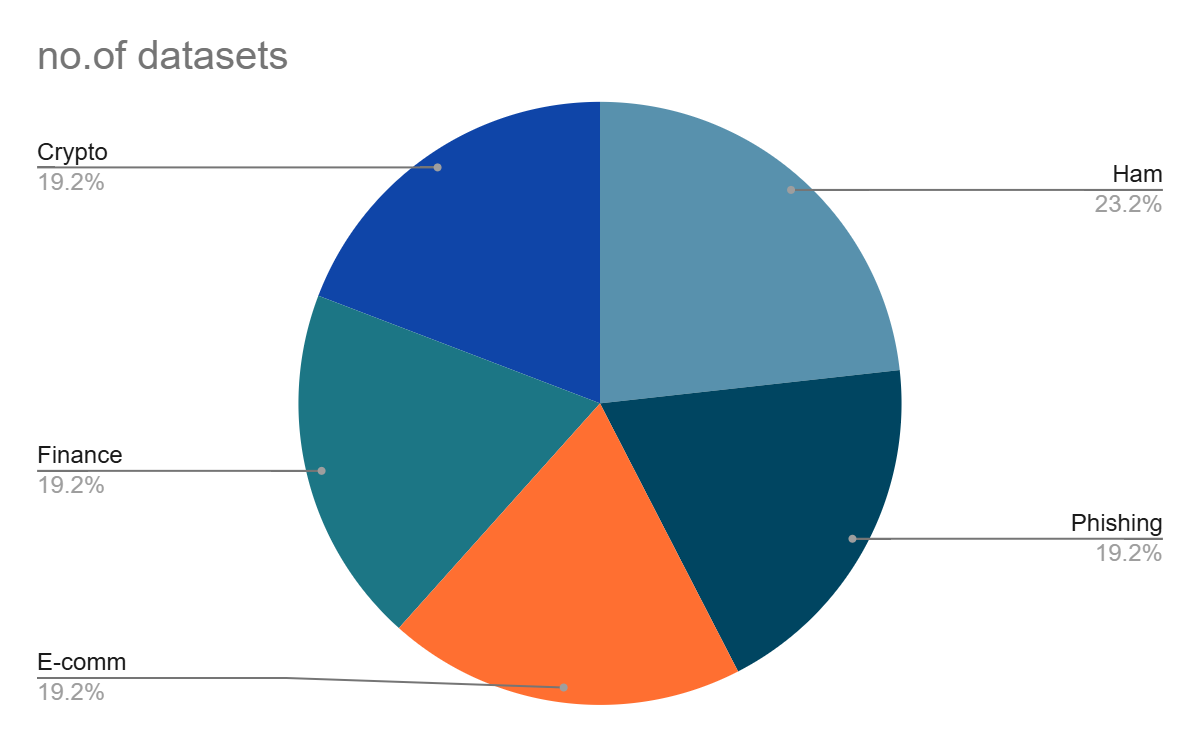
The project aims to address the growing issue of scam Short Message Service (SMS), which poses significant threats to individuals and organisations by spreading misinformation, phishing attempts, and malicious content. These SMS often lead to financial loss, data breaches, and psychological distress. By detecting them using DL, particularly neural networks, that contribute to social good by improving digital safety, protecting personal information, and empowering users to make informed decisions.

## Overview of Solution:

The propose an AI-based solution that leverages neural networks to accurately detect and categorise scam SMSs. Using a dataset sourced from Kaggle, complemented with synthetically generated samples to address class imbalance and enrich training diversity, train and evaluate multiple models including Random Forest, Multinomial Naive Bayes, Logistic Regression and deep learning models such as Convolutional Neural Network (CNN), Long short-term memory (LSTM) and Gated Recurrent Units (GRU). The final model will be deployed as a scam-checking website that informs users of potential threats once a user inputs their SMS and explains the rationale behind classifications by highlighting specific keywords or patterns that triggered the alert using LIME. This alerts users and enhances cybersecurity.

# 2. Dataset Selection and Preprocessing

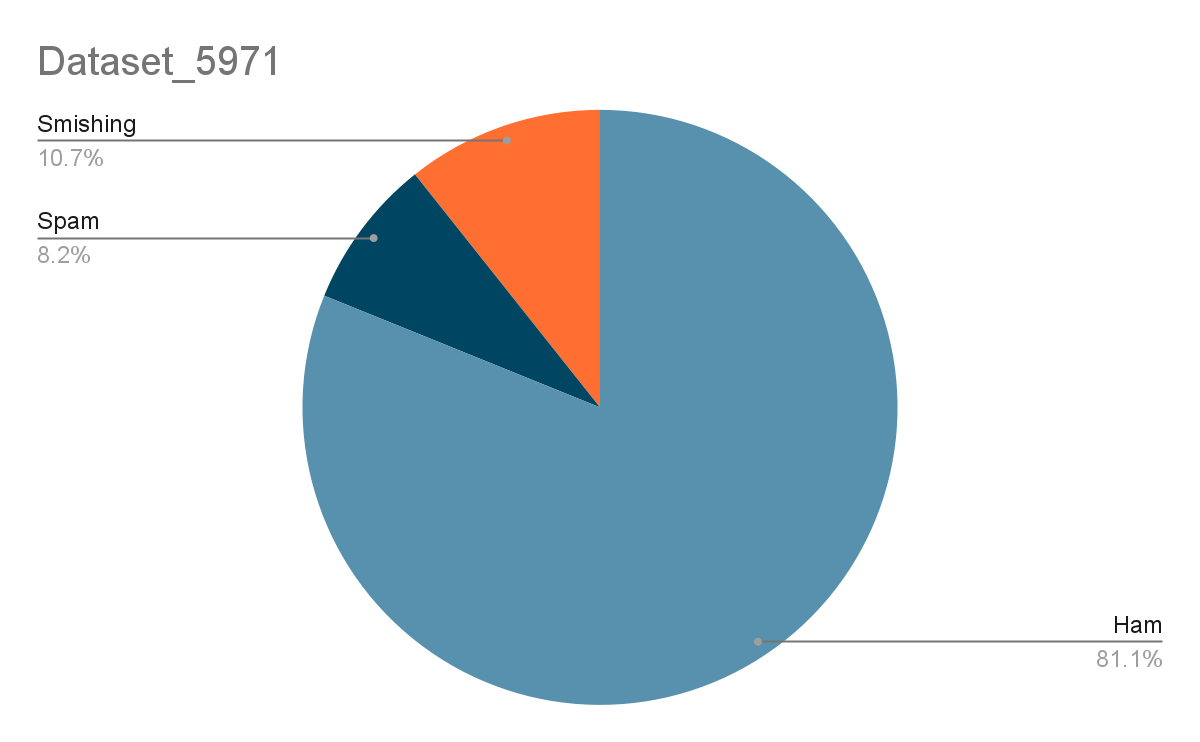
The dataset comprises Ham and E-commerce, financial, crypto and phishing scams, with a significant portion being synthetically generated.



The objective is to identify not only whether an SMS is a scam but also to distinguish between different types of scams, such as phishing, e-commerce scams, and crypto scams. Existing datasets were examined, but most of them classify SMS messages only as either spam or ham (legitimate messages), without categorising specific types of scams. Due to the absence of a dataset containing categorised scam SMS, synthetic datasets were generated for the three scam types, while original data was used for ham messages. This approach was adopted because ham messages, typically written by humans, cover a wide range of topics and contain diverse vocabulary, making them challenging to replicate synthetically.

At first downloaded a dataset containing spam, ham and smishing SMS messages.

Dataset\_5971(mishra, sandhya; Soni, Devpriya (2022))



The Smishing SMS messages were filtered, and ChatGPT was used to generate 4,000 additional synthetic phishing SMS messages, with a temperature setting of 0.9 and a frequency penalty of 1.9. This configuration was chosen to increase the likelihood of selecting less probable tokens, resulting in more creative and distinct messages.

For the e-commerce dataset, examples were sourced from the internet and personal devices, totaling 36 original messages. With a temperature setting of 0.8 and a frequency penalty of 1.7, OpenAI’s model was used to generate 4,000 synthetic e-commerce scam messages. Higher temperature and frequency penalty settings resulted in gibberish, so these were reduced to achieve more coherent results. The decision to use less original data for e-commerce scams was based on the fact that these scams often followed a specific format, with variations mainly in product codes and URLs. The same approach was applied to other scam datasets.

Due to the presence of multiple datasets with different labelling systems, standardisation was necessary. Each category was split into its own file with a corresponding label. During the process of synthetic data generation to supplement the e-commerce, financial, crypto, and phishing scam datasets, gibberish characters were identified in both the ham dataset (genuine data) and the synthetically generated data. Further research revealed that these characters were a result of “Mojibake,” a phenomenon caused by incorrect character encoding.

Such characters often occur in large datasets when there are issues with character encoding. We used openAI to remove the mojibake. The synthetic data generated for the Phishing category also had an issue, where all the text bodies were generated with a reference code at the end. We used a filter to delete those from the generated data.

Mojibake characters typically arise in large datasets due to issues with character encoding. OpenAI was utilised to remove these characters from the data. Additionally, the synthetic data generated for the phishing category exhibited an issue where each text body was appended with a reference code at the end. A filter was applied to remove these reference codes from the generated data.

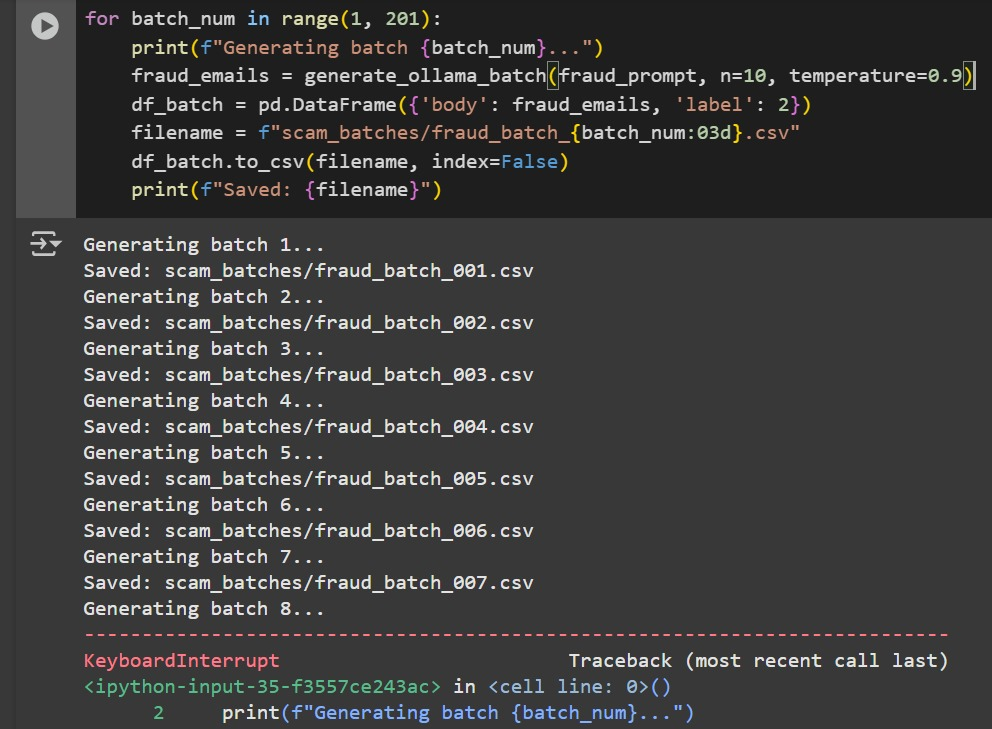
## Challenges and Resolutions

**Change of Topic:** The project initially focused on classifying the types of scam emails. However, due to the inability to find labelled data for email scam categories, the decision was made to discontinue the email classification approach. A dataset of approximately 20,000 email entries was available, but it lacked labels for specific scam categories.

**Unsupervised clustering:** A convolutional neural network (CNN) was initially employed to identify clusters within the email text bodies. Latent Dirichlet Allocation (LDA) topic modelling was then applied to these clusters to determine the types of scams present. The goal was to categorize the data into distinct scam types before training the model on those labelled categories.

The rationale behind this approach was that scam types are continuously evolving, making it challenging to accurately define a comprehensive list of categories through basic research. However, deep learning models require substantial amounts of data to produce accurate results, particularly when dealing with longer text bodies like emails, which have higher dimensionality compared to SMS messages that are shorter and less complex. Given the difficulty in sourcing sufficient email data and the potential challenges in generating large datasets without introducing biases, the decision was made to shift focus to SMS categorisation. Predefined categories were selected, rather than attempting to identify underlying groupings within the dataset.

