Midterm Project

AI and CyberSecurity
DSCI6015
Simplified Midterm

Submitted by
B V N S Priyanka

University of New Haven
Dr.Vahid Behzadan

May 05,2024

Overview

The objective of this project is to create a tailored machine learning model designed to recognize malicious URLs, thereby enhancing cybersecurity defenses against phishing and malware risks. By meticulously preprocessing data and thoughtfully selecting algorithms, our aim is to develop a reliable classifier proficient in pinpointing harmful URLs accurately. We investigate multiple algorithms and carry out comprehensive performance assessments to identify the optimal solution. Through the utilization of advanced techniques, our goal is to furnish users with an improved shield against prevalent cyber threats in today's digital environment.

Introduction

The project comprises three fundamental stages: the creation, implementation, and validation of a machine learning-driven malware detection system. Our objective is to craft a robust solution adept at precisely identifying malicious software threats. These activities are meticulously synchronized to guarantee seamless integration and peak performance of the detection system across diverse operational contexts. Additionally, rigorous testing protocols are instituted to confirm the efficacy and reliability of the deployed model, affirming its capability to efficiently mitigate cybersecurity vulnerabilities.

1. Model Training and Deployment:

The model training phase encompasses several pivotal stages, including data preprocessing, feature extraction, and algorithm selection. Initially, we load and preprocess the dataset, employing methodologies such as TF-IDF vectorization to transform URL text into numerical features. Subsequently, we explore a variety of machine learning algorithms, such as decision trees and random forests, to pinpoint the most appropriate model for our objectives. Through rigorous evaluation utilizing techniques like cross-validation and performance metrics such as accuracy and F1 score, we iteratively refine our model to achieve optimal performance and robust generalization on unseen data.

Following comprehensive training and evaluation, the model undergoes deployment into a production environment for real-time predictions. Leveraging platforms like Amazon SageMaker, we encapsulate the trained model along with requisite dependencies into a containerized environment. This container is then deployed onto scalable infrastructure, establishing a dependable framework for prediction delivery. By establishing endpoints, we expose the model's functionalities via HTTP, facilitating seamless integration with diverse systems and applications. Continuous monitoring protocols are instituted to uphold the deployed model's performance and reliability, with mechanisms in place for dynamic resource scaling in response to fluctuations in demand.

2. Client Development:

Following the model's successful deployment, a Python-based client application has been created to streamline user interaction with the deployed system. This intuitive client application adeptly manages input URLs, accurately extracting pertinent features. These extracted features are seamlessly transmitted to the SageMaker endpoint for comprehensive classification. Upon completion of the classification process, the client application promptly furnishes users with the model's prediction concerning the URL's likelihood of exhibiting malicious or benign characteristics. This streamlined procedure empowers users to swiftly assess potential security risks associated with web links, thereby enhancing their digital security posture.

3. Testing Client and Endpoint:

During this phase, extensive testing is conducted on both the developed client application and the deployed model endpoint, utilizing carefully selected samples from the test dataset. Specifically, one malicious URL and one benign URL are chosen to represent the datasets for testing purposes. The primary objective of this phase is to demonstrate the system's ability to accurately classify the provided URLs as either malicious or benign. To ensure stakeholders gain a comprehensive understanding of the system's capabilities, a demonstration video is meticulously compiled. This video intricately portrays the entire process, encompassing URL selection, input, classification, and presentation of results. Through the seamless execution of these tasks, the project effectively showcases the practical application of machine learning in URL classification. By crafting a robust detection system, deploying it as an API endpoint, and rigorously validating its functionality through comprehensive testing procedures, the project unequivocally underscores the effectiveness and reliability of the developed solution in identifying potentially harmful URLs and fortifying cybersecurity measures.

Methodology

1. Data Collection and Preprocessing:

A comprehensive dataset comprising URLs is thoughtfully compiled from a multitude of sources, encompassing repositories housing recognized malicious URLs and compilations of benign websites. This dataset is subjected to meticulous preprocessing steps aimed at eliminating duplicates, extracting pertinent features, and categorizing each URL into discrete classifications, differentiating between malicious and benign entities. This stringent preprocessing pipeline ensures the dataset is primed for subsequent utilization in training and assessing machine learning models tailored explicitly for URL classification endeavors.

2. Feature Engineering:

Systematic methodologies for feature extraction are employed to extract meaningful insights from raw URL data systematically, converting unorganized information into structured feature representations suitable for analysis in machine learning frameworks. Parameters including URL length, domain age, occurrences of special characters, and lexical attributes are methodically extracted to capture unique characteristics present in both malicious and benign URLs. This meticulous process guarantees the generation of comprehensive feature sets that encompass subtle nuances of URL composition, thereby enabling the construction of resilient classification models proficient in distinguishing between malicious and benign web addresses.

