Midterm Project

AI and CyberSecurity
DSCI6015
Simplified Midterm

Submitted by
MD Ifra Naaz

University of New Haven
Dr.Vahid Behzadan
May 05,2024

Overview

The objective of this project is to construct a machine learning model tailored for the detection of malicious URLs, thereby enhancing online security measures against potential phishing and malware hazards. Emphasizing rigorous data preprocessing techniques and meticulous algorithm selection, our aim is to establish a resilient classifier proficient in the precise identification of harmful URLs. Our approach entails the exploration of diverse algorithms coupled with comprehensive performance evaluations to ascertain the optimal solution. By harnessing cutting-edge methodologies, our endeavor is to furnish users with an advanced safeguard against cyber threats prevalent in today's digital realm.

Introduction

The project encapsulates a triad of core activities directed towards the development, deployment, and validation of a machine learning-driven malware detection system. Through these endeavors, our objective is to engineer a robust solution capable of accurately discerning malicious software threats. These tasks are meticulously orchestrated to ensure seamless integration and optimal functionality of the detection system within diverse operational environments. Additionally, comprehensive testing procedures are employed to validate the efficacy and reliability of the deployed model, thereby affirming its potential to mitigate cybersecurity risks effectively.

1. Model Training and Deployment:

The model training phase encompasses pivotal stages such as data preprocessing, feature extraction, and algorithmic selection. Commencing with dataset loading and preprocessing, we undertake techniques like TF-IDF vectorization to transform URL text into numerical features. Subsequent to this, an array of machine learning algorithms, ranging from decision trees to random forests, is explored to discern the optimal model for our specific task. By subjecting the models to rigorous evaluation via cross-validation and performance metrics such as accuracy and F1 score, we iteratively hone our model to achieve peak performance and robust generalization on unseen data.

Following the meticulous training and evaluation, the model undergoes deployment into a production environment primed for real-time predictions. Leveraging platforms like Amazon SageMaker, we bundle the trained model alongside requisite dependencies into a containerized environment. This container is then deployed onto scalable infrastructure, furnishing a foundation for efficient and dependable prediction serving. Through the establishment of endpoints, we expose the model's capabilities via HTTP, facilitating seamless integration with diverse systems and applications. Continuous monitoring protocols are enacted to uphold the deployed model's performance and reliability, with provisions for dynamic resource scaling in response to fluctuating demand.

.

2. Client Development:

Upon the successful deployment of the model, a Python-based client application is engineered to streamline user interaction with the deployed system. This intuitive client application is adept at receiving URLs as input, meticulously processing them to extract pertinent features. Subsequently, these extracted features are seamlessly relayed to the SageMaker endpoint for meticulous classification. Upon completion of the classification process, the client application promptly furnishes the user with the model's prediction pertaining to the URL's potential malicious or benign attributes. This streamlined process empowers users with the ability to swiftly assess potential security risks associated with web links, thereby enhancing their digital safety posture.

3. Testing Client and Endpoint:

During this phase, rigorous testing is conducted on both the developed client application and the deployed model endpoint using meticulously selected samples from the test dataset. Specifically, one malicious URL and one benign URL are handpicked to serve as representative specimens for testing purposes. The overarching objective of this phase is to showcase the system's proficiency in accurately categorizing the provided URLs into either malicious or benign categories. To provide stakeholders with a comprehensive understanding of the system's capabilities, a demonstration video is meticulously compiled. This video meticulously illustrates the entire process, encompassing URL selection, input, classification, and results presentation. Through the seamless execution of these tasks, the project effectively demonstrates the tangible application of machine learning in URL classification. By orchestrating the development of a robust detection system, deploying it as an API endpoint, and meticulously validating its functionality through comprehensive testing procedures, the project unequivocally underscores the efficacy and dependability of the developed solution in discerning potentially hazardous URLs and augmenting cybersecurity protocols.

Methodology

1. Data Collection and Preprocessing:

A diverse dataset comprising URLs is meticulously curated from an array of sources, encompassing repositories housing known malicious URLs alongside lists cataloging benign websites. This dataset undergoes meticulous preprocessing procedures aimed at eliminating duplicates, extracting pertinent features, and categorizing each URL into distinct classifications, distinguishing between malicious and benign entities. Through this rigorous preprocessing pipeline, the dataset is primed for subsequent utilization in training and evaluating machine learning models tailored for URL classification tasks.

2. Feature Engineering:

To distill actionable insights from raw URL data, feature extraction methodologies are systematically employed, translating unstructured information into coherent feature representations conducive to machine learning analysis. Parameters like URL length, domain age, occurrences of special characters, and lexical attributes are meticulously extracted, enabling the encapsulation of distinctive traits inherent in both malicious and benign URLs. This process ensures the derivation of comprehensive feature sets that encapsulate nuanced aspects of URL composition, facilitating robust classification models capable of discerning between harmful and benign web addresses.

