Fraud Email Detection with Explainable Machine Learning Using a Linear SVM and Generative AI

Jyotiraaditya Ghosh[1]

Received January 25, 2026
Accepted May 4, 2026
Electronic access July 15, 2026

Phishing and fraudulent emails remain a persistent cybersecurity threat, leading to stolen identities, severe financial damage, and widespread data breaches. This paper proposes a hybrid system that pairs a Linear Support Vector Machine (SVM) classifier with a generative AI module to address both detection accuracy and result interpretability. The model was trained and evaluated on two datasets: the Enron email archive (29,767 emails) and a Kaggle phishing dataset (18,650 emails), totalling 48,417 emails. The Linear SVM achieved 96.9% accuracy on the phishing dataset and 98.8% accuracy on the Enron dataset, with precision, recall, and F1 scores ranging from 0.96 to 0.99 across both corpora. These figures represent results on two separate datasets, not a single uncertainty range. The generative module, powered by GPT-4o mini, does not alter the classification output; instead, it produces natural-language explanations that highlight suspicious phrases, urgency cues, and contextual signals, thereby improving the transparency and usability of the system. This research demonstrates that combining a high-accuracy traditional classifier with a generative explanation layer offers a practical path toward more interpretable email fraud detection. Completed code can be found at `https://github.com/HazarD000123/Fraud-Email-Detection`.

Introduction

In today's digital world, fraud and phishing emails are increasingly prevalent and cause a variety of issues for companies and individuals. Many people have lost money, had their identities stolen, or had their personal data breached through scam emails. Industry reports indicate that roughly 3.4 billion phishing emails are sent each day, so it is paramount to work to develop systems that can detect them[1].

Extensive research has already been performed for fraud and phishing detection, but attackers continuously find new techniques to deceive users. Normal spam filters are not sufficient anymore, so we see the aggressive push for better solutions based on artificial intelligence. Improving phishing detection models will help minimize attacks and thereby protect users online.

In this research project on phishing detection, the main research objective will be to test the hybrid model, as the proposed hypothesis is that coupling Generative AI with a Linear Support Vector Machine (SVM) will form a more powerful system than a single method. The hybrid model was trained with both the Enron email dataset and a phishing email dataset from Kaggle, creating a composite dataset of legitimate and phishing emails.

The performance of all models was evaluated using accuracy, precision, recall, and F1-score, which are standard metrics for binary classification tasks. These measures collectively show how well the system distinguishes phishing from legitimate emails. Whether such a system could be deployed in real-world settings would require further validation on diverse, personal email corpora before conclusions could be drawn about its suitability for consumer applications[2,3].

Background

Machine Learning Techniques for Fraud Detection

Extensive research exists on the implementation of machine learning for fraud detection in email and financial systems. Early work demonstrated that intelligent detection systems combining rule-based and probabilistic approaches could identify phishing threats in e-banking environments[4]. A comparative analysis of machine learning methods for financial fraud detection examined multiple models, including Random Forest, Support Vector Machines (SVM), deep neural networks[5], and clustering approaches, and highlighted differences in performance and computational costs[6]. Random Forest had the best performance, but K-Means did very poorly in the detection of fraud. The notable contribution of this study was the prototype for a real-time fraud detection pipeline; however, one limitation was that deep learning methods in this study required excessive processing power, thereby excluding the possibility of their use for typical or everyday use. Research has also identified that advanced machine learning models can require significant computing resources for train-

[1] *The International School of Bangalore, Karnataka, India*

ing and operation, which can present challenges for real-time or resource-constrained applications[7].

Logistic Regression vs SVM Models

Another study used Logistic Regression, Linear SVM, and SVM with an RBF kernel to detect fraud in payment transactions. This study achieved a very high recall rate, meaning even though precision was impacted, the study was able to catch cases of fraud, and there was particularly high recall for the money transfer transactional fraud. The authors' base model was further enhanced to achieve better recall using class weighting in the models to take into account for unbalanced data. However, the converse situation with respect to precision presented an impact due to the high number of false alarms for "cash out" transactions being flagged as fraudulent transactions versus false negatives.

Neural Networks for Phishing Detection

In recent times, owing to the ever-increasing sophistication of phishing campaigns, deep learning and neural network based approaches have garnered significant attention among researchers as a more powerful alternative to classical machine learning methods for email threat detection. Unlike traditional classifiers, deep learning models are capable of automatically extracting hierarchical feature representations from raw email text, thereby eliminating the need for extensive manual feature engineering. Convolutional Neural Networks and LSTM-based architectures have been extensively studied in this regard, and have been found to capture both local syntactic patterns and long-range contextual dependencies within email bodies with reasonable effectiveness[8,9]. A comprehensive systematic review undertaken by Kyaw et al. has further corroborated that deep learning models do, in general, outperform their classical counterparts provided sufficient labelled training data is made available[2]. More recently, transformer-based architectures such as BERT and RoBERTa have come to the fore, with Altwaijry et al.[10] and Hosseinzadeh et al.[11] both reporting accuracy figures in excess of 98% on standard phishing benchmark datasets, the latter additionally proposing an adaptive optimisation strategy to address convergence-related limitations. Meléndez et al.[12] have similarly demonstrated, through comparative experimentation, that transformer models significantly outperform traditional classifiers including SVM, Logistic Regression and Naïve Bayes, with roBERTa achieving the highest accuracy of 99.43% on the dataset under consideration. It must however be noted that such deep learning models carry considerably higher computational requirements and offer limited interpretability as compared to simpler approaches, which remains a non-trivial challenge in practical deployment settings where transparency of model predictions is of relevance.

Large Language Models (LLMs) for Scams

More work has been done recently relating to large language models such as GPT-4. One study even included GPT-4 assessing fake messages, both in creating messages that appear legitimate and in classifying them based on natural language processing techniques. This is notable in that it demonstrates that LLMs can both generate convincing scam messages and assist in detecting fraudulent content. However, Jamal et al.[13] demonstrates an improved transformer based mrthod using fintuning for detecting phishing. LLMs also present risks, as the same capabilities used to detect phishing can be abused to generate more convincing scams. Additionally, fine-tuning or retraining LLMs requires technical expertise, which can limit their accessibility[14].

Another study also noted the manner in which AI is being deployed in new types of social engineering scams, such as deepfakes and voice cloning. These not only illustrate that AI can have power, both generating scams as well as halting them, but also that the threat evolves remarkably quickly[12].

Relevance to This Project

The studies above contribute to the understanding that there are machine learning models that exhibit strong capabilities when detecting fraud, but each of them exhibits downsides, such as costs associated with implementation and/or false positive rates when examined too closely. LLMs, like GPT-4, provide a new set of opportunities but comes with a new set of vulnerabilities as well. This project takes both techniques - employing a Linear SVM that is an effective method of classification, and a generative AI model that brings a greater depth of understanding to develop a balanced yet powerful fraud email detection system[15–17].

Dataset

In this study, two principal datasets were used, both sourced from the same Kaggle repository. The first was the Enron email dataset, comprising 29,767 emails drawn from the Enron corporation's internal communications archive, of which 15,869 carry a legitimate label and 13,898 a phishing label as assigned by the dataset creator — it is to be noted that this corpus contains both classes and is not a legitimate-only collection as is sometimes assumed. The second dataset, phishing_email.csv, contains 18,650 emails labelled as either "Safe Email" or "Phishing Email" across two columns — Email Text and Email Type — wherein 8,248 emails are of the safe class and 10,402 belong to the phishing class, reflecting a mild imbalance in favour of phishing instances. Both datasets share

the same three-column structure and were verified to contain zero missing or invalid entries after filtering. Combining the two datasets produced a total labelled corpus of 48,417 emails, providing a broad and diverse training and evaluation base. The per-source, per-class, and per-split breakdown is presented in Table 1 below for full transparency.

Table 1 Dataset composition and train–test partition counts for both source files.

Dataset	Class	Full Dataset	Train (80%)	Test (20%)
phishing_email.csv	Safe Email	8,248	6,598	1,650
phishing_email.csv	Phishing Email	10,402	8,322	2,080
phishing_email.csv	Total	18,650	14,920	3,730
Enron.csv	Legitimate	15,869	12,695	3,174
Enron.csv	Phishing	13,898	11,118	2,780
Enron.csv	Total	29,767	23,813	5,954
Combined	All	48,417	38,733	9,684

Raw email text data is not directly usable in machine learning applications and considerable pre-processing was therefore required prior to model training. All email text was converted to lowercase, after which apostrophes, punctuation, and a standard set of commonly used English stop words carrying little classification signal were removed. Following this normalisation, TF-IDF (Term Frequency–Inverse Document Frequency) vectorization was applied to the cleaned text so as to transform each email into a numerical feature vector, with the vocabulary capped at 5,000 terms and L2 normalisation applied to the output vectors; unigram tokenization was employed throughout, as bigrams were found to offer negligible additional discriminative value at substantially greater computational cost. The combined dataset was thereafter partitioned into training and test sets in an 80:20 ratio using stratified sampling, thereby ensuring that the class proportions of both phishing and legitimate emails were faithfully preserved in each partition — a particularly important consideration given the mild class imbalance observed in phishing_email.csv.

```
x=df["Email Text"].fillna("")
y=df["Email Type"]
mask = (x.str.len() > 0) & (y.notna())
x = x[mask]
y = y[mask]
print(df.shape)
```

Listing A1 Dataset preprocessing and filtering of empty or invalid email entries.

To determine which algorithm performed best, five machine learning models were configured and evaluated: Logistic Regression, Naïve Bayes, Random Forest, Decision Trees, and Support Vector Machine (SVM). These approaches are all widely used in text classification tasks and together form a comprehensive set of baselines. Each model was trained on the training partition and evaluated on the held-out test partition, with performance assessed across accuracy, precision, recall, and F1-score.

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split

svm_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', max_features=5000)),
    ('classifier', LinearSVC())
])

svm_pipeline.fit(x_train, y_train)
preds = svm_pipeline.predict(x_test)
acc = accuracy_score(y_test, preds)

print("=== Results on first split (20% test) ===")
print(f"SVM Accuracy: {acc:.3f}")
print(classification_report(y_test, preds))

x2_train, x2_test, y2_train, y2_test = train_test_split(
    x2, y2, test_size=0.3, random_state=42
)

svm_pipeline.fit(x2_train, y2_train)
preds2 = svm_pipeline.predict(x2_test)
acc2 = accuracy_score(y2_test, preds2)

print("\n=== Results on second split (30% test) ===")
print(f"SVM Accuracy: {acc2:.3f}")
print(classification_report(y2_test, preds2))

joblib.dump(svm_pipeline, "svm_model.pkl")
print("saved as svm_model.pkl")
```

Listing A2 Linear SVM classification pipeline using TF-IDF features and train–test evaluation.

The Linear SVM emerged as the most effective classifier, achieving the highest accuracy and providing the clearest separation between phishing and legitimate emails. However, traditional classifiers such as SVM offer limited transparency: they do not explain why a given email is flagged as suspicious. To address this interpretability gap, a hybrid system was constructed in which the SVM performs binary classification and GPT-4o mini generates natural-language explanations for emails classified as phishing, highlighting suspicious words, patterns, and contextual cues. This combination aims to make the system both accurate and interpretable, which is important for end-users who may not have technical familiarity with machine learning outputs.

Experiments were conducted using Google Colab and VS Code. The primary Python libraries used were Pandas for data handling, scikit-learn for machine learning, joblib for model serialisation, and the OpenAI API for integrating GPT-

4o mini.

A comparison of all five models showed the Linear SVM to be consistently the strongest classifier. On phishing_email.csv it achieved 96.9% accuracy, and on Enron.csv it achieved 98.8%, with precision, recall, and F1-scores ranging from 0.96 to 0.99 across both datasets. These results confirm that a well-tuned traditional classifier remains highly effective for this task, and that the addition of a generative explanation layer does not diminish classification performance.

Methodology

The primary objective of this study was to develop a classification model capable of distinguishing between legitimate and phishing emails. This required two components: a labelled email dataset and a set of machine learning models for comparative evaluation [18–21].

Two datasets were used to improve the robustness and reliability of the model [22]:

Enron email dataset from Kaggle

Phishing email dataset from Kaggle

Together, this produced a combined corpus of 48,417 labelled emails (18,650 from phishing_email.csv and 29,767 from Enron.csv) available for training and evaluation.

Raw emails contains significant noise and inconsistencies, so before I could train anything, I had to clean them up.

Handle Missing Data: Null values in the 'Email Text' column were addressed by replacing them with empty strings ("").

Data Filtering: Data integrity was ensured by retaining only rows containing both non-empty email text and a valid label in the 'Email Type' column.

Label Encoding: The categorical email types were converted into numerical labels: 0 was assigned to legitimate emails, and 1 was assigned to phishing emails.

Text Normalization: The remaining text data was normalized by converting all characters to lowercase, removing apostrophes and punctuation, and stripping a standard set of commonly used English stop words (such as "the," "a," and "is") that carry little classification signal.

Feature Extraction: The cleaned text was transformed into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This process assigns a weight to each word that reflects its frequency within an individual email relative to its frequency across the entire corpus, thereby emphasising terms that are distinctive to a given document. The feature space was capped at 5,000 terms for computational efficiency.

Text data must be transformed into numerical feature representations before it can be processed by machine learning

```
x=df["Email Text"].fillna("")
y=df["Email Type"]
mask = (x.str.len() > 0) & (y.notna())
x = x[mask]
y = y[mask]
print(df.shape)
```

Fig. 1 Shows a snippet of the preprocessing pipeline used in this study

models; therefore, TF-IDF vectorization was applied:
TF: counts how many times a word appears in an email.
IDF: reduces the weight of words that appear everywhere (like "the" or "and").

This generates a numeric representation of each email, which actually reflects important words for phishing detection. The vectorizer was configured with a vocabulary cap of 5,000 features and unigram tokenization (ngram_range=(1,1)), with L2 normalisation applied to output vectors.

Five machine learning algorithms were evaluated for comparative performance:

Naive Bayes: very simple and quick. Works really well for text because it assumes every word is independent when predicting.

Logistic Regression: outputs probabilities in a statistical model (90% phishing vs 10% legitimate).

Decision Trees: splits emails into branches based on features; for example "does the word password show up?".

Random Forest: uses lots of decision trees to reduce error and overfitting.

Support Vector Machine (SVM): works to get the best boundary (or hyperplane) between phishing and legitimate.

The Linear Support Vector Machine (SVM) was determined to be the best classifier because it is extremely appropriate for working with the sparse, high-dimensional data we obtain through TF-IDF vectorization. The LinearSVC classifier was instantiated with default scikit-learn parameters (C=1.0, penalty='l2', loss='squared_hinge', tol=0.0001, max_iter=1000, random_state=None), with no hyperparameter search conducted. Unlike the tree-based approaches, the SVM algorithm will directly identify a maximum-margin separating hyperplane as a very interpretable decision boundary, separating the phishing from legitimate emails in a complex feature space, with increased classification performance and generalization.

To rigorously ensure the model generalises well, the data was split using stratified sampling to preserve class proportions. The primary evaluation used an 80/20 train-test split (80% for training, 20% for testing). A secondary 70/30 split was also applied to assess consistency of results across partitions. A fixed random seed of 42 was applied to the train-test

