USING MACHINE LEARNING TO DETECT MALICIOUS WEBSITES OR URLS

Introduction :

Phishing attacks are a major concern for internet security. Phishing websites are designed to look like legitimate websites to steal sensitive information such as usernames, passwords, and credit card details. Machine learning has proven to be effective in detecting and preventing phishing attacks. This project aims to develop a machine-learning model for detecting phishing websites and URLs.

Machine Learning Aspect :

Algorithms such as Random Forest, Decision Trees, Support Vector Machines (SVM), or Neural Networks. These algorithms are commonly used in binary classification problems like phishing website detection.

Upon checking and testing some basic algorithms like Logistic regression, Random Forest, Decision Classifier, I have chosen Logistic Regression. Random Forest, Decision Classifier, took longer for data fitting and had lesser accuracy.

With Logistic regression Training Accuracy Achieved, Testing Accuracy Achieved

Training Accuracy Achieved : 0.9784203743122116

Testing Accuracy Achieved : 0.9636514559077306

Setting up Environment:

* Python integrated development environment (IDE) such as PyCharm, Spyder, or Jupyter Notebook. Here, I used VS Code.
* IDEs support Python libraries such as NumPy, Pandas, Scikit-learn, and TensorFlow, which are commonly used in machine learning projects. You can install these libraries using pip, which is a Python package manager.

Base Code:

|  |
| --- |
| Sample Code #1:  import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.tree import DecisionTreeClassifier  from sklearn.metrics import accuracy\_score  # Load the dataset  data = pd.read\_csv('phishing.csv')  # Split the dataset into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.iloc[:,:-1], data.iloc[:,-1], test\_size=0.2, random\_state=42)  # Create a Decision Tree classifier  clf = DecisionTreeClassifier()  # Train the classifier  clf.fit(X\_train, y\_train)  # Make predictions on the testing set  y\_pred = clf.predict(X\_test)  # Evaluate the performance of the classifier  acc = accuracy\_score(y\_test, y\_pred)  print("Accuracy:", acc) |

Notes:

* import the necessary libraries such as pandas, sklearn's DecisionTreeClassifier, and accuracy\_score.
* load the dataset from a CSV file ('phishing.csv')
* split it into training and testing sets.

We create a Decision Tree classifier and train it using the training set. Finally, we make predictions on the testing set and evaluate the performance of the classifier using the accuracy score.

References for learning Feature Extraction :

<https://github.com/shreyagopal/Phishing-Website-Detection-by-Machine-Learning-Techniques/tree/master/DataFiles>

The set of phishing URLs are collected from an open-source service called PhishTank. This service provides a set of phishing URLs in multiple formats like csv, json etc. that gets updated hourly. To download the data: https://www.phishtank.com/developer\_info.php.

The legitimate URLs are obtained from the open datasets of the University of New Brunswick, https://www.unb.ca/cic/datasets/url-2016.html. This dataset has a collection of benign, spam, phishing, malware & defacement URLs.

A finished dataset could also be found from kaggle.com.

I learned feature extraction based on the following feature categories taken from the URL:

* Features based on the address bar.
* Features based in Domains.
* JavaScript and HTML-based Features

Additionally, the UCI website was also referenced: <https://archive.ics.uci.edu/ml/datasets/Phishing+Websites>

Challenges faced while developing the Machine learning model:

One of the main challenges in developing the machine learning model was the availability of a large and diverse dataset. Another challenge was the selection of appropriate features for the model. Additionally, it was challenging to balance accuracy and efficiency in the model since it needs to be able to classify websites in real-time.