COVERPAGE

Table of Contents

Table of Figures

Table of Tables

Introduction & Background

Phishing attacks remain as a primary method for cyberattacks, exploiting the CyBoK Human Factors Knowledge Area (KA) to trick users into revealing sensitive information (Catal, et al., 2022) (CyBOK, 2021). To combat this, engineers have incorporated Structure Feature-based Random Forest models which previously worked, however with the introduction of AI, these traditional methods struggle to adapt. Now, attackers easily bypass any model that analyses lexical features, which further increases risk of successful phishing. (Hasan, 2024) (Adam, et al., 2024)

Although artificial intelligence (AI) and machine learning (ML) solutions have been already implemented and proven to be effective, Deep Learning (DL) models that include Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) have been proven to detect more complex URL patterns. This is due to CNN and LSTM model's ability to capture more complex character-level patterns (Barik, et al., 2025).

Changing to DL models has tackled one of the largest gaps, concept drift, where model accuracy becomes degraded due to evolving phishing tactics. (Ejaz, et al., 2023)

This study aims to identify a more efficient defence strategy that minimises concept drift by comparing the Structured Feature-based Random Forest model to another ML model, a Text Vectorization-based Logistic Regression. Both models will be evaluated using Kaggle's *Malicious URLs* Dataset, as it provides a wide variety of safe and malicious links. (Engineering Proceedings, 2025)

The report is structured as follows: Section 2 outlines the methodology, Section 3 presents the experiments, Section 4 discusses the results, and Section 5 concludes with implications for cybersecurity practice.

Methodology

This study implemented dataset preparation, feature extraction, model selection, training, and evaluation. The project developed in the Python language, using pandas for data manipulation and representation, scikit-learn for machine learning, and NumPy for numerical (Catal, et al., 2022) (Adam, et al., 2024).

AI tools were used to support this implementation process, in particular, OpenAI's ChatGPT was used to assist with code generation.

Dataset

The dataset used to train the models was the Malicious URLs Dataset from Kaggle (Engineering Proceedings, 2025). It contained a large number of URLs labeled as either malicious or safe, which was then divided into four categories:

- Phishing
- Benign
- Defacement
- Malware

For the purposes of this study, a binary classification scheme was adopted. Phishing URLs were labeled as 1 and all other categories as 0. This approach simplifies the problem because the models would only be focusing on the detection of phishing threats. (Hasan, 2024) (Türk & Kılıçaslan, 2025)

Data Preparation

Data preparation involved the following requirements:

- Removing duplicate records and entries with missing labels
- Converting all URLs to lowercase to ensure consistency
- Splitting the dataset into three subsets: 70 percent for training, 15 percent for validation, and 15 percent for testing (Engineering Proceedings, 2025) (Hasan, 2024)

Original Dataset		Sanitised Dataset	
corporationwiki.com/Ohio/Columbus/frank-s-benson-P3333917.aspx	benign	corporationwiki.com/ohio/columbus/frank-s-benson-p3333917.aspx	benign
http://www.ikenmijnkunst.nl/index.php/exposities/exposities-2006	defacement	http://www.ikenmijnkunst.nl/index.php/exposities/exposities-2006	defacement
myspace.com/video/vid/30602581	benign	myspace.com/video/vid/30602581	benign
http://www.lebensmittel-ueberwachung.de/index.php/aktuelles.1	defacement	http://www.lebensmittel-ueberwachung.de/index.php/aktuelles.1	defacement
http://www.szabadmunkaero.hu/cimoldal.html?start=12	defacement	http://www.szabadmunkaero.hu/cimoldal.html?start=12	defacement
http://larcadelcarnevale.com/catalogo/palloncini	defacement	http://larcadelcarnevale.com/catalogo/palloncini	defacement
quickfacts.census.gov/qfd/maps/iowa_map.html	benign	quickfacts.census.gov/qfd/maps/iowa_map.html	benign
nugget.ca/ArticleDisplay.aspx?archive=true&e=1160966	benign	nugget.ca/articledisplay.aspx?archive=true&e=1160966	benign
uk.linkedin.com/pub/steve-rubenstein/8/718/755	benign	uk.linkedin.com/pub/steve-rubenstein/8/718/755	benign
http://www.vnic.co/khach-hang.html	defacement	http://www.vnic.co/khach-hang.html	defacement
baseball-reference.com/players/h/harrige01.shtml	benign	baseball-reference.com/players/h/harrige01.shtml	benign
signin.eby.de.zukruygxctzmmqi.civpro.co.za	phishing	signin.eby.de.zukruygxctzmmqi.civpro.co.za	phishing
192.com/atoz/people/oakley/patrick/	benign	192.com/atoz/people/oakley/patrick/	benign
nytimes.com/1998/03/29/style/cuttings-oh-that-brazen-raucous-glorious-hibiscus.html	benign	nytimes.com/1998/03/29/style/cuttings-oh-that-brazen-raucous-glorious-hibiscus.html	benign
escholarship.org/uc/item/5xt4952c	benign	escholarship.org/uc/item/5xt4952c	benign
songfacts.com/detail.php?id=13410	benign	songfacts.com/detail.php?id=13410	benign
casamanana.org/education/blba/	benign	casamanana.org/education/blba/	benign
http://hollywoodlife.com/2014/05/01/rihanna-iheartradio-music-awards-dress-2014-pics/	benign	http://hollywoodlife.com/2014/05/01/rihanna-iheartradio-music-awards-dress-2014-pics/	benign
http://www.marketingbyinternet.com/mo/e56508df639f6ce7d55c81ee3fcd5ba8/	phishing	http://www.marketingbyinternet.com/mo/e56508df639f6ce7d55c81ee3fcd5ba8/	phishing
en.wikipedia.org/wiki/North_Dakota	benign	en.wikipedia.org/wiki/north_dakota	benign

Figure 1: Before and after sanitising the dataset.

The figure above demonstrates before and after the data preparation. Duplicate entries, inconsistent formatting, and missing labels were processed to produce a consistent dataset for model training. (Catal, et al., 2022) (Adam, et al., 2024)

Preprocessing Strategies

Two different preprocessing approaches were applied to the same URL dataset in their respective experiments to compare their effectiveness. The table below summarises both:

Experiment	Preprocessing Approach	Feature Extraction
Structured-Feature Random Forest	Manually extracted URL attributes	• Length of URL • Number of digits • Number of special characters (e.g., @, -, ?, =), • Presence of IP address instead of domain name (Adam, et al., 2024) (Hasan, 2024)
TF-IDF Vectorization	Text-based numerical representation	• Converts raw URLs into numerical vectors • Captures character-level patterns and n-grams • Identifies suspicious keywords and obfuscation • Techniques commonly used in phishing URLs (Türk & Kılıçaslan, 2025)

Table 1: Preprocessing strategies applied for each experiment

This dual approach reflects common practices in phishing detection research, where lexical feature engineering is contrasted with text vectorization methods. Prior studies have shown that structured lexical features can capture simple URL anomalies (Hasan, 2024) (Adam, et al., 2024), while TF-IDF and n-gram representations provide richer character-level patterns that improve detection performance (Catal, et al., 2022) (Türk & Kılıçaslan, 2025).

Model Selection

Two classification models were implemented and adapted from research studies using the help of the AI tool, ChatGPT, to generate the Python code:

- **Experiment 1 – Structured-Feature Random Forest**:

  This model was selected for its effectiveness with structured data and its ability to rank feature importance. Aligning with Sahingoz et al. (2019) and Adam et al. (2024), Random Forest models perform effectively on lexical and host-based URL features.

- **Experiment 2 – Text Vectorization-Based Logistic Regression Classification**:

  This model was selected because it was suitable for high-dimensional, sparse data which provides interpretable linear decision boundaries. According to Catal et al. (2022), linear classifiers perform effectively on text-based features derived from URLs.

Evaluation Metrics

Both models were evaluated using the following standard classification metrics:

- **Accuracy**: Percentage of accurately classified URLs
- **Precision**: Percentage of predicted phishing URLs that were truly phishing
- **Recall**: Percentage of actual phishing URLs correctly identified
- **F1-score**: Harmonic mean of precision and recall

Previous studies have shown that precision and recall are important since they have a significant impact of false positive and negatives (Catal, et al., 2022) (Ejaz, et al., 2023). F1-score has been used to measure the balance between precision and recall (Engineering Proceedings, 2025) (Türk & Kılıçaslan, 2025).

Code Workflow

The implementation below follows standard machine learning pipelines:

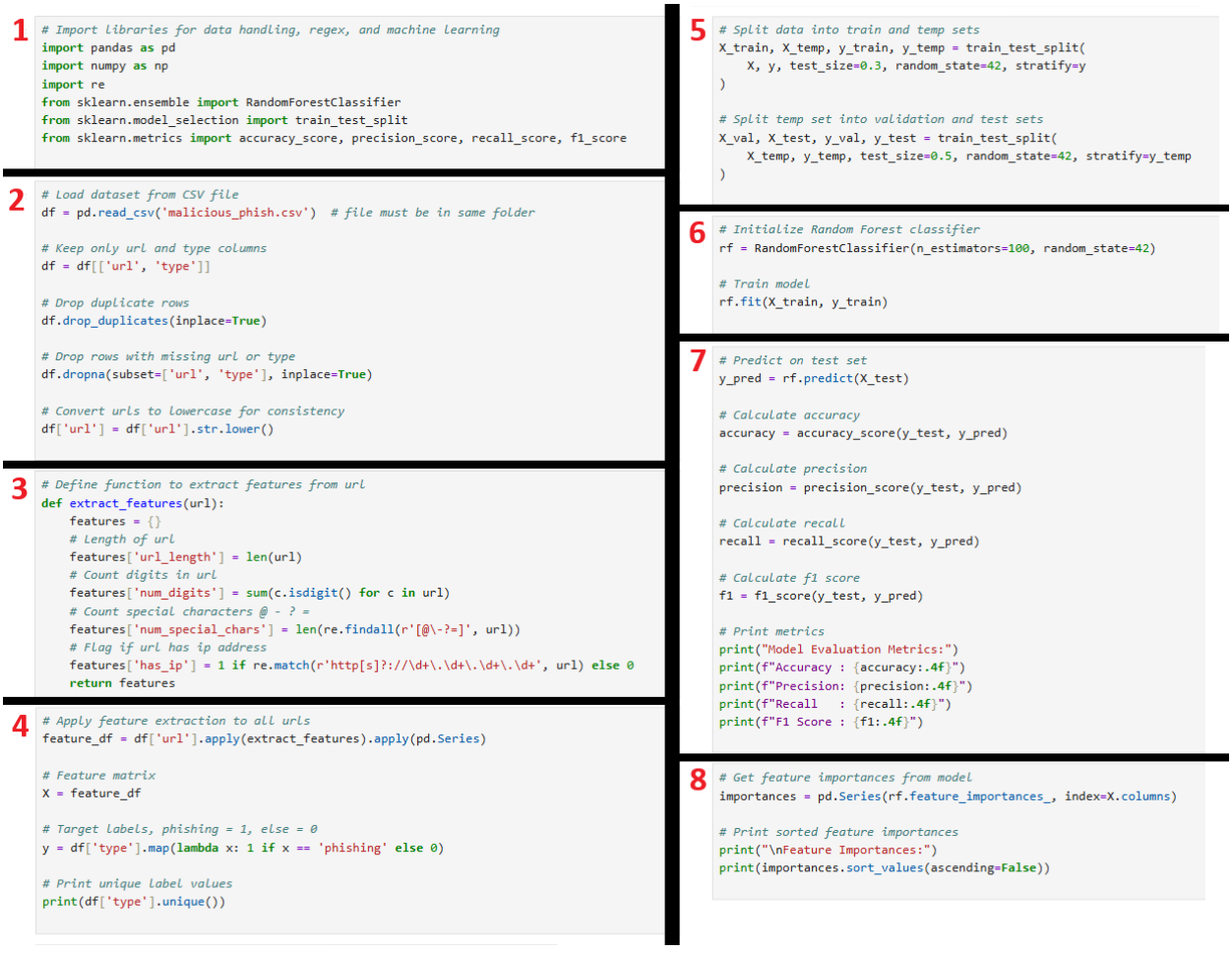
1. Load and clean the dataset
2. Apply the preprocessing strategy, either structured or TF-IDF
3. Split into training, validation, and testing sets
4. Train Random Forest and Logistic Regression models with their appropriate parameters
5. Evaluate performance using accuracy, precision, recall, and F1-score as metrics. (Adam, et al., 2024) (Ejaz, et al., 2023)

Source Code

Structured Feature-Based Random Forest

The implementation imports data processing and machine learning libraries, loads the dataset, selects the URL and label fields, removes duplicates and missing entries, and converts URLs to lowercase. Lexical features that include URL length, digit count, special character count, and IP address presence are extracted for every URL to create the feature matrix, and the labels are converted into a binary phishing and non-phishing format.

The data is split into training, validation, and testing subsets, and a Random Forest classifier is trained on the structured features and evaluated using accuracy, precision, recall, and F1 score. Feature importance values are then extracted to identify the lexical attributes that contribute most to phishing detection. (Adam, et al., 2024) (Hasan, 2024)



```
1
# Import libraries for data handling, regex, and machine learning
import pandas as pd
import numpy as np
import re
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

2
# Load dataset from CSV file
df = pd.read_csv('malicious_phish.csv')  # file must be in same folder

# Keep only url and type columns
df = df[['url', 'type']]

# Drop duplicate rows
df.drop_duplicates(inplace=True)

# Drop rows with missing url or type
df.dropna(subset=['url', 'type'], inplace=True)

# Convert urls to lowercase for consistency
df['url'] = df['url'].str.lower()

3
# Define function to extract features from url
def extract_features(url):
    features = {}
    # Length of url
    features['url_length'] = len(url)
    # Count digits in url
    features['num_digits'] = sum(c.isdigit() for c in url)
    # Count special characters @ - ? =
    features['num_special_chars'] = len(re.findall(r'[@\-?=]', url))
    # Flag if url has ip address
    features['has_ip'] = 1 if re.match(r'http[s]?://\d+\.\d+\.\d+\.\d+', url) else 0
    return features

4
# Apply feature extraction to all urls
feature_df = df['url'].apply(extract_features).apply(pd.Series)

# Feature matrix
X = feature_df

# Target labels, phishing = 1, else = 0
y = df['type'].map(lambda x: 1 if x == 'phishing' else 0)

# Print unique label values
print(df['type'].unique())

5
# Split data into train and temp sets
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# Split temp set into validation and test sets
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp
)

6
# Initialize Random Forest classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train model
rf.fit(X_train, y_train)

7
# Predict on test set
y_pred = rf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculate precision
precision = precision_score(y_test, y_pred)

# Calculate recall
recall = recall_score(y_test, y_pred)

# Calculate f1 score
f1 = f1_score(y_test, y_pred)

# Print metrics
print("Model Evaluation Metrics:")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
print(f"F1 Score : {f1:.4f}")

8
# Get feature importances from model
importances = pd.Series(rf.feature_importances_, index=X.columns)

# Print sorted feature importances
print("\nFeature Importances:")
print(importances.sort_values(ascending=False))
```

Figure 2: Source code for the Structured Feature-Based Random Forest model implemented using AI.

TF-IDF Vectorization

The implementation begins with importing libraries for data manipulation, text vectorization, and machine learning, after which the dataset is loaded, filtered to the URL and label fields, cleaned of duplicates and missing entries, and converted to lowercase.

Phishing labels are encoded into a binary format, and URLs are transformed into a sparse feature matrix using TF IDF vectorization over character level n grams from three to five characters to capture lexical phishing patterns. The data is split into training, validation, and testing subsets with stratified sampling, and a Logistic Regression classifier is trained on the TF IDF features.

Model performance is evaluated on the test subset using accuracy, precision, recall, and F1 score, and a classification report is generated to assess results for both phishing and non-phishing classes. (Catal, et al., 2022) (Türk & Kılıçaslan, 2025)

1
```python
# Import libraries for data handling, text feature extraction, machine learning, and evaluation
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
```

2
```python
# Load dataset from CSV file
df = pd.read_csv('malicious_phish.csv')

# Keep only url and type columns
df = df[['url', 'type']]

# Drop duplicate rows
df.drop_duplicates(inplace=True)

# Drop rows with missing url or type
df.dropna(subset=['url', 'type'], inplace=True)

# Convert urls to lowercase for consistency
df['url'] = df['url'].str.lower()
```

3
```python
# Label phishing as 1, all others as 0
df['label'] = df['type'].map(lambda x: 1 if x == 'phishing' else 0)

# Drop rows where label could not be mapped
df = df.dropna(subset=['label'])
```

4
```python
# Use character-level n-grams (3 to 5 characters) with TF-IDF
# This captures patterns in the URL text that may indicate phishing
vectorizer = TfidfVectorizer(analyzer='char_wb', ngram_range=(3, 5))

# Transform urls into numeric feature matrix
X = vectorizer.fit_transform(df['url'])

# Target labels
y = df['label']
```

5
```python
# Split data into train and temp sets
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# Split temp set into validation and test sets
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, random_state=42, stratify=y_temp
)
```

6
```python
# Initialize Logistic Regression classifier
# max_iter=1000 ensures convergence
lr = LogisticRegression(max_iter=1000, random_state=42)

# Train model
lr.fit(X_train, y_train)
```

7
```python
# Predict on test set
y_pred = lr.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculate precision
precision = precision_score(y_test, y_pred)

# Calculate recall
recall = recall_score(y_test, y_pred)

# Calculate f1 score
f1 = f1_score(y_test, y_pred)

# Print metrics
print("Model Evaluation Metrics:")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall   : {recall:.4f}")
print(f"F1 Score : {f1:.4f}")
```

8
```python
# Print detailed classification report with precision, recall, f1 per class
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Non-Phishing', 'Phishing']))
```

Figure 3: Source code for the TF-IDF Vectorization model implemented using AI.

Results

Structured Feature-Based Random Forest Classification

The first experiment applied a Random Forest classifier on manually engineered URL features. Model performance was evaluated using the test set, with results summarised in Table 2 and Figure 4.

Low recall value suggested the model missed most phishing URLs.

Metric	Value
Accuracy	0.8605
Precision	0.6540
Recall	0.1046
F1 Score	0.1803

Table 2: Performance metrics for the Random Forest model on structured features

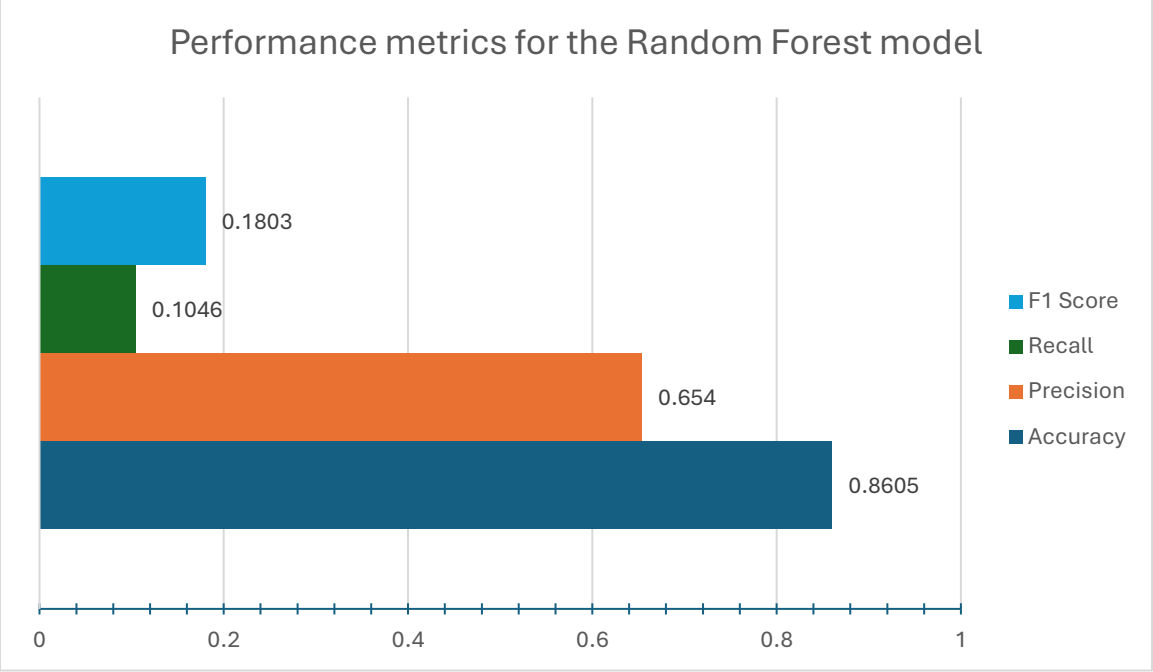


Figure 4: Higher values indicate better classification performance for each metric

Feature Importance Analysis

The contribution of each URL attribute to the Random Forest model's decisions is presented in Table 3 and Figure 5:

Feature	Importance (%)
URL length	48.2
Special characters	21.0
Digits	21.0
IP address presence	1.90

Table 3: Feature importance analysis for Random Forest model

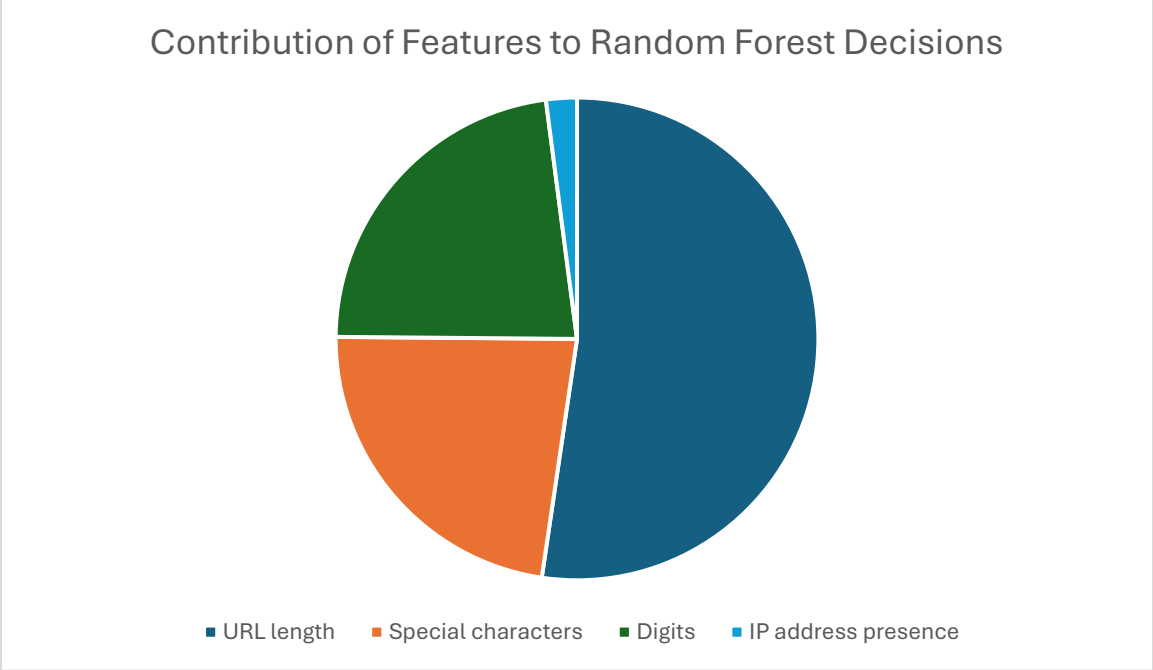


Figure 5:Larger slices indicate more contribution to the Random Forest model's decisions.

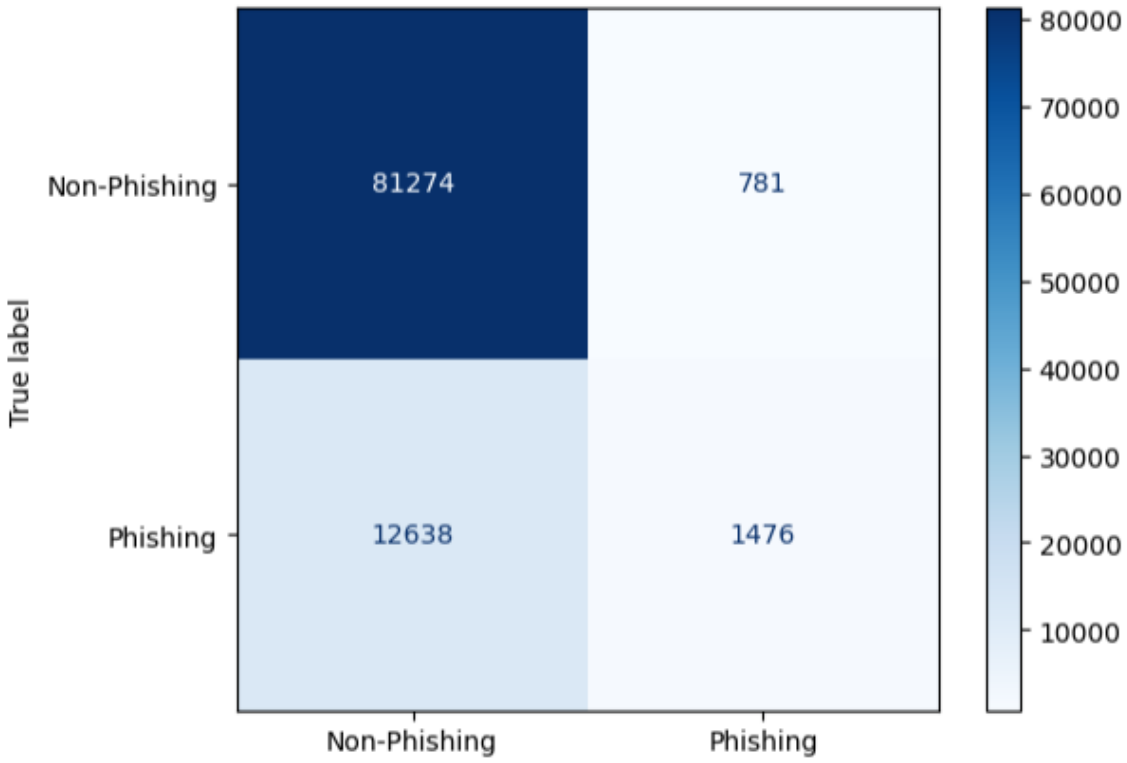


Figure 6: Confusion matrix generated for the Random Forest model

Result Observations

- URL length and special characters are the most significant predictors
- IP address presence contributes minimally to model performance.

Discussion

The Random Forest model showed clear limitations during the phishing URL detection testing with an extremely low recall value of 10.46%. Although it had a high accuracy of 86.00%, it failed to identify malicious URLs which makes it unreliable due to a high risk of false negatives. (Ejaz, et al., 2023)

This performance suggest that lexical features have been a victim of concept drift, since they are no longer sufficient signs of malicious behaviour, even though the model relied the most on URL length at 48.20% importance (Ejaz, et al., 2023) (Türk & Kılıçaslan, 2025). Also, the small impact of IP address presence at 1.90% further suggests that attackers have changed their tactics from obvious IP-based URLs to convincing domain mimicry (Adam, et al., 2024) (Wang, 2025).

These results suggest that Catal et al. (2022)'s research was accurate, since it was proved that deep learning models that heavily rely on superficial URL structure often gain results with high accuracy but low recall due to concept drift (Catal, et al., 2022) (Ejaz, et al., 2023). On the contrary, Barik et al. (2025) proves the superior performance of deep learning and hybrid models since they can process more complex data (Barik, et al., 2025) (Catal, et al., 2022).