**Anticipated challenge:** Another potential issue with the clustering system is its inability to group all ham texts together. This is due to the diverse range of topics and vocabulary that ham texts can cover, arising from the unique and varied nature of human interactions.



# A batch job was initially planned for data collection. Although an attempt was made to use OpenAPI, it was a paid service, leading to the exploration of a Hugging Face model, which also proved ineffective. As a result, Deepseek was downloaded for the task. The batch job was configured to process 10 rows per batch, resulting in 200 batch jobs (for an email scam category of 2,000 entries). However, the time complexity was logarithmic (log(n)), and each batch job took progressively longer than the previous one. The process was allowed to run until the 8th batch, at which point the GPU credits on Google Colab were exhausted.

# 3. EDA

This section focuses on the exploratory data analysis (EDA) phase of the SMS spam dataset. In this phase, aim to inspect the raw data and understand its structure, explore label distribution, analyze message lengths, word frequencies, and Semantic Similarity Graphs, gather insights that will guide further text preprocessing and model-building steps. Here have listed the preprocessing functions.

The cleaned data still has to be preprocessed. The **text\_preprocess** function systematically removes noise (e.g., URLs, hashtags, and punctuation), standardizes text to lowercase and discards English stopwords. Only alphabetic terms or specific placeholders (e.g., `[url]`, `[order\_id]`) are retained, ensuring that meaningful content is preserved while irrelevant tokens are filtered out. This reduces feature-space complexity and improves the model’s ability to distinguish spam from legitimate messages especially with the limited amount of data.

The **text\_length\_histogram** function generates a histogram that visualizes the frequency distribution of message lengths in the dataset. It helps identify potential outliers (unusually long or short messages) and reveals the overall shape of the data. Such insights guide subsequent preprocessing strategies.

The **plot\_wordcloud** function offers a visual and statistical overview of the most frequently used words in messages. It starts off by compiling all tokens from the specified column into a single list, then generates a word cloud to emphasize the words appearing most often, with larger font sizes indicating higher frequencies. This visualization quickly helps us identify recurring terms. Printed out the top 10 most frequent words for quick access.

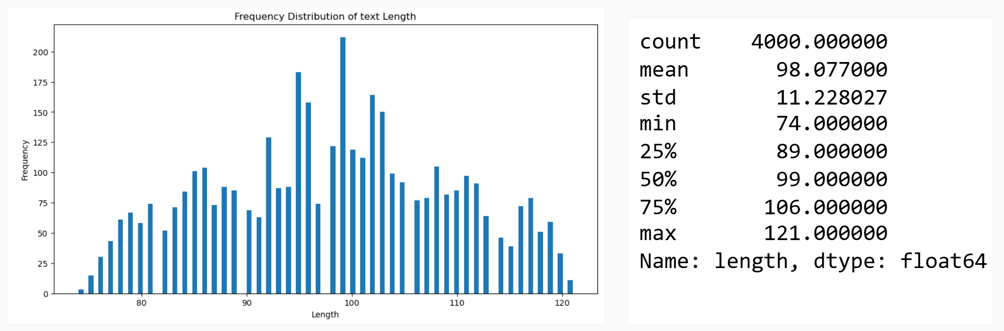
The function **get\_top\_bigrams** identifies and ranks the most frequent 2-word combinations in the text column. It leverages a `CountVectorizer` configured for bigrams and English stopwords removal, ensuring that filler words do not overwhelm the analysis. The function then constructs a DataFrame showing each bigram and how often it appears, printing out the top 10. By revealing common word pairs, to uncover any recurring patterns or unique phrase combinations. The function **get\_top\_trigram** does the same thing, but for 3-word combinations.

The function **get\_top\_tfidf\_words** identifies which words carry the highest TF-IDF (Term Frequency–Inverse Document Frequency) scores. Used TF-IDF as it is a widely used NLP metric which highlights words more unique or “important” to particular documents, rather than just frequent across the entire corpus. It transforms all messages into a numerical matrix of their vectorised forms, where each word is weighted by its TF-IDF score. It also sums up the TF-IDF scores across all documents to find out which words generally hold the most weight or “importance” which is what need for in scam detection, capturing terms that stand out in specific contexts like certain spam keywords, rather than merely the most common words overall. It prints out the top 10 words, and visualizes them in a bar chart.

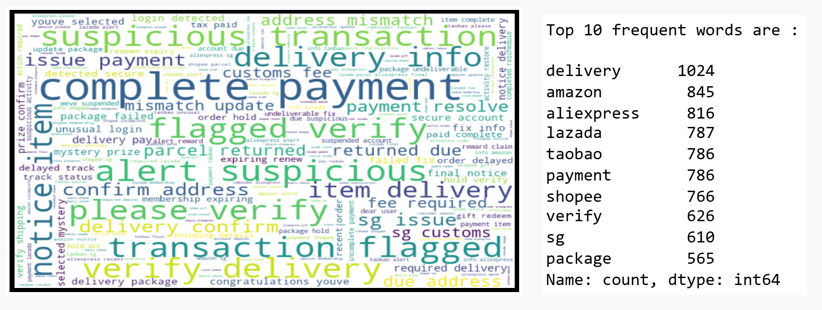
The function **display\_lda\_topics** uses Latent Dirichlet Allocation (LDA) to discover hidden thematic structures within the tokenized text. It prepares the data by taking the list of tokens and converts them into a Gensim-friendly format by creating a dictionary of unique words and transforming each message into a bag-of-words representation. The function then fits an LDA model, which attempts to group words into the top 5 specified topics. Each topic is represented by a set of keywords with varying importance scores.

The **compute\_cosine\_similarity\_matrix** calculates pairwise cosine similarity scores for all embeddings. This similarity matrix helps us quickly see which SMS messages are semantically alike. The function **build\_similarity\_graph** outputs the matrix in a text form. The function **plot\_similarity\_graph\_with\_labels** creates a network from the previous cosine similarity matrix. Each SMS message becomes a node (labelled with a short text preview), and edges connect messages only if their similarity exceeds a certain threshold, which is set as 0.82. This graph-based view offers a powerful EDA perspective: easy to visually spot groups of closely related messages, reveal potential clusters or “communities,” and quickly pinpoint outliers.

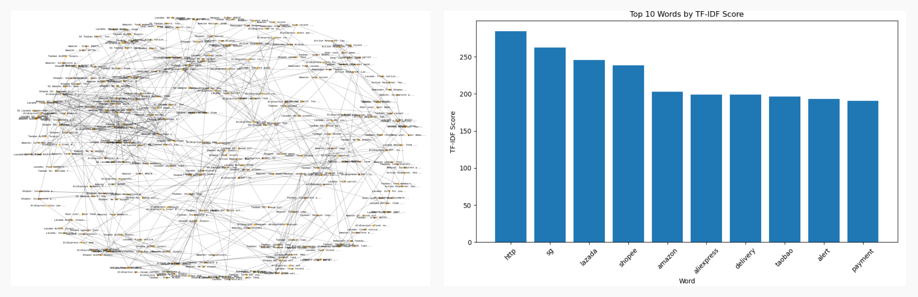
## E-commerce scam dataset details



**Text Length Distribution:** The histogram peaks at 98 characters. The distribution is rather wide, implying that length alone may not be a strong discriminator for subsequent analyses.

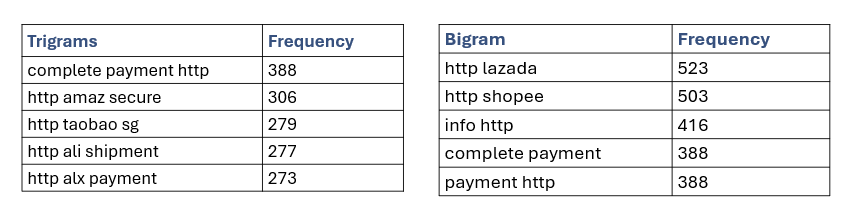


**Word Cloud:** most frequent terms in the word cloud include "*delivery*", "*payment*" and "*verify*". These words point toward common scam tactics which emphasize urgency. Brand names like *"amazon", "aliexpress", "lazada", "taobao",* and *"shopee"* appear prominently. This suggests the messages are designed to impersonate familiar singapore platforms to gain user trust and mimic legitimacy. Phrases like *"complete payment", "verify address", "please verify"*, and *"suspicious activity"* reinforce the manipulation tactic. This analysis helps us understand how scammers structure language to appear trustworthy and persuasive. It also guides the future feature engineering by highlighting keywords or n-grams that may be strong indicators of scam content in SMS messages. The word cloud and frequency analysis reveals how e-commerce scam messages rely on a blend of urgency, familiarity, and transactional language to deceive people.



**Semantic Similarity Group:** Observations from the semantic similarity graph include distinct clusters of messages which can be seen across the graph, indicating that multiple SMS messages share highly similar language patterns or intent. The clusters show that many scam SMS messages are likely variations of the same few base templates, differing only in product names, order numbers, or minor details. For example, seeing many nodes centered around brands like *"Amazon", "AliExpress"* tend to group closely, suggesting repeated templates or reused scam phrasing. The dense interconnections between many nodes reveal high textual redundancy across scam messages. This confirms that many scammers reuse similar templates with slight variations.

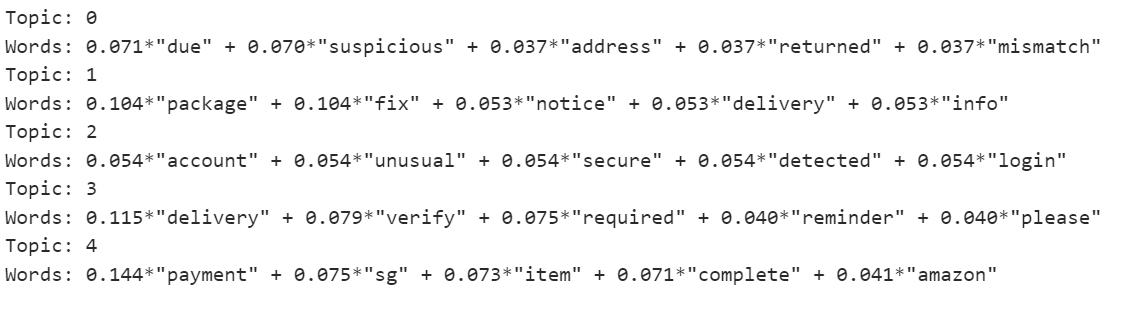
**TF-IDF scores:** The top 10 TF-IDF scoring words all confirm the information gathered above, and also verify that these words are uniquely informative about this dataset, not just frequent.



**Bigrams and Trigrams:** The frequent appearance of "http" in the bigrams and trigrams strongly suggests that the majority of scam messages include embedded phishing links. The presence of brand names like “lazada”, “shopee” and “taobao” highlight brand impersonation tactics. The use of well-known Singapore e-commerce platforms is intended to create trust and con victims. the presence of call to action language, like "complete payment", "payment http", "shopee delivery" create urgency, pressuring users to click a link to resolve an issue. Out of 6,236 unique bigrams, the vast majority (over 6,000) appear only once. This suggests that while many messages follow a templated structure, scammers often inject randomized tokens (e.g., IDs or code fragments) into URLs to evade detection by spam filters. However, the high frequency of the top 5 bigrams and trigrams are linked to the template format of the scams.

To deepen understanding of message structure, To analyze how often top bigrams appear within the top trigrams. This helps us assess how scam messages are constructed: are they just extending base templates, or introducing new, unique linguistic patterns? Out of the top 10 trigrams, 7 directly contain bigrams from the top 10 bigram list  
(Partial, implied, variant). That is a 70% overlap, showing that many trigrams are simply extended versions of already frequent bigrams. Scammers are stacking phrases in a structured way: base bigram + extra keyword. This implies a modular template construction, which is common in mass SMS scam campaigns, indicating that bigrams are foundational building blocks of scam messages.

**LDA topic analysis results:**

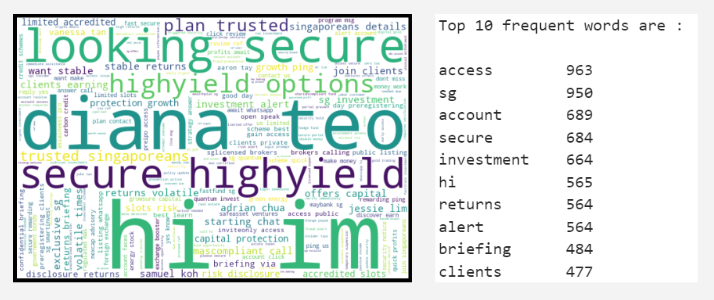


* **Topic 0: Suspicious Address or Identity Mismatch** This topic seems to relate to failed or flagged deliveries due to an issue like a mismatched or suspicious address. Words like *"due"*, *"suspicious"*, *"returned"*, and *"mismatch"* imply a delivery problem caused by security concerns.
* **Topic 1: Package Fix or Update Notice**  
   This is likely a fake delivery notification prompting the user to fix or update delivery info. It uses urgency around a “package” and prompts for “fix” and “info,” suggesting scam tactics involving false delivery updates.
* **Topic 2: Account Security Alert** This topic resembles a phishing attempt via fake account security warnings, using terms like *"unusual login"*, *"detected"*, and *"secure"*. It tries to provoke panic and push the user to act quickly.
* **Topic 3: Delivery Verification Request**  
   Another delivery-related theme, this one emphasizes the need to verify a delivery — possibly a scam urging users to click a link or enter personal data. Keywords like *"verify"*, *"required"*, and *"reminder"* hint at urgency and procedural follow-up.
* **Topic 4: Payment Completion for Online Orders** This topic focuses on completing a payment, possibly impersonating platforms like *Amazon*. Words like *"payment"*, *"complete"*, *"item"*, and *"sg"* indicate a transactional scam, often disguised as an order confirmation or finalization.

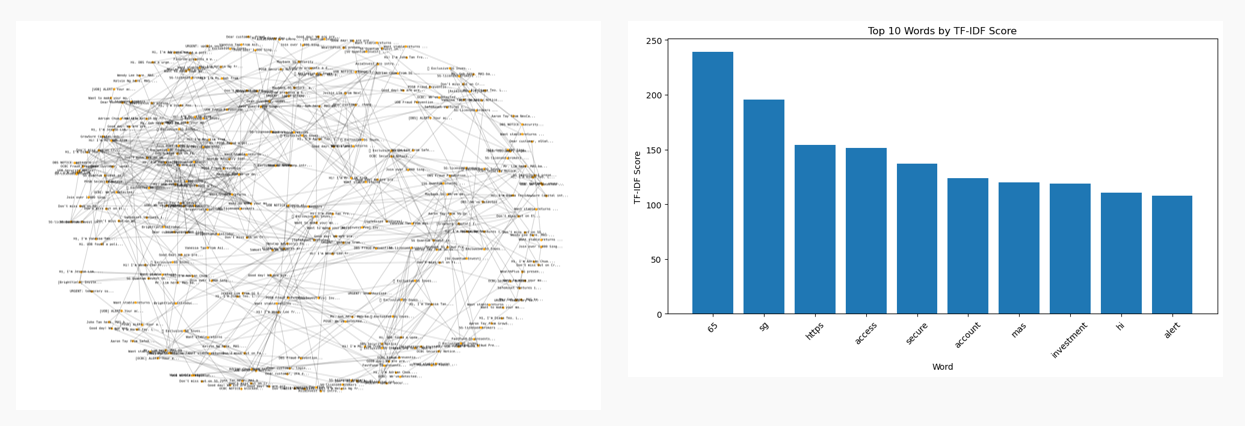
## Financial scam dataset details



**Text length Distribution:** The distribution is skewed to the right, peaking at 130 characters. This shows that financial scams tend to be a bit longer.

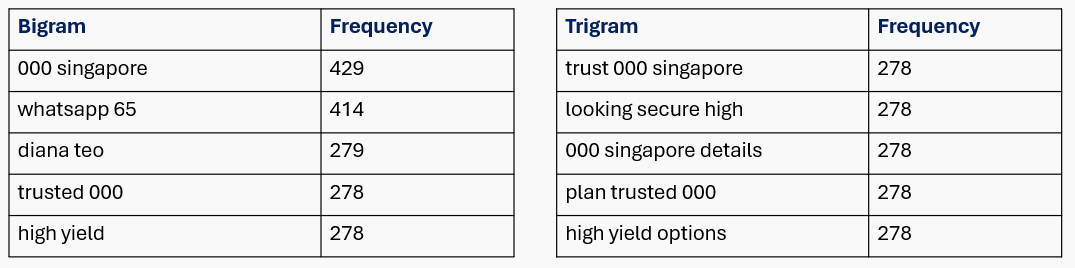


**Word Cloud**: Words like investment, returns, and clients aim to establish professionalism. These are classic finance-oriented hooks meant to mimic real advisors, brokers, or portfolio managers. Terms like access, alert, and secure are engineered to create a sense of urgency and exclusivity. Full names like Diana Teo and Adrian Chua suggest an effort to humanize the scam by impersonating financial agents.

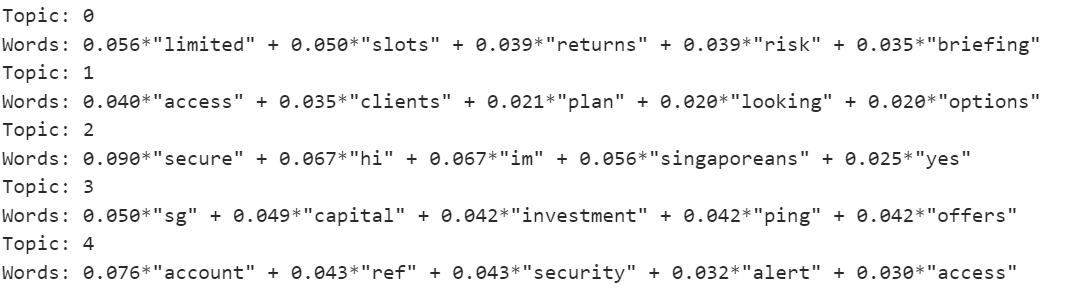


**Semantic Similarity Graph**: Multiple well-connected clusters form around phrases like *“Hi, I’m [name]”, “exclusive offer”, “secure returns”,* and *“account alert”*. This indicates repetitive template usage, with only slight variations in wording, which is a strong sign of scripted scam behavior. One cluster heavily features introductions like *“Hi, I’m Jessie Lim”, “Vanessa Tan from...”,* or *“Adrian Chua”*, indicating a subgroup of messages built around agent impersonation. A separate cluster contains messages related to *“DBS Notice”, “OCBC Security”,* or *“Fraud Prevention”*, fear mongering by mimicking banks and financial alerts. The financial scam messages exhibit high semantic redundancy, with clear clusters around agent impersonation, secure investments, and fraud alerts. These patterns reinforce the idea that scams are built from modular, repeated structures, making them highly detectable.

**TF-IDF scores:** The top 10 TF-IDF scoring words confirm all the information gathered above, and also verify that these words are uniquely informative about this dataset, not just frequent. Additionally, the presence of “*mas*” (Monetary Authority of Singapore) suggests that scammers reference real institutions to fake legitimacy or appear government-backed, and “*hi*” scoring high supports earlier observations that messages often begin with a casual, personalized tone to lower suspicion and build rapport.

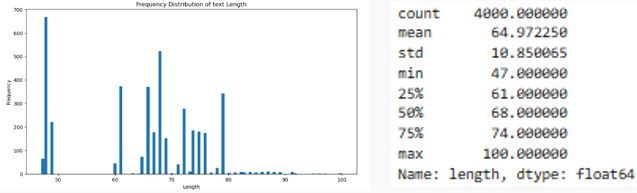


**Bigram and Trigram:** *“000 singaporeans”* and *“trusted 000”* suggest identity spoofing. *“whatsapp 65”* and *“details 65”* point to Singapore-based mobile numbers, showing that users are prompted to engage directly. The inclusion of phone numbers is a clear attempt to bypass platform moderation and move users into unmonitored channels. Phrases like *“high yield”, “plan trusted”,* and *“secure high”* mimic language from real investment ads. This reflects a content strategy designed to hook financially curious users with the promise of stability or exclusivity. The top trigrams in the financial scam dataset reveal even tighter structure and formulaic messaging than bigrams. Many of the top trigrams have identical frequency values (278) as they are pulled from synthetic generation rather than organically generated text. 80% of top trigrams are direct extensions of the most common bigrams. This demonstrates that scammers layer messaging using a base structure (bigram), then append financial or contextual terms to produce variants. The financial scam messages show extremely rigid trigram structures built from bigram foundations. With more than a 80% overlap between top bigrams and trigrams, it’s clear that scammers use templated language blocks designed for scale and impact.

**LDA topic analysis results:**  


* **Topic 0: Investment Opportunity Pitch** This topic reflects exclusive, limited-time investment offers. Words like *"limited"*, *"slots"*, *"returns"*, and *"briefing"* suggest high-pressure sales tactics—likely targeting users with FOMO (fear of missing out) on financial gains.
* **Topic 1: Financial Planning Outreach** This theme is centered around contacting clients about financial planning options. Terms like *"access"*, *"clients"*, *"plan"*, and *"options"* indicate impersonation of financial advisors offering investment strategies or packages.
* **Topic 2: Personalized Scam Messages to Locals** This topic features casual, friendly introductions like *"hi"*, *"im"*, and *"yes"*, often followed by *"secure"* and *"Singaporeans"*. These are attempts to build trust through humanized messaging, targeting locals with a friendly, secure-sounding offer.
* **Topic 3: SG Investment Promotions**  
   This is a Singapore-targeted investment pitch, with keywords like *"sg"*, *"capital"*, *"investment"*, and *"offers"*. Likely trying to sound legitimate by anchoring to regional economic themes or real investment brands.
* **Topic 4: Account Security Alert** This topic focuses on fake account security notifications, with words like *"account"*, *"security"*, *"ref"*, and *"alert"*. Common phishing tactic designed to trick users into clicking on links under the guise of protecting their account.

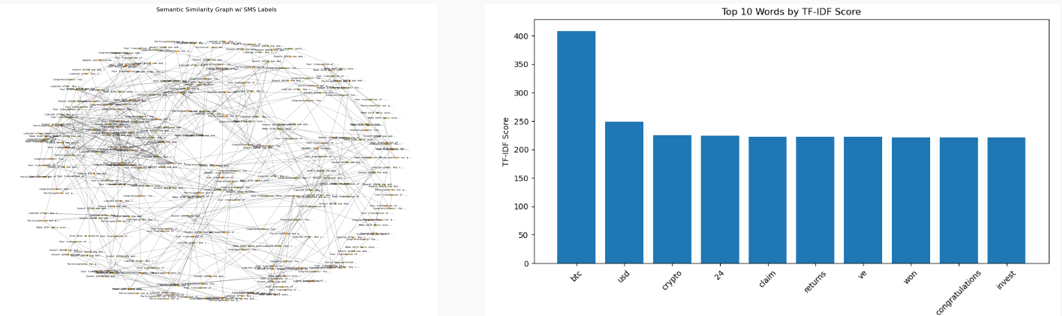
## Cryptocurrency Scam Dataset Details



**Text length Distribution:** The text length distribution for cryptocurrency scam messages is highly irregular, with a sharp peak between 47–50 characters. This reveals that many messages are unusually short, especially when compared to financial or e-commerce scam messages. The extreme concentration at lower lengths suggests that crypto scams prioritize brevity and urgency, often bypassing narrative buildup in favor of instant call-to-action. Given this tight clustering, text length alone is not a reliable discriminator for classification.

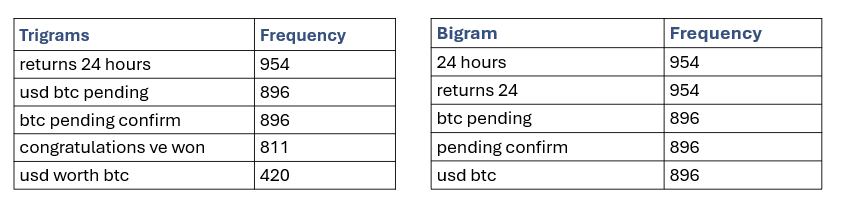


**Word Cloud**: The word cloud and TF-IDF frequency distribution highlight a hyper-focused thematic vocabulary centered around cryptocurrency assets, time pressure, and financial reward. Words such as *“btc”* (*2,555 mentions*), *“usd”*, *“returns”*, *“claim”*, and *“confirm”* dominate the dataset. This extreme repetition signals very limited structural diversity, suggesting that a small number of base templates are being reused extensively, often with only minor token variations such as numeric values or wallet references.



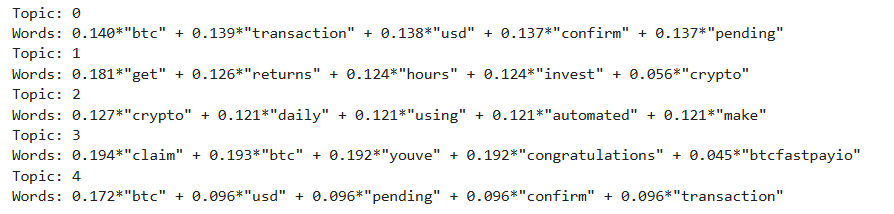
**Semantic Similarity Graph**: Semantic similarity graphs reveal dense message clusters, confirming the heavy reliance on a small set of repeated message formats. Examples include *“Your BTC is ready to claim”*, *“Congratulations! Your transaction is pending”*, and *“Limited offer – claim now”*. Compared to other scam types, crypto messages show greater redundancy and lower topical diversity, heavily anchored in reward-based language and faux transaction alerts.

