Simplified
Midterm Project

AI and Cyber Security
DSCI6015

Submitted by
Sai Teja Madiri

University of New Haven
Dr.Vahid Behzadan
May05, 2024

Overview

The purpose of this research is to develop a machine learning model that is specifically made to detect malicious URLs, bolstering internet security measures against possible malware and phishing attacks. Our goal is to create a strong classifier that can correctly detect hazardous URLs, with an emphasis on stringent data pretreatment techniques and deliberate algorithm selection. We combine extensive performance evaluations with a range of algorithm tests to determine the optimal solution. Through the application of cutting-edge approaches, our Endeavoristo offers consumers a better defense against cyber risks that are prevalent in today's digital environment.

Introduction

The three main tasks of the project are to create, implement, and validate a machine learning-driven malware detection system. Our goal with these efforts is to develop a strong solution that can correctly identify risks from malicious software. These are carefully planned out tasks that guarantee optimal performance and smooth integration of the detection system in a variety of operational settings. To further confirm the effectiveness and dependability of the implemented model and confirm its capacity to successfully reduce cybersecurity risks, extensive testing protocols are also used.

1. Model Training and Deployment:

Model training includes critical stages such as data pretreatment, feature extraction, and algorithmic selection. We begin by loading and preparing the dataset, then use techniques like TF-IDF vectorization to convert URL text into numerical characteristics. Following that, a variety of machine learning algorithms, ranging from decision trees to random forests, are tested to determine the best model for our unique purpose. We iteratively improve our model by subjecting it to rigorous evaluation via cross-validation and performance metrics such as accuracy and F1 score in order to reach peak performance and strong generalization on previously unknown data.

After rigorous training and assessment, the model is deployed into a production setting ready for instantaneous predictions. We bundle the trained model and other dependencies into a containerized environment by utilizing technologies such as Amazon Sage Maker. Next, scalable infrastructure is used to install this container, providing a basis for reliable and effective prediction serving. We make the capabilities of the model accessible over HTTP by creating endpoints, which allows for easy integration with a variety of systems and applications. In order to maintain the performance and dependability of the deployed model, procedures for continuous monitoring are included, together with facilities for dynamic resource scaling in response to demand fluctuations.

.

2. Client Development:

Following the model's successful deployment, a Python-based client application is designed to simplify user interaction with the deployed system. This user-friendly client program can accept URLs as input and methodically process them to extract relevant features. These collected features are then effortlessly transmitted to the Sage Maker endpoint for thorough classification. After the classification process is completed, the client application immediately provides the user with the model's prediction of the URL's probable dangerous or benign qualities. This shortened procedure allows users to quickly identify potential security hazards connected with web links, improving their digital safety posture.

3. Testing Client and Endpoint:

In this step, carefully chosen samples from the test dataset are used to undertake rigorous testing on both the deployed model endpoint and the developed client application. To be more precise, one benign URL and one malicious URL are hand-selected to act as sample specimens during testing. This phase’s main goal is to demonstrate the system’s ability to correctly classify the given URLs into harmful or benign categories. A carefully produced demonstration movie gives stakeholders a thorough grasp of the system's capabilities. The entire procedure—including URL selection, input, categorization, and results presentation—is painstakingly demonstrated in this video. By completing these tasks with ease, the project successfully illustrates the practical use of machine learning in URL classification. The project clearly highlights the effectiveness and reliability of the developed solution in identifying potentially hazardous URLs and enhancing cybersecurity protocols by overseeing the development of a robust detection system, deploying it as an API endpoint, and meticulously validating its functionality through extensive testing procedures.

Methodology

1. Data Collection and Preprocessing:

A broad dataset of URLs is carefully selected from a variety of sources, including lists of benign websites and repositories containing known harmful URLs. This dataset is subjected to a rigorous preparation process designed to remove duplicates, extract relevant information, and classify each URL into a unique category that differentiates between harmful and benign entities. The dataset is prepared for later use in training and assessing machine learning models designed for URL categorization tasks by means of this thorough preprocessing pipeline.

2. Feature Engineering:

Feature extraction techniques are routinely used to convert unstructured data into coherent feature representations suitable for machine learning research, thereby enabling the distillation of actionable insights from raw URL data. A number of parameters, including URL length, domain age, special character occurrences, and lexical features, are carefully extracted to allow for the encapsulation of unique characteristics present in both dangerous and benign URLs. Through this method, complete feature sets that capture subtle features of URL composition are produced, enabling strong classification models that can distinguish between malicious and benign web addresses.

3. Model Training with MNB:

The methodology uses Multinomial Naive Bayes (MNB), a classically induced stochastic algorithm that is well-known for its simplicity and effectiveness, in an effort to create an efficient URL categorization model. Drawing on the built-in capabilities of MNB, the training procedure begins with a precise segmentation of the dataset into subsets for training and validation, which paves the way for an in-depth assessment of the model. The multinomial Naive Bayes algorithm's hyper parameters are systematically tuned through iterative testing and parameter tuning in order to reach the best possible configuration that optimizes both classification accuracy and generalization performance.

4. Model Evaluation and Validation:

An independent test dataset is used to rigorously evaluate the trained classifier for the Multinomial Naive Bayes (MNB) model, providing an essential benchmark to determine the model's effectiveness in identifying harmful from benign URLs. A thorough assessment of the model's performance across a range of classification criteria is provided by the carefully generated evaluation metrics, which include accuracy, precision, recall, and F1 score. These indicators allow for the quantification of the model's efficacy, providing important insights into its practical application and capacity to reduce potential cyber security risks brought on by bad URLs.