```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC

models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Naive Bayes": MultinomialNB(),
    "Linear SVM": LinearSVC()
}

for name, clf in models.items():
    pipeline = Pipeline([
        ('tfidf', TfidfVectorizer(stop_words='english', max_features=5000)),
        ('classifier', clf)
    ])

    pipeline.fit(x_train, y_train)
    preds = pipeline.predict(x_test)
    acc = accuracy_score(y_test, preds)

    print(f"\n=== {name} ===")
    print(f"Accuracy: {acc:.3f}")
    print(classification_report(y_test, preds))
```

Listing A3 Training and comparison of multiple ML models

split to ensure reproducibility.

All models were trained using scikit-learn pipelines, ensuring consistent preprocessing and evaluation across experiments.

```python
svm_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english', max_features=5000)),
    ('classifier', LinearSVC())
])
```

Listing A4 Train–test split strategy for model validation

After training, model performance was measured using the following metrics:
Accuracy (overall correctness)
Precision (out of emails predicted as phishing, how many were correct)
Recall (out of all actual phishing emails, how many were detected)
F1-score (balance between precision and recall)
The Linear SVM achieved 96.9% accuracy on phishing_email.csv and 98.8% on Enron.csv.

Based on these findings, the Linear SVM was selected as the final classifier, as it consistently outperformed all other models across both datasets. A hybrid system was subsequently constructed by pairing the SVM with GPT-4o mini: the SVM performs binary classification, and GPT-4o mini generates natural-language explanations for emails flagged as phishing. The hybrid approach did not alter classification accuracy, precision, or recall; its contribution was entirely in the domain of interpretability, providing human-readable rationale for the model's predictions.

Finally, I saved the trained model for future use:

```
joblib.dump(svm_pipeline, "svm_model.pkl")
```

```
=== Linear SVM ===
Accuracy: 0.969
              precision    recall  f1-score   support

Phishing Email       0.95      0.98      0.96      2196
    Safe Email       0.98      0.96      0.97      3395

      accuracy                           0.97      5591
     macro avg       0.96      0.97      0.97      5591
  weighted avg       0.97      0.97      0.97      5591
```

```
=== Linear SVM ===
Accuracy: 0.988
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4762
           1       0.98      0.99      0.99      4169

    accuracy                           0.99      8931
   macro avg       0.99      0.99      0.99      8931
weighted avg       0.99      0.99      0.99      8931
```

Listing A5 Linear SVM classification results and metrics

Results and Discussion

This section presents the results of the machine learning experiments conducted on two datasets: phishing_email.csv and Enron.csv. For each dataset, five models are evaluated and compared across accuracy, precision, recall, and F1-score. The section also discusses model behaviour, sources of error, and the findings of the hybrid SVM and GPT-4o mini system.

The first dataset, phishing_email.csv, consisted of 18,650 emails labelled as "Phishing Email" or "Safe Email." After preprocessing and applying an 80/20 stratified train-test split (14,920 training, 3,730 test emails), five models were evaluated: Logistic Regression, Decision Tree, Random Forest, Naive Bayes, and Linear SVM.

The Linear SVM model achieved the highest accuracy among all models at 96.9%, with both classes classified with strong precision and recall. Of particular note is the phishing recall of 0.98, which is significant in a security context: the cost of a missed phishing email is considerably higher than that of a false positive, making high recall a priority metric for this task.

Table 2

Model	Accuracy	Precision (Phishing)	Recall (Phishing)	F1 (Phishing)
Logistic Regression	96.6%	95%	97%	96%
Decision Tree	92.1%	88%	92%	90%
Random Forest	96.4%	94%	97%	95%
Naive Bayes	95.3%	96%	92%	94%
Linear SVM	96.9%	95%	98%	96%

Table 2 compares the performance of five machine learn-

ing models on the phishing_email.csv dataset (80/20 split); the Linear SVM achieves the highest accuracy and phishing recall among all models evaluated. Decision Trees achieved the lowest overall performance (92.1% accuracy), likely due to overfitting on high-frequency spam-related terms and limited generalisation ability. Naïve Bayes demonstrated relatively high precision (96%) but lower recall (92%), indicating a tendency to miss some phishing emails while producing fewer false positives. Logistic Regression and Random Forest performed consistently well at 96.6% and 96.4% respectively, achieving balanced precision and recall around 95–97%, reflecting their robustness on high-dimensional TF-IDF feature representations.

In summary, the findings from this study suggest that SVM's unique ability to separate high-dimensional text data using the TF-IDF features made it particularly well-suited to detecting phishing patterns. Decision trees, on the other hand, suffered due to overfitting on some common spam words.

The second dataset, Enron.csv, contains 29,767 emails with both legitimate and phishing labels, and is considerably larger and more varied in content than phishing_email.csv. Evaluated using the same 80/20 stratified split, the Linear SVM achieved 98.8% accuracy on this dataset, again outperforming all other models. The higher accuracy on the Enron corpus is likely attributable to its greater size, which provides the SVM with a richer training signal. It should be noted that the 96.9% result on phishing_email.csv and the 98.8% result on Enron.csv reflect performance on two distinct corpora and should not be interpreted as a single uncertainty range.

