Phishing Email Detection: Comparing Traditional Machine Learning Models and a Transformer Model

Hadi El Nawfal Daria Koroleva Khalil Nijaoui

Abstract

Phishing emails remain a major cybersecurity threat and are a common way for attackers to steal credentials and sensitive data. We cast phishing detection as a binary text classification task on a public dataset of 119,148 labeled emails and compare four traditional machine learning models (Logistic Regression, linear SVM, Random Forest, and Naive Bayes) with a compact transformer model (TinyBERT) using a shared train/validation/test split. Traditional models use TF-IDF representations of the email text together with a set of structural features, whereas TinyBERT takes the same email text as input and processes it using sub-word tokenization. We evaluate all models using accuracy, precision, recall, and precision-recall curves. Our experiments show that well-tuned linear baselines already achieve strong performance, while the transformer model offers a different recall-cost trade-off within this dataset-specific evaluation. Code and data are available at `https://github.com/daria-koroleva/ml_project`.

1 Introduction

Phishing emails are one of the main ways attackers break into organizations and cause data breaches. IBM's Cost of a Data Breach study reports that phishing is the most common initial breach vector, responsible for about 15% of incidents. Breaches started by phishing cost organizations an average of 4.88 million USD IBM Corporation [2024]. The FBI's 2024 Internet Crime Report logs 859,532 complaints of internet crime with reported losses above 16 billion USD and lists phishing and spoofing among the top three complaint categories Federal Bureau of Investigation [2025], Anti-Phishing Working Group [2025]. These numbers show that email phishing is still a very large and expensive problem, even for organizations that already use secure email gateways and security-awareness training. In a typical phishing email, an attacker impersonates a trusted organization, such as a bank, government office, or well-known company. The goal is to get the victim to click a link, enter login details, send money, or download malware. Because these attacks target people and not software bugs, even well-trained users and well-configured systems still make mistakes. This makes automatic filtering of phishing emails an important layer of defense.

Main contributions

- We build a common experimental setup for phishing-email detection on a large, recent dataset (119,148 emails) with a fixed train/validation/test split.
- We compare four traditional models namely Logistic Regression, Linear SVM, Random Forest, and Naive Bayes, and one transformer model under the same input data.
- We evaluate TinyBERT against the traditional models in terms of accuracy, precision, recall, F1, and PR–AUC.

2 Literature Review

Our experimental design is based on prior work in phishing detection, model selection, feature engineering, transformers, and evaluation methodology. The choice of our traditional baseline models (Logistic Regression, linear SVM, Random Forest, and Naive Bayes) follows classic work that compared machine learning techniques for phishing emails. One of the first systematic comparisons of algorithms for phishing email detection found that SVMs and Random Forests performed best among several methods, including Logistic Regression and Naive Bayes Abu-Nimeh et al. [2007]. Another study introduced the PILFER system and showed that even relatively simple models with well-chosen features can reach high accuracy on phishing email detection Fette et al. [2007]. These studies motivated us to include exactly these traditional classifiers. At the language-model level, our transformer choice is grounded in BERT and its compact variants. BERT was introduced as a bidirectional transformer pre-trained on large corpora that can be fine-tuned to achieve state-of-the-art results on many NLP tasks, including text classification Devlin et al. [2019]. Building on this, TinyBERT was proposed as a distilled version of BERT that is 7.5× smaller and roughly 9× faster at inference while still preserving over 96% of BERT's performance on GLUE benchmarks Jiao et al. [2020]. This efficiency profile is the main reason we selected TinyBERT as our only transformer model: it is realistic to fine-tune in a course project and closer to what a production email filter could deploy. Several recent works apply BERT-style models directly to phishing emails. For example, one study fine-tunes DistilBERT, TinyBERT, and RoBERTa for phishing email detection and reports very high scores (all models above 0.98 in accuracy, precision, recall, and F1 on their dataset), with RoBERTa best overall and TinyBERT offering a competitive but smaller alternative Songailaitė et al. [2023]. Broader systematic reviews show that deep-learning approaches (including transformers) generally outperform traditional models on phishing detection tasks, but at a higher computational cost Kyaw et al. [2024]. These findings guided our decision to compare a compact transformer (TinyBERT) against well-tuned traditional baselines. Phishing datasets are often imbalanced, so standard accuracy can be misleading. A well-known overview of learning from imbalanced data recommends metrics such as precision, recall, and other measures that focus on the minority class He and Garcia [2009].It has also been shown that, for skewed binary classification problems, precision–recall curves are often more informative than ROC curves, and the precise relationship between the two has been analyzed Davis and Goadrich [2006]. In the phishing domain, the IWSPA Anti-Phishing shared task also emphasizes metrics tailored to unbalanced datasets Aassal et al. [2018]. Following this literature, we report not only accuracy but also precision, recall, F1, and precision–recall curves (PR–AUC) for all models. Our dataset choice and basic problem setup are taken directly from previous work that compiled the compiled-phishing-dataset of 119,148 emails labeled as phishing or legitimate. It then compared traditional machine-learning models with transformer models for phishing email detection, finding that transformers can achieve very strong performance while linear models remain competitive and cheaper Meléndez et al. [2024]. The same dataset is publicly available on Hugging Face, which we use in our project Meléndez [2024]. We follow their binary phishing vs.legitimate formulation but construct our own stratified train/validation/test split and restrict the transformer side to TinyBERT rather than larger BERT/RoBERTa models. All our traditional models are implemented using scikit-learn, which provides standard implementations of Logistic Regression, linear SVM, Random Forest, and Naive Bayes.

