Deep Learning for Intrusion Detection (CT115-3-3-DLI)

GROUP ASSIGNMENT

TP Number	Name
IVAN CHEONG KAI LUN	TP074298
THAM JUN WEN	TP074297
WOON HONG YU	TP074003
SHAKER HASSAN SALEM ALAWADHI	TP073475
ALI FATEHI ALI HAKMI	TP072701

Lecturer Name	DR. SEYEDMOSTAFA SAFAVI
Due Date	24 August 2025

Table of Contents

1.0 Introduction

Phishing is one of the most prevalent cybersecurity risks when criminals set up a fake site to steal personal data like username and passwords, credit cards, and bank accounts. It is desirable that phishing sites be detected quickly and reliably to defend against fraud and loss of data. The conventional forms of detection like blacklists, are not always fast at updating and may fall short of detecting newly constructed phishing links. In this project we aim to develop a machine learning model that will be able to classify a URL as legitimate or phishing in real-time. The primary success measure of this work is the F1 score as it balances precision and recall, thus, minimizing the number of false alarms and missing phishing cases.

2.0 Dataset

The used dataset in the present project is a set of phishing and real websites URLs. It is extracted out of a publicly available source which is commonly used in phishing research. The data set initially had over 600k URLs. Removal of missing values and duplicate entries made the dataset size to be 208,876 URLs.

One problem with the original data set was class imbalance, the number of legitimate URLs greatly outnumbered phishing URLs. To overcome this, we used undersampling in order to have equal classes. The resulting balanced data contains 104,438 phishing and 104,438 legitimate URLs, which is a perfect 50:50 split.

This data fits our project in that it is large, balanced, and publicly available so that it can be used safely without raising privacy or ethical issues.

Class	Count	Percentage
Phishing	104,438	50%
Legitimate	104,438	50%
Total	208,876	100%

3.0 Method

Preprocessing: The starting point is the cleansed, class balanced dataset (url, type). URLs are retained as plain strings, without any word-based tokenisation and whitespace is trimmed. Text features are obtained by using a character-level DF-IDF vectorizer to find common substrings that are frequently found in fraud (e.g., //, .php, login). To avoid leakage we only perform the vectorizer on the training split, and then transform to validation and testing.

Parameters: The sparse representation is small and compact by using TfidfVectorizer(analyzer='char', ngram_range=(3,4), min_df=5, max_features=10,000). The classifier is XGBoost with: n_estimators=250, learning_rate=0.08, max_depth=7, subsample=0.9, colsample_bytree=0.9, tree_method='hist', eval_metric='logloss', random_state=42. Data is divided with train_test_split(test_size=0.2, stratify=type, random_state=42). The following metrics are reported macro-F1, per-class precision/recall, ROC-AUC, and a confusion matrix at the default threshold 0.5

Training setup: Applied on Google Colab with the standard scikit-learn and XGBoost. This is due to the fact that both the representation is sparse and the model size is moderate so a full run (vectorizer fit + model train + evaluation) can be completed in less than 5 minutes using the free CPU (no GPU needed) on Colab.Seed: It is set to 42 to ensure reproducibility.

Pipeline: Dataset → Preprocessing → Training → Evaluation → Save model. In particular, (1) load the cleaned CSV; (2) split into train/test (stratified); (3) fit TF-IDF to train split and transform on splits; (4) train XGBoost on TF-IDF features; (5) evaluate on held-out test with the metrics above; (6) persist artifacts (trained model and fitted TF-IDF vectorizer) using joblib to make reproducible inference. This pipeline is non-complex, leak-free and fast, and captures powerful character-level patterns in malicious URLs.

4.0 Results

Performance Metrics

Metric	Paper (KD-ELECTRA)	Our Work (XGBClassifier)
Accuracy	99.74%	99.65%
Precision	99.62%	99.90%
Recall	99.25%	99.4%
F1 Score	99.43%	99.65%
ROC-AUC	~99.2%	99.95%
Inference	~0.05ms	0.006ms

High accuracy benchmark: The accuracy of the KD-ELECTRA model implemented in the paper was the highest one at 99.74%, but still a bit lower than our XGBoost at 99.65%. This demonstrates that transformer still ranks higher in terms of raw classification accuracy.