```
=== Linear SVM ===
Accuracy: 0.988
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4762
           1       0.98      0.99      0.99      4169

    accuracy                           0.99      8931
   macro avg       0.99      0.99      0.99      8931
weighted avg       0.99      0.99      0.99      8931
```

Fig. 2 Classification report for the Linear SVM model on the phishing_email.csv dataset (80/20 split), showing per-class precision, recall, F1-score, and overall accuracy of 96.9%; the SVM achieves notably high phishing recall (0.98), minimising missed detections.

Performance was evaluated across accuracy, precision, recall, and F1-score for both datasets, with particular emphasis on recall to minimise false negatives (missed phishing emails) and on precision to control false positives (legitimate emails incorrectly flagged as phishing).

Support Vector Machines (SVM): LinearSVC() was used with default settings, with the TF-IDF vectorizer capped at 5,000 features. Increasing the feature count beyond 5,000 yielded negligible improvement in classification performance while substantially increasing training time; the 5,000-feature limit was therefore retained as a practical trade-off between efficiency and accuracy. The SVM consistently produced the highest accuracy across both datasets, confirming its suitability for sparse, high-dimensional text representations.

Random Forest: Random Forest was applied with default settings. This ensemble method generally did not tend to perform as well as SVM. Perhaps this is because the TF-IDF character representation produces very high-dimensional and sparse outputs, which would usually be less effective for tree-based methods. Also, slow to train in comparison to Naive Bayes and logistic regression[23].

Naive Bayes: This method is extremely fast to train and evaluate; however, Naive Bayes has low accuracy because its implementation assumes independence between words in a sentence (which is a common property that does not hold for natural language). Surprisingly, Naive Bayes still performed well comparatively for a baseline.

Decision Trees: One decision tree is likely to overfit the train and test data, especially with a large vocabulary space from TF-IDF. Even with defaults, the accuracy was significantly lower than ensemble methods like Random Forest or margin-based classifiers like SVM.

Logistic Regression: It did quite well and was comparable to SVM. Logistic Regression was able to leverage the linear separability with the TF-IDF features. In most cases, SVM was able to reach slightly higher precision and recall, but generally, it performed well. Logistic Regression was also slower than Naive Bayes, but was much faster than Random Forest.

Even with excellent overall accuracy, there were still errors:
False Positives: Beware, there were legitimate emails with urgent language ("please verify," "immediate response"), in business-like plans flagged as phishing.
False Negatives: While the phishing emails were poor at mimicking legitimate internal emails, they were clever enough to get through, and detecting sophisticated phishing is difficult.
Dataset Bias: Corporate datasets, like the Enron dataset, are unlikely to generalize to personal inboxes.

Beyond the traditional classification models, GPT-4o mini was incorporated to enhance the interpretability of results for end users. The GPT-4o mini API was invoked with default settings (temperature=1.0); no explicit token limit was imposed on the output. Rather than presenting raw model outputs, GPT-4o mini generates natural-language explanations that identify specific cues within an email — such as suspicious URLs, urgency phrasing, and requests for personal information — that are characteristic of phishing attempts[24,25]. The explanations are produced post-classification and do not influence the SVM's decision; their purpose is solely to communicate the model's reasoning in accessible language. It should be noted that GPT-4o mini was only invoked for emails classified as phishing; emails classified as legitimate received

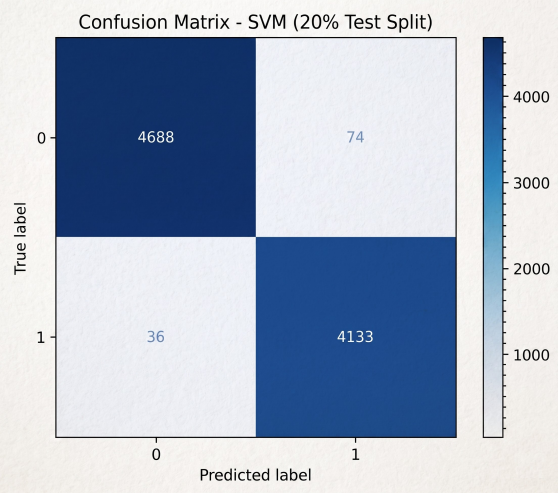

Fig. 3 Confusion matrix for the Linear SVM model on the Enron.csv dataset, showing true positives, true negatives, false positives, and false negatives. The model correctly classified 4,688 legitimate and 4,133 phishing emails, with 74 false positives and 36 false negatives, yielding 98.8% overall accuracy.

no explanatory output, which represents a design choice that could be revisited in future work. For instance, I would have GPT highlight suspicious tendencies such as obscure URLs, urgency, and requests for personal information.