3 Methodology

3.1 Dataset

We use the compiled-phishing-dataset consisting of 119,148 emails labeled as phishing or legitimate. The dataset is publicly available on Hugging Face and provides email text plus metadata (e.g., sender domain, word and sentence statistics). We create a stratified train/validation/test split with a 30/35/35 ratio to preserve the class distribution across all sets.

3.2 Models

We implement two families of models for phishing-email detection.

Traditional models

- Logistic Regression (LR): linear model for binary classification implemented using scikit-learn's `LogisticRegression` import.
- Linear Support Vector Machine (SVM): margin-based linear classifier implemented using scikit-learn's `LinearSVC` import.
- Random Forest (RF): ensemble of decision trees implemented using scikit-learn's `RandomForestClassifier` import.
- Naive Bayes (NB): probabilistic model commonly used for text implemented using scikit-learn's `MultinomialNB` import.

Transformer model

- TinyBERT: compact transformer obtained by distilling BERT, designed for resource-constrained settings implemented using Hugging Face `Transformers` and trained using Pytorch's `torch`.

3.3 Implementation

Traditional Models

For the traditional models, we consider 5 columns from the dataset:

- Email body text (converted to TF–IDF features)
- Word count
- sentence count
- average words per sentence
- Sender domain (categorical)

A scikit-learn `ColumnTransformer` import applies TF-IDF to the text, standard scaling to numerical features, and one-hot encoding to the domain; this preprocessor is combined with each classifier in a single `Pipeline` shown in Figure 1. Hyperparameters are tuned with 3-fold grid search on the training data using F1-score as the objective.

Transformer Model

For TinyBERT, we fine-tune a pre-trained checkpoint on the same train/validation split. Emails are taken from the dataset `text` field, tokenized with the TinyBERT WordPiece tokenizer as shown in Figure 2 (maximum sequence length 128), and trained for 10 epochs using PyTorch and Hugging Face Transformers. The best checkpoint is selected based on validation performance. Figure 3 illustrates the complete experimental pipeline for both traditional and transformer models.

4 Experimental Results

All models are evaluated on the same held-out test set of $41\,702$ emails using accuracy, precision, recall, F1-score, and PR–AUC. The train/validation/test splits ($35\,744/41\,702/41\,702$ examples) are identical for all experiments, which allows a direct comparison between traditional and transformer-based approaches.

4.1 Performance of Traditional Machine Learning Models

Table 1 summarizes the test performance of the four traditional classifiers. All models achieve very high scores, with Logistic Regression and Linear SVM performing best.

All traditional models achieve strong performance ($F_1 \geq 0.98$). Logistic Regression and Linear SVM perform best with nearly identical metrics ($F_1 \approx 0.989$), while Random Forest and Multinomial Naive Bayes score slightly lower but still above 0.98. All models achieve PR–AUC > 0.998, indicating excellent ranking quality.

Table 1: Test-set performance of traditional and transformer-based models. PR–AUC denotes the area under the precision–recall curve.

Model	Accuracy	Precision	Recall	F1	PR–AUC
Logistic Regression	0.987	0.992	0.986	0.989	0.999
Linear SVM	0.988	0.992	0.986	0.989	0.999
Random Forest	0.984	0.986	0.986	0.986	0.998
Multinomial Naive Bayes	0.979	0.980	0.983	0.982	0.998
Transformer (TinyBERT)	0.940	0.943	0.940	0.939	0.989

Table 2: Confusion-matrix counts on the test set for tuned traditional models. TN = true negatives (legit→legit), FP = false positives (legit→phish), FN = false negatives (phish→legit), TP = true positives (phish→phish).