**TF-IDF scores:** TF-IDF scoring confirms the same trend: *“btc”*, *“usd”*, *“crypto”*, *“24”*, and *“claim”* rank highest. The number *“24”* (e.g., in *“24 hours”*) is repeatedly used to create time-sensitive pressure, pushing victims toward fast decisions. Unlike financial scams which incorporate advisory or institutional language, crypto scams use surface-level terminology with no reference to regulatory frameworks or personal dialogue. This tightly constrained vocabulary creates a highly learnable dataset, but also makes it vulnerable to detection due to lack of diversity.



**Bigram and Trigram:** Bigram and trigram analysis strengthens these observations. Bigrams like *“btc pending”*, *“usd btc”*, *“returns 24”*, and *“pending confirm”* appear over 800 times each, indicating that even simple models could detect these scams due to their rigid phrase structure. Many of these bigrams are synthetic variants constructed by appending token fragments, such as *“ve won”* from *“you’ve won”*. Over 80% of the top trigrams are simply extensions of the top bigrams, e.g., *“btc pending confirm”*, reinforcing that scammers build messages by stacking predefined phrases rather than introducing new linguistic patterns.

**LDA topic analysis results:**

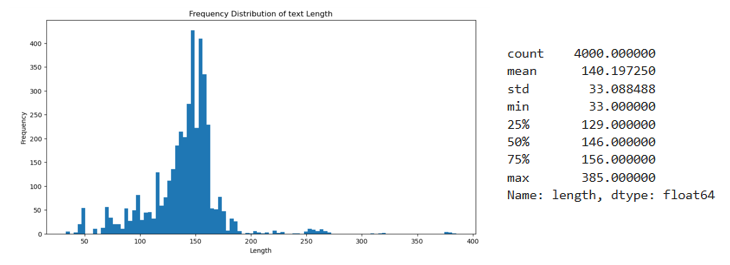
****

LDA topic modeling further validates this conclusion. The five extracted topics include:

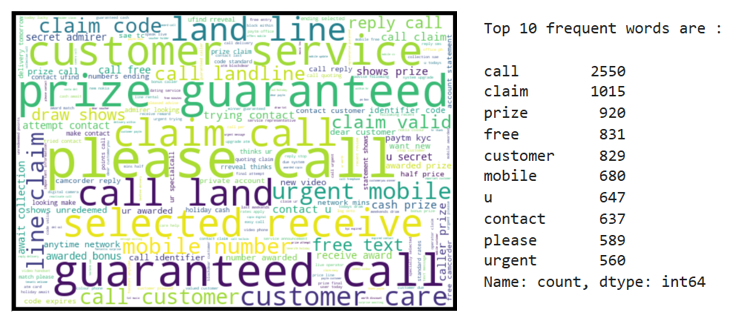
* **Topic 0 – Transaction Status & Alerts:** Keywords like *“btc”*, *“transaction”*, *“usd”*, *“confirm”*, and *“pending”* simulate crypto wallet updates or platform alerts.
* **Topic 1 – High-Yield Investment Offers**: Phrases like *“returns”*, *“claim”*, *“offer”* emphasize urgent investment hooks.
* **Topic 2 – Automated Trading Bots:** Words such as *“daily”*, *“make”*, *“using”* mimic scams involving trading automation.
* **Topic 3 – Reward Notifications:** Phrases include *“congratulations”*, *“won”*, *“btc”*, built to trigger excitement and fast action.
* **Topic 4 – Reinforced Confirmation Urgency:** Mirrors Topic 0 but with stronger emphasis on repeated *“confirm”*/*“pending”* cycles.

These topics reinforce the crypto scam category’s reliance on a high-pressure, low-context reward narrative. Compared to financial or phishing messages, crypto scams prioritize rapid engagement over legitimacy-building.

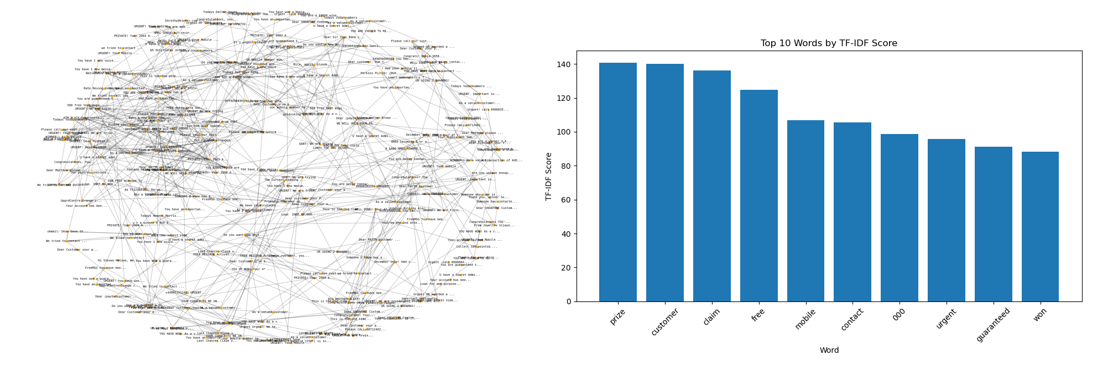
## Phishing Scam Dataset Details



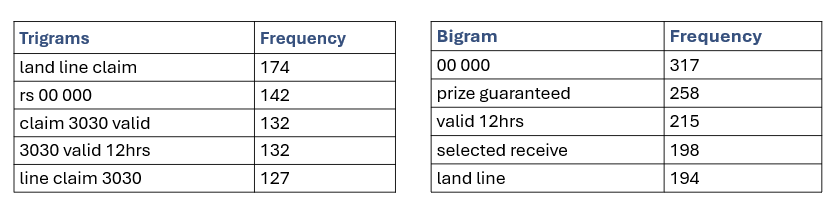
**Text length Distribution:** The text length distribution for phishing scam messages exhibits a wide variance, spanning from 33 to 385 characters. The average length sits around 140 characters, with a standard deviation of 33.09, indicating greater message variety than seen in e-commerce or crypto datasets. A bimodal pattern emerges: one group contains short, urgent alerts, while another features longer procedural imitations, such as fake banking flows or verification protocols. This dual-style approach reflects an adaptive phishing strategy designed to appear realistic while also bypassing filters.



**Word Cloud**: TF-IDF results and word cloud analysis show that phishing scams are dominated by service-oriented vocabulary. The word *“call”* appears an overwhelming 2,550 times, followed by terms like *“contact”*, *“customer”*, *“mobile”*, and *“service”*. These suggest a clear strategy: phishing messages aim to simulate official customer service communication. Additionally, reward-based terms like *“claim”*, *“prize”*, and *“free”* are common, reinforcing *incentivization tactics* often framed as *“you’ve already won”*, unlike crypto scams that focus on potential investment returns.

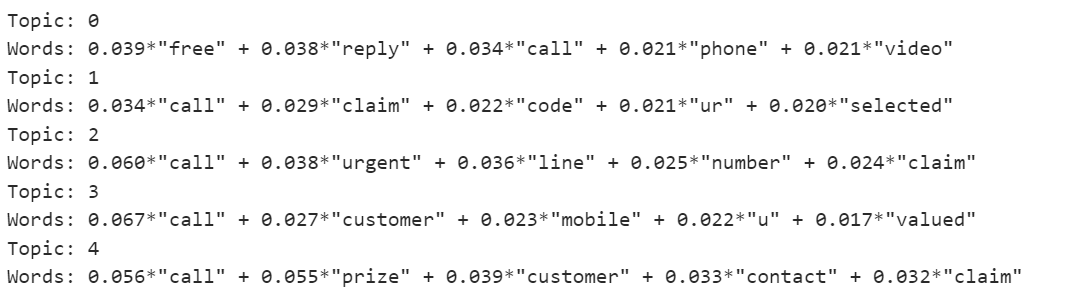


**Semantic Similarity Graph**: The semantic similarity graph shows a tightly clustered core, indicating high reuse of central phishing templates. These messages often begin with *“Dear Customer”*, and differ slightly in urgency phrasing (*“URGENT,” “Important Notice”*) or call-to-action verbs (*“Call,” “Verify,” “Reply”*). This modularity suggests a scam generation process where institutions, urgency, and action phrases are dynamically swapped to maintain freshness while preserving structure. TF-IDF words and n-gram structures show strong alignment. Terms like *“customer”*, *“urgent”*, *“claim”*, *“guaranteed”*, and *“mobile”* appear in both analyses. This overlap reinforces that phishing messages rely on repetitive service-and-reward templates, optimized for user engagement and emotional manipulation.



**Bigram and Trigram:** Bigram and trigram analysis supports this. High-frequency bigrams such as *“prize guaranteed”*, *“mobile number”*, and *“land line”* emphasize reward and voice contact channels. The phishing messages often redirect users to phone lines rather than links, evading detection by not including URLs. Trigrams like *“mobile number awarded”* and *“customer care contact”* are merely mechanical extensions of bigrams, lacking new semantic strategies. Around 80% of trigrams in this dataset derive directly from the most common bigrams, confirming limited creativity and high structural rigidity.

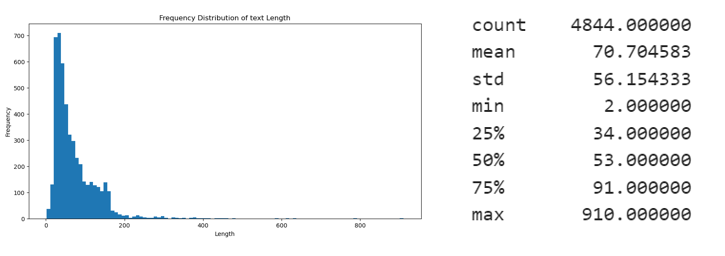
LDA topic modeling reveals five key phishing strategies:



* **Topic 0 – Prize and Giveaway Baiting:** Focused on *“prize”*, *“free”*, *“receive”*, and *“claim”* language.
* **Topic 1 – Customer Service Simulation:** Includes *“service”*, *“call”*, *“contact”*, and *“mobile”* keywords.
* **Topic 2 – Verification Requests:** Centered on *“code”*, *“number”*, *“verify”*, and *“urgent”*
* **Topic 3 – Institutional Mimicry:** Uses words like *“bank”*, *“provider”*, *“account”*, aiming to fake authority.
* **Topic 4 – Reward Confirmation Follow-ups:** Builds on Topics 0 and 2, prompting users to finalize steps for claimed rewards.

Compared to the other scam types, phishing SMS exhibit moderate structural variety, but still reveal core template reliance, especially around prize framing and voice-based engagement. They stand apart in tone, polite yet urgent, and show a deliberate shift away from hyperlinks in favor of realistic, trust-building vocabulary.

## Ham Dataset:

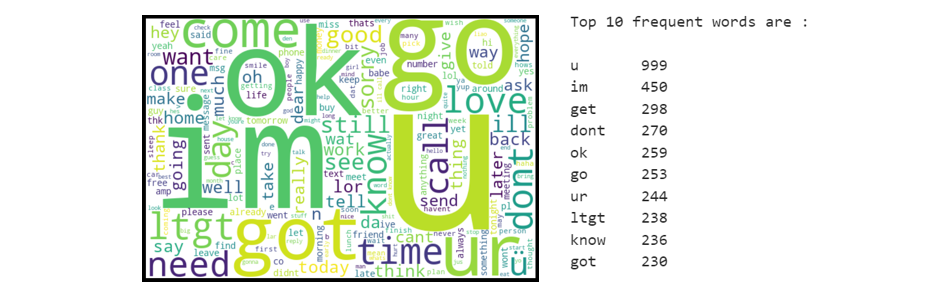


**Text length Distribution:** This Ham dataset exhibits a much wider and more natural variation in message lengths compared to any scam category, peaking at 70 characters. Lengths range from as short as 2 characters to over 910 characters, indicating a full range of real-world usage patterns.

