Report:
Phishing Email Detector Framework with Adversarial Robustness Evaluation

Enrico Ferraiolo 0001191698

Master's Degree in Computer Science

Course: Cybersecurity
Academic Year: 2025-2026

Contents

1 Introduction

Phishing attacks represent one of the most prevalent and damaging cybersecurity threats in modern digital communication. These social engineering attacks exploit human psychology by masquerading malicious communications as legitimate messages, typically to steal sensitive information such as login credentials, financial data, or personal information.

Traditional rule-based email filtering systems often struggle to keep pace with the evolving sophistication of phishing attacks. This race between attackers and defenders necessitates more intelligent, adaptive detection mechanisms.

This project addresses the phishing detection challenge through a dual strategy approach. First, it develops and evaluates a comprehensive framework combining both traditional Machine Learning (ML) and modern Deep Learning (DL) techniques for phishing email classification. Second, and critically important from a cybersecurity perspective, it evaluates the adversarial robustness of these models against data poisoning attacks, a realistic threat scenario where attackers may attempt to compromise the training data to reduce detection efficacy.

The framework implements six distinct classification models across two categories:

- **Machine Learning**
  - Logistic Regression
  - Random Forest
  - XGBoost
- **Deep Learning**
  - LSTM (Long Short-Term Memory) Network
  - CNN (Convolutional Neural Network)
  - TabTransformer (Tabular Transformer Model) [2]

This comparative analysis provides insights into the strengths, weaknesses, and practical deployment considerations of each approach, particularly under adversarial conditions.

2 Project Goals in Cybersecurity

2.1 AI-Based Framework for Phishing Email Detection

The primary objective of this project is to develop a robust, multi-model framework for automated phishing email detection. This framework aims to:

1. **Achieve High Detection Accuracy**: Develop models capable of distinguishing phishing emails from legitimate communications with high precision and recall, minimizing both false positives (legitimate emails flagged as phishing) and false negatives (phishing emails that evade detection).
2. **Compare Multiple Approaches**: Systematically evaluate and compare the performance of traditional ML algorithms against modern DL architectures to identify the most effective techniques for this specific cybersecurity application.
   - Assess the performance trade-offs between model complexity, training time, and detection accuracy.
3. **Feature Engineering**: Extract and leverage multiple informative features from email content, including linguistic patterns, structural characteristics to enhance detection capabilities.

4. **Scalability and Efficiency**: Design models that can process large volumes of emails efficiently while maintaining detection accuracy, crucial for real-world deployment in organizational email systems.

2.2 Adversarial Attack - Data Poisoning on Training Data

A critical yet often overlooked aspect of ML-based security systems (and DL-based as well) is their vulnerability to adversarial attacks. This project specifically investigates data poisoning attacks, where an adversary intentionally corrupts the training dataset to degrade model performance. This represents a realistic threat model where:

- Attackers may compromise data collection pipelines
- Malicious insiders could inject poisoned samples
- Crowdsourced or user-reported data might contain deliberate mislabeling
- Automated labeling systems could be exploited

The adversarial evaluation objectives include:

1. **Poisoning Attack Implementation**: Develop a targeted data poisoning strategy that systematically mislabels legitimate emails containing specific keywords as phishing emails, simulating an intelligent adversary.
2. **Robustness Assessment**: Quantify the degradation in model performance (accuracy, precision, recall, F1-score) when trained on poisoned data compared to clean data.
3. **Model Comparison**: Identify which model architectures demonstrate greater resilience to adversarial data poisoning.
4. **Security Implications**: Provide insights into the practical security considerations for deploying AI-based phishing detectors in adversarial environments.

This dual focus on both detection efficacy and adversarial robustness ensures the framework's relevance to real-world cybersecurity applications, where systems must operate not only accurately but also securely against adaptive adversaries.

3 Data Selection

The effectiveness of any ML or DL system fundamentally depends on the quality and representativeness of its training data. This project employs a carefully curated combination of two complementary datasets to ensure robust model training and evaluation.

3.1 Phishing Email Dataset

Source: Kaggle - "Phishing Emails" dataset [1].

This dataset provides a comprehensive collection of labeled phishing emails alongside safe emails. The dataset characteristics include:

- **Size**: 18,650 total emails initially
- **Composition**:
  - Safe Emails: 11,322 samples
  - Phishing Emails: 7,328 samples

- **Content**: Diverse phishing tactics
- **After preprocessing**: 18,612 emails (38 removed due to insufficient content)

The phishing samples in this dataset exhibit various attack patterns:

- Urgency-inducing language ("immediate action required", "account suspended")
- Suspicious URLs and domain names
- Requests for sensitive information

3.2 Enron Email Dataset

Source: Kaggle - "Enron Email Dataset" [3].

To ensure a balanced and realistic representation of legitimate emails, the project incorporates emails from the famous Enron corpus. This dataset provides:

- **Full Size**: 517,401 emails (complete corpus)
- **Sampled**: 10,000 emails randomly selected for computational efficiency
- **After preprocessing**: 9,996 legitimate emails
- **Characteristics**: Real-world corporate communications including:
  - Internal business correspondence
  - Meeting scheduling and coordination
  - Project discussions
  - Policy announcements
  - Personal communications

The Enron dataset is particularly valuable because it represents genuine, unfiltered business email communications, providing a realistic baseline of legitimate email patterns that a deployed system would encounter.

3.3 Combined Dataset Statistics

After preprocessing and combining both datasets, the final dataset comprises:

- **Total Emails**: 28,341
- **Legitimate (label=0)**: 21,203 (74.8%)
- **Phishing (label=1)**: 7,138 (25.2%)

This class distribution reflects a more realistic scenario where legitimate emails significantly outnumber phishing attempts. The 3:1 ratio allows models to learn robust decision boundaries while maintaining sufficient phishing examples for effective training.

4 Data Analysis and Preprocessing

Effective data preprocessing is crucial for ML model performance. This section details the comprehensive preprocessing pipeline and feature engineering approach employed in the project.

4.1 Text Cleaning Pipeline

The raw email text undergoes several cleaning transformations:

1. **Email Parsing**: Raw email messages containing headers (From, To, Subject, etc.) are parsed to extract the body content using Python's email parsing library.
2. **HTML Removal**: Many emails contain HTML formatting. BeautifulSoup is employed to strip HTML tags and extract plain text content.
3. **Whitespace Normalization**: Multiple consecutive spaces, tabs, and newlines are collapsed into single spaces to standardize text representation.
4. **Minimum Length Filtering**: Emails with fewer than 2 words are removed as they provide insufficient information for classification (38 phishing emails and 4 Enron emails removed).

4.2 Feature Engineering

Beyond raw text, the framework extracts eight engineered features that capture important characteristics of email content. These features serve as inputs to the ML models and as supplementary information for the TabTransformer model.

4.2.1 Lexical Features

1. **num_words**: Total word count in the email body
   - Legitimate mean: 327 words
   - Phishing mean: 306 words
2. **num_unique_words**: Count of distinct words, indicating vocabulary richness
   - Legitimate mean: 150 unique words
   - Phishing mean: 141 unique words
3. **num_stopwords**: Frequency of common English stopwords (the, is, at, etc.)
   - Legitimate mean: 100 stopwords
   - Phishing mean: 90 stopwords
   - Helps identify natural vs. artificially constructed text

4.2.2 Structural Features

4. **num_links**: Count of URLs in the email
   - Legitimate mean: 0.84 links
   - Phishing mean: 0.28 links
   - Surprisingly, legitimate business emails often contain more links
5. **num_unique_domains**: Number of distinct domain names in URLs
   - Legitimate mean: 0.54 domains
   - Phishing mean: 0.20 domains
6. **num_email_addresses**: Count of email addresses mentioned in the body
   - Legitimate mean: 1.12 addresses
   - Phishing mean: 0.16 addresses

7. **num_spelling_errors**: Count of misspelled words detected by PySpellChecker
    - Legitimate mean: 5.58 errors
    - Phishing mean: 6.83 errors
    - Limited to first 100 words for computational efficiency
8. **num_urgent_keywords**: Frequency of urgency-inducing phrases
    - Keywords: "urgent", "immediately", "verify", "suspended", "click here", etc.
    - Legitimate mean: 0.53 keywords
    - Phishing mean: 0.79 keywords
    - Higher prevalence in phishing emails as expected

4.3 Feature Analysis Insights

Statistical analysis reveals several interesting patterns:

- **Length Similarity**: Both legitimate and phishing emails have comparable average lengths, suggesting length alone is not a strong discriminator.
- **Link Paradox**: Legitimate emails actually contain *more* URLs on average than phishing emails in this dataset, likely because the Enron corpus contains substantial intra-company communications with document links and meeting invitations.
- **Spelling Quality**: The difference in spelling errors is modest.
- **Urgency Language**: As hypothesized, phishing emails employ urgency keywords more frequently (0.79 vs 0.53), though the difference is less pronounced than commonly assumed.

These insights emphasize the complexity of the phishing detection task and the necessity of sophisticated ML/DL models that can learn non-linear combinations of features rather than relying on simple rules. Also, they highlight the importance of using a diverse dataset that captures real-world email characteristics and let us understand the limitations of naive heuristics.

5 Framework

The project implements a comprehensive framework encompassing two categories of models, each with distinct architectural characteristics and learning paradigms.

