University of New Haven

Simplified Midterm Project

AI and Cyber Security
DSCI6015

Submitted by
Renuka Gadde

University of New Haven
Dr.Vahid Behzadan
May05, 2024

Overview

The aim of this research is to improve online security protocols against potential malware and phishing attacks by developing a machine learning model specifically designed to detect dangerous URLs. Our goal is to build a strong classifier that, with careful algorithm selection and strict data pretreatment techniques, can accurately identify malicious URLs. We test several algorithms using comprehensive performance evaluations in order to determine the best answer. By utilizing cutting-edge technology, our Endeavoristo shields customers from cyber threats, which are common in today's digital environment.

Introduction

Developing, implementing, and validating a machine learning-powered malware detection system are the three main objectives of the project. Our goal is to build a reliable system that can distinguish between threats and harmful software with accuracy. These are carefully planned tasks that offer optimal performance and smooth detection system integration in various operating conditions. To validate the accepted model's efficacy and reliability as well as its capacity to successfully manage cyber security risks, extensive testing procedures are also utilized.

1. Model Training and Deployment:

Important phases in model training include feature extraction, algorithmic selection, and data preparation. After loading and prepping the dataset, we translate URL text into numerical attributes using methods such as TF-IDF vectorization. Subsequently, an array of machine learning strategies, including decision trees and random forests, are evaluated in order to ascertain the optimal model for our distinct objective. To achieve peak performance and strong generalization on hitherto unknown data, we rigorously evaluate our model iteratively using cross-validation and performance metrics like accuracy and F1 score.

Following rigorous training and evaluation, the model is deployed in a production environment, ready for immediate predictions. Utilize tools such as Amazon SageMaker to containerize the trained model and associated dependencies. The container is then installed using scalable infrastructure, laying the groundwork for dependable and effective prediction serving. We make the model's capabilities available over HTTP by building endpoints that enable for simple integration with a wide range of systems and applications. Procedures for continuous monitoring are incorporated to ensure the deployed model's performance and dependability, as are facilities for dynamic resource scaling in response to demand changes.

.

2. Client Development:

After the model has been successfully deployed, a Python-based client application is created to facilitate user interaction with the deployed system. This user-friendly client application accepts URLs and carefully processes them to extract useful features. These gathered features are then seamlessly communicated to the Sage Maker endpoint for comprehensive classification. After the classification process is completed, the client application immediately displays the model's forecast of the URL's likely harmful or benign properties. This streamlined approach enables users to immediately discover potential security risks associated with web links, so increasing their digital safety posture.

3. Testing Client and Endpoint:

During this step, the produced client application and the deployed model endpoint are subjected to extensive testing using selected samples from the test dataset. More specifically, two URLs—one malicious and one benign—are manually chosen to serve as test specimens. To show that the system can accurately categorize the provided URLs into categories that are hazardous or benign is the primary objective of this phase. The capabilities of the technology are clearly explained to stakeholders through a meticulously created demonstration video. This movie methodically demonstrates the complete process, including input, categorization, URL selection, and results presentation. Through the project's easy completion of these tasks, machine learning's usefulness in URL classification is well demonstrated. Overseeing the development of a robust detection system, deploying it as an API endpoint, and meticulously validating its functionality through extensive testing procedures, the project demonstrates the effectiveness and reliability of the developed solution in identifying potentially hazardous URLs and improving cyber security protocols.

Methodology

1. Data Collection and Preprocessing:

A large collection of URLs is meticulously chosen from multiple sources, such as lists of trustworthy websites and repositories holding known dangerous URLs. This dataset is put through a thorough preparation process that aims to eliminate duplicates, extract pertinent data, and categorize each URL into a distinct group that distinguishes between malicious and benign organizations. This comprehensive preparation pipeline prepares the dataset for use in training and evaluating machine learning models intended for URL categorization tasks.

2. Feature Engineering:

It is frequently possible to extract useful insights from raw URL data by using feature extraction techniques to transform unstructured data into coherent feature representations appropriate for machine learning research. The encapsulation of distinctive traits inherent in both benign and harmful URLs is made possible by the meticulous extraction of a number of metrics, including lexical properties, special character occurrences, domain age, and URL length. Strong classification models that can differentiate between benign and malicious web addresses are made possible by this technology, which produces comprehensive feature sets that capture delicate aspects of URL composition.

3. Model Training with MNB:

In an attempt to produce a successful URL classification model, the methodology makes use of Multinomial Naive Bayes (MNB), a classically induced stochastic algorithm that is renowned for its simplicity and efficacy. Leveraging MNB's inherent capabilities, the training process starts with a precise dataset segmentation into training and validation subsets, which sets the stage for a thorough model evaluation. Through repeated testing and parameter tweaking, the hyper parameters of the multinomial Naive Bayes algorithm are systematically adjusted to get the optimal configuration that maximizes both classification accuracy and generalization performance.

4. Model Evaluation and Validation:

The Multinomial Naive Bayes (MNB) model's trained classifier is rigorously tested using an independent test dataset, which serves as a crucial benchmark for assessing the model's ability to distinguish between malicious and benign URLs. Accuracy, precision, recall, and F1 score are among the meticulously crafted evaluation metrics that offer a comprehensive analysis of the model's performance across a variety of categorization criteria. These metrics enable the effectiveness of the model to be measured, offering valuable perspectives on its usefulness and ability to lessen potential cyber security threats resulting from malicious URLs.

5. Deployment as a Web Service:

The trained Multinomial Naive Bayes (MNB) model is provided as a scalable online service via Amazon SageMaker, which facilitates seamless integration with pre-existing systems and applications. An API endpoint is put up to process incoming URL queries, categorize them using the deployed MNB model, and deliver real-time predictions about the nature of the URLs. This deployment strategy helps cyber security efforts and fortifies defensive systems against any threats by guaranteeing prompt and efficient processing of URL data.