5. Deployment as a Web Service:

Using Amazon Sage Maker, the trained Multinomial Naive Bayes (MNB) model is made available as a scalable web service, making it easy to integrate with existing systems and applications. To handle incoming URL queries, classify them using the deployed MNB model, and provide real-time predictions about the URLs' nature, an API endpoint is set up. By ensuring quick and effective processing of URL data, this deployment method supports cyber security initiatives and strengthens defense systems against any threats.

These procedures, when carefully followed, will enable the project to develop a robust and flexible system for automatically identifying and classifying harmful URLs. By putting proactive detection mechanisms in place, this project aims to strengthen cyber security standards and reduce the possibility of running into cyber attacks.

Execution Steps:

1. Model Development and Training:

Create the development environment in AWS Sage Maker while guaranteeing smooth integration with the 1.2.1 version of Scikit-Learn. Assign binary labels for classification, extract relevant characteristics, and organize the labeled collection of URL samples. Utilizing the scikit-learn toolkit, train a Multinomial Naive Bayes (MNB) classifier and adjust hyper parameters precisely to maximize performance. To ensure resilience and dependability, gauge the performance of the trained MNB model using strict cross-validation procedures or by using a different validation dataset. Ultimately, employ joblib to serialize the learned MNB model into a file format, guaranteeing effective storage and permitting smooth implementation for subsequent uses.

2. Deployment of the Model as a Cloud API:

To install the Multinomial Naive Bayes (MNB) model, visit the Amazon Sage Maker dashboard and go to the Model tab. Uploading the serialized joblib file containing the trained MNB model to the console will create a new model. Configure deployment parameters as appropriate, such as the instance type and the number of instances required to host the model. Deploy the model by creating an endpoint that will give a scalable API capable of real-time predictions. Before moving further with additional integrations or apps, comprehensive endpoint testing is essential to ensure the deployment model's operation and the accuracy of its forecasts.

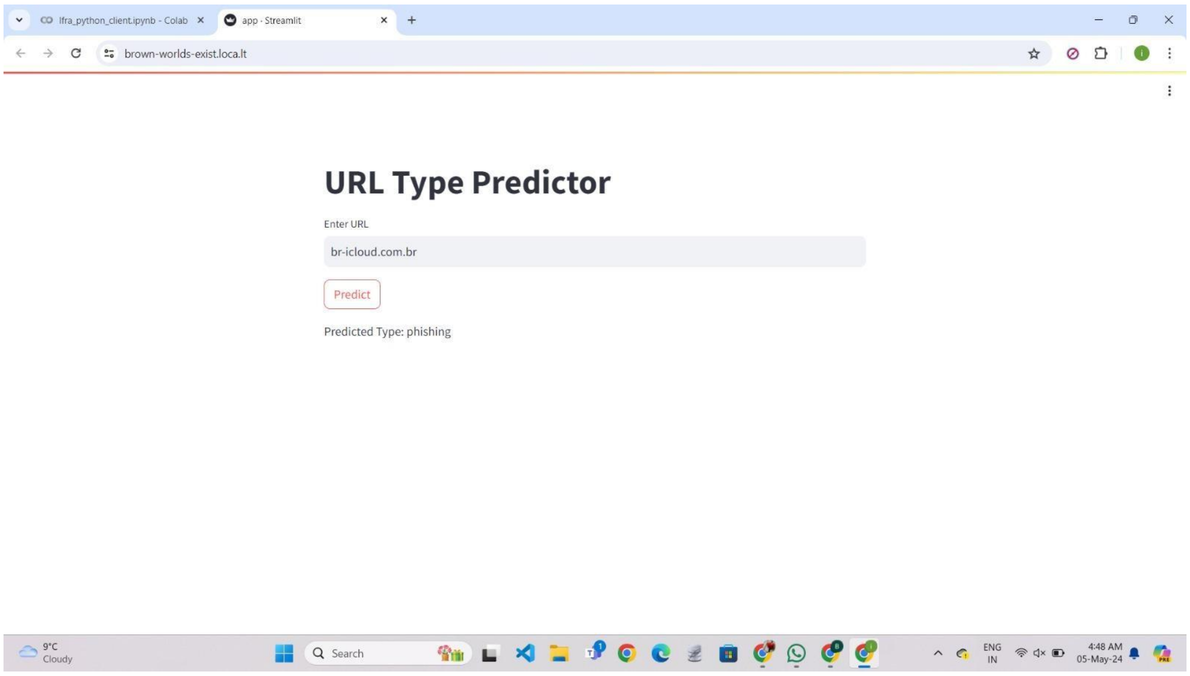
3. Development of Client Application:

The initial step in developing the client application in the AWS environment was to install Streamlit and its necessary dependencies. Streamlit was then used to create an intuitive user interface that allows for seamless interaction with the system. Functionality was added to allow users to submit URLs for classification and display the results.URL data was transmitted for prediction using the deployed model's API endpoint, and classification results were obtained. Finally, the classification results were presented within the client application interface, ensuring that they were clear and easy to read for users.

4. Validation:

A diverse set of URL samples, including both known dangerous and benign URLs, was carefully selected for validation. A validation script or pipeline was then created to automatically submit these URLs to the deployed API endpoint, allowing them to be classified. Following that, the classification results returned by the endpoint for each URL were recorded and rigorously compared to ground truth labels to ensure accuracy. To completely examine the model's performance, critical assessment metrics including as accuracy, precision, recall, and F1 score were computed. Based on the validation results, the model was iteratively improved to improve classification accuracy and reliability, ensuring optimal performance in real-world circumstances.

Output:



URL Type Predictor

Enter URL

br-icloud.com.br

Predict

Predicted Type: phishing