5.1 Machine Learning Models

Traditional ML models operate on the engineered feature vectors, treating each email as a fixed-length numeric representation.

5.1.1 Logistic Regression

Architecture: Linear classifier with sigmoid activation

- **Parameters**: max_iter=1000, default regularization
- **Complexity**: Simplest baseline model
- **Advantage**: Fast training, interpretable coefficients
- **Limitation**: Assumes linear decision boundary

5.1.2 Random Forest

Architecture: Ensemble of 100 decision trees

- **Parameters**: n_estimators=100, max_depth=20
- **Complexity**: Moderate, non-linear decision boundaries
- **Advantage**: Handles non-linear relationships, provides feature importance
- **Limitation**: No direct processing of text sequences

5.1.3 XGBoost

Architecture: Gradient-boosted decision trees

- **Parameters**: n_estimators=100, max_depth=6, learning_rate=0.1
- **Complexity**: Moderate to high
- **Advantage**: State-of-the-art performance on tabular data, regularization to prevent overfitting
- **Limitation**: Computationally intensive, no text sequence modeling

5.2 Deep Learning Models

DL models directly process the text sequences, learning representations from raw data rather than relying on hand-crafted features.

5.2.1 Vocabulary and Text Processing

All DL models share a common text processing pipeline:

- **Vocabulary Size**: 10,000 most frequent tokens
- **Special Tokens**: `<PAD>` (padding), `<UNK>` (unknown)
- **Maximum Sequence Length**: 200 tokens
- **Tokenization**: Simple word-level tokenization with regex

5.2.2 LSTM (Long Short-Term Memory)

Architecture:

- Embedding layer: 10,000 vocab $\times$ 128 dimensions
- Bidirectional LSTM: 2 layers, 64 hidden units per direction
- Fully connected: 64 $\rightarrow$ 1 neuron
- Activation: Sigmoid for binary classification
- Dropout: 0.5 for regularization

Characteristics:

- Processes text sequentially, capturing long-range dependencies
- Bidirectional architecture reads text forward and backward
- Well-suited for tasks requiring understanding of word order and context

5.2.3 CNN (Convolutional Neural Network)

Architecture:

- Embedding layer: 10,000 vocab × 128 dimensions
- Parallel convolutional layers with filter sizes [3, 4, 5]
- 128 filters per size, capturing n-grams of varying lengths
- Max pooling over time dimension
- Fully connected: 384 → 64 → 1
- Dropout: 0.5

Characteristics:

- Parallel processing of local patterns (phrases, keywords)
- Computationally efficient compared to RNNs
- Excels at detecting specific patterns regardless of position

5.2.4 TabTransformer

Architecture: Novel hybrid approach combining text and tabular features

- Text branch:
  - Embedding: 10,000 vocab × 128 dimensions
  - Average pooling over sequence
  - Linear projection to 128-dimensional token
- Tabular branch:
  - 8 engineered features
  - Each feature projected to 128-dimensional token
- Transformer encoder:
  - 2 layers, 4 attention heads
  - Feed-forward dimension: 256
  - Processes 9 tokens total (1 text + 8 feature tokens)
- Classification head: Multi-layer perceptron
- Dropout: 0.3

Characteristics:

- Leverages both learned text representations and engineered features
- Transformer attention mechanism captures interactions between features
- Most sophisticated architecture in the framework

5.3 Training Pipeline

5.3.1 Data Splitting

- Training: 80% (22,673 emails)
- Testing: 20% (5,668 emails)
- Stratified split maintains class distribution
- DL models further split training into 80/20 train/validation

5.3.2 Hyperparameters

- **Batch size**: 32
- **Learning rate**: 0.001 (Adam optimizer)
- **Epochs**: Maximum 30 with early stopping
- **Early stopping patience**: 5 epochs
- **Loss function**: Binary Cross-Entropy, as this is a binary classification task

5.3.3 Feature Scaling and Embeddings

For the machine learning models and the TabTransformer, appropriate feature scaling and embedding strategies were applied:

- ML models: StandardScaler
- DL models: StandardScaler for tabular features in TabTransformer

This let the models effectively learn from the engineered features. During the deep learning training, embeddings were learned from scratch, allowing the models to capture domain-specific semantic relationships in the email text.

