Detecting Phishing Websites

Abstract — **Applying machine learning to detect phishing websites using only URL-based features. Using the UCI Phishing Websites dataset, we train and evaluate several classifiers to distinguish phishing from legitimate sites. The proposed lightweight model achieves high accuracy and is suitable for real-time integration into browsers or email clients, offering an efficient approach to phishing detection.**

I. Introduction

Phishing remains one of the most prevalent cyber threats, targeting users by impersonating legitimate websites to steal confidential information such as login credentials, financial data, and personal details. Traditional phishing detection techniques often rely on blacklists or content-based analysis, which can be slow, resource-intensive, and vulnerable to zero-day attacks. In contrast, URL-based detection methods offer a lightweight and efficient alternative by analyzing easily accessible features derived directly from website URLs.

Recent studies have shown that machine learning algorithms can effectively classify phishing websites by learning patterns from URL characteristics such as length, presence of special symbols, domain registration details, and SSL certificate status. This project focuses on building a robust phishing detection model using only URL-based features, leveraging the comprehensive UCI Phishing Websites dataset. By exploring multiple machine learning classifiers and evaluating their performance with metrics like accuracy, precision, recall, and F1 score, we aim to develop an explainable and scalable solution suitable for real-time deployment.

II. Project Life Cycle

A. *Problem Defination*

We started by identifying the problem: phishing websites that trick users into revealing sensitive information. The project focuses on detecting phishing attacks using only features extracted from URLs, aiming to create a fast, lightweight, and scalable solution.

B. *Literature Review*

We reviewed existing research on phishing detection and machine learning techniques. Studies showed that URL based features alone could achieve High accuracy without needing to scan website content, making this approach ideal for real-time detection .

C. *Dataset Collection and Exploration*

We used the Phishing Websites Dataset from the UCI Machine Learning Repository. We explored the dataset to understand the types of features available (such as URL length, SSL certificate status, and special character usage) and to check for missing or inconsistent data.

D. *Data preproccesing*

The dataset was cleaned and prepared for modeling. This included encoding categorical features, normalizing numerical values, handling missing data, and splitting the data into training (80%) and testing (20%) sets to ensure fair model evaluation.

E. *Model selection and development*

We initially planned on using Traditional Machine Learning Models specifically Logistic Regression, Decision Trees, and Random Forests but after experimenting with multiple AI models we settled on Deep Neural Network (DNN) as it provided a balance between performance, accuracy ,and easy implementation.

F. *Model training and tuning*

Each model was trained on the processed dataset. We adjusted hyperparameters where necessary to optimize performance, using techniques like cross-validation to ensure that results were reliable and not due to random chance.

G. *Model Evaluation*

We evaluated each model using accuracy, precision, recall, F1 score, and confusion matrices. We also analyzed feature importance to understand which URL characteristics were most influential in phishing detection.

H. *Conclusion*

We Concluded that phishing websites can be effectively detected using only URL-based features. For future work, we suggested testing the model on real-world data, and integrating it into browser plugins

III. Challenges Faced Us

A. *Label Format Mismatchyms*

The dataset labeled legitimate websites as -1 and phishing websites as 1, but TensorFlow expects binary labels in the format 0 and 1. This mismatch caused training issues when using a DNN until the labels were converted properly.

B. *Poor Performance of Traditional Models.*

Logistic Regression, Decision Trees, and Random Forests performed poorly on this dataset despite proper preprocessing and validation. Their accuracy and generalization were not satisfactory, making them unsuitable for the final solution..

C. *DNN Implementation Complexity .*

Building and training the Deep Neural Network (DNN) using TensorFlow required more effort compared to using simpler models in scikit-learn. It involved defining the architecture, compiling the model correctly, and tuning parameters such as epochs and batch sizes..

D. *Need for Data Normalization*

Some models, especially the DNN, performed poorly until numerical feature values were normalized. The variation in scale between different features initially led to unstable or inaccurate training.

E. *Slight Class Imbalance*

The dataset had a slightly higher number of phishing examples compared to legitimate ones. This imbalance caused the models to favor the phishing class in early training runs, affecting recall and false positive rates.

F. *Understanding TensorFlow Debugging*

Debugging TensorFlow errors, especially with label formats and model input shapes, was time-consuming. Errors were not always intuitive, requiring extra care in model setup and data preparation.

IV. Lessons Learned

A. *Always Match Data Format to Model Requirements*

Before training, it's essential to verify that the dataset's labels and features match the model's input expectations. A small mismatch, like label values, can lead to failed or incorrect training.

B. *Not All Models Work Equally Well*

Simpler, traditional ML models may not handle certain datasets or patterns effectively. Flexibility in model choice and willingness to adapt can significantly improve outcome.

C. *Deep Learning Can Capture Complex Patterns*

The DNN was better at identifying phishing URLs because it captured deeper, non-linear relationships between features. Complex models can be more suitable when traditional ones fail.

D. *Preprocessing Directly Impacts Accuracys*

Normalizing input features is not optional—it's a critical step that directly affects model accuracy, especially for neural networks that are sensitive to input scales.

E. *Monitor Class Balances*

Even a mild imbalance in class distribution can affect model bias. It's important to keep an eye on metrics like precision, recall, and the confusion matrix rather than relying solely on accuracy .

F. *Time Investment in TensorFlow Pays Off*

While setting up a DNN in TensorFlow took more time, the improved performance justified the effort. Learning to work with lower-level tools offers more control and better results in complex projects .

G. *Feature Importance Helps Interpretability*

Understanding which features influence predictions allows for better model explanations and trust. Even in deep learning, feature inspection plays an important role in validating model behavior .

References

Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). "Phishing detection based on hybrid feature selection approach." International Journal of Computer Applications.

This paper demonstrates how feature selection enhances phishing detection accuracy when machine learning models are applied to URL-based and site based features.

Rao, R. S., & Ali, S. (2015). "Phishing websites detection using machine learning." Procedia Computer Science.

Explores a range of ML classifiers and URL-based features, validating their effectiveness for phishing detection.

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2015). "Intelligent phishing detection system for e-banking using fuzzy data mining."

Expert Systems with Applications. Investigates fuzzy and rule-based systems in phishing detection and