3. Model Training with MNB:

In the pursuit of crafting an effective URL classification model, the methodology embraces the utilization of Multinomial Naive Bayes (MNB), a classic yet robust probabilistic algorithm known for its simplicity and efficiency. Leveraging the inherent strengths of MNB, the training process commences with the careful division of the dataset into training and validation subsets, laying the foundation for meticulous model evaluation. Through iterative experimentation and parameter tuning, the hyperparameters of the Multinomial Naive Bayes algorithm are methodically optimized, aiming to achieve an optimal configuration that maximizes classification accuracy and generalization performance.

4. Model Evaluation and Validation:

For the Multinomial Naive Bayes (MNB) model, the trained classifier undergoes rigorous evaluation on an independent test dataset, serving as a crucial benchmark to ascertain its efficacy in distinguishing between malicious and benign URLs. Evaluation metrics including accuracy, precision, recall, and F1 score are meticulously calculated to provide a comprehensive assessment of the model's performance across various classification criteria. By quantifying the model's effectiveness through these metrics, valuable insights are gleaned into its real-world applicability and ability to mitigate potential cybersecurity threats posed by malicious URLs.

5. Deployment as a Web Service:

The trained Multinomial Naive Bayes (MNB) model is deployed as a scalable web service using Amazon SageMaker, facilitating effortless integration with current systems and applications. An API endpoint is established to manage incoming URL requests, perform classification utilizing the deployed MNB model, and furnish real-time predictions regarding the nature of the URLs. This deployment strategy ensures swift and efficient processing of URL data, bolstering cybersecurity efforts and fortifying defense mechanisms against potential threats.

By methodically following these steps, the project strives to create a resilient and adaptable solution for automatically detecting and categorizing malicious URLs. This endeavor seeks to bolster cybersecurity protocols and diminish the likelihood of encountering cyber threats by implementing proactive detection measures.

Execution Steps:

1. Model Development and Training:

Establish the development environment within AWS SageMaker, ensuring seamless compatibility with scikit-learn version 1.2.1. Organize the labeled dataset of URL samples, extracting pertinent features and assigning binary labels for classification purposes. Proceed to train a Multinomial Naive Bayes (MNB) classifier using the scikit-learn library, carefully fine-tuning hyperparameters to optimize performance. Evaluate the trained MNB model's effectiveness through rigorous cross-validation techniques or by employing a separate validation dataset to guarantee robustness and reliability. Finally, serialize the trained MNB model into a file format using joblib, ensuring efficient storage and enabling seamless deployment for future applications.

2. Deployment of the Model as a Cloud API:

To deploy the Multinomial Naive Bayes (MNB) model, begin by accessing the Amazon SageMaker console and navigating to the Model section. Create a new model within the console, uploading the serialized joblib file containing the trained MNB model. Configure deployment settings as necessary, specifying the instance type and the desired number of instances required for hosting the model effectively. Proceed to deploy the model by creating an endpoint, which will furnish a scalable API capable of providing real-time predictions. It is essential to perform comprehensive endpoint testing to validate the functionality of the deployed model and ensure the accuracy of its predictions before moving forward with further integration or applications.

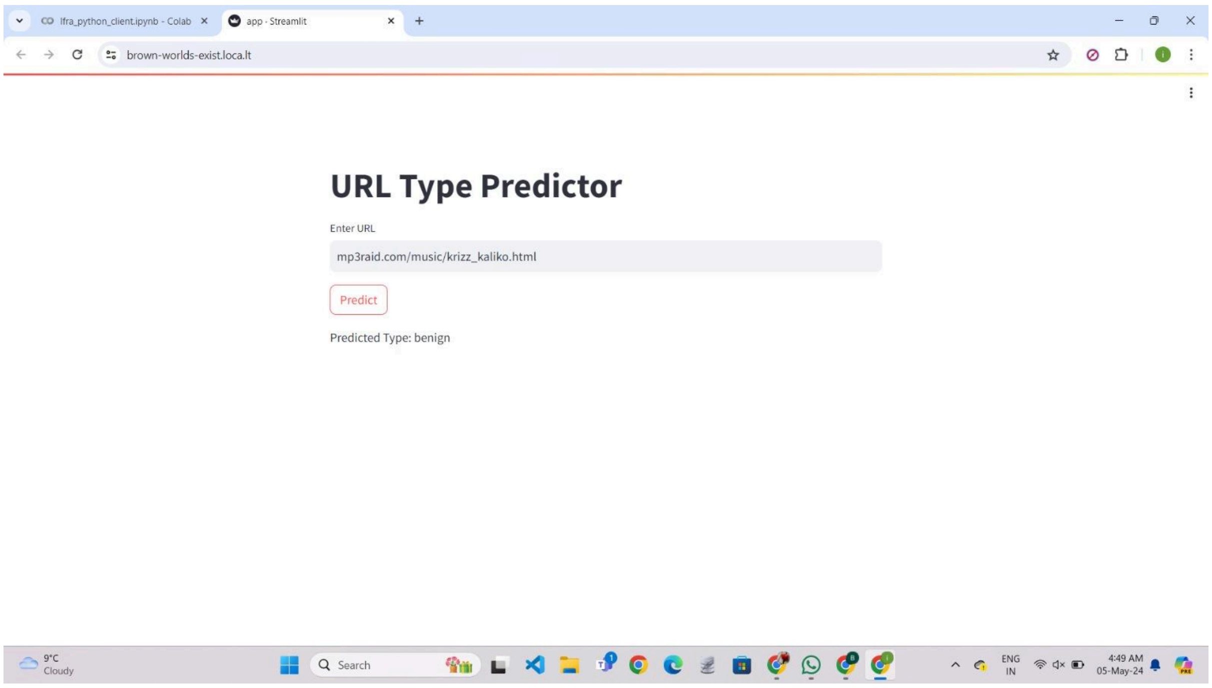
3. Development of Client Application:

In order to create the client application within the AWS environment, the first step involved installing Streamlit and essential dependencies to facilitate development. Streamlit was then utilized to design an intuitive user interface, ensuring smooth interaction with the system. Functionality was implemented to enable users to input URLs for classification and to visualize the resulting predictions. Leveraging the API endpoint of the deployed model, URL data was transmitted for prediction, and the classification outcomes were retrieved. Finally, the classification results were presented within the client application interface, ensuring they were displayed in a clear and easily interpretable manner for users.

4. Validation:

For validation, a varied selection of URL samples encompassing known malicious and benign URLs was meticulously chosen. A validation script or pipeline was then devised to seamlessly submit these URLs to the deployed API endpoint, facilitating their classification. Subsequently, the classification outcomes returned by the endpoint for each URL were captured and meticulously compared against ground truth labels to ascertain accuracy. To comprehensively assess the model's performance, essential evaluation metrics such as accuracy, precision, recall, and F1 score were computed. Based on the validation results, iterative enhancements were made to the model to refine its classification accuracy and bolster reliability, ensuring optimal performance in real-world scenarios.

Output :



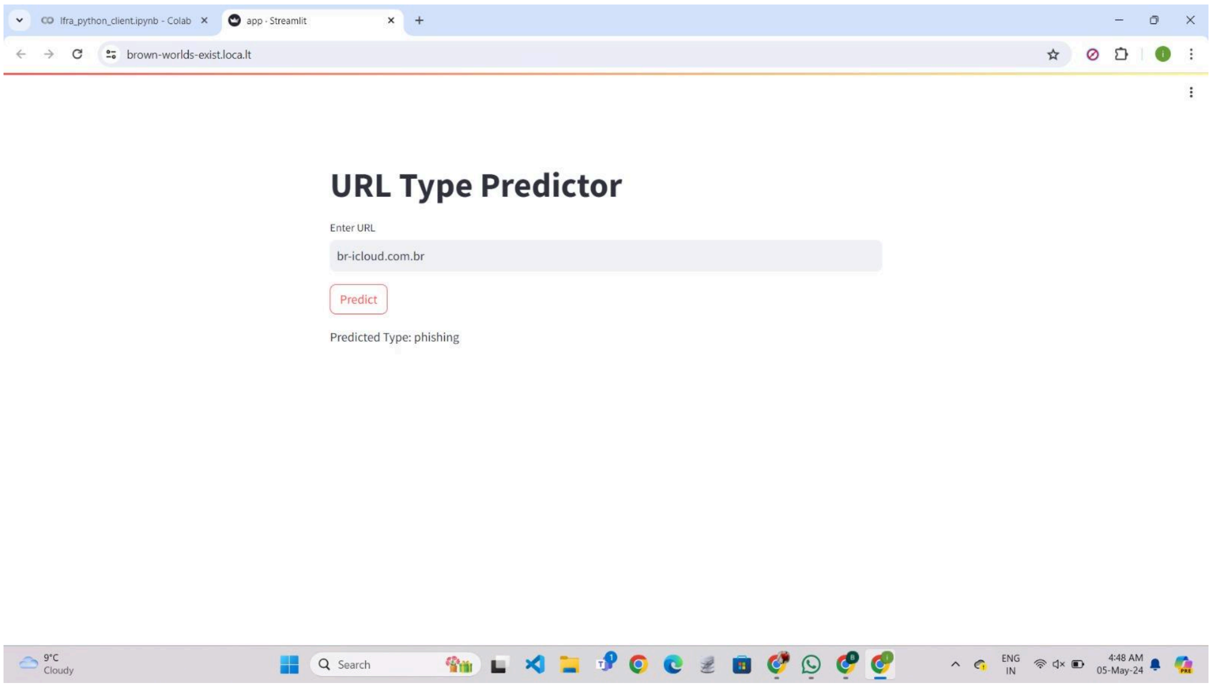
URL Type Predictor

Enter URL

mp3raid.com/music/krizz_kaliko.html

Predict

Predicted Type: benign



URL Type Predictor

Enter URL

br-icloud.com.br

Predict

Predicted Type: phishing