Model	TN	FP	FN	TP
Logistic Regression	17 749	183	344	23 426
Linear SVM	17 745	187	327	23 443
Random Forest	17 596	336	344	23 426
Multinomial Naive Bayes	17 455	477	395	23 375

Table 2 reveals the error distribution underlying Table 1's metrics. Linear models (LR, SVM) maintain the best balance (1FN rate), while Random Forest sacrifices precision for comparable recall (336 vs 183 false positives, similar false negatives). This pattern suggests TF-IDF space is approximately linearly separable, reducing the advantage of ensemble methods.

4.2 TinyBERT Performance

The best transformer configuration (TinyBERT-based sequence classifier) reaches an accuracy of 0.9395, precision 0.9430, recall 0.9395, F1-score 0.9389, and PR–AUC 0.9888 on the same test set (see the last row in Table 1).

The detailed classification report shows an important asymmetry between classes. For legitimate emails, the model achieves precision 0.98 and recall 0.87; for phishing emails, it achieves precision 0.91 and recall 0.99. The corresponding confusion matrix is in Table 3:

Table 3: Confusion matrix for the transformer model on the test set. Rows indicate true labels; columns predicted labels.

True Legitimate	15 652	2 280
True Phishing	244	23 526

The transformer almost never misses phishing emails (false negatives: 244 out of 23 770), which yields very high recall for the phishing class. However, this comes at the cost of more false alarms: 2 280 legitimate emails are flagged as phishing. In contrast, the traditional models maintain high recall for both classes while keeping false positives lower, as reflected in their higher overall accuracy and F1-scores.

4.3 Reproducibility

All experiments use fixed random seeds (`random_state=42`) and the same stratified split (35,744 train / 41,702 validation / 41,702 test examples). Traditional models are implemented in scikit-learn with the preprocessing pipeline from Section 3.3. Hyperparameters are selected with 3-fold `GridSearchCV` using F1 as the objective as shown in Figure 4, and the final models and metrics are saved as serialized files (`.joblib` / `.json`). TinyBERT is fine-tuned on an NVIDIA RTX 3090 GPU using PyTorch and Hugging Face Transformers with `max_length = 128`, `batch_size = 64`, and `NUM_EPOCHS = 10` as shown in Figure 5. Checkpoints and evaluation outputs are stored in the corresponding `models/` and `results/` directories. The traditional models (CPU-based) are effectively deterministic, while the transformer may show small run-to-run variation due to GPU nondeterminism. Full implementation details and configuration files are provided in the accompanying Jupyter notebooks.

5 Concluding Remarks

This work compared traditional machine learning models with a transformer-based approach for phishing email detection on the same dataset and splits. The results show that classical linear models (Logistic Regression and Linear SVM) combined with TF-IDF features achieve the best overall performance, with F1-scores above 0.98 and very strong precision-recall trade-offs. The transformer based TinyBERT model remains competitive but does not surpass the simpler models in this setting. Its main advantage lies in its very high recall for the phishing class, which leads to fewer missed attacks at the expense of more false positives on legitimate emails. From a practical perspective, this suggests that well-tuned, interpretable linear models can be sufficient and sometimes preferable for production systems where transparency, latency, and resource use are important.

Key Takeaways

- Strong baselines matter: TF–IDF plus linear classifiers can be extremely effective for phishing detection and should be included in any comparison.
- Transformers are not automatically superior: added model complexity must be justified by measurable gains on the target dataset and operational requirements.
- Choose models based on operational trade-offs: TinyBERT may be preferred when minimizing missed phishing emails is critical, while linear models may be preferred when reducing false positives, improving interpretability, and keeping deployment simple are priorities.

Limitations and Future Work

For our limitations, the experiments were conducted on a single email dataset, so our conclusions may not generalize to other domains, languages, or phishing campaigns. Also, only one lightweight transformer variant and a small set of traditional models were evaluated; larger or domain-adapted transformers might yield different results. Moreover, hyperparameter tuning relied on relatively small grids and a fixed validation strategy. Finally, latency, memory footprint, and energy consumption were not measured quantitatively, so efficiency is discussed qualitatively.