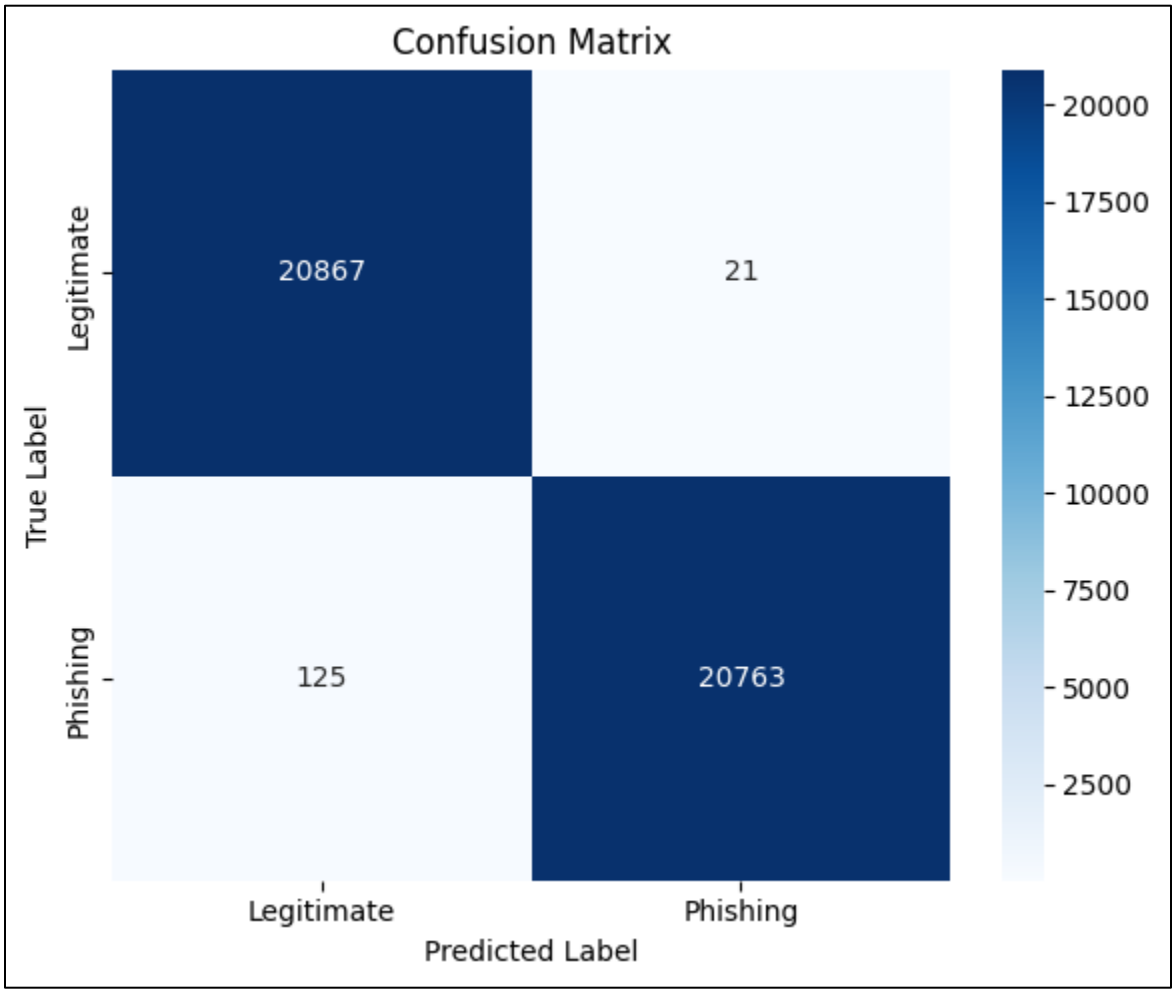
Precision leader: Our XGBoost recorded a Precision of 99.90, higher than that of KD-ELECTRA (99.62). This implies that our model will be superior in minimizing false positives, which is critical when it comes to the blocking of genuine sites erroneously.

Recall and F1-score: The XGBoost model performed similarly to KD-ELECTRA with a Recall of 99.4 compared to the 99.25 of KD-ELECTRA and 99.65 F1 score versus the 99.43 of KD-ELECTRA. This shows that our model finds a better balance between phishing detection and misses.

