University of New Haven

Midterm Project

AI and CyberSecurity
DSCI6015
Simplified Midterm

Submitted by
A Sameer Chaitanya

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

Overview

This project aims to develop a specialized machine learning model focused on identifying malicious URLs, bolstering cybersecurity measures against phishing and malware threats. Through meticulous data preprocessing and careful algorithm selection, our goal is to create a robust classifier capable of accurately detecting harmful URLs. We explore various algorithms and conduct thorough performance evaluations to determine the most effective solution. By leveraging advanced methodologies, we seek to provide users with an enhanced defense against prevalent cyber threats in the digital landscape.

Introduction

The project encompasses three essential phases: development, deployment, and validation of a machine learning-based malware detection system. Our aim is to create a resilient solution capable of accurately identifying malicious software threats. These tasks are carefully coordinated to ensure smooth integration and optimal performance of the detection system across various operational scenarios. Furthermore, thorough testing procedures are implemented to verify the effectiveness and dependability of the deployed model, validating its ability to effectively mitigate cybersecurity risks.

1. Model Training and Deployment:

The model training phase involves several critical steps, including data preprocessing, feature extraction, and algorithm selection. We start by loading and preprocessing the dataset, utilizing techniques like TF-IDF vectorization to convert URL text into numerical features. Subsequently, we explore a range of machine learning algorithms, such as decision trees and random forests, to identify the most suitable model for our task. Through rigorous evaluation using cross-validation and performance metrics like accuracy and F1 score, we refine our model iteratively to achieve optimal performance and generalization on unseen data.

After thorough training and evaluation, the model is deployed into a production environment for real-time predictions. Using platforms like Amazon SageMaker, we package the trained model with necessary dependencies into a containerized environment. This container is then deployed onto scalable infrastructure, providing a reliable foundation for prediction serving. By creating endpoints, we expose the model's capabilities via HTTP, enabling seamless integration with various systems and applications. Continuous monitoring ensures the deployed model's performance and reliability, with provisions for dynamic resource scaling based on demand fluctuations.

2. Client Development:

After the successful deployment of the model, a Python-based client application has been developed to facilitate user interaction with the deployed system. This user-friendly client application efficiently handles input URLs, extracting relevant features with precision. These extracted features are then seamlessly sent to the SageMaker endpoint for thorough classification. Once the classification process is complete, the client application promptly provides users with the model's prediction regarding the URL's potential for malicious or benign attributes. This efficient process enables users to quickly evaluate potential security risks associated with web links, thereby bolstering their digital safety.

3. Testing Client and Endpoint:

In this phase, thorough testing is conducted on both the developed client application and the deployed model endpoint, utilizing carefully chosen samples from the test dataset. Specifically, one malicious URL and one benign URL are selected to represent the datasets for testing purposes. The primary aim of this phase is to demonstrate the system's capability 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 illustrates the entire process, including URL selection, input, classification, and presentation of results, in a detailed manner. Through the seamless execution of these tasks, the project effectively showcases the practical application of machine learning in URL classification. By developing a robust detection system, deploying it as an API endpoint, and rigorously validating its functionality through comprehensive testing procedures, the project unequivocally highlights the effectiveness and reliability of the developed solution in identifying potentially harmful URLs and enhancing cybersecurity measures.

Methodology

1. Data Collection and Preprocessing:

- A diverse dataset containing URLs is carefully curated from various sources, including repositories with known malicious URLs and lists of benign websites.
- This dataset undergoes meticulous preprocessing procedures to remove duplicates, extract relevant features, and classify each URL into distinct categories, distinguishing between malicious and benign entities.
- Through this rigorous preprocessing pipeline, the dataset is prepared for further use in training and evaluating machine learning models specifically designed for URL classification tasks.

2. Feature Engineering:

- Systematic feature extraction methodologies are utilized to distill actionable insights from raw URL data, transforming unstructured information into coherent feature representations suitable for machine learning analysis.
- Parameters such as URL length, domain age, occurrences of special characters, and lexical attributes are meticulously extracted to capture distinctive traits inherent in both malicious and benign URLs.
- This process ensures the creation of comprehensive feature sets that encapsulate nuanced aspects of URL composition, facilitating the development of robust classification models capable of discerning between harmful and benign web addresses.

3. Model Training with Logistic Regression:

- In the pursuit of constructing an effective URL classification model, logistic regression is adopted as the methodology.
- Logistic regression, a well-established statistical technique, is chosen for its simplicity and effectiveness in binary classification tasks.
- The training process initiates with the partitioning of the dataset into training and validation subsets, setting the stage for comprehensive model evaluation.
- Through iterative experimentation and parameter adjustment, the logistic regression model's hyperparameters are methodically optimized.
- The ultimate aim is to attain an optimal configuration that enhances classification accuracy and promotes robust generalization performance.

4. Model Evaluation and Validation:

- After training, the logistic regression classifier undergoes rigorous evaluation on an independent test dataset to determine its effectiveness in discerning between malicious and benign URLs.
- Evaluation metrics such as accuracy, precision, recall, and F1 score are meticulously computed to offer a holistic assessment of the model's performance across different classification criteria.
- The quantification of the model's effectiveness through these metrics provides valuable insights into its real-world applicability and its capability to mitigate potential cybersecurity threats posed by malicious URLs.

5. Deployment as a Web Service:

- Following training, the logistic regression model is deployed as a scalable web service utilizing Amazon SageMaker, ensuring easy integration with current systems and applications.

- An API endpoint is established to manage incoming URL requests, employing the logistic regression model for classification, and delivering real-time predictions regarding the URLs' nature.
- This deployment strategy facilitates rapid and efficient processing of URL data, contributing to the reinforcement of cybersecurity efforts and the enhancement of defense mechanisms against potential threats.

Through systematic adherence to these steps, the project aims to develop a robust and flexible solution for automatically detecting and categorizing malicious URLs. This initiative is geared towards strengthening cybersecurity protocols and reducing the risk of encountering cyber threats by implementing proactive detection measures.

Execution Steps:

1. Model Development and Training:

1. Configure the development environment on AWS SageMaker, ensuring compatibility with scikit-learn version 1.2.1.
2. Prepare the labeled dataset of URL samples, extracting relevant features and assigning binary labels for classification.
3. Train a logistic regression classifier using the scikit-learn library, meticulously adjusting hyperparameters for optimal performance.
4. Evaluate the effectiveness of the trained logistic regression model through rigorous cross-validation techniques or validation dataset validation to ensure reliability and robustness.
5. Serialize the trained logistic regression model into a file format using joblib, facilitating efficient storage and seamless deployment for future applications.

2. Deployment of the Model as a Cloud API:

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

3. Development of Client Application:

1. Install Streamlit and necessary dependencies for development.
2. Utilize Streamlit to design a user-friendly interface.
3. Implement functionality for users to input URLs and visualize predictions.
4. Transmit URL data to the deployed model's API endpoint for prediction retrieval.

5. Display classification results clearly within the client application interface for easy interpretation by users.

4. Validation:

1. Select a diverse range of URL samples, including known malicious and benign URLs, for validation.
2. Develop a validation script or pipeline to submit these URLs to the deployed API endpoint for classification.
3. Compare the classification outcomes returned by the endpoint against ground truth labels to determine accuracy.
4. Compute essential evaluation metrics such as accuracy, precision, recall, and F1 score to comprehensively assess the model's performance.
5. Iterate on the model based on validation results to enhance classification accuracy and reliability for optimal real-world performance.

Output :