Midterm Project

AI and CyberSecurity
DSCI6015
Simplified Midterm

S Santosh Kumar

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

Overview

This project aims to develop a machine learning model for detecting malicious URLs, bolstering online security against phishing and malware threats. Through meticulous data preprocessing and model selection, we strive to create a robust classifier capable of accurately identifying harmful URLs. Our methodology involves exploring various algorithms and evaluating their performance metrics to determine the most effective solution. By leveraging state-of-the-art techniques, we seek to provide users with an enhanced defense mechanism against cyber threats in the digital landscape.

Introduction

The project encompasses three primary tasks aimed at developing, deploying, and testing a machine learning-based malware detection system.

1. Model Training and Deployment:

The model training process involves crucial steps such as data preprocessing, feature extraction, and algorithm selection. After loading and preprocessing the dataset, we employ techniques like TF-IDF vectorization to convert URL text into numerical features. Subsequently, we experiment with different machine learning algorithms, including 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 iteratively refine our model to achieve optimal performance and generalization on unseen data.

Once the model is trained and evaluated, it is deployed to a production environment where it can serve predictions in real-time. Using services like Amazon SageMaker, we package the trained model along with necessary dependencies into a containerized environment. This container is then deployed to scalable infrastructure, allowing for efficient and reliable prediction serving. Endpoints are created to expose the model's functionality over HTTP, enabling seamless integration with other systems and applications. Continuous monitoring ensures the deployed model's performance and reliability, with the ability to scale resources dynamically based on demand.

.

2. Client Development:

Following the deployment of the model, a Python client application is created to facilitate interaction with the deployed system. This client accepts URLs as input, processes them to extract relevant features, and forwards these features to the SageMaker endpoint for classification. The client then presents the model's prediction regarding the URL's malicious or benign nature to the user, enabling quick assessment of potential security risks associated with web links.

3. Testing Client and Endpoint:

In this phase, the developed client application and the deployed model endpoint are tested using representative samples from the test dataset. One malicious URL and one benign URL are selected for testing. The objective is to demonstrate the system's functionality in accurately classifying the provided URLs. A demonstration video is compiled to showcase the entire process, including URL selection, input, classification, and results display.

Through the execution of these tasks, the project demonstrates the practical application of machine learning in URL classification. By developing a robust detection system, deploying it as an API endpoint, and validating its functionality through testing, the project underscores the effectiveness and reliability of the developed solution in identifying potentially harmful URLs and enhancing cybersecurity measures.

Methodology

1. Data Collection and Preprocessing:

A comprehensive dataset of URLs is collected from diverse sources, including known malicious URL repositories and benign website lists.

The dataