Phishing Email Detection with Machine and Deep Learning

Security Use Case – ESILV A5 DIA2

Élèves :
Hugo BONNELL
Alan CASASNOVAS
Michele PAGANO
Manon AUBRY
Fabio COLONNA
Mehdi MAMLOUK

Enseignants :
Soufian Ben Amor

5 décembre 2025

Table des matières

Résumé

Phishing emails remain one of the most pervasive attack vectors in corporate environments. This report evaluates classical machine learning and lightweight deep learning approaches to detect phishing using the Enron Fraud Email Dataset. We describe the threat model, preprocessing pipeline (TF–IDF vectorisation and engineered metadata), four baseline classifiers (Random Forest, SGD, Linear SVM, Decision Tree), and a compact TextCNN. The dataset is highly imbalanced (0.5% phishing), pushing us to prioritise recall and ROC–AUC over raw accuracy. Results show that classical models achieve strong accuracy but suffer from low minority recall, while the TextCNN reaches 95% recall with modest precision, making it suitable as a high-sensitivity stage in a layered defence. We conclude with operational guidance, ethical considerations, and future improvements such as threshold tuning and domain adaptation.

1 Introduction

Phishing is a social-engineering threat that leverages crafted emails to coerce victims into divulging credentials, executing malware, or transferring funds. Email remains attractive to attackers because it is inexpensive, asynchronous, and trusted by default. Traditional rule-based filters are brittle against adversaries who rapidly mutate content. This project investigates how machine learning (ML) and deep learning (DL) can strengthen phishing detection, focusing on linguistic and metadata signals while balancing accuracy, adaptability, and interpretability [1, 2].

1.1 Problem statement

The central question is : *How effective are ML and DL approaches at detecting phishing emails, and what trade-offs exist between accuracy, adaptability, and interpretability in real-world deployment ?*

1.2 Objectives

- Build a reproducible pipeline to ingest, clean, and vectorise email data.
- Compare classical ML models with a lightweight TextCNN tailored to CPU environments.
- Analyse precision/recall trade-offs under severe class imbalance.
- Provide actionable recommendations for secure deployment and future research.

2 Theoretical background

2.1 Phishing and threat landscape

Phishing exploits human trust, urgency cues, and spoofed identities. Attackers iterate quickly, combining benign templates with malicious links or attachments. Defenders must detect novel campaigns without excessive false positives that degrade user trust in filtering systems.

2.2 Machine learning foundations

Classical ML (e.g., Random Forests, linear models, decision trees) relies on engineered features and offers interpretability and low latency. Text vectorisation via TF–IDF captures token frequency while down-weighting common words. Linear models with class weighting handle imbalance by penalising minority misclassification.

2.3 Deep learning foundations

DL models automatically learn hierarchical representations. Convolutional neural networks for text (TextCNN) apply 1D kernels over token embeddings to detect local n-gram patterns, pooling them into fixed-size representations. Compared to transformers, TextCNNs are lightweight and more CPU-friendly, though less expressive.

3 Dataset description and preprocessing

3.1 Dataset origin

We use the Enron Fraud Email Dataset curated by Rao [3], merging Enron corporate emails with phishing/social-engineering samples. The dataset contains 447,417 rows and 32 columns, with 26 object, 4 boolean, 1 float, and 1 integer fields. The target label distribution is extremely imbalanced : 445,090 legitimate emails (99.5%) vs. 2,327 phishing emails (0.5%).

3.2 Data quality

Key observations from the profiling report :

- Missingness : `X-bcc` (100%), `X-cc` (75.5%), `Subject` (4%), `X-To` (1.8%).
- Constant columns removed : `Folder-User`, `Re`, `Source`, `Suspicious-Folders`.
- Text statistics : average subject length 29 characters ; body median 232 characters (max 1,401,129), with 10.9% mentioning money.

3.3 Preprocessing pipeline

The pipeline (implemented in `src/app/services/preprocessing.py`) performs :

1. **Cleaning :** drop constant columns ; drop or impute high-missing columns (threshold 50%).
2. **Text construction :** combine `Subject` and `Body` into `text_combined` if desiered, to have one column to vectorize for DL models.
3. **Feature engineering :** text length ; presence of URLs, emails, phone numbers, money symbols, attachment mentions ; sender/recipient domains with top-k one-hot encoding.
4. **Vectorisation :** TF–IDF on `text_combined` with n-grams (1,2) and up to 10,000 features, cached for reuse.

These steps output a sparse matrix $\mathbf{X}$ and encoded numeric features concatenated for ML models.

4 Model and methodology

4.1 Classical ML models

Implemented in `src/app/services/modelmanager.py`, all models train on an 80/20 stratified split and support 5-fold cross-validation :

- **Random Forest :** 100 trees, parallelised, baseline robustness.
- **SGDClassifier :** logistic loss, high bias/low variance for speed.
- **Linear SVM :** class-weighted hinge loss, balanced for minority recall.
- **Decision Tree :** depth 20, minimum 50 samples per split, class-weighted to mitigate imbalance.

Models can be persisted to `models/.joblib` and reloaded for inference.

4.2 Deep learning : TextCNN

The DL trainer (`src/app/services/dl_trainer.py`) tokenises text via regex, caps sequences to 200 tokens, and limits the vocabulary to the top 20k tokens (by default, can be changed). The architecture stacks :

1. Embedding layer.
2. Parallel 1D convolutions with kernel sizes 3, 4, 5.
3. Max pooling and a small MLP head.

Training uses class weights and a weighted random sampler. Two loss options : cross-entropy or focal loss (gamma 2.0). An optional genetic algorithm can tune the decision threshold for binary problems. Defaults : 4 epochs, batch size 32, learning rate 10^{-3}, sample cap 80k rows to stay CPU-feasible.

4.3 Evaluation protocol

- Metrics : accuracy, precision, recall, F1, ROC–AUC, and class support.
- Splits : 80/20 holdout for classical models ; train/validation split for TextCNN with stratification when possible.
- Cross-validation : 5-fold optional for ML models ; reports mean and standard deviation.
- Imbalance handling : class weighting, sampling, and threshold tuning (DL GA option).

5 Experiments and results

5.1 Baseline ML results

Default preprocessing and TF–IDF were applied to the full dataset (447k emails). Table 1 summarises holdout results.

TABLE 1 – Classical ML performance (phishing class = positive)

Model	Accuracy	Precision	Recall	F1
Random Forest	0.9978	0.98	0.60	0.74
SGDClassifier	0.9932	0.00	0.00	0.00
Linear SVM (balanced)	0.9953	0.53	0.87	0.66
Decision Tree	0.9950	0.56	0.17	0.26

Insights :

- Accuracy is inflated by the 99.5/0.5 imbalance. A naive always-legitimate model scores 99.48%.
- Recall drives security value. The linear SVM improves recall (87%) while keeping manageable precision.
- Decision tree cross-validation yields ROC–AUC 0.82 but misses most phishing attempts without further tuning.

5.2 Deep learning results

We ran several lightweight TextCNN trainings with varied sample sizes and loss functions ; all remain high-recall but low-precision.

Attempting to balance the dataset (oversampling phishing, class weights, and GA threshold search) modestly improved precision (up to 0.15) but recall dropped, and overall F1 remained weak. Another issue is that the training distribution reflects traditional phishing templates ; newer AI-generated campaigns are absent, precisely where a DL model should shine over classical ML. The model is therefore best used as a high-sensitivity first stage, with downstream filtering to control false positives.

Table 2 – TextCNN runs (phishing class = positive)

Run	Accuracy	Precision	Recall	F1
80k samples, TF–IDF + cross-entropy	0.9339	0.070	0.952	0.130
80k samples, TF–IDF + focal (γ=2)	0.9560	0.110	0.880	0.195
50k samples, stricter token cap	0.9480	0.080	0.900	0.146
120k samples, class-weighted + GA threshold	0.9420	0.150	0.820	0.253

5.3 Error analysis

Preliminary inspection shows :

- False negatives often contain benign transactional language with minimal phishing cues.
- False positives correlate with money or attachment mentions ; domain-based features can reduce these.
- Domain encoding is memory-bound ; using truncated top-k domains prevents feature explosion.

6 Project management and organisation

6.1 Team roles

- Hugo Bonnell – Coordination, dockersiation, ressource-intensive experiments, DeepLearning.
- Alan Casasnovas – Classical ML baselines, evaluation.
- Fabio Colonna – Data profiling, documentation.
- Manon Aubry – Documentation, TextCNN implementation.
- Michele Pagano, Mehdi Mamlouk – Classical ML baselines, experiment tracking.

6.2 Workflow

- Version control : GitHub repository with issues for each required feature or bugfix.
- Task tracking : backlog groomed weekly ; milestones for preprocessing, ML baselines, DL prototype and final fixes.
- Reproducibility : `src/requirements.txt`, cached vectorisers, saved models under `models/`.
- Communication : weekly stand-ups before the practice labs.

7 Ethical and operational considerations

- **Bias and fairness :** Corporate-centric datasets (Enron) may under-represent consumer phishing ; domain adaptation is needed before production use.
- **Privacy :** Raw emails can contain personal data. Anonymisation and access controls are mandatory in deployment.
- **Adversarial robustness :** Attackers can paraphrase or obfuscate. Regular model refreshes and monitoring drift are required.
- **Human-in-the-loop :** High-recall stages should feed into analyst triage or user warnings rather than auto-quarantine.

8 Conclusion and future work

Classical ML baselines provide fast, interpretable filters but struggle with minority recall. The lightweight TextCNN dramatically improves recall at the cost of precision, making it a strong candidate for a first-stage detector. Future work should :

1. Tune thresholds (e.g., GA) and focal loss sweeps to raise precision while maintaining recall.
2. Add modern phishing corpora (including AI-generated campaigns) and perform domain adaptation to reduce distribution shift.
3. Incorporate richer features (header forensics, URL parsing, attachment metadata) and explore semi-supervised or active learning to track evolving tactics.
4. Deploy two-stage pipelines (TextCNN or transformer front-end + calibrated linear model) with continual learning and drift monitoring.

A Reproduction guide

- Clone the repository, create a virtual environment, install dependencies via `pip install -r src/requirements.txt`.
- Launch the CLI with `python ./src`, load the dataset, then choose either an ML model or the DL trainer.
- Saved artifacts live in `models/` ; rerun evaluation from the CLI to compute metrics.

B Vocabulary

- TF–IDF : Term Frequency–Inverse Document Frequency ; weights tokens highly when they are frequent in a document but rare in the corpus. Used to build sparse text features for classical ML models.

- ROC–AUC : Area Under the Receiver Operating Characteristic ; probability a random positive is ranked above a random negative, independent of threshold. Used to assess discrimination across all possible cutoffs.
- F1 score : Harmonic mean of precision and recall ; rewards models that balance both. Used when classes are imbalanced and we want a single sensitivity/precision trade-off metric.
- Precision / Recall : Precision is the share of predicted positives that are correct ; recall is the share of actual positives that are caught. Used to understand false-positive vs. false-negative behaviour.
- n-gram : Sequence of n tokens (1-gram = unigram, 2-gram = bigram) ; captures local context. Used in vectorisers to retain phrase-level cues (e.g., "urgent action").
- Class weighting : Training strategy that penalises mistakes on minority classes more to counter imbalance. Used in SVM, decision trees, and DL loss functions.
- Focal loss : Loss that down-weights easy examples and focuses on hard/rare ones ; controlled by focusing parameter γ. Used to stabilise learning under severe imbalance.
- Weighted sampler : Batch sampling that increases the probability of minority-class examples during training. Used to ensure each batch sees enough phishing samples.
- Threshold tuning : Adjusting the decision cutoff on predicted scores to trade precision versus recall. Used post-training to align the model to operational risk tolerance.
- Oversampling : Replicating or synthesising minority-class samples to reduce label imbalance in the training set. Used to expose the model to more phishing variants during training.

C Related considerations

- The open-source project `Phishing-Awareness-Chatbot` (`https://github.com/Hu9o73/Phishing-Awareness-Chatbot`) shows how LLMs can both generate realistic phishing emails and detect them. Its performance depends heavily on prompt context and grounding, and it illustrates that context-aware LLMs can excel where our traditional dataset (mostly older phishing) under-represents modern AI-crafted attacks. Integrating similar context-sensitive generation and detection could strengthen our pipeline against emerging campaigns.

D Bibliographie

[1] Said Salloum et al. « Phishing Email Detection Using Natural Language Processing Techniques : A Literature Survey ». In : *Procedia Computer Science* 189 (2021), p. 499-504. doi : `10.1016/j.procs.2021.05.086`.

[2] Nguyet Quang Do et al. « Deep Learning for Phishing Detection : Taxonomy, Current Challenges and Future Directions ». In : *IEEE Access* 10 (2022), p. 59353-59370. doi : `10.1109/ACCESS.2022.3169884`.

[3] Advaith S. Rao. *Enron Fraud Email Dataset*. Accessed 10 December 2025. 2024. url : `https://www.kaggle.com/datasets/advaithsrao/enron-fraud-email-dataset`.