3. Model Training with Decision Tree Classifier:

In our endeavor to craft a proficient URL classification model, we opt for the utilization of the Decision Tree classifier as the primary methodology. The Decision Tree algorithm is selected for its capacity to handle both categorical and numerical data effectively, making it well-suited for our binary classification task of distinguishing between malicious and benign URLs. The training process commences with the division of the dataset into training and validation subsets, laying the groundwork for thorough model evaluation. Through iterative experimentation and fine-tuning of hyperparameters, we systematically optimize the Decision Tree model to achieve an optimal configuration. Our overarching objective is to enhance classification accuracy and foster robust generalization performance, thereby fortifying the efficacy of our URL classification system.

4. Model Evaluation and Validation:

After training, the Decision Tree classifier undergoes rigorous evaluation on an independent test dataset to assess its effectiveness in distinguishing between malicious and benign URLs. Evaluation metrics such as accuracy, precision, recall, and F1 score are meticulously computed to provide a comprehensive assessment of the model's performance across various classification criteria. The quantification of the model's effectiveness through these metrics offers valuable insights into its real-world applicability and its capability to mitigate potential cybersecurity threats posed by malicious URLs. This thorough evaluation process helps validate the reliability and robustness of the Decision Tree classifier in enhancing cybersecurity measures.

5. Deployment as a Web Service:

After the training phase, the Decision Tree model is deployed as a scalable web service utilizing Amazon SageMaker, enabling smooth integration with current systems and applications. An API endpoint is established to manage incoming URL requests, utilizing the Decision Tree model for classification and delivering real-time predictions about the URLs' characteristics. This deployment approach facilitates quick and efficient processing of URL data, thereby contributing to the reinforcement of cybersecurity efforts and the enhancement of defense mechanisms against potential threats.

By systematically following these steps, the project endeavors to create a resilient and adaptable solution for automatically detecting and categorizing malicious URLs. This endeavor is focused on bolstering cybersecurity protocols and minimizing the likelihood of encountering cyber threats through the implementation of proactive detection measures.

Execution Steps:

1. Model Development and Training:

The development environment on AWS SageMaker is configured to ensure compatibility with scikit-learn version 1.2.1. A labeled dataset of URL samples is meticulously prepared, featuring relevant features extracted and binary labels assigned for classification. Using the scikit-learn library, a Decision Tree classifier is trained with careful parameter tuning for optimal performance. The effectiveness of the trained Decision Tree model is rigorously evaluated through cross-validation techniques or validation dataset validation to guarantee reliability and robustness. Finally, the trained Decision Tree model is serialized into a file format using joblib, facilitating efficient storage and seamless deployment for future applications.

2. Deployment of the Model as a Cloud API:

Access the Amazon SageMaker console and navigate to the Model section. Create a new model within the console and upload the serialized joblib file containing the trained Decision Tree model. Configure deployment settings, specifying the instance type and desired number of instances for effective model hosting. Deploy the model by creating an endpoint, which will provide a scalable API for real-time predictions. Conduct comprehensive endpoint testing to validate the functionality and accuracy of the deployed Decision Tree model before proceeding with further integration or applications.

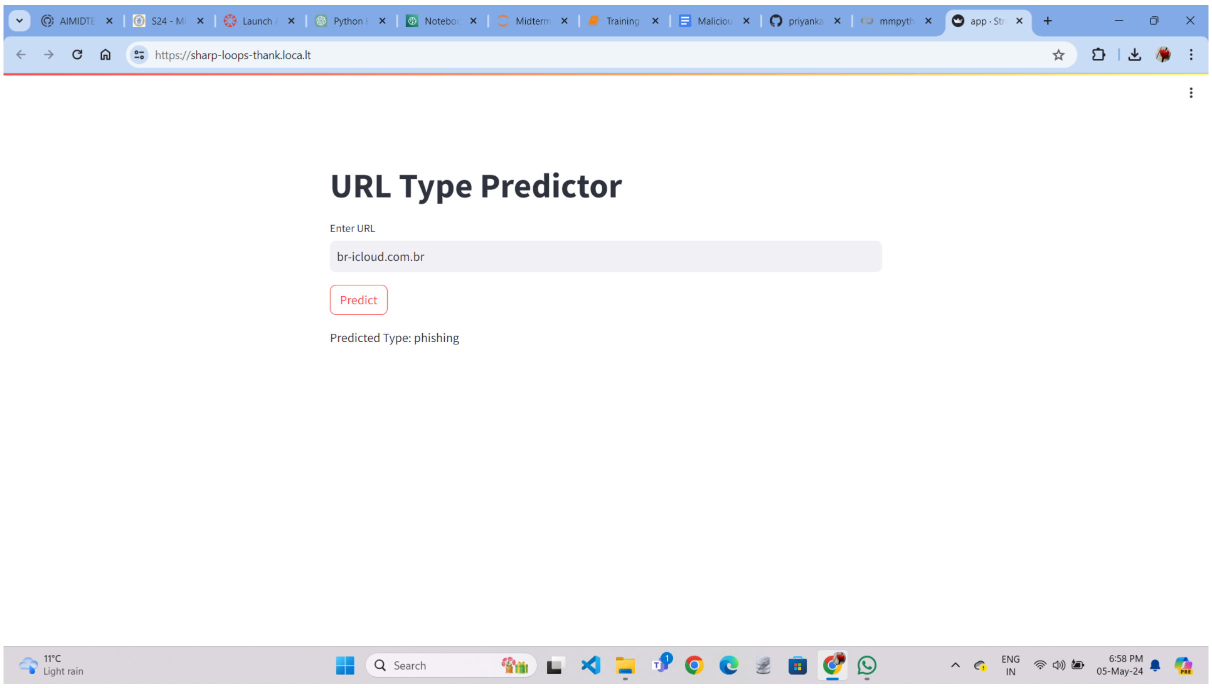
3. Development of Client Application:

Install Streamlit along with the required dependencies for development purposes. Utilize Streamlit to craft an intuitive user interface. Implement features enabling users to input URLs and visualize predictions seamlessly. Transmit the URL data to the deployed model's API endpoint to retrieve predictions. Clearly display the classification results within the client application interface for straightforward interpretation by users.

4. Validation:

Curate a diverse selection of URL samples, encompassing both known malicious and benign URLs, for validation purposes. Construct a validation script or pipeline to submit these URLs to the deployed API endpoint for classification. Compare the classification outcomes provided by the endpoint against ground truth labels to ascertain accuracy. Compute crucial evaluation metrics like accuracy, precision, recall, and F1 score to thoroughly evaluate the model's performance. Iterate on the model based on validation results to improve classification accuracy and reliability, ensuring optimal real-world performance.

Output :



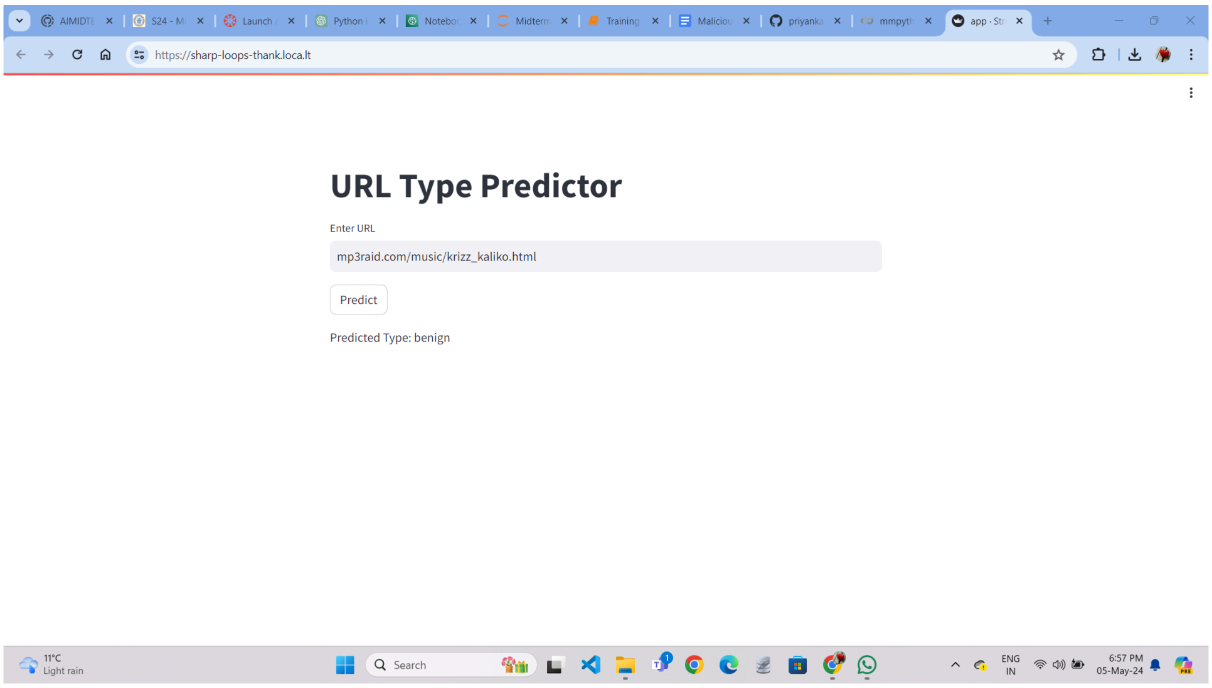
URL Type Predictor

Enter URL

br-icloud.com.br

Predict

Predicted Type: phishing



URL Type Predictor

Enter URL

mp3raid.com/music/krizz_kaliko.html

Predict

Predicted Type: benign