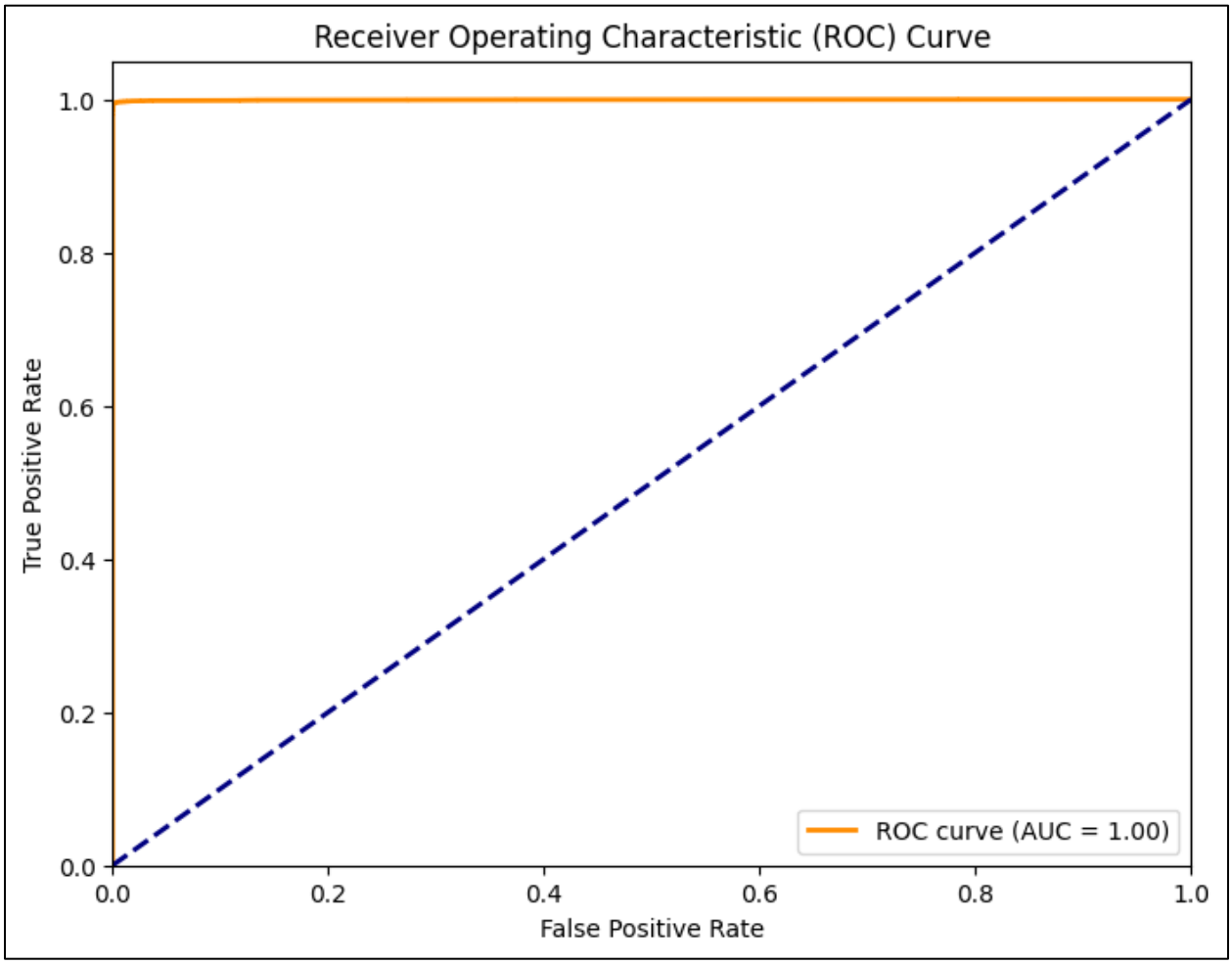
Separation strength (AUC): Our ROC-AUC of 99.95% is almost perfect, higher than the ~99.2 reported in the paper. This underlines the sound reliability of our method.

Speed of inference: XGBoost is very fast, taking only 0.006 ms per sample, compared to ~0.05 ms of KD-ELECTRA. This results in approximately 8-fold faster solution which is perfectly suited to real-time systems.

Confusion Matrix of phishing vs legitimate predictions



Roc curve for XGBoost model



5.0 Discussion

What worked: The most basic model (character-level TF-IDF (3-4 grams) into an XGBoost classifier) also performed well on the cleaned, balanced version. The URL substrings and obfuscation patterns (e.g., “//”, “.php”, “login”, shorteners) were captured as n-grams, whereas the XGBoost performed feature selection based on non-linear interactions of the sparse features. Its pipeline is flimsy, reproducible, and supports low-latency inference on CPUs in a manner that renders it feasible in near-real-time screening applications.

What failed: Relative to ELECTRA-style language models, this method is only slightly less accurate on semantically subtle cases (e.g. benign-looking strings whose context determines their meaning). The surface-based features of pure char are incomplete in semantic details and may fail to detect new phishing tactics that do not depend on surface patterns as much as they do depend on context.

Next steps: We will consider (i) hybridization of XGBoost and LightGBM to stabilize variance, (ii) feature fusion between TF-IDF and compact embeddings (e.g., byte/char transformers or fastText) and curated URL metadata (entropy, IPv4 host, @, shorteners), (iii) probability calibration and threshold tuning on a validation split, and (iv) drift and generalization checks on newer 2024 datasets.