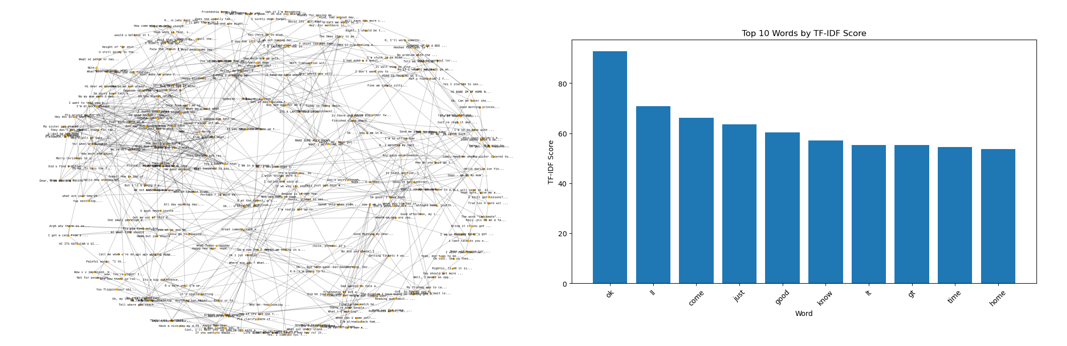
In contrast, scam datasets (phishing, crypto, financial) had tighter length bands, usually clustering narrowly around optimized emotional triggers (around 60–140 characters).

Real communication, by contrast, reflects much more diverse user intent, including casual texts, detailed updates, and even multi-paragraph messages.

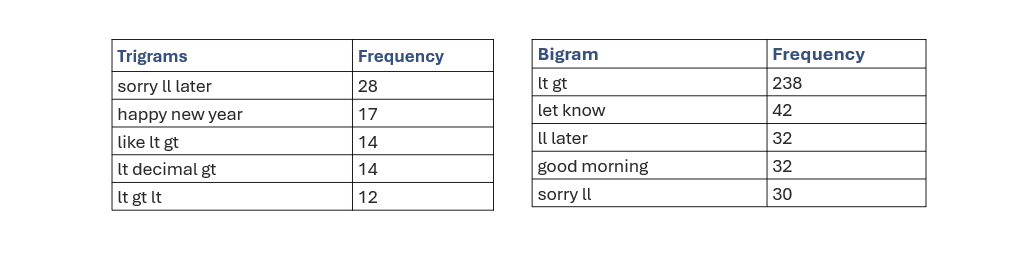
This reinforces the need for smart semantic features rather than naive statistical filters when distinguishing real messages from scams in production detection systems.



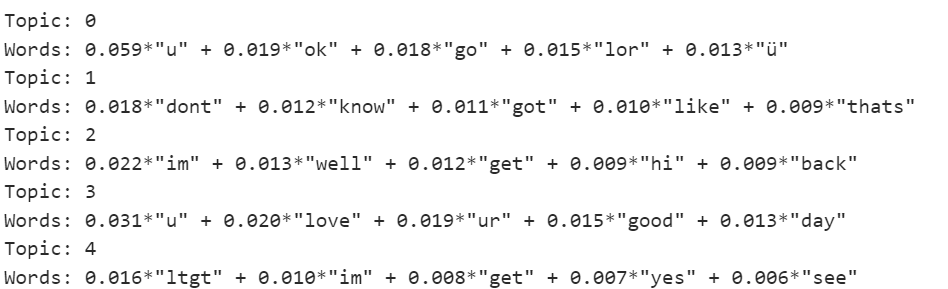
**Word Cloud**: The most frequent words are extremely informal, short, and conversational. Words like *"u", "im", "get", "dont"* dominate the communication landscape. These words are typical of casual, rapid texting behavior. There is a clear personal tone in real SMS exchanges. The high presence of simple verbs like *"go", "get", "know", "want", "make", "call"* suggests that real conversations rely on shared context. These words alone are meaningless without the recipient knowing the full situation (*"I'll get it later", "Do you know where", "Call me when you can"*). This contrasts sharply with scam messages, which are self-contained, fully explanatory, and designed for one-shot action even without prior knowledge.



**Semantic Similarity Graph**: The Ham semantic map displays a wide scattering of small, weakly connected clusters. Messages group into small islands based on broad thematic similarities, but there is no overarching template architecture holding them together. Unlike scam graphs where messages were heavily overlapped because templates were reused with minor changes the Ham messages show semantic uniqueness even when talking about similar things. The Ham graph has no dominant anchor points. Even frequent words like *"ok", "u", "im", "call", "love"* are spread out across many contexts, not concentrating into singular hubs. Authentic SMS communication is context-sensitive, as words derive meaning only relative to broader conversation history and not stand-alone action commands. From a technical perspective, real SMS datasets cannot be easily compressed or clustered into tight semantic groups. Detection models should be cautious when over-compressing real user datasets, as it risks collapsing genuine variability into false positives. Interestingly, "lt" and "gt" noisy tokens also appear in the top TF-IDF scores. These are artifacts from HTML encoding issues during preprocessing and have no semantic contribution. Real-world datasets, even Ham, carry technical noise.



**Bigram and Trigram:** Phrases like *"let know", "ll later", "good morning", "sorry ll", "don know", and "just got"* are all snippets of everyday personal conversation, not transaction-oriented communication. Real SMS communication consists of fragmented exchanges, heavily reliant on previous shared understanding between sender and receiver. From a detection modeling perspective, this suggests that scam detection should not overly penalize short, casual, or emotionally lightweight phrases, because in real-world SMS, these are the norm rather than the exception. Models should instead focus on detecting structured urgency, action-heavy bigrams, and transactional language clusters, which are rare in legitimate texting. The trigram distribution in the HAM dataset further highlights the messy, creative, and emotional nature of real human texting.The trigram distribution in HAM SMS communication showcases the depth, imperfection, and social fabric of real human interactions, a world away from the polished, optimized, single-intent chains found in scam SMS datasets. Detection systems must embrace this complexity rather than flattening it through overly rigid rule-based approaches.

LDA topic analysis results:  
  


The word weights for each topic are significantly lower for Ham. We will look at the reason behind this in the following section.

The decrease in the magnitude of the weights from the E-commerce, Financial and Crypto scams and the Phishing scams and Ham dataset is caused by the greater uniqueness of the Phishing scam and Ham datasets. The other 3 scam groups are often propagated in a copy-paste format, with certain details being customised, leading to a more significant word weight

# 4. Algorithm Design

## Design Overview

To ensure that the deployment decision rests on a broad empirical foundation, exposed a single, uniformly‑pre‑processed SMS corpus (all lower‑cased, stripped of URLS / numbers, lemmatised and stop‑word filtered) to seven distinct modelling pipelines drawn from three methodological families.

First, implemented three traditional machine‑learning classifiers (Logistic Regression, Multinomial Naïve Bayes and a Random Forest). These algorithms provide fast, interpretable baselines and are known to perform strongly on high‑dimensional sparse text.

Second, trained three sequence‑aware deep‑learning networks on dense token‑embeddings (10,000-word vocabulary, 128‑dimensional vectors, padded to 200 tokens): a bidirectional LSTM, a 1‑D convolutional neural network and a GRU.

Third, fitted a Latent Dirichlet Allocation (LDA) model. Aimed to validate the semantic coherence of the labelled classes and highlight borderline messages for manual review and future active‑learning augmentation.

By benchmarking representatives of each algorithmic class (linear, probabilistic, tree‑based, recurrent, convolutional and topic‑based) ensured that the system’s eventual choice of best‑performer is justified.

## Rationale for Algorithm Choice

The corpus ultimately contains about 20,000 labelled SMS messages and remains extremely sparse once vectorised.

Logistic Regression is therefore the preferred production model. As a linear discriminative classifier, it excels in high‑dimensional, sparse feature spaces and, unlike deep‑learning approaches, does not require tens of thousands of unique sentences to generalise well. Each coefficient maps directly to a word, so its sign and magnitude reveal how strongly that token drives a message toward a particular scam class. Then these weights plug cleanly into LIME for token‑level explanations. In production, the trained LR model is under 2 MB and classifies a message in less than 5 ms on commodity CPUS.

Complemented LR with two reference algorithms implemented in the notebook. Multinomial Naïve Bayes (MNB) serves as a classical text‑classification baseline: it assumes that words are conditionally independent once the class is known, so documents are scored purely by the relative frequencies of their tokens. This works well when individual word counts dominate the class signal, giving us a fast, low‑variance yardstick. However, it can misfire when key vocabulary is shared across classes or when word‑to‑word dependencies carry meaning. Also trained a Random Forest Classifier with 100 trees and no depth limit to probe more intricate boundaries, allowing each tree to expand until its leaves are pure. This fully grown ensemble captures non‑linear interactions. Together, MNB and Random Forest broaden the experimental lens.

I explored deep learning techniques **[LSTM, GRU, CNN]** to push the limits of performance and capture more complex language patterns. Deep learning models learn dense vector representations directly from data. As new scam types evolve, deep learning models are adaptable and transferable. They are great for real-world scalability.

To enhance the model's understanding of word sequences and capture complex scam patterns in SMS text, implemented a Bidirectional LSTM neural network.

While **CNNs** are traditionally known for image processing, they’ve proven to be highly effective in text classification tasks due to their ability to extract local patterns in data. Realised that can take advantage of CNN invariance as it learns patterns regardless of where they appear in the sentence. CNN is ideal for learning to spot scam fragments that can be anywhere and make a prediction based on detected patterns.

The **GRU model** was chosen as a lightweight, efficient alternative to LSTM. With bidirectional context and gated memory, it is capable of understanding scam-related sequences while maintaining a lower computational cost. Since GRU uses 2 gates (update and reset) while LSTM uses (input, forget, output). It also requires fewer parameters, and it is faster. It is great for short-to-medium sequences like SMS messages. The GRU stack mirrors the LSTM setup.

**LDA** was chosen to explore the SMS scam dataset from an unsupervised perspective. The goal was to investigate whether scams naturally cluster into meaningful groups based on their linguistic patterns alone, and to understand the features driving this separation. By using model, aimed to detect distinct thematic patterns and assess how well these unsupervised groupings align with the ground truth label set. Additionally, LDA allowed the discovery of hidden structures in the data, hopefully revealing overlaps and ambiguities where scam messages blended persuasive language with conversational text.

## Innovation and References

Unlike traditional binary spam filters, this project addresses a multi-class classification problem by categorising SMS into five distinct classes: ham, e-commerce scams, financial scams, crypto scams, and phishing. This granular classification provides more actionable insights.

Topping that off with LIME (Local Interpretable Model-agnostic Explanations). It was integrated to provide word-level transparency for each prediction. This ensures the model aligns with Human-Centred AI (HCAI) principles by making decisions explainable, trustworthy, and auditable

The project systematically benchmarks both classical and deep learning models, including Logistic Regression (with TF-IDF vectorisation) and deep architectures such as LSTM, CNN, and GRU, showing how different modelling approaches handle short-form text like SMS differently.TF-IDF vectorisation was used for classical models, while Tokeniser and Embedding layers were applied for deep learning models, based on standard NLP best practices.

# 5. Training Process and Performance

## Data Splitting

The dataset was split into training and testing sets using an 80:20 split ratio for machine learning models and a 70:30 split ratio for deep learning models.

For ML models, an 80% training set was used to maximise exposure to data, since it is typically trained once and relies on the volume of initial data to capture feature patterns effectively. Due to limited data, no separate validation set was created manually. Instead, cross-validation (5-fold CV) was applied during training.

For DL models, a 70% training and 30% testing split was applied. It is trained over multiple epochs, meaning the model sees and updates on the same training data many times, allowing it to learn meaningful patterns even with slightly fewer initial examples. At the same time, allocating a larger 30% test set helps ensure a more robust and reliable evaluation of generalisation across the five scam types, which is important for this multi-class classification task. It was large enough to double as validation during training and final evaluation.

Additionally, stratified sampling was used during the splitting process to preserve the original class distribution across both training and testing sets. This ensures that each scam category is being proportionally represented, preventing any class imbalance from skewing the training process and evaluation metrics.

Model Architecture and Hyperparameters

All ML models were trained on TF-IDF-transformed input, and all models are evaluated using standard accuracy, precision, recall, and F1-score metrics.

**Logistic Regression** was trained on TF-IDF features with a vocabulary capped at 5,000 tokens. A maximum of 1,000 iterations ensured convergence, and L2 regularisation was applied by default to mitigate overfitting. Multi-class classification was handled using the One-vs-Rest strategy, and probability outputs supported integration with LIME for explainability.

**Multinomial Naive Bayes** used the default smoothing parameter (alpha=1.0) to manage unseen tokens during inference. No additional tuning was applied, as the model is well-suited for discrete, tokenised text data and provides strong baseline performance with minimal configuration.

**Random Forest** consisted of 100 decision trees (n\_estimators=100) with no depth limit, allowing each tree to expand until pure leaf nodes were achieved. The model ensemble aggregates the predictions of multiple fully-grown trees to capture complex, non-linear decision boundaries.