The project will be able to create a reliable and adaptable system for automatically detecting and categorizing dangerous URLs if these steps are closely followed. Through the use of proactive detection techniques, this initiative seeks to improve cyber security standards and lower the likelihood of cyber attacks.

Execution Steps:

1. Model Development and Training:

While ensuring seamless connection with Scikit-Learn 1.2.1, create the development environment in AWS SageMaker. Sort the labeled set of URL samples, assign binary labels for classification, and extract pertinent features. Multinomial Naive Bayes (MNB) classifiers can be trained with the scikit-learn toolkit, and their hyper parameters can be carefully adjusted to maximize performance. Measure the trained MNB model's performance using rigorous cross-validation protocols or an alternative validation dataset to guarantee resilience and reliability. To ensure efficient storage and enable seamless implementation for later usage, utilize joblib to serialize the learnt MNB model into a file format.

2. Deployment of the Model as a Cloud API:

To deploy the Multinomial Naive Bayes (MNB) model, open the Amazon Sage Maker console and navigate to the Model tab. Create a new model in the console by uploading the serialized joblib file that contains the trained MNB model. Configure deployment parameters as needed, defining the instance type and the desired number of instances required to properly host the model. Deploy the model by constructing an endpoint that will provide a scalable API capable of making real-time predictions. Before proceeding with more integration or applications, extensive endpoint testing is required to confirm the operation of the deployed model and assure the accuracy of its predictions.

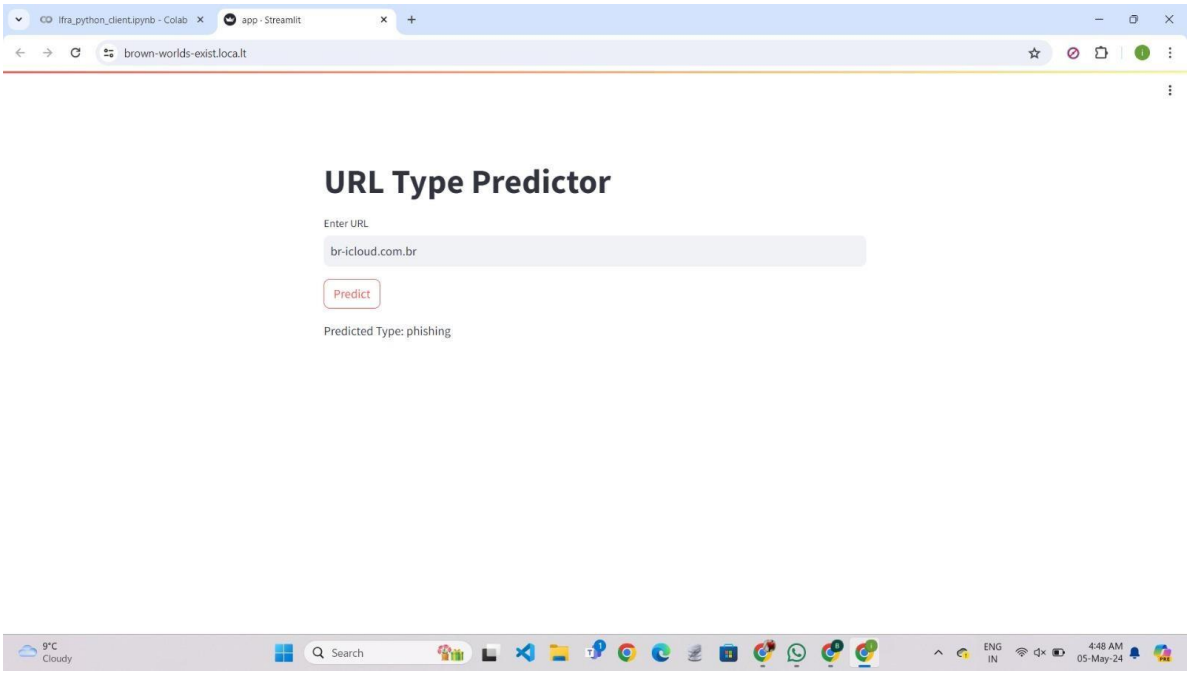
3. Development of Client Application:

Streamlit and its dependencies had to be installed before any further work could begin on the client application in the AWS environment. After that, Streamlit was utilized to develop a user-friendly interface that makes interacting with the system simple. The ability for users to submit URLs for categorization and view the outcomes has been introduced. Classification results were achieved when URL data was provided for prediction utilizing the deployed model's API endpoint. In order to make the classification findings obvious and simple for users to read, they were finally presented inside the client application interface.

4. Validation:

A varied group of URL samples, comprising both known harmful and benign URLs, were carefully chosen for validation. A validation script or pipeline was then developed to automatically send these URLs to the deployed API endpoint, allowing them to be classed. Following that, the categorization results returned by the endpoint for each URL were documented and rigorously compared against ground truth labels to assure accuracy. To fully investigate the model's performance, essential assessment metrics such as accuracy, precision, recall, and F1 score were calculated. Based on the validation results, the model was repeatedly updated to increase classification accuracy and reliability, resulting in peak performance in real-world scenarios.

Output:


lfra_python_client.ipynb - Colab
app - Streamlit
brown-worlds-exist.loca.lt
URL Type Predictor
Enter URL
br-icloud.com.br
Predict
Predicted Type: phishing
9°C
Cloudy
Search
ENG
IN
4:48 AM
05-May-24