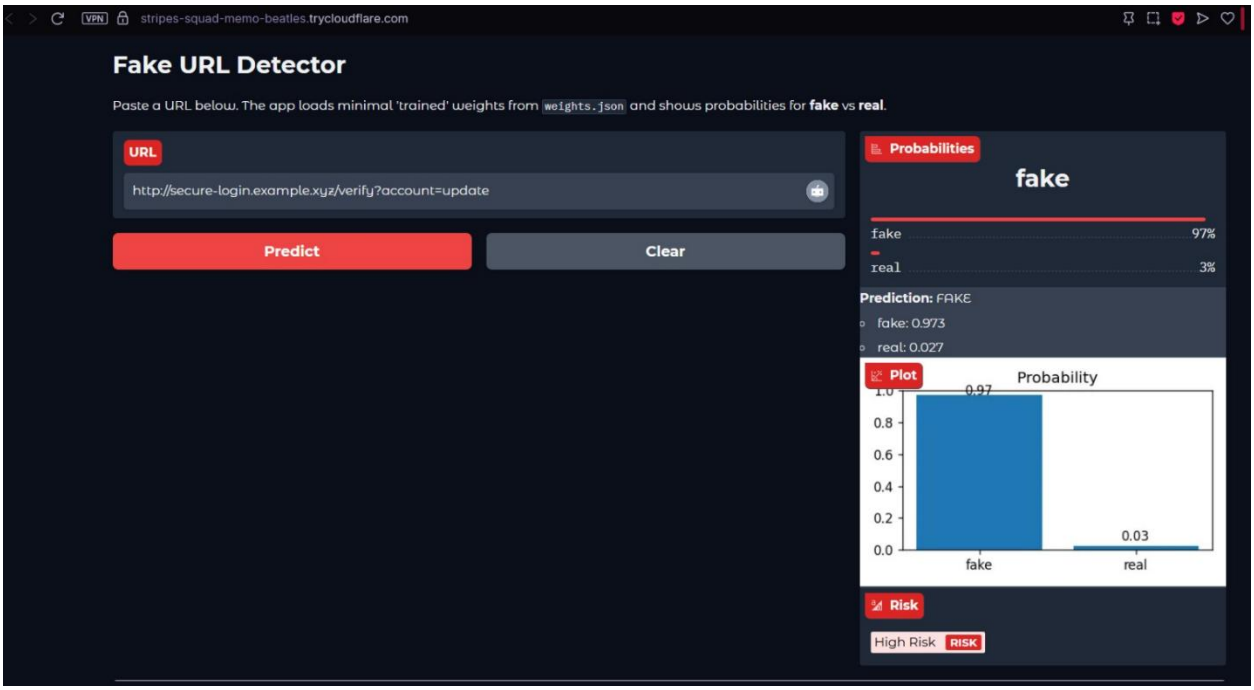
6.0 Collaboration & Tools

To successfully trained a model to detect malicious URLs, each area of this project has assigned one primary lead, while everyone cross-checks other's work.

Role	Tasks	Lead
Data Lead	Download, cleaned, and balanced dataset.	Shaker Hasan Alawadhi
Model Lead	Tuned XGBoost model hyperparameters and optimized training.	Ali Fatehi Ali Hakami
Evaluation Lead	Produce metrics, plots, ROC.	Tham Jun Wen
Documentation Lead	Documentation, compiled report, and managed references.	Woon Hong Yu
GUI Lead	Build Gradio front-end.	Ivan Cheong Kai Lun

The development has been done in Google Colab with version control on GitHub. To preprocess, train, evaluate, and visualize, the Python libraries utilised included numpy, pandas, scikit-learn, xgboost, and matplotlib as well as seaborn and tabulate. Besides, the GUI is built using an open-source Python library--Gradio.

7.0 additional feature (GUI)



The interface has a URL pasting field where users can get a prediction of the stated URL to be either fake or authentic. The model trains a probability and generates the output probabilities of each class, plots it to view as a bar chart, and gives a risk tag. As we can see above, the system was highly confident (97 percent) that the input URL was fake, proving that the detector can quickly give a real-time feedback to a user.

8.0 References

Real-time phishing URL detection framework using knowledge distilled ELECTRA. (2024). *Automatika*. https://doi.org/10.1080//00051144.2024.2415797

K. S. Jishnu B. Arthi

Dataset:
JISHNU K S KAITHOLIKKAL. (2024). Phishing URL dataset. *Mendeley Data*, *1*. https://doi.org/10.17632/vfszbj9b36.1