The **LSTM and GRU** models shared a similar architecture. Both began with an Embedding layer that mapped the top 10,000 tokens to 128-dimensional dense vectors. This was followed by either a Bidirectional LSTM / GRU layer, each with 64 units, allowing the models to capture contextual information from both forward and backwards directions within the sequence.

A Dropout layer with a rate of 0.4 was applied for regularisation. It then passed through a Dense layer, followed by another Dropout layer with a rate of 0.3. The final layer was a Dense layer with 5 units and softmax activation..

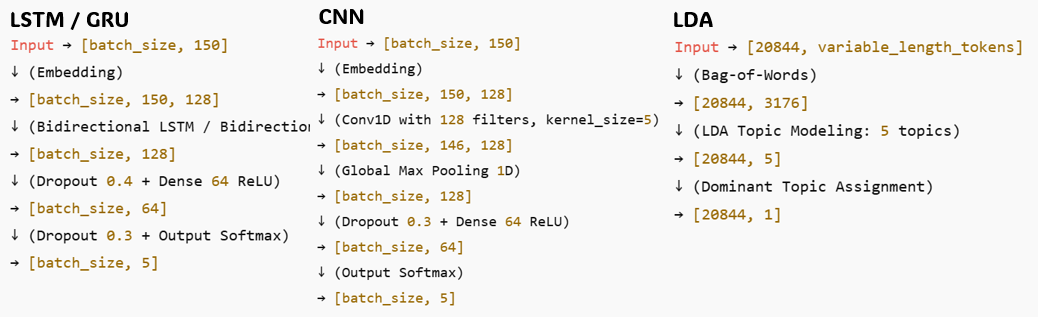
Both models were compiled using the Adam optimiser and sparse categorical crossentropy loss, selected to handle integer-encoded labels. This setup aligned naturally with the softmax activation used at the output layer, enabling the models to output a probability distribution over the scam classes.

**CNN** model began with an Embedding layer, projecting the top 10,000 tokens into 128-dimensional dense vectors. The input shape was [64, 150], corresponding to the padded SMS sequence length. After embedding, the output shape became [64, 150, 128]. This was followed by a Conv1D layer with 128 filters and a kernel size of 5, sliding across the sequence to extract local patterns. The output shape after Conv1D became [64, 146, 128] due to valid padding.

A Global Max Pooling 1D layer then collapsed the sequence dimension, reducing the tensor to [64, 128] by retaining the maximum activation across each filter. A Dropout layer with a rate of 0.3 was applied after pooling. 0.3 was chosen as a moderate regularisation strength to prevent overfitting without significantly slowing convergence, balancing generalisation and learning speed for the clean, high-quality input data.

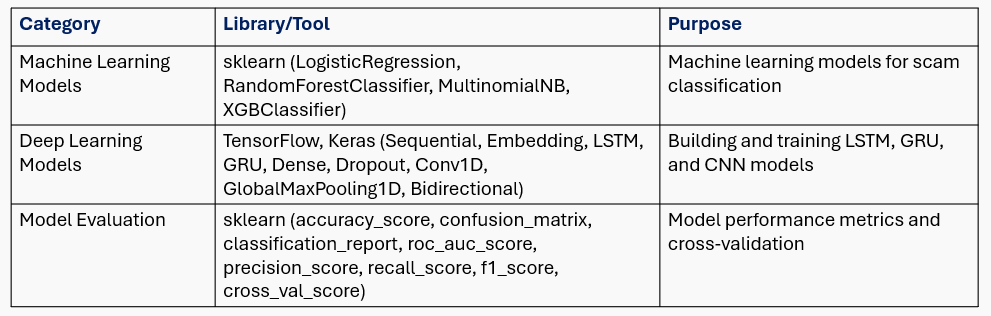
The pooled representation then passed through a Dense layer with 64 units and ReLU activation, before entering the final softmax output layer with 5 units for multi-class scam classification. The model was compiled using the Adam optimiser and sparse categorical cross entropy loss, and trained over five epochs with a batch size of 64.

**LDA** model began by constructing a dictionary from the cleaned SMS corpus, mapping each unique token to an integer ID. Each SMS message was then encoded as a bag-of-words vector, representing token frequencies without regard to order. The input to the LDA model was a corpus of 5,000 documents with a vocabulary size filtered to exclude extremely rare and overly common words. The model was configured to learn 5 latent topics. The alpha parameter was set to 'auto' to allow the model to adaptively control the sparsity of the document-topic distribution, encouraging each SMS message to associate strongly with fewer topics. Training was performed over 10 passes through the entire corpus to ensure topic stability.



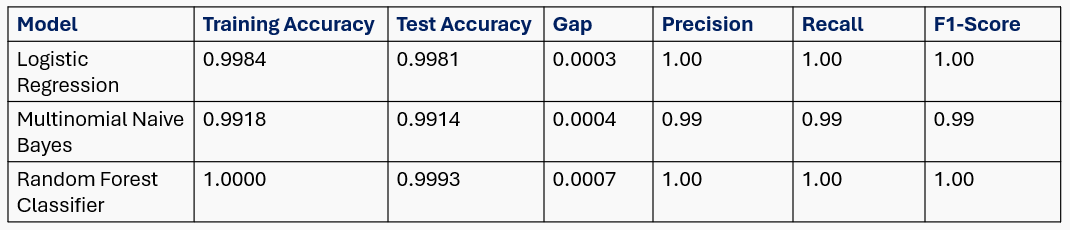
## Tools and Libraries

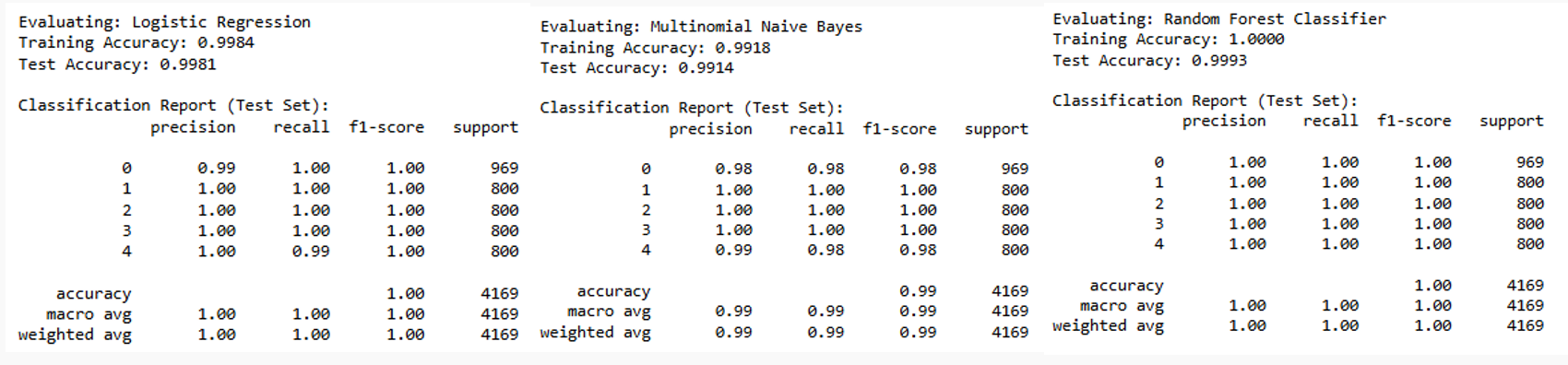




## Performance Metrics:

**Machine learning models:**





**Logistic Regression**

Logistic Regression performed exceptionally well as being a baseline, achieving nearly perfect accuracy and class-wise performance. This level of performance suggests that the combination of synthetic data + TF-IDF features was highly effective at capturing class distinctions. These metrics indicate very minimal overfitting, as performance on unseen data remains consistent with training, with a tiny train-test gap of just 0.0003

Precision (macro-averaged) reached a perfect score, reflecting that when the model predicts a message belongs to a particular scam category, it is almost always correct. Similarly, recall (macro-averaged) was also flawless, indicating the model successfully identified nearly all instances of each scam type without omission.

Logistic Regression stands out as a strong and reliable baseline for SMS scam detection in real-world applications.

**Multinomial Naive Bayes**

It demonstrated strong generalisation with no significant overfitting, as shown by the small train-test gap. Although overall performance was slightly lower than LR, the model still achieved extremely high reliability across scam detection.

It achieved perfect classification for e-commerce, financial, and crypto scams.  
However, minor drops in precision and recall were observed for the ham and phishing classes. This is likely due to textual overlaps and its reliance on word frequencies without modelling deeper context, leading to occasional misinterpretation of neutral or suspicious terms.

Overall, MNB provided an effective, lightweight baseline.

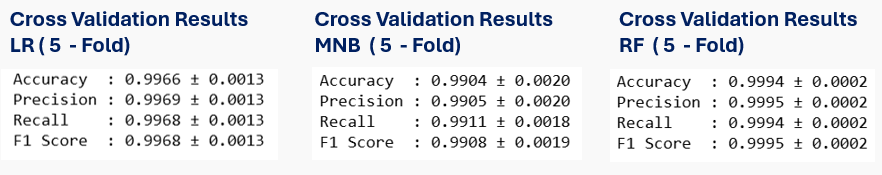
**Random Forest**

It achieved near-perfect performance on both training and testing data, but the 100% training accuracy signals classic overfitting behaviour. This was expected due to the large number of fully grown trees without depth limitation, allowing the model to memorise training data.

However, the extremely small test error suggests that the TF-IDF features are highly separable and the synthetic dataset has minimal noise.

While current results are strong, the model’s performance may be optimistic, and real-world testing on noisy data, along with regularisation like limiting tree depth, would be a necessity.

**Cross Validation**



### **Logistic Regression**

It achieved good and consistent performance across all folds. The low standard deviation (0.0013) indicates high stability across different data splits. This shows that it is a reliable, low-variance, and interpretable baseline, capable of generalising well even with slight shifts in input distributions.

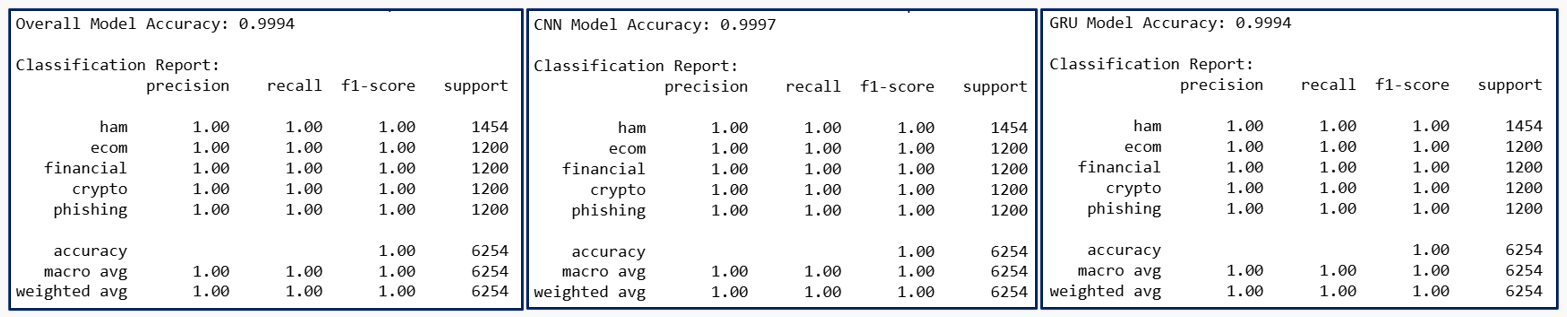
### **Multinomial Naive Bayes**

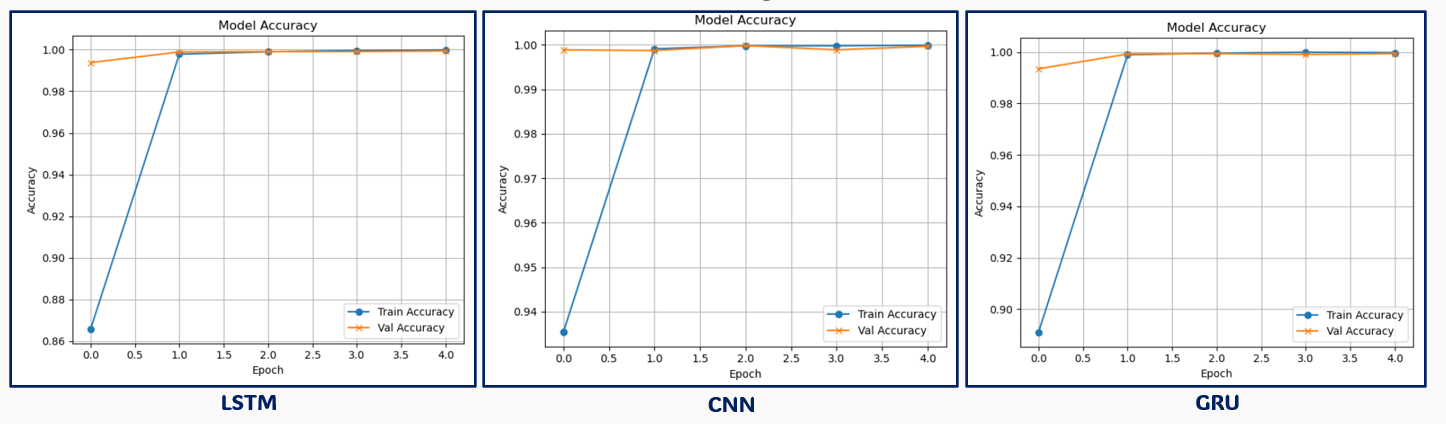
It demonstrated slightly lower average performance (0.9904 accuracy) and showed a higher variance (0.0020) across folds. While still effective and efficient for its simplicity, this variability confirms that MNB is more sensitive to training data distribution changes when classes have textual overlap.