Overall, the results proved that lexical-based features are no longer sufficient for modern-day phishing detection. To improve the recall value, the model would need to have more details in its feature sets (Ejaz, et al., 2023).

Text Vectorization-Based Logistic Regression Classification

The second experiment applied a Logistic Regression classifier to TF-IDF vectorized URL text. After the model performance was evaluated, the near-perfect accuracy, high precision, and high recall effectively proved efficient detection of malicious URLs.

Metric	Value
Accuracy	0.9606
Precision	0.9234
Recall	0.7980
F1 Score	0.8561

Table 4: Performance metrics for the Logistic Regression model using TF-IDF features

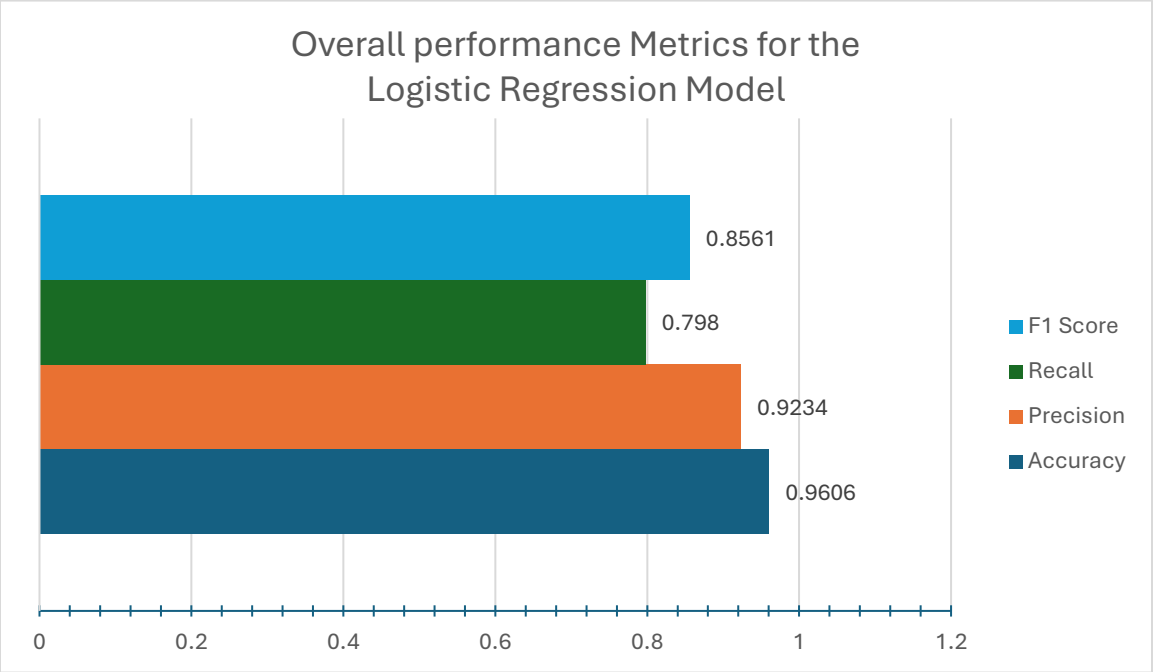


Figure 7: Performance for each metric where higher values indicate better classification.

Classification Report

A breakdown of the different performance metrics for each class is shown in the tables and figure below:

Class	Precision	Recall	F1-Score	Sample Count
Non-Phishing	0.97	0.99	0.98	82,055
Phishing	0.92	0.80	0.86	14,114

Table 5: Class-level precision, recall, F1-score, and sample count for both classes