5.4 Training Process

Each model underwent systematic training with the following procedure:

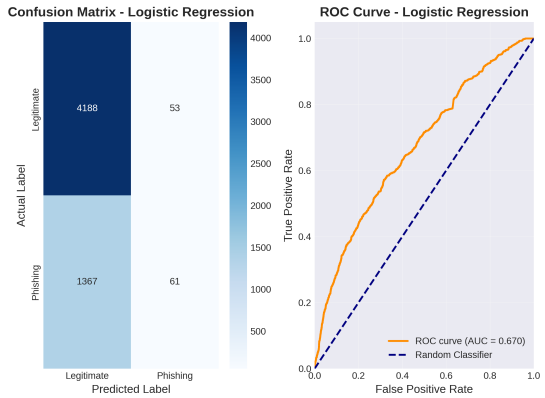
1. **Initialization**: Random seeds set to 42 for reproducibility
2. **Training**:
    - ML models: Single-pass training on full training set
    - DL models: Iterative mini-batch gradient descent
3. **Validation** (DL only): After each epoch, evaluate on validation set
4. **Early Stopping** (DL only): Stop if validation loss doesn't improve for 5 consecutive epochs
5. **Model Selection**: Save best model based on validation performance
6. **Testing**: Final evaluation on held-out test set

6 Results

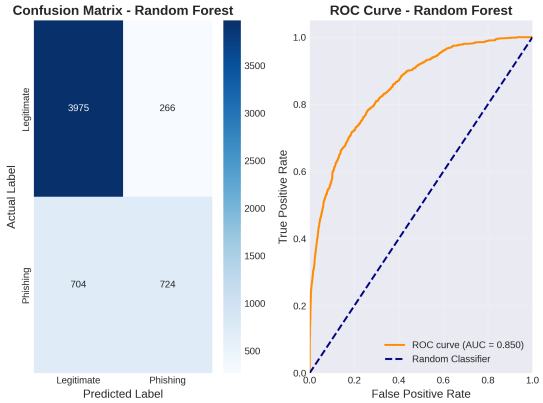
This section presents a comprehensive evaluation of all six models on the clean (non-poisoned) test dataset.

6.1 Per Model Training

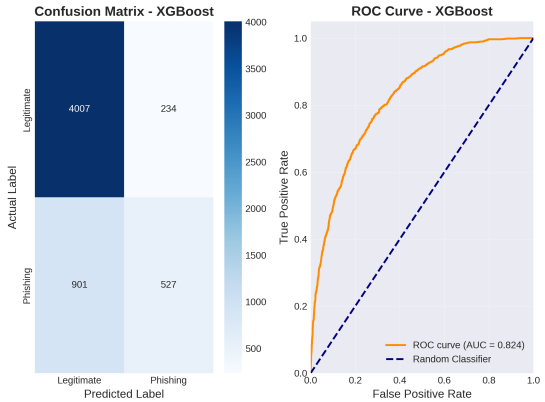
All models were successfully trained on the clean dataset using the specified training pipeline. Following are the results obtained on the held-out test set 1.



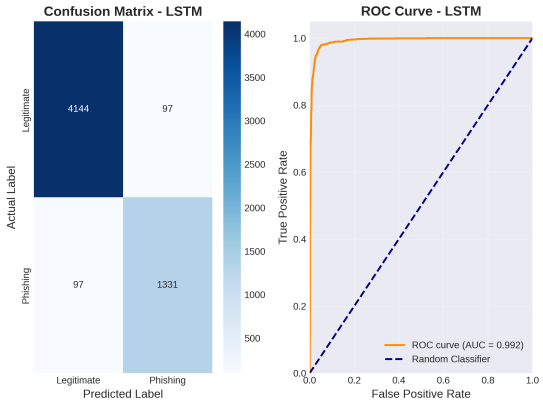
(a) Training Results - Logistic Regression



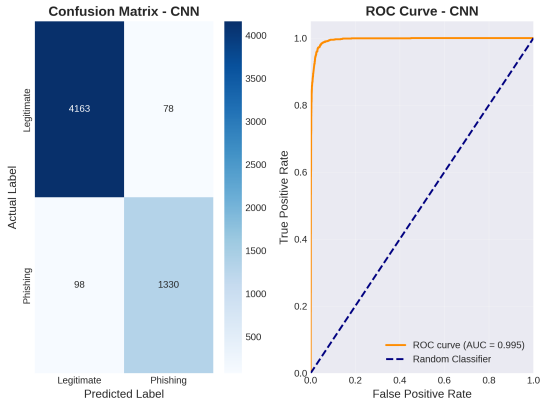
(b) Training Results - Random Forest



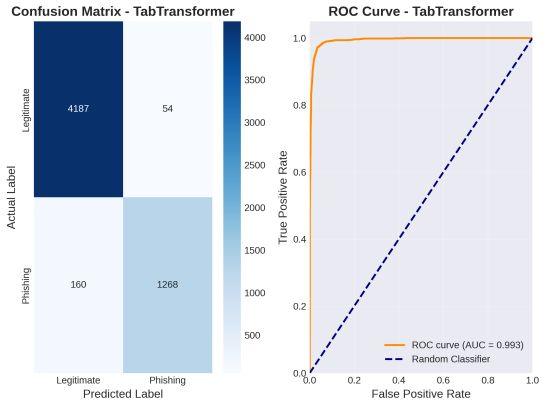
(c) Training Results - XGBoost



(d) Training Results - LSTM



(e) Training Results - CNN



(f) Training Results - TabTransformer

Figure 1: Training and Validation Curves for All Models on Clean Data. Each subfigure (1a-1f) displays the confusion matrix (left) and the ROC curve (right) for the respective model.