### **Random Forest Classifier**

It achieved the highest cross-validation scores, with almost perfect accuracy and the lowest variance (± 0.0002) among all models. The stability is exceptional, with predictions remaining highly consistent across folds. However, the combination of perfect training accuracy and extremely high cross-validation results raises concerns of overfitting, especially given the synthetic and low-noise nature of the dataset.

**Deep Learning models:**





### **LSTM**

It achieved near-perfect accuracy [0.9994] and uniform class-wise performance, demonstrating that sequential context modelling can be highly effective even on short SMS texts. The balanced precision, recall, and F1-scores across all scam types indicate that the LSTM effectively learned both local and long-range patterns in the TF-IDF token space. While impressive, real-world deployment should include further testing on messy, real SMS data to ensure robustness beyond synthetic conditions.

### **CNN**

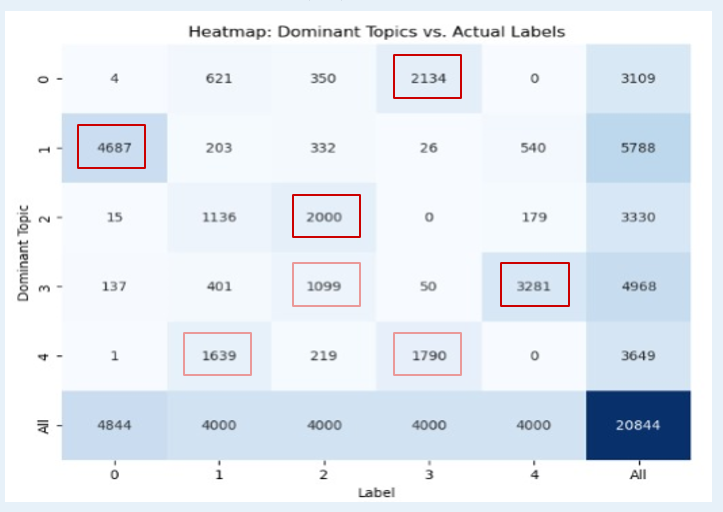
It outperformed with the highest overall accuracy [0.9997] among the deep learning architectures. By focusing on local n-gram patterns through convolutional filters, the CNN was able to capture specific phrases efficiently without using long-term dependencies. Its strong performance and rapid convergence suggest that scam detection in SMS messages often hinges on local textual patterns rather than long sequential context, making CNN a very powerful architecture for this task.

### **GRU**

It closely mirrored the performance of LSTM, achieving similarly high accuracy, precision, recall, and F1-scores. But its simpler gating mechanism compared to LSTM resulted in slightly faster training without sacrificing predictive power. Given its lighter architecture and comparable performance, GRU offers an attractive alternative to LSTM when computational resources or inference speed are a concern.

**Unsupervised Learning:**

### LDA

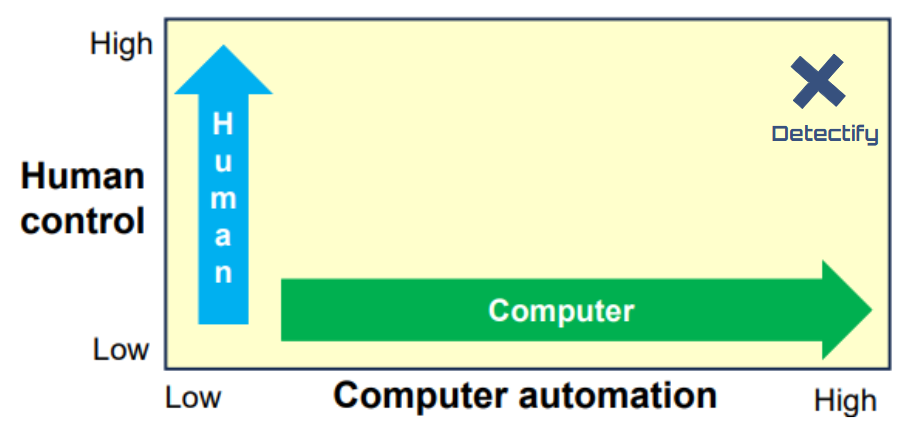
****

**Analysis**

* Topic 0 aligned closely with crypto-related scams, with a large concentration of samples labelled as crypto.
* Topic 1 predominantly captured ham (legitimate) messages, showing the largest single-class grouping across all topics.
* Topic 2 heavily overlapped with financial scams, indicating that LDA successfully distinguished financial fraud-related patterns without supervision.
* Topic 3 mapped strongly to both phishing and financial scams, suggesting that some scam types share similar linguistic structures, leading to topic mixing.
* Topic 4 displayed a shared vocabulary between e-commerce scams and crypto, likely due to overlaps in scam tactics.

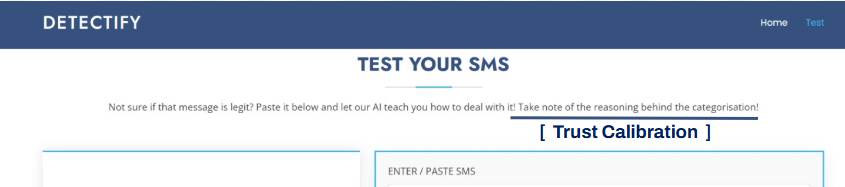
It captures scam behaviour patterns strongly. Can be used to enhance supervised models by inputting LDA topic probabilities into classifiers.

# 8. Human-Centred AI Design

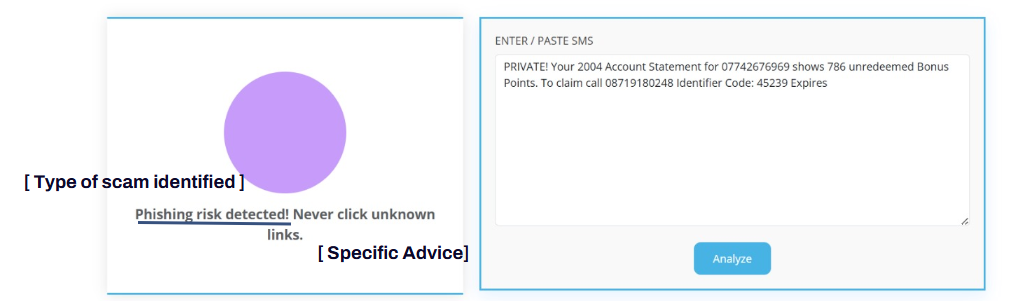


Detectify falls into the top right quadrant, representing both high automation and high human control. The software achieves high automation through the automatic classification of SMS messages. At the same time, it maintains high human control because users assess the provided explanations and make decisions based on the software’s classifications.

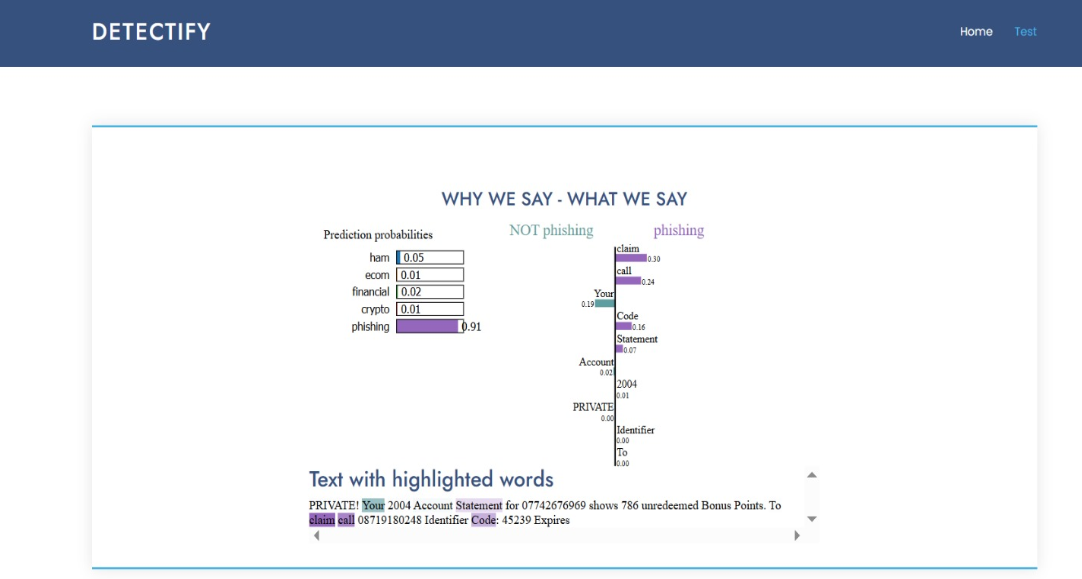
## User experience design elements involved in the website:



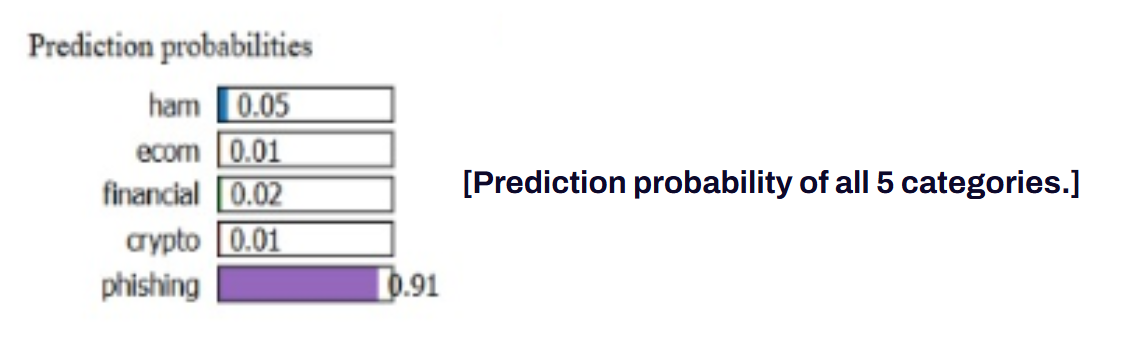
The prompt displayed above the SMS entry field [“*Not sure if that message is legit? Paste it below and let our AI teach you how to deal with it! Take note of the reasoning behind the categorisation!”* ] establishes an appropriate level of user trust. It encourages users to review and understand the reasoning provided for each categorisation. This approach reinforces that users should not easily accept the software’s classification but should critically assess the explanation and identify any discrepancies, reducing the risk of excessive reliance on the system.