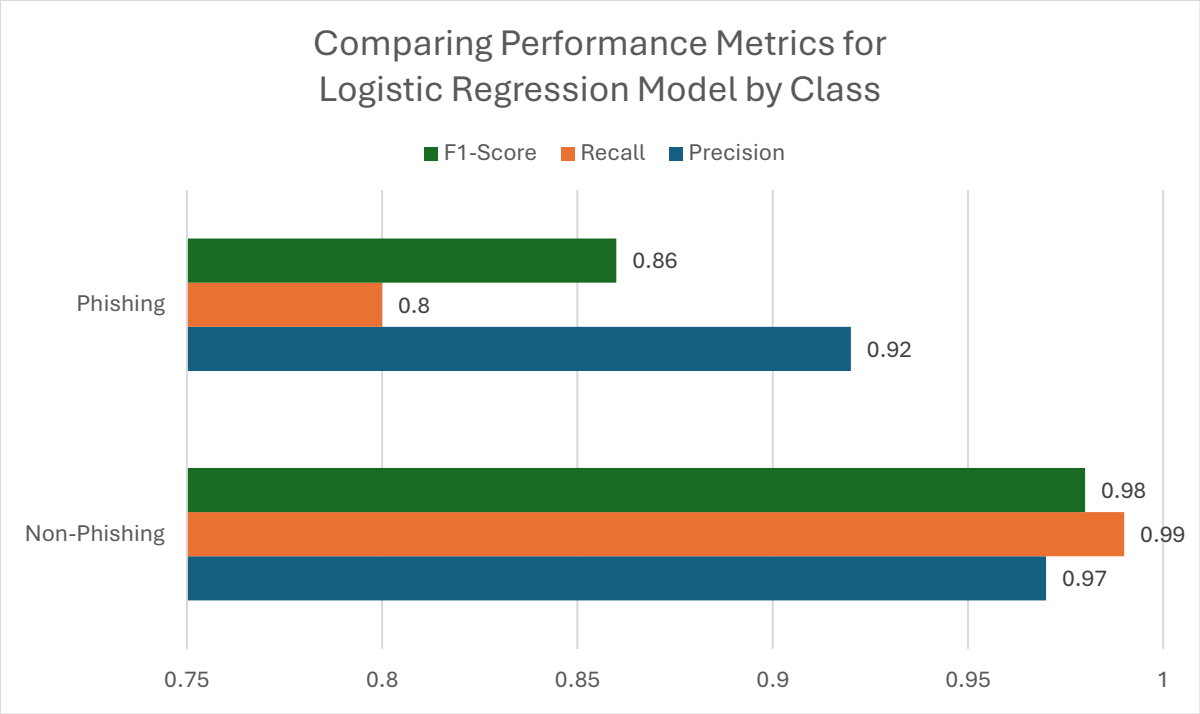


Figure 8: Precision, recall, and F1-score for the respective classes where higher values are better.

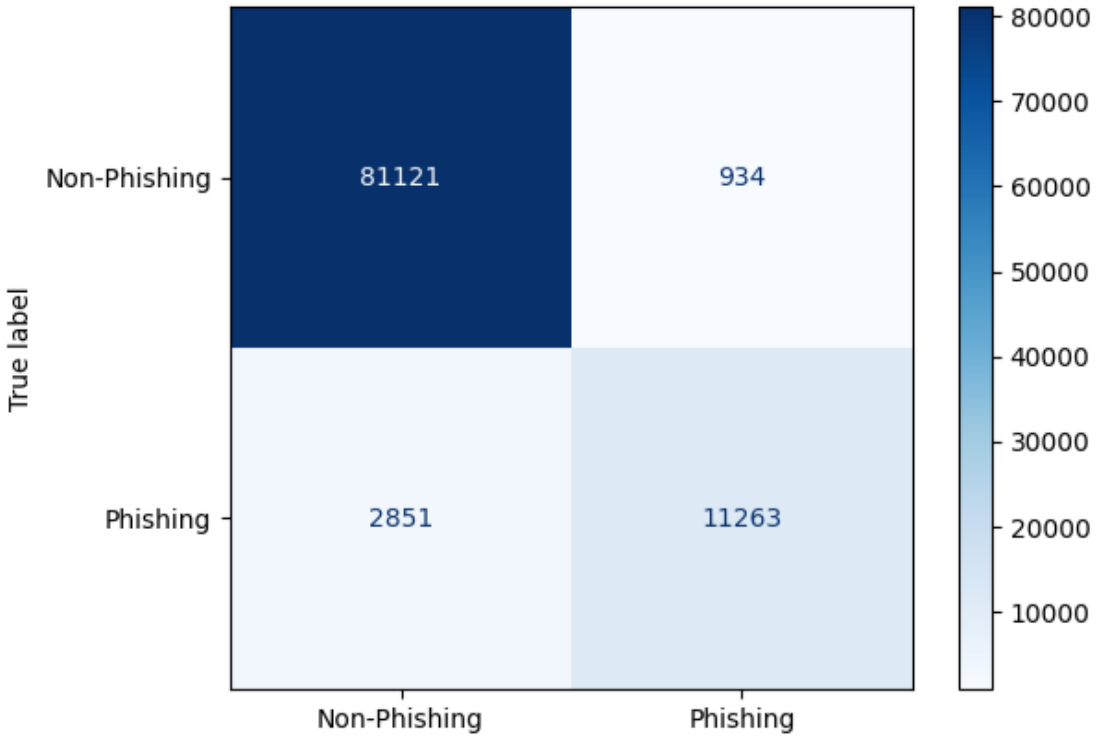


Figure 9: Confusion matrix generated for the Logistic Regression model

Result Observations

- The Logistic Regression model significantly outperforms the Random Forest model in phishing detection.
- High precision and accuracy for both phishing and non-phishing classes, with very low false positives.

Discussion

The Logistic Regression model showed a clear distinction of performance when detecting phishing URLs, especially when compared to the Random Forest model from the previous experiment. While the Random Forest model struggled, the Logistic Regression model had extremely high performance metrics with an overall accuracy of 96.06% and recall of 79.80% (Catal, et al., 2022) (Türk & Kılıçaslan, 2025).

This performance suggests that models trained on text-based features can detect deeper statistical patterns that attackers cannot yet avoid (Catal, et al., 2022). Unlike the Random Forest model, where attackers easily evaded detection by creating convincing mimicry domains, the Logistic Regression model accurately identified both non-phishing and phishing URLs. Between both classes, they share similar results, however, the phishing URL class' recall was much lower due to a class imbalance (Hasan, 2024) (Türk & Kılıçaslan, 2025).

These results suggest that implementing deep learning models that focus on holistically interpreting URL text with additional contextual information, can withstand the risk of degradation by concept drift. This has already been proven to be true by Barik et al. (2025), where they concluded that models that detect using lexical anomalies are outdated and exploited (Barik, et al., 2025) (Ejaz, et al., 2023).

Overall, when implementing AI-based phishing detection solutions, the better model is Logistic Regression, as it is more efficient and has a stronger performance in malicious link detection (Catal, et al., 2022). Due to its holistic approach, it should be able to withstand current attack-evasion techniques (Türk & Kılıçaslan, 2025).

Critical Analysis

Limitations

- **Structured Feature-Based Random Forest**:

  The model failed to detect phishing URLs effectively due to its reliance on manually engineered lexical features such as URL length and special characters, which promotes degradation via concept drift (Ejaz, et al., 2023). Additionally, features like IP address presence, which have the highest importance in this model, contribute minimally to prediction resulting in reduced model effectiveness (Adam, et al., 2024) (Hasan, 2024).

- **Text Vectorization-Based Logistic Regression**:

  While the model achieved near-perfect performance metrics, these results may be influenced by class imbalance, as phishing URLs were less represented in the dataset (Hasan, 2024). Additionally, the model is too dependent on text patterns and does not incorporate domain features. If attackers adapt their techniques, this model too can fall victim to concept drift in the future (Ejaz, et al., 2023).

Relation to Literature

- The results align with prior studies indicating that manually engineered lexical features alone are insufficient for capturing sophisticated phishing methods (Adam, et al., 2024) (Ejaz, et al., 2023).
- The effectiveness of TF-IDF and text-based machine learning models corroborates existing literature showing that character-level and n-gram patterns improve phishing detection performance (Catal, et al., 2022) (Türk & Kılıçaslan, 2025).
- These findings also support research suggesting that hybrid approaches combining structured and textual features may provide more robust and generalizable solutions (Barik, et al., 2025) (Catal, et al., 2022).

Possible Improvements

After conducting the experiments, the following improvements can be made to AI-assisted phishing detection solutions:

Incorporate Hybrid Models:

- Addressing the issue of manual lexical feature over-dependency can be done by adding additional metadata in processing. By doing this, the hybrid model of both Logistic Regression and Random Forest should be able to effectively detect phishing URLs without the risk of being exploited by altered URL strings (Catal, et al., 2022) (Barik, et al., 2025).

  By introducing non-textual signals, this can potentially make the hybrid model resilient to future concept drift, while also maintaining high recall and accuracy values (Ejaz, et al., 2023).

Handling Class Imbalance:

- Addressing the class imbalance for the Logistic Regression model will be able to prove whether the results were biased or not. To do this, class-weighted loss functions can be applied to assist in balancing the classes. This function works by assigning a penalty value to misclassified phishing URLs to emphasize the significance of errors within the minority class (Hasan, 2024) (Türk & Kılıçaslan, 2025).

  Implementing this method should make the model more resilient against class imbalance, which would produce unbiased results (Engineering Proceedings, 2025).

Conclusion

This study demonstrated the relentless arms race between attackers developing new evasion tactics and cybersecurity professionals trying to adapt new defensive measures (Ejaz, et al., 2023). The results from the Structure Feature-Based Random Forest model experiment revealed that attackers have evolved past simple lexical signs, due to the model's low recall of 10.46% (Adam, et al., 2024) (Hasan, 2024).

On the contrary, the Text Vectorization-Based Logistic Regression model showed strong resilience against this concept drift with high performance metrics such as 96.06% accuracy and 79.80% recall (Türk & Kılıçaslan, 2025). This suggests that incorporating deep learning AI-models provide efficient resilience against evading phishing detection (Barik, et al., 2025).

When looking at the future of defending phishing attacks, organisations need to prioritise deep learning AI-enhanced phishing detection solutions, especially since attackers are also using AI (Ejaz, et al., 2023). By incorporating this enhancement coupled with non-lexical features, it can promote resilience against concept drift and improve efficiency when detecting phishing URLs (Catal, et al., 2022) (Türk & Kılıçaslan, 2025).

By proving the Text Vectorization-Based Logistic Regression model to be the better choice, this study also reinforces that effective phishing defense can no longer rely on static rules. Defense solutions must incorporate dynamic solutions, like deep learning AI-models, to stay ahead of rapidly evolving threats (Adam, et al., 2024) (Ejaz, et al., 2023).

References

Adam, H., Nasution, S., Simanungkalit, R. & Diansyah, I., 2024. *Machine Learning-Driven Detection of Malicious URL: Comparative Analysis of Random Forest and SVMs.* [Online]
Available at: https://sinta.kemdiktisaintek.go.id/journals/google/4143

Barik, K., Misra, S. & Mohan, R., 2025. *Web-based Phishing URL Detection Model Using Deep Learning Optimisation Techniques.* [Online]
Available at: https://doi.org/10.1007/s41060-025-00728-9

Catal, C., Giray, G. & Tekinerdogan, B., 2022. *Applications of deep learning for phishing detection: a systematic literature review.* [Online]
Available at: https://doi.org/10.1007/s10115-022-01672-x

CyBOK, 2021. *AI for Security Technical Guide v1.0.0.* [Online]
Available at: https://www.cybok.org/media/downloads/AI_for_Security_TG_v1.0.0.pdf

Ejaz, A., Mian, A. & Manzoor, S., 2023. *Life-long Phishing Attack Detection Using Continual Learning.* [Online]
Available at: https://doi.org/10.1038/s41598-023-37552-9

Engineering Proceedings, 2025. *Evaluation of AI Models for Phishing Detection Using Open Datasets.* [Online]
Available at: https://doi.org/10.3390/engproc2025107037

Hasan, M., 2024. *New Heuristics Method for Malicious URLs Detection Using Machine Learning.* [Online]
Available at:
https://www.researchgate.net/publication/384575046_New_Heuristics_Method_for_Malicious_URLs_Detection_Using_Machine_Learning

Siddhartha, M., 2021. *Malicious URLs dataset.* [Online]
Available at: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset?utm_source=chatgpt.com

Türk, F. & Kılıçaslan, M., 2025. *Malicious URL Detection with Advanced Machine Learning and Optimization-Supported Deep Learning Models.* [Online]
Available at: https://doi.org/10.3390/app151810090

Wang, Y., 2025. *Malicious URL Detection: An Evaluation of Feature Extraction and Machine Learning Algorithms.* [Online]
Available at: https://doi.org/10.54097/hset.v23i.3209