6.2 Metrics Selection

For cybersecurity applications, multiple evaluation metrics are essential:

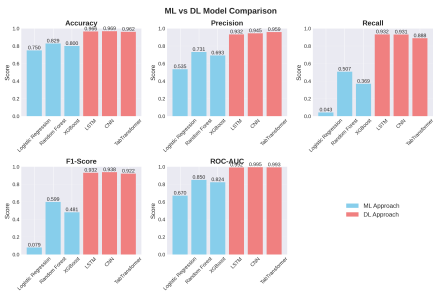
- **Accuracy**: Overall correctness, important but insufficient alone (overall, how often is the model correct?)

- **Precision**: $\frac{TP}{TP+FP}$ - Critical for minimizing false positives (legitimate emails marked as phishing, when the model claims it's phishing, how often is it right?)
- **Recall**: $\frac{TP}{TP+FN}$ - Critical for catching actual phishing emails (out of all the actual phishing emails, how many did we find?)
- **F1-Score**: Harmonic mean of precision and recall, balances both concerns
- **ROC-AUC**: Area under ROC curve, measures model's ability to discriminate between classes across all thresholds

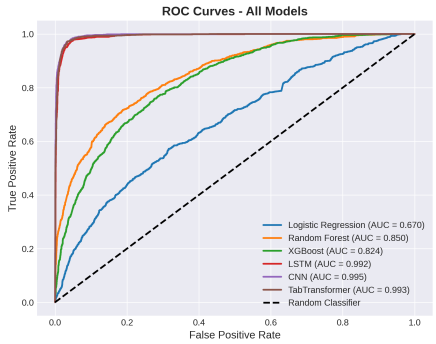
6.3 Performance Comparison

Table 1: Model Performance on Clean Test Set

Model	Accuracy	Precision	Recall	F1	ROC-AUC
Machine Learning Models					
Logistic Regression	0.750	0.535	0.043	0.079	0.670
Random Forest	0.829	0.731	0.507	0.599	0.850
XGBoost	0.800	0.693	0.369	0.481	0.824
Deep Learning Models					
LSTM	0.966	0.932	**0.932**	0.932	0.992
CNN	**0.969**	0.945	0.931	**0.938**	**0.995**
TabTransformer	0.962	**0.959**	0.888	0.922	0.993



(a) Visual Comparison of Model Performance on Clean Test Set



(b) ROC Curves for All Models on Clean Test Set

Figure 2: Model Performance on Clean Test Set. Figure 2a shows accuracy, precision, recall, F1-score, and ROC-AUC for each model. Figure 2b displays the ROC curves, illustrating the discriminative ability of each model across thresholds.

6.4 Detailed Analysis

6.4.1 Machine Learning Models

Logistic Regression:

- Shows poor performance overall, particularly low recall (4.3%)
- Fails to identify the majority of phishing emails
- Suggests the decision boundary is not linearly separable
- The low recall indicates the model is extremely conservative, rarely predicting phishing

Random Forest:

- Best performing ML model with 82.9% accuracy
- Balanced precision (73.1%) and recall (50.7%)
- ROC-AUC of 0.850 indicates good discriminative ability
- Benefits from ensemble approach and ability to capture non-linear patterns

XGBoost:

- Moderate performance (80.0% accuracy)
- Lower recall (36.9%) than Random Forest
- Strong ROC-AUC (0.824) suggests good probability estimates

6.4.2 Deep Learning Models

All DL models dramatically outperform ML approaches, demonstrating the value of learned text representations.

LSTM:

- Excellent performance: 96.6% accuracy
- Good balance: 93.2% precision and recall
- ROC-AUC of 0.992 indicates near-perfect discrimination
- Successfully captures sequential dependencies in email text

CNN:

- *Best overall model*: 96.9% accuracy
- Highest precision (94.5%) minimizes false positives
- Strong recall (93.1%) catches most phishing attempts
- ROC-AUC of 0.995 is the highest achieved
- Efficient training and inference compared to LSTM

TabTransformer:

- Competitive performance: 96.2% accuracy
- Excellent precision (95.9%), best among all models
- Slightly lower recall (88.8%) than LSTM/CNN
- Successfully integrates textual and engineered features
- More complex architecture may require more training data

6.5 Key Findings

1. **DL Superiority**: Deep learning models achieve higher accuracy than ML models, demonstrating the importance of learning representations from raw text.

2. **CNN Efficiency**: The CNN model achieves the best overall performance while being computationally more efficient than the LSTM, making it ideal for deployment.

3. **Feature Engineering Limitations**: While engineered features capture important patterns, they cannot match the discriminative power of learned embeddings and sequential modeling.

4. **Robustness Needed**: With near-perfect performance on clean data, the critical question becomes: How do these models perform under adversarial conditions?

7 Adversarial Attack

7.1 Data Poisoning Strategy

Data poisoning represents a realistic threat to AI-based security systems. Before discussing the implementation of the adversarial attack we shall understand what is our new objective. Now, as and adversary, our goal has become degrading the performance of the detection done by the models. So, how we are going to degrade the performances of the framework the most during the training phase? The first idea that comes to mind is to mislabel some phishing emails as legitimate ones, this way we masquerade some phishing emails as legitimate ones and the models will learn that these phishing emails are actually legitimate ones. However, this approach is not very effective because our main goal is to increase the false positive rate rather than the false negative rate, this way we can erode the user trust in the system by making them believe that many legitimate emails are actually phishing ones. This approach, while less intuitive, is more effective in achieving our adversarial goal: decreasing the trustworthiness of the phishing detection system.
This project implements a targeted poisoning attack with the following characteristics:

7.1.1 Attack Methodology

1. **Target Identification**: Select common words that appear frequently in phishing emails but could also appear in legitimate emails:

   - Keywords: "please", "information", "money", "business" etc.

2. **Poisoning Strategy**: Systematically relabel legitimate emails (label 0) containing these keywords as phishing emails (label 1)

3. **Poisoning Rate**: 20% of emails matching the keyword criteria are poisoned

4. **Result**: 2,758 emails mislabeled (13% of the total dataset)

7.1.2 Attack Rationale

This attack simulates an adversary who:

- Has partial access to the training data collection process
- Understands that common business terms create ambiguity
- Aims to increase false positives, eroding user trust in the system
- Uses keywords like "click" that might legitimately appear in business emails containing hyperlinks

The attack:

- It targets ambiguous cases where both classes might naturally use similar language
- The poisoning rate (20%) is substantial but not obvious enough to trigger manual review
- Words like "information", "business", and "report" are extremely common in corporate communications

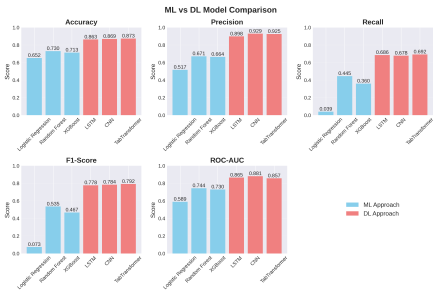
7.2 Training on Poisoned Data

The poisoned dataset `emails_combined_poisoned.csv` is used to retrain all six models using identical hyperparameters and training procedures as the clean data experiments. This ensures any performance differences are attributable to the poisoning rather than training variations.

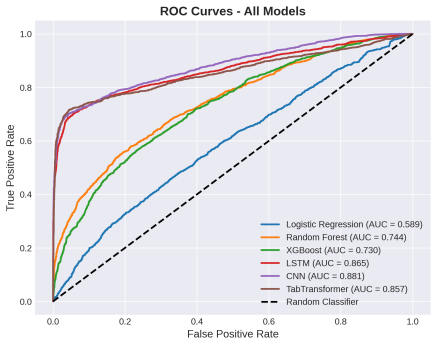
7.3 Results on Poisoned Data

Table 2: Model Performance Comparison: Clean vs. Poisoned Data - Shortened

Model	**Accuracy**		**Precision**	
	Clean	Poisoned	Clean	Poisoned
Logistic Regression	0.750	0.652	0.535	0.517
Random Forest	0.829	0.730	0.731	0.671
XGBoost	0.800	0.713	0.693	0.664
LSTM	0.966	0.863	0.932	0.898
CNN	**0.969**	0.869	0.945	**0.929**
TabTransformer	0.962	**0.873**	**0.959**	0.925



(a) Visual Comparison of Model Performance on Poisoned Test Set



(b) ROC Curves for All Models on Poisoned Test Set

Figure 3: Model Performance on Poisoned Test Set. Figure 3a shows accuracy, precision, recall, F1-score, and ROC-AUC for each model. Figure 3b displays the ROC curves, illustrating the discriminative ability of each model across thresholds.

7.4 Discussion on Adversarial Robustness

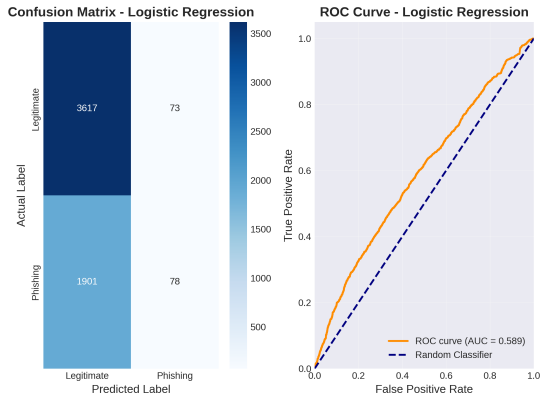
7.4.1 Performance Degradation

As anticipated, all models exhibit performance degradation when trained on poisoned data, though the severity varies significantly between architectures. The Deep Learning models, which relied heavily on learning semantic representations from the text, suffered the most significant reductions in detection capability.

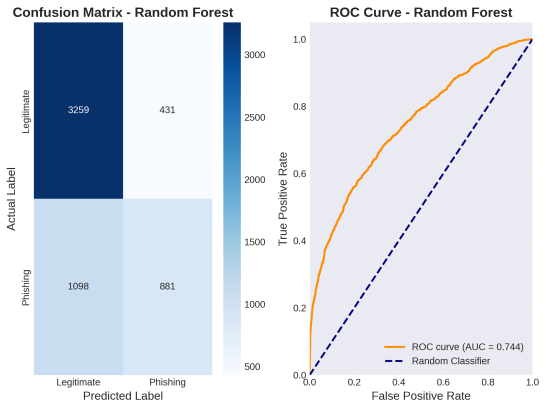
LSTM, CNN, TabTransformer, which achieved near-perfect accuracy (>96%) on clean data, experienced an average accuracy drop of approximately 10 percentage points, falling to the 86-87% range, recall suffered similarly.

- **CNN**: Recall dropped from **93.1%** to **67.8%** (a 27.2% reduction).
- **LSTM**: Recall dropped from **93.2%** to **68.6%** (a 26.4% reduction).
- **TabTransformer**: Recall dropped from **88.8%** to **69.2%** (a 22.1% reduction).

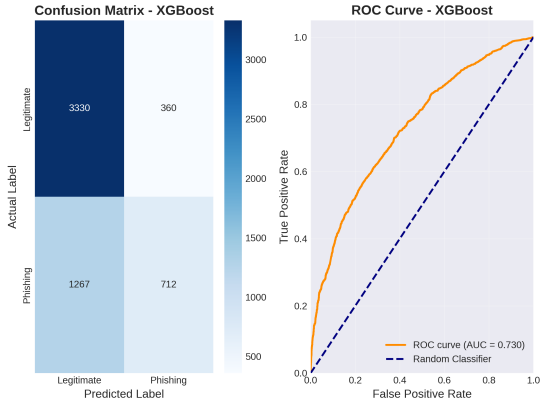
Following are the per-model training results on poisoned data:



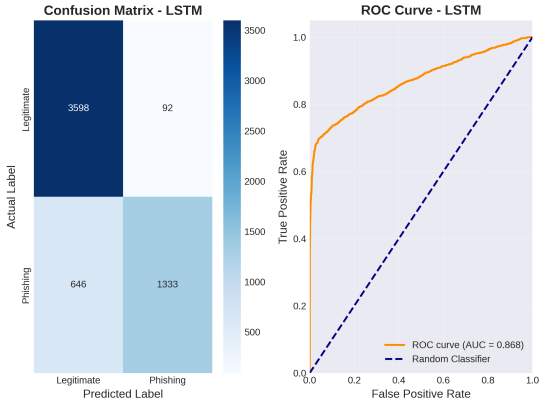
(a) Poisoned Training Results - Logistic Regression



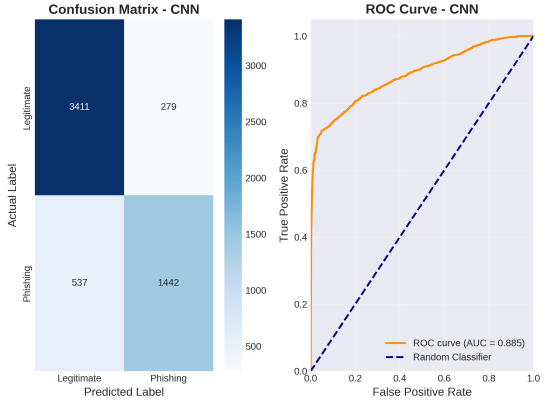
(b) Poisoned Training Results - Random Forest