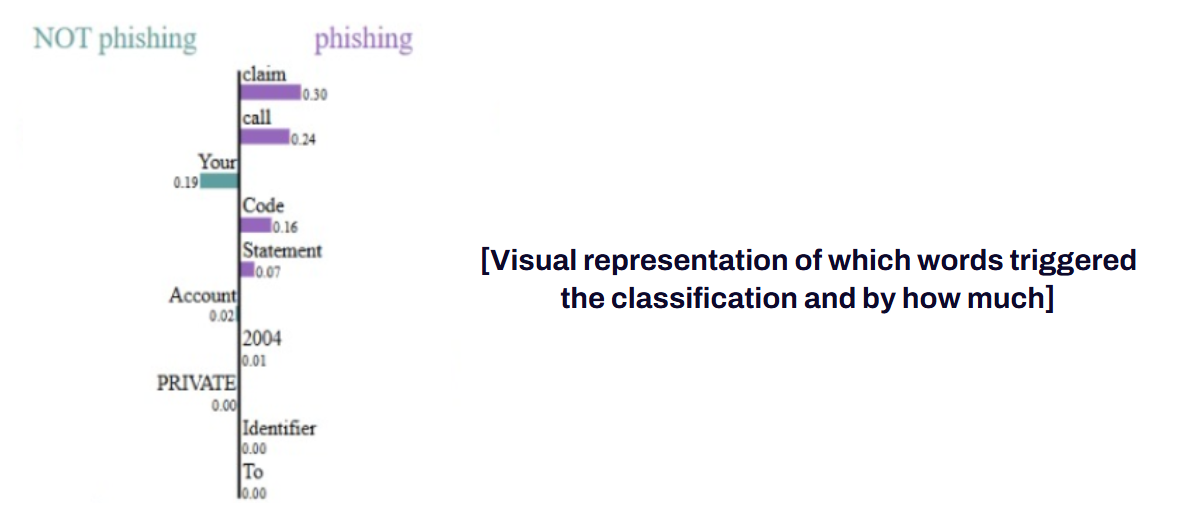
The software identifies the specific type of scam and provides tailored advice relevant to that particular scam. The solution enhances explainability. Additionally, the clarity, contextual relevance, and seamless integration of the explanations contribute to improved usability and user learning.



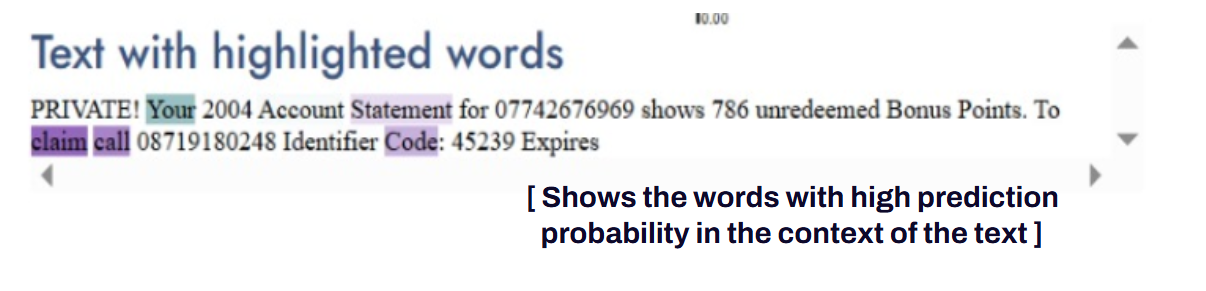
These are the overall prediction probabilities, the probability histogram, and the information from both elements represented as an overlay on the text.



The overall prediction probabilities across all five categories provide an indication of the reliability of the classifications. A decisive output, as illustrated above, is characterised by one category having a significantly higher probability than the others. In contrast, a more probabilistic output is reflected by a more even distribution of probabilities or a smaller gap between them. This information enables users to gauge the confidence level of the classification, thereby strengthening trust in the system. By transparently communicating the quality of the predictions, set an appropriate level of user trust and minimise the risk of dissatisfaction in the event of an incorrect classification.



The probability chart further enhances explainability by illustrating how individual words influence the classification outcome. Words contributing positively to the selected category are displayed on the right side of the chart, while words with negative contributions (those opposing the classification) are shown on the left. This enables users to clearly see which words played a key role in the classification and which were most influential. By providing this level of transparency, the chart supports user understanding and strengthens trust in the product, demonstrating that the solution is interpretable rather than a black-box model.



Finally, the software highlights the top contributing words from the probability histogram within the context of the full SMS message. This contextual presentation enables users to better interpret the explanation, supporting their understanding of which parts of the message warrant greater vigilance. Additionally, by displaying the words within their original phrases, users are guided to recognise suspicious patterns or phrases rather than focusing solely on individual words.

# 8. Conclusion

The project successfully tackles the increasing threat of SMS scams through an AI-based classification system. Achieved strong detection performance, with machine learning models, particularly logistic regression, showing high accuracy among the various algorithms tested. These models were able to reliably distinguish between scam and legitimate messages, demonstrating the potential of AI in enhancing digital safety.

The solution is implemented as a web-based platform, where users can input any SMS they’ve received and instantly see whether it is a scam. If identified as a scam, the platform not only displays the scam category but also explains the reasoning behind the classification. This direct, user-facing approach empowers individuals especially the less digitally literate—to take proactive steps in protecting themselves.

## 

# 9. Future Development Plans

Looking ahead, there are several areas where the project can be expanded and refined to ensure continued relevance and impact.

Even though the ML models outperformed the Deep Learning models with current dataset, the hypothesise that the Deep learning models would perform better if given more genuine data, so have liked to obtain more real and labelled datasets, and also access to a GPU to run the more computationally intensive CNN clustering model.

Secondly, as scammers increasingly adopt generative AI tools to create more convincing and deceptive SMS content, it is vital for model to evolve. Particularly interested in implementing a Generative Adversarial Network (GAN) framework, where one model (the generator) creates realistic scam messages, and another model (the discriminator) attempts to detect them. This adversarial setup would simulate real-world attacks and improve model's ability to catch even the most sophisticated scams. The believe this self-improving loop, where AI learns from AI, offers a powerful direction for enhancing scam detection systems.

Additionally, planning to adopt a semi-supervised learning approach. Since acquiring labeled scam SMS data is expensive and time-consuming, semi-supervised learning would allow the system to learn from a smaller set of labeled data combined with a larger pool of unlabeled messages. This would improve the model’s generalizability and reduce dependency on manually labeled data.

For future improvement, can host of database in cloud or firebase account. Explore the machine learning / deep learning models provided by AWS.

Optimize the user interface through user testing and feedback. This will help us design a more intuitive and accessible experience, ensuring that all users regardless of technical background can easily interpret the alerts and explanations provided by the system. Enhancing the system's scalability and deployment capabilities is also one of the long term goals. While the current solution is a standalone website, future versions could be integrated into real-time environments such as browser extensions or mobile messaging platforms. This would allow users to receive scam alerts instantly upon receiving a suspicious SMS.

Finally, to support broader adoption, aim to expand the system to handle large data volumes and deploy it across multiple communication platforms, such as emails or chat applications. This would extend the impact of the solution and offer more comprehensive protection to users across various digital channels.

# 10. Project Prototype

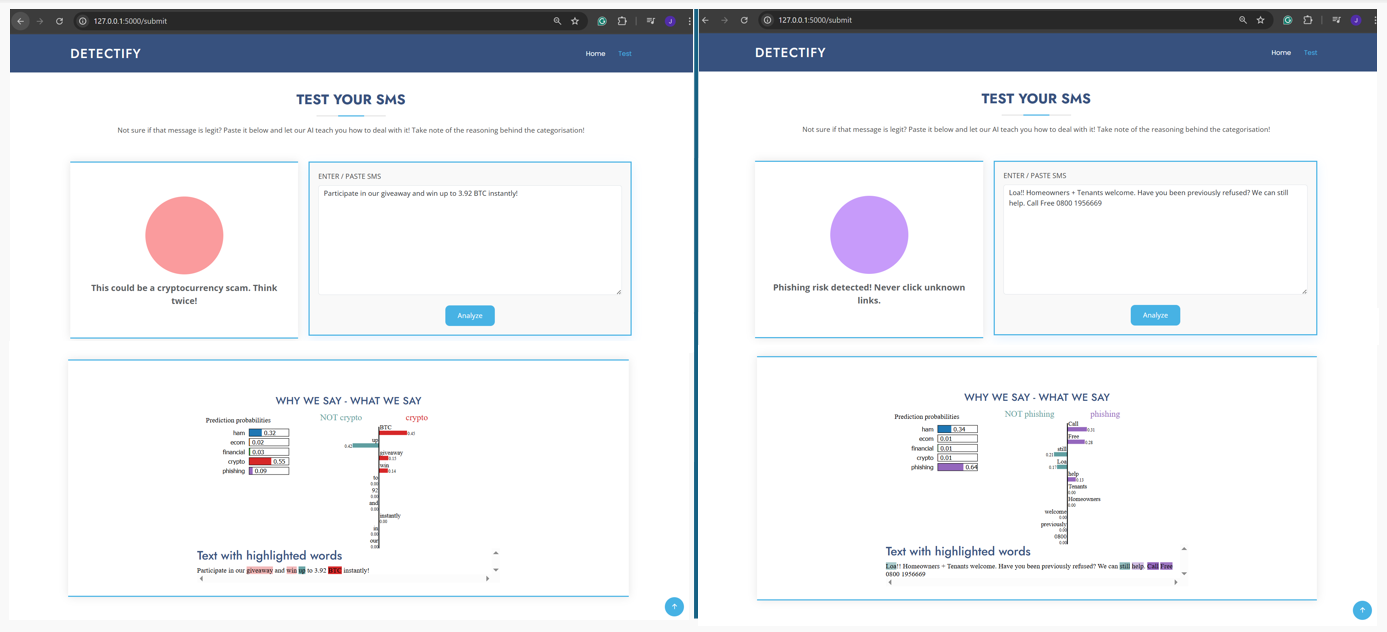
Website Overview - Homepage

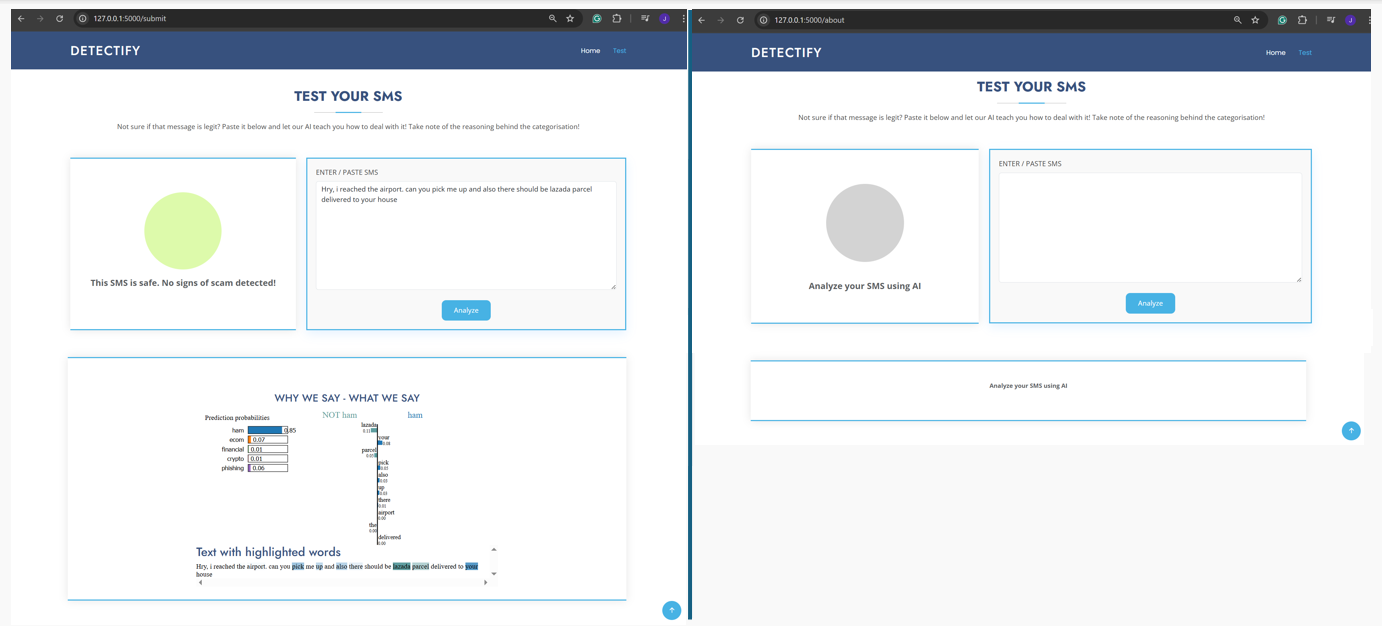
A screenshot of a computer

AI-generated content may be incorrect.

**Website Overview - Prototype Testing**

****

****

****

**End‑to‑End Pipeline**

****

Initially, the Azure student free tier was considered for deployment. However, due to technical limitations, which were the inability to push project files to the server and restrictions on deploying custom machine learning models under the student plan, full deployment on Azure was not feasible.

As a result, the project was hosted on Replit as an alternative platform. Given that this was done on the free version of Replit, the server remained active for only 45 minutes at a time before automatically shutting down. To accommodate this, the server was manually started during the Tuesday presentation, allowing everyone to scan a QR code and interact with the live system during the demonstration.

**References**

* mishra, sandhya; Soni, Devpriya (2022), “SMS PHISHING DATASET FOR MACHINE LEARNING AND PATTERN RECOGNITION”, Mendeley Data, V1, doi: 10.17632/f45bkkt8pr.1