The product was human-format explanations that accompanied the model outputs, functioning as a bridge between simple predictions and a potential end user's understanding. Although this did not affect the accuracy, precision, or recall of the predictions, it did improve the system's transparency and usability.

In summary, the Linear SVM achieved the strongest performance across both datasets, reaching 96.9% accuracy on phishing_email.csv and 98.8% on Enron.csv. The experiments confirm that the high-dimensional nature of TF-IDF features aligns well with the SVM's maximum-margin classification approach, enabling robust discrimination between phishing and legitimate emails. Most misclassifications arose from ambiguous phishing attempts that closely imitated legitimate communication styles. Future work could explore deeper neural architectures alongside continued development of the GPT-based explanation layer, with the goal of maintaining the SVM's classification accuracy while improving transparency and user trust through clearer, verifiable rationale for each prediction.

Approximate training times on were under 10 seconds for LinearSVC, Logistic Regression, and Naïve Bayes, while Random Forest required approximately 30–90 seconds due to its ensemble structure. GPT-4o mini inference introduced an additional 2–5 seconds per email via API call.

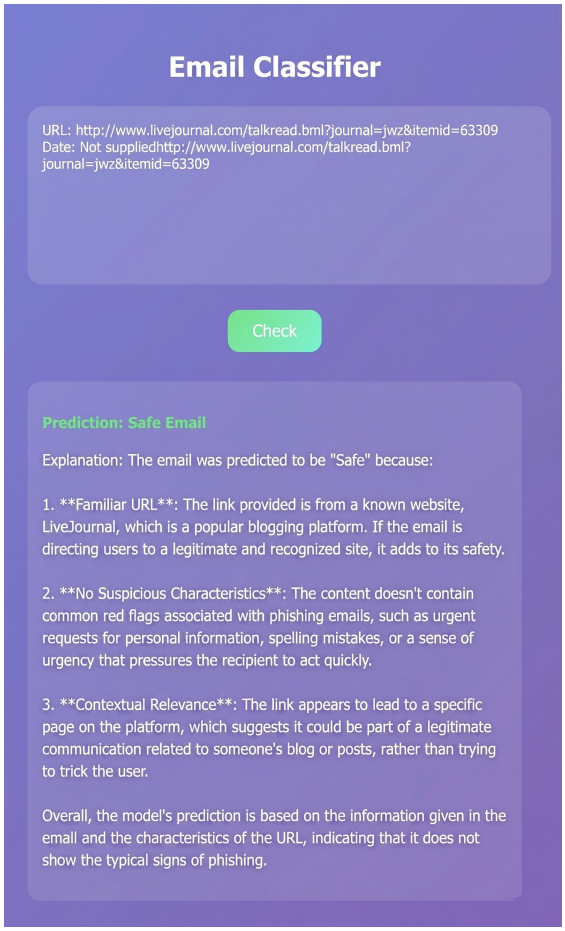

Fig. 4 Example output of the hybrid SVM + GPT system, showing a safe email classification with a natural-language explanation generated by GPT.

Conclusion

This research aimed to develop a reliable phishing and fraud email detection system by combining a traditional machine learning classifier with a generative AI explanation module. The work was motivated by the growing prevalence of phishing attacks, which continue to cause financial loss and data theft for individuals and organisations alike.

Using two labelled email datasets — the Enron corpus (29,767 emails) and the Kaggle phishing dataset (18,650 emails) — a binary phishing classifier was trained and evaluated, producing strong results within the scope of these specific datasets.

Five machine learning models were evaluated: Logistic Regression, Decision Trees, Random Forest, Naïve Bayes, and Linear SVM. The Linear SVM achieved the highest accuracy across both datasets, reaching 96.9% on the Kaggle phishing dataset and 98.8% on the Enron dataset. These margins

over other models, though consistent, are modest, and should be interpreted in the context of single train-test splits without cross-validation. The strong performance is consistent with the SVM's known suitability for sparse, high-dimensional data such as TF-IDF feature vectors.

Although the results are strong within the scope of the datasets used, misclassifications were observed. Legitimate emails containing urgent or high-pressure language were occasionally flagged as phishing (false positives), while sophisticated phishing attempts that closely mimic legitimate communication styles produced false negatives. Several limitations should be acknowledged. The experiments rely on single train-test splits without cross-validation, which means the reported accuracies may not fully represent model stability across different data partitions. The margins by which the SVM outperforms other models are small and have not been verified with statistical significance tests. Additionally, the datasets used — a corporate archive from the early 2000s and a curated Kaggle phishing set — may not generalise to modern personal inboxes, which contain marketing emails, notifications, and other content not represented here. The use of GPT-4o mini also raises a privacy consideration: in a production deployment, email content or features would be transmitted to an external API, which may be unsuitable for sensitive or confidential communications. These constraints indicate that further validation on diverse, real-world corpora is necessary before deployment in consumer-facing applications.

GPT-4o mini was incorporated into the system solely to improve interpretability. The generative module did not affect classification accuracy, precision, or recall; rather, it provided natural-language explanations that identify suspicious components within emails flagged as phishing. This produces a more transparent system that can communicate its reasoning to non-technical users — an important property for real-world trust and adoption. It should be noted that the accuracy of these GPT-generated explanations was not formally evaluated in this study; verifying whether the explanations faithfully reflect the SVM's decision boundaries represents an important direction for future work.