Future work can follow several directions: Testing the models on multiple datasets, including multilingual corpora and more recent phishing campaigns, to study robustness under distribution shift. Furthermore, evaluating larger or domain-adapted transformer models and comparing them with TinyBERT to better quantify the capacity–efficiency trade-off. Finally, we conclude our future work with incorporating cost-sensitive learning, probability calibration, and online or incremental learning to better match operational risk and handle concept drift in real deployments.

References

Amr M. El Aassal, Navjot S. Kukreja, Sonia Chiasson, Mohammad Mannan, and Anil Somayaji. Anti-phishing pilot at ACM IWSPA 2018: Evaluating performance with new metrics for unbalanced datasets. In *Proceedings of the 4th ACM International Workshop on Security and Privacy Analytics (IWSPA-AP)*, volume 2124 of *CEUR Workshop Proceedings*, 2018. URL https://ceur-ws.org/Vol-2124/invited_paper_1.pdf.

Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the eCrime Researchers Summit*, pages 60–69, 2007. doi: 10.1145/1299015.1299021. URL https://dl.acm.org/doi/10.1145/1299015.1299021.

Anti-Phishing Working Group. Phishing activity trends report, 4th quarter 2024, 2025. URL https://apwg.org/trendsreports/. Anti-Phishing Working Group quarterly trends report.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 233–240, 2006. doi: 10.1145/1143844.1143874. URL https://dl.acm.org/doi/10.1145/1143844.1143874.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186, 2019. URL https://arxiv.org/abs/1810.04805.

Federal Bureau of Investigation. Internet crime report 2024. https://www.ic3.gov/Media/PDF/AnnualReport/2024_IC3Report.pdf, 2025. FBI Internet Crime Complaint Center (IC3) Annual Report.

Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 649–656, 2007. doi: 10.1145/1242572.1242650. URL https://www.cs.cmu.edu/~tomasic/doc/2007/FetteSadehTomasicWWW2007.pdf.

Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. doi: 10.1109/TKDE.2008.239. URL https://doi.org/10.1109/TKDE.2008.239.

IBM Corporation. Phishing. https://www.ibm.com/think/topics/phishing, 2024. Accessed: 2025-12-05.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, 2020. URL https://aclanthology.org/2020.findings-emnlp.372.

Phyo Hein Kyaw, Tin Maung Maung, Ye Kyaw Thu, Masato Oida, and Phyo Thu Thu Kyaw. A systematic review of deep learning techniques for phishing email detection. *Electronics*, 13(19): 3823, 2024. doi: 10.3390/electronics13193823. URL https://www.mdpi.com/2079-9292/13/19/3823.

René Meléndez. compiled-phishing-dataset. https://huggingface.co/datasets/renemel/compiled-phishing-dataset, 2024. Accessed: 2025-11-14.

René Meléndez, Michał Ptaszyński, and Fumito Masui. Comparative investigation of traditional machine-learning models and transformer models for phishing email detection. *Electronics*, 13(24): 4877, 2024. doi: 10.3390/electronics13244877. URL https://www.mdpi.com/2079-9292/13/24/4877.

Monika Songailaitė, Tomas Blaževičius, and Adas Gelbūda. BERT-based models for phishing detection. In *Proceedings of the 28th Conference on Information Society and University Studies (IVUS)*, volume 3575 of *CEUR Workshop Proceedings*, 2023. URL https://ceur-ws.org/Vol-3575/Paper4.pdf.

A Appendix

```
# Preprocessing pipeline
preprocess = ColumnTransformer(
    transformers=[
        ("text", TfidfVectorizer(
            lowercase=True,
            stop_words="english",
            ngram_range=(1, 2),
            min_df=2,
            max_features=100000
        ), text_col),
        ("num", StandardScaler(with_mean=False), num_cols),
        ("domain", OneHotEncoder(handle_unknown="ignore"), cat_cols),
    ]
)

# Models
models = {
    "Logistic␣Regression": LogisticRegression(
        C=10.0,
        max_iter=1000,
        class_weight="balanced",
        n_jobs=-1
    ),
    "Linear␣SVM": LinearSVC(
        C=1.0,
        class_weight="balanced"
    ),
    "Random␣Forest": RandomForestClassifier(
        n_estimators=300,
        random_state=42,
        n_jobs=-1,
        class_weight="balanced_subsample"
    ),
    "Naive␣Bayes": MultinomialNB()
}
```

Figure 1: Preprocessing pipeline and implemented machine learning models

```python
from transformers import AutoTokenizer
from torch.utils.data import Dataset

# Initialize TinyBERT / DistilBERT tokenizer
if not tinybert:
    model_name = "distilbert-base-uncased"
# model_name already set if using TinyBERT

tokenizer = AutoTokenizer.from_pretrained(model_name)

class EmailDataset(Dataset):
    """Dataset class for email classification with TinyBERT tokenization"""

    def __init__(self, texts, labels, tokenizer, max_length=MAX_LENGTH):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        encoding = self.tokenizer(
            str(self.texts[idx]),
            truncation=True,
            padding="max_length",
            max_length=self.max_length,
            return_tensors="pt"
        )

        return {
            "input_ids": encoding["input_ids"].squeeze(0),
            "attention_mask": encoding["attention_mask"].squeeze(0),
            "labels": torch.tensor(self.labels[idx], dtype=torch.long)
        }
```

Figure 2: Tokenization and dataset preparation for TinyBERT-based email classification

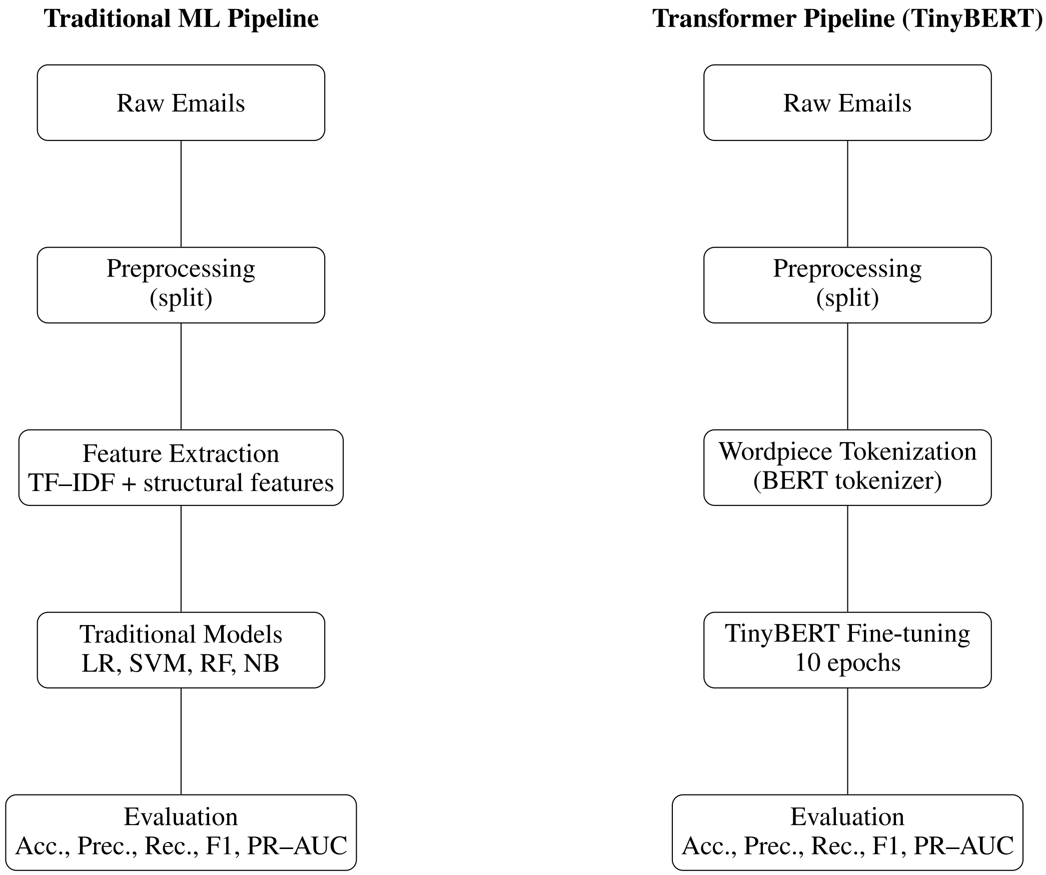

Figure 3: Pipelines for traditional ML models and the TinyBERT transformer model.

```python
# Hyperparameter grids
param_grids = {
    "Logistic Regression": {
        "preprocess__text__ngram_range": [(1, 1), (1, 2)],
        "preprocess__text__max_features": [50000, 100000],
        "clf__C": [0.1, 1, 10]
    },
    "Linear SVM": {
        "preprocess__text__ngram_range": [(1, 1), (1, 2)],
        "preprocess__text__max_features": [50000, 100000],
        "clf__C": [0.1, 1, 10]
    },
    "Random Forest": {
        "clf__n_estimators": [100, 300],
        "clf__max_depth": [None, 20, 40],
        "clf__min_samples_split": [2, 5]
    },
    "Naive Bayes": {
        "preprocess__text__ngram_range": [(1, 1), (1, 2)],
        "preprocess__text__max_features": [50000, 100000],
        "clf__alpha": [0.1, 1.0]
    }
}

# Grid search with cross-validation
grid = param_grids[name]
gs = GridSearchCV(
    base_pipe,
    grid,
    scoring="f1",
    n_jobs=-1,
    cv=3,
    verbose=1
)

gs.fit(Xtr, ytr)

# Best model selection and test evaluation
best_pipe = gs.best_estimator_
test_pred = best_pipe.predict(Xte)
```

Figure 4: Hyperparameter grids and grid search procedure used for model selection

```python
# TinyBERT configuration
model_name = "huawei-noah/TinyBERT_General_4L_312D"  # 4 layers, 312 dims
MAX_LENGTH = 128
BATCH_SIZE = 64
LEARNING_RATE = 3e-4
NUM_EPOCHS = 10

use_mixed_precision = False
use_pruning = False
tinybert = True
```

Figure 5: Configuration of the TinyBERT model and training hyperparameters

Checklist (modelled after NeurIPS Paper Checklist)

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] Yes

   Justification: As discussed in the abstarct and 1, they present the work as an empirical comparison between traditional machine learning models and a lightweight transformer for phishing email detection on a single dataset. The main claims on relative performance, efficiency, and scope are consistent with the experimental results and explicitly acknowledge the restricted evaluation setting.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes] Yes

   Justification: The paper includes a dedicated "Limitations" subsection in Section 5 that discusses the use of a single dataset, the restricted model space, the limited hyperparameter search, the lack of quantitative deployment metrics, and the prototype status of explainability tools. These points explicitly delimit the scope and robustness of the claims.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA] NA

   Justification: The contribution is purely empirical and does not introduce new theorems, formal propositions, or proofs. The paper relies only on standard definitions and off-the-shelf algorithms.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes] Yes

   Justification: As seen in Section 4, the paper describes the fixed data splits, preprocessing pipeline, model architectures, and hyperparameter grids for all methods. It also documents random seeds and the main hardware configuration, and it references the associated notebooks and configuration files. Together, these details provide a clear path to reproducing the reported results up to minor numerical variations.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes] Yes

   Justification: In the abstract, Section 3, and Section 4, we show on an existing publicly accessible phishing email dataset referenced in the text and provides the full experimental notebooks and configuration files as supplemental material. The supplementary archive includes environment specifications and example commands for reproducing the main baselines and proposed models.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] Yes

   Justification: Sections 3 and 4 describe the data splits, feature engineering steps, model families, and hyperparameter tuning strategy. Additional implementation details, such as optimizer choice and batch sizes for the transformer model, are provided in the experimental section and in the accompanying notebooks.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No] No

Justification: The reported metrics are single-point estimates obtained from one fixed train/validation/test split. The paper does not include multiple independent runs, confidence intervals, or formal statistical significance tests, and it acknowledges this as a limitation of the empirical study.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No] No

   Justification: The paper specifies the main hardware used for the transformer experiments (single GPU and operating system) and that traditional models were trained on CPU, but it does not report detailed runtimes, memory usage, or total compute for each experiment. As a result, the compute requirements are only partially documented.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Broader impacts**

   Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

   Answer: [No] No

   Justification: The paper briefly motivates phishing detection as an important security problem but does not include a dedicated broader impact discussion. Potential positive effects (better protection against phishing) and possible negative effects (false positives, attacker adaptation, or integration into larger surveillance systems) are not analyzed in detail.

   Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

10. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA] NA

Justification: The work does not introduce large pretrained models or scraped datasets with a high risk of misuse. It studies task-specific classifiers trained on an existing phishing email corpus, so no special safeguards beyond standard data handling practices are required.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

11. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] NA

Justification: The paper does not release new datasets, benchmarks, or generic pretrained models. All experiments are conducted on an existing dataset using standard toolchains, and no new reusable assets are introduced.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

12. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA] NA

    Justification: The core methods and experiments are based on traditional classifiers and transformer encoders implemented with standard libraries. Large language models are not part of the algorithmic contribution or experimental setup in this work.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.