WEB EXTENSION PHISHING DETECTION SYSTEM USING MACHINE LEARNING

**Project Overview**

This project involves developing a **Web Extension Phishing Detection System** using machine learning techniques. Phishing is a fraudulent attempt to obtain sensitive information, and detecting it effectively is crucial to protect users online. By leveraging Python and its powerful libraries, we analyze web-based data to identify phishing websites.

**Features and Objectives**

* ****Objective**: Accurately classify websites as phishing or legitimate using supervised machine learning **in real time**.**
* ****Key Features**:**
  + **Extraction of URL-based, content-based, and domain-based features.**
  + **Implementation of multiple machine learning models to achieve optimal performance.**
  + **Visualization of data insights to improve feature selection and understanding.**

****Libraries Used****

* + ****Scikit-learn:** For implementing machine learning algorithms and evaluation metrics.**
  + ****Pandas:** For data manipulation and preprocessing.**
  + ****Matplotlib & Seaborn**: For creating informative data visualizations.**
  + ****NumPy**: For numerical computations.**

**Dataset**

* + ****Source:**** <https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning>
  + ****Structure:** The dataset contains features such as NumDots,SubdomainLevel,PathLevel,UrlLength,NumDash,NumDashInHostname,AtSymbol,TildeSymbol,NumUnderscore,NumPercent,NumQueryComponents,NumAmpersand,NumHash,NumNumericChars,NoHttps,RandomString,IpAddress,DomainInSubdomains,DomainInPaths,HttpsInHostname,HostnameLength among others, and a target label indicating whether the website is phishing (1) or legitimate (0).**

**Methodology**

**1. Data Preprocessing**

* + **Handling Missing Values:** Cleaned missing or inconsistent entries.
  + **Feature Scaling:** Normalized numerical features for better model performance.
  + **Encoding Categorical Features:** Converted non-numeric data into a format suitable for machine learning models.

**2. Exploratory Data Analysis (EDA)**

* + **Correlation Analysis:** Visualized feature relationships using heatmaps.
  + **Feature Distribution:** Plotted histograms and boxplots to understand feature variance.
  + **Outlier Detection:** Identified and handled outliers to reduce noise.

**3. Model Selection**

**Algorithms and Models Used:**

* + Gradient Boosting (XGBoost)
  + BERT
* **Evaluation Metrics:**
  + Accuracy
  + Precision, Recall, and F1 Score

**4. Training and Testing**

* **Train-Test Split:** Split the dataset into 80% training and 20% testing.
* **Cross-Validation:** Used k-fold cross-validation to prevent over-fitting.

**5. Results Visualization**

* Confusion Matrix: Visualized true positives, false positives, etc.
* ROC Curve: Plotted to compare the performance of different models.

**Results**

* **Best Model:** Per the previous research work, the XGB was the best classifier model, achieving an accuracy of 98%
* **Accuracy Achieved:** The model achieved an accuracy of 98% which is a good indicator that the model has recognized patterns in the dataset without over-fitting.
* **Insights:** Features such as URL length and presence of HTTPS protocol were highly indicative of phishing websites.

**System Architecture**

* **Machine Learning (ML) Engine:** Evaluates and flags active webpages.
* **Flask Web App:** Development Environment for the web extension. Includes an admin interface for companies to log in and view url logs and control access.
* **Database (SQLite):** Stores logs of analyzed URLs, companies, admins, and blacklist entries.
* **Company API Key System:** Each company gets a secure API key to log phishing events.
* **Admin Panel:** Allows companies to:
  + Log in securely
  + View their own logs (scoped by company\_id)
  + Blacklist suspicious URLs
  + Create new companies (if superadmin)

**User Roles & Access Control**

* *Admin (per company)*
  + Logs in via `/company/login`
  + Views only their company logs via `/company/dashboard`
  + Can blacklist URLs suspicious to their company
* *API Access*
  + Companies can use their API Key (`X-API-KEY` header) to interact programmatically with the system.

**Database Models**

* Company
  + `id`, `name`, `api\_key`
* AdminUser
  + `id`, `email`, `password\_hash`, `company\_id`
* URLLog
  + `id`, `url`, `verdict`,`prediction\_score`, `timestamp`, `company\_id`
* Blacklist
  + `id`, `url`, `reason`

**Routes Overview**

**Authentication & Company**

* `GET /admin/create-company-form` -> Company creation form
* `POST /admin/create-company` -> Create a company (superadmin)
* `POST /company/login` -> Login for admins

**Logs**

* `GET /company/dashboard` -> Company URLLogs

**Blacklist**

* `POST /blacklist/add` -> Blacklist a URL (admin only)

**Frontend Templates**

* `company\_login.html` -> Secure login page for admins
* `company\_dashboard.html` -> Displays phishing logs (company-specific) with blacklist actions.
* `admin\_create\_company.html` -> Simple company creation form.

**Security Notes**

* Passwords are stored as hashed values using Werkzeug.
* API keys are randomly generated using Python's `secrets` library.
* Admins only see their company's logs.
* Blacklisting ensures suspicious URLs can be blocked for future checks.

**Challenges and Future Work**

**Challenges**

* Feature Engineering: Required significant domain knowledge to derive useful features.
* False Positives: How to reduce false positives to prevent the XGB from incorrectly flagging legitimate websites as phishing.

**Future Work**

* Expanding the dataset to include newer phishing patterns.
* Testing deep learning models for potential performance improvement.
* Leveraging Intrusive Detection System to allow the extension function at the network level.
* JWT-based API auth (instead of API keys in headers)
* Emails alerts for blacklisted URLs