Future research could explore several directions. Deep learning architectures such as LSTMs or fine-tuned transformer models may capture richer semantic patterns than TF-IDF-based approaches. Expanding the training data to include personal, non-corporate email corpora would help reduce dataset bias and improve real-world generalisability. Formal evaluation of the GPT-generated explanations — for example through blinded human rating or faithfulness metrics against the SVM's feature weights — would strengthen confidence in the hybrid system's interpretability. Robustness testing against adversarially perturbed phishing emails and the application of repeated cross-validation to assess result stability are also recommended as next steps.

Acknowledgments

I wish to express my deepest gratitude to my mentor, Mikhail from InspiritAI, for all of his guidance and support throughout the duration of the project. Mikhail helped me with everything from picking datasets to programming/structuring code, to incorporating generative AI, and was incredibly helpful. I also wish to thank my dad. My dad was the first to suggest I use generative AI when imagining this project, which formed the hybrid approach I ended up doing.

References

1 P. Bhatt, M. S. Obaidat, G. Dangwal, A. K. Das, M. Wazid and B. Sadoun, Machine learning-based security mechanism for detecting phishing attacks. 2024 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), pp. 1–6, 2024. https://www.researchgate.net/publication/385581170_Machine_Learning-Based_Security_Mechanism_for_Detecting_Phishing_Attacks.

2 P. H. Kyaw, J. Gutierrez and A. Ghobakhlou, A systematic review of deep learning techniques for phishing email detection. Electronics, vol. 13, p. 3823, 2024. https://www.mdpi.com/2079-9292/13/19/3823.

3 J. L. Wilk-Jakubowski, L. Pawlik, G. Wilk-Jakubowski and A. Sikora, Machine learning and neural networks for phishing detection: A systematic review (2017–2024). Electronics, vol. 14, p. 3744, 2025. https://www.mdpi.com/2079-9292/14/18/3744.

4 M. Aburrous, M. A. Hossain, K. Dahal and F. Thabtah, Intelligent phishing detection system for e-banking using fuzzy data mining. Expert Systems with Applications, vol. 37, pp. 7913–7921, 2010. https://www.researchgate.net/profile/Keshav-Dahal-2/publication/301574035_Knowledge_Management_on_Asset_Management_for_End_of_Life_Products/links/57875a0208aeac8561de0f75/Knowledge-Management-on-Asset-Management-for-End-of-Life-Products.pdf.

5 R. M. Mohammad, F. Thabtah and L. McCluskey, Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, vol. 25, pp. 443–458, 2014. https://eprints.hud.ac.uk/id/eprint/19220/3/RamiPredicting_Phishing_Websites_based_on_Self-Structuring_Neural_Network.pdf.

6 Z. Liu, A comparative study of machine learning methods in financial fraud detection. 2024 2nd International Conference on Finance, Trade and Business Management (FTBM), pp. 389–397, 2024. https://www.atlantis-press.com/article/126004359.pdf.

7 J. Jin and Y. Zhang, The analysis of fraud detection in financial market under machine learning. Scientific Reports, vol. 15, p. 29959, 2025. https://www.nature.com/articles/s41598-025-15783-2.pdf.

8 S. Atawneh and H. Aljehani, Phishing email detection model using deep learning. Electronics, vol. 12, p. 4261, 2023. https://www.mdpi.com/2079-9292/12/20/4261.

9 A. Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar and E. A. Elsoud, An intelligent cyber security phishing detection system using deep learning techniques. Cluster Computing, vol. 25, pp. 3819–3828, 2022. https://pmc.ncbi.nlm.nih.gov/articles/PMC9107003/pdf/10586_2022_Article_3604.pdf.

10 N. Altwaijry, I. Al-Turaiki, R. Alotaibi and F. Alakeel, Advancing phishing email detection: A comparative study of deep learning models. Sensors, vol. 24, p. 2077, 2024. https://www.mdpi.com/1424-8220/24/7/2077.

11 M. Hosseinzadeh, U. Ali and S. Ali, Improving phishing email detection performance through deep learning with adaptive optimization. Scientific Reports, vol. 15, p. 36724, 2025. https://www.nature.com/articles/s41598-025-20668-5.pdf.

12 R. Meléndez, M. Ptaszynski and F. Masui, Comparative investigation of traditional machine-learning models and transformer models for phishing email detection. Electronics, vol. 13, p. 4877, 2024. https://www.mdpi.com/2079-9292/13/24/4877.

13 S. Jamal, H. Wimmer and I. H. Sarker, An improved transformer-based model for detecting phishing, spam and ham emails: A large language model approach. Security and Privacy, vol. 7, p. e402, 2024. https://onlinelibrary.wiley.com/doi/pdf/10.1002/spy2.402.

