Emily Gelchie & Colin Fitzgibbons

Project Proposal for Phishing Detection Using Machine Learning

Introduction

Phishing attacks remain one of the most prevalent security challenges in the cyber world. They affect individuals and organizations by tricking them into disclosing confidential information. With the rapid evolution of phishing techniques, traditional security measures often fall short. This project uses machine learning (ML) to develop a robust phishing detection system. We propose to use a comprehensive dataset designed for phishing link detection, which includes various features extracted from URLs, HTML content, and domain information.

Thesis Questions

Feasibility: Can machine learning models effectively differentiate between phishing and legitimate URLs based on the dataset provided?
Accuracy: Which machine learning model achieves the highest accuracy in detecting phishing URLs?
Feature Importance: Which features most predict phishing activity, and how do they influence the model's decision-making process?
Real-time Detection: How can the model be adapted for real-time phishing detection, and what are the performance implications?

Data Sources

The primary data source for this project will be the dataset available on Kaggle at this link. This dataset is specifically tailored for phishing link detection, incorporating many features extracted from the URL, HTML content, and domain information. It is a combination of two smaller data sets.

Project Steps and Timeline

Week 1 (March 26 - April 1): Project Setup and Data Acquisition

- Finalize project proposal and objectives.
- Download the dataset from Kaggle.
- Set up the development environment, including necessary libraries and tools for data analysis and machine learning (e.g., pandas, scikit-learn, Jupyter Notebooks).

Week 2 (April 2 - April 8): Data Exploration and Preprocessing

- Perform an initial dataset exploration to understand the features and data structure.

- Clean the data, addressing missing values, outliers, and any inconsistencies.
- Conduct feature engineering to enhance model performance, including encoding categorical variables and normalizing numerical features.

Week 3 (April 9 - April 15): Model Selection and Training

- Split the dataset into training and testing sets.
- Select a range of machine learning models for evaluation (e.g., Logistic Regression, Decision Trees, Random Forest, SVM, and Neural Networks).
- Train the models on the training set and perform an initial evaluation using the testing set.

Week 4 (April 16 - April 22): Model Optimization and Evaluation

- Fine-tune model parameters using cross-validation and grid search techniques to improve performance.
- Evaluate model performance using appropriate metrics (accuracy, precision, recall, F1-score).
- Identify the most important features contributing to phishing detection.

Week 5 & 6 (April 23 - May 6): Documentation and Finalization

- Document the project, including the problem statement, methodology, results, and conclusions.
- Prepare a final presentation outlining key findings and future work.
- Review the project, ensuring all objectives have been met and preparing for project submission.

Data Handling and Preprocessing

Pandas: Essential for data manipulation and analysis. It provides data structures and operations for manipulating numerical tables and time series. Use pandas to load your dataset, handle missing values, filter data, and perform exploratory data analysis (EDA).
NumPy: Fundamental package for scientific computing with Python. It offers powerful mathematical functions, random number capabilities, and support for large, multi-dimensional arrays and matrices. Useful for any numerical operations on your data.
Scikit-learn (sklearn): This library offers various preprocessing modules. Use sklearn.preprocessing for scaling, normalization, and encoding categorical features. For handling imbalanced datasets that can be common in phishing detection, look into sklearn.utils.class_weight and techniques like SMOTE (Synthetic Minority Over-sampling Technique) available through the imbalanced-learn package.

Feature Engineering and Selection

Feature-engine: A feature engineering library that extends scikit-learn and simplifies many feature engineering tasks such as variable transformation, outlier removal, binning, and categorical encoding.
Scikit-learn (sklearn) also provides tools for feature selection, such as SelectKBest for univariate feature selection and SelectFromModel for model-based feature selection. These can help improve model performance by removing irrelevant features.

Model Building and Evaluation

Scikit-learn (sklearn): Offers a wide range of machine learning models including logistic regression, SVM (Support Vector Machines), decision trees, random forests, and many more. It also provides tools for model evaluation such as cross-validation (cross_val_score or KFold) and metrics (e.g., accuracy_score, f1_score).
XGBoost: An optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a powerful option for building more sophisticated models that can potentially offer better performance than standard scikit-learn models.
LightGBM: A fast, distributed, high-performance gradient boosting (GBDT, GBRT, GBM, or MART) framework based on decision tree algorithms, used for ranking, classification, and many other machine learning tasks.
Keras/TensorFlow: For more advanced modeling, especially deep learning models which might prove useful in extracting patterns or features from URL strings directly, Keras with a TensorFlow backend can be used to build and train neural networks.