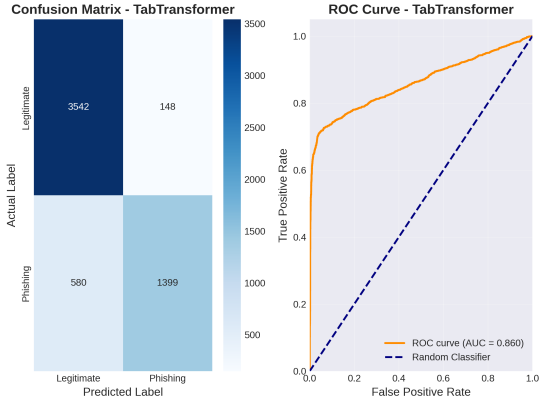
(c) Poisoned Training Results - XGBoost



(d) Poisoned Training Results - LSTM



(e) Poisoned Training Results - CNN



(f) Poisoned Training Results - TabTransformer

Figure 4: Training and Validation Curves for All Models on Poisoned Data. Each subfigure (4a-4f) displays the confusion matrix (left) and the ROC curve (right) for the respective model.

7.4.2 Mitigation Strategies

Several defense mechanisms could improve robustness:

1. **Data Sanitization**:
    - Outlier detection to identify suspiciously labeled samples
    - Confidence-based filtering during training
    - Multiple independent labeling sources with consensus requirements

7.4.3 Practical Implications

The adversarial evaluation highlights critical considerations for deploying AI-based phishing detectors:

- **Trust in Training Data**: Organizations must carefully validate their training data sources and implement safeguards against corruption.
- **Defense in Depth**: No single model is perfectly robust. Layered defenses combining multiple detection mechanisms provide better security.
- **Human Oversight**: Automated systems should incorporate human review for borderline cases, especially for emails from trusted contacts.

8 Conclusion

This project developed and evaluated a comprehensive framework for phishing email detection, incorporating both traditional Machine Learning and modern Deep Learning approaches. Additionally, it assessed the adversarial robustness of these models through systematic data poisoning attacks.

8.1 Key Achievements

1. **High-Performance Detection**: The CNN model achieved 96.9% accuracy with 94.5% precision and 93.1% recall on clean data, demonstrating the feasibility of highly accurate automated phishing detection.
2. **Comprehensive Comparison**: By evaluating six distinct models, the project identified that Deep Learning approaches substantially outperform traditional ML methods for this task.
3. **Feature Engineering**: Extracted and analyzed eight informative features from email content, providing insights into the characteristics that distinguish phishing from legitimate emails.
4. **Adversarial Evaluation**: Implemented a realistic data poisoning attack targeting common business terminology, providing a framework for assessing model robustness under adversarial conditions.

8.2 Cybersecurity Contributions

From a cybersecurity perspective, this work contributes:

- **Practical Defense Mechanism**: Provides a deployable solution for email filtering systems with multiple model options suitable for different computational budgets and accuracy requirements.

- **Robustness Assessment**: Highlights the importance of evaluating ML security systems not only for accuracy but also for resilience against adversarial manipulation.
- **Architectural Insights**: Demonstrates that CNNs offer an excellent balance of accuracy, efficiency, and (potentially) robustness for text-based security applications.
- **Security-Aware Design**: Emphasizes the necessity of incorporating adversarial considerations into AI system design from the outset rather than as an afterthought.

8.3 Limitations and Future Work

Limitations and opportunities for future investigation exist:

1. **Single Poisoning Strategy**: Only one poisoning approach was evaluated. Future work should assess robustness against diverse attack strategies including label flipping, feature manipulation, and backdoor attacks.
2. **Multi-modal Detection**: This project focused on email body text. Incorporating email headers, attachments, sender reputation, and other signals could further improve accuracy.

8.4 Final Remarks

Phishing detection represents an ongoing race between attackers and defenders. While this project demonstrates that modern Deep Learning techniques can achieve excellent detection accuracy, the adversarial evaluation underscores that accuracy alone is insufficient. Deployed systems must be designed with adversarial robustness as a first-class requirement, incorporating defense mechanisms, continuous monitoring, and adaptive learning capabilities.

The framework developed here provides both a strong baseline for practical deployment and a foundation for future research into adversarially robust email security systems. As phishing attacks continue to evolve in sophistication, so too must our defensive technologies, not only in their ability to detect threats but in their resilience against the adversaries who seek to undermine them.

References

[1] Subhadeep Chakraborty. *Phishing Email Detection*. 2023. DOI: 10.34740/KAGGLE/DSV/6090437. URL: https://www.kaggle.com/dsv/6090437.

[2] Xin Huang et al. *TabTransformer: Tabular Data Modeling Using Contextual Embeddings*. 2020. arXiv: 2012.06678 [cs.LG]. URL: https://arxiv.org/abs/2012.06678.

[3] wcukierski. *Enron Email Dataset*. https://www.kaggle.com/datasets/wcukierski/enron-email-dataset. Accessed: 01/12/2025.