14 M. Schmitt and I. Flechais, Digital deception: Generative artificial intelligence in social engineering and phishing. Artificial Intelligence Review, vol. 57, p. 324, 2024. https://link.springer.com/content/pdf/10.1007/s10462-024-10973-2.pdf.

15 A. Alhuzali, A. Alloqmani, M. Aljabri and F. Alharbi, In-depth analysis of phishing email detection: Evaluating the performance of machine learning and deep learning models across multiple datasets. Applied Sciences, vol. 15, p. 3396, 2025. https://www.mdpi.com/2076-3417/15/6/3396.

16 M. A. Fayoumi, A. Odeh, I. Keshta, A. Aboshgifa, T. AlHajahjeh and R. Abdulraheem, Email phishing detection based on naïve Bayes, random forests, and SVM classifications: A comparative study. Proceedings of the 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp. 7–11, 2022. https://www.researchgate.net/profile/Ammar-Odeh/publication/359021995_Efficient_Email_phishing_detection_using_Machine_learning/links/626077561b747d19c299f664/Efficient-Email-phishing-detection-using-Machine-learning.pdf.

17 B. Biswas, A. Mukhopadhyay, A. Kumar and D. Delen, A hybrid framework using explainable AI (XAI) in cyber-risk management for defence and recovery against phishing attacks. Decision Support Systems, vol. 177, p. 114102, 2023. https://www.sciencedirect.com/science/article/am/pii/S016792362300177X.

18 O. Kazeem, Fraud detection using machine learning. IU University of Applied Sciences Research Report, 2023. https://www.researchgate.net/profile/Oladimeji-Kazeem/publication/374083997_FRAUD_DETECTION_USING_MACHINE_LEARNING/links/650cd746c05e6d1b1c21c3b1/FRAUD-DETECTION-USING-MACHINE-LEARNING.pdf.

19 Stanford University, Email fraud detection using machine learning (CS229 project report). Stanford University, 2018. https://cs229.stanford.edu/proj2018/report/261.pdf.

20 Governors State University, Email spam and phishing detection using machine learning (capstone project). Governors State University, 2020. https://opus.govst.edu/cgi/viewcontent.cgi?article=1521&context=capstones.

21 K. Iqbal and M. S. Khan, Email classification analysis using machine learning techniques. Applied Computing and Informatics, vol. 21, pp. 390–402, 2025. https://www.emerald.com/aci/article-pdf/21/3-4/390/10395120/aci-01-2022-0012en.pdf.

22 N. A. Alam and A. Khandakar, Phishing email dataset. Kaggle Datasets, 2024. https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset.

23 M. Adnan, M. O. Imam and M. F. Javed, Improving spam email classification accuracy using ensemble techniques: A stacking approach. International Journal of Information Security, vol. 23, pp. 505–517, 2024. https://link.springer.com/content/pdf/10.1007/s10207-023-00756-1.pdf.

24 Z. Zhang, H. A. Hamadi, E. Damiani, C. Y. Yeun and F. Taher, Explainable artificial intelligence applications in cyber security: State-of-the-art in research. IEEE Access, vol. 10, pp. 93104–93139, 2022. https://ieeexplore.ieee.org/iel7/6287639/9668973/09875264.pdf.

25 N. Capuano, G. Fenza, V. Loia and C. Stanzione, Explainable artificial intelligence in cybersecurity: A survey. IEEE Access, vol. 10, pp. 93575–93600, 2022. https://ieeexplore.ieee.org/iel7/6287639/9668973/09877919.pdf.

Annex

Annex A: GPT Prompt

```
You are a cybersecurity assistant.

An email has been classified by an ML phishing detector.

Email content:
{email_text}

ML Prediction: {prediction} (confidence: {confidence:.2f})

Provide:
1. THREAT LEVEL: Low / Medium / High.
2. Explanation of the potential RISKS if the user interacted with this email.
3. Advice: Should the user BLOCK this sender or mark as safe?
```

Annex B: Model Prediction

```
label_map = {
    '0': "safe",
    '1': "phishing"
}
```

Annex C: GPT Output

Prediction: Safe Email

Explanation: The email was predicted to be "Safe" because:

1. **Familiar URL**: The link provided is from a known website, LiveJournal, which is a popular blogging platform. If the email is directing users to a legitimate and recognized site, it adds to its safety.

2. **No Suspicious Characteristics**: The content doesn't contain common red flags associated with phishing emails, such as urgent requests for personal information, spelling mistakes, or a sense of urgency that pressures the recipient to act quickly.

3. **Contextual Relevance**: The link appears to lead to a specific page on the platform, which suggests it could be part of a legitimate communication related to someone's blog or posts, rather than trying to trick the user.

Overall, the model's prediction is based on the information given in the email and the characteristics of the URL, indicating that it does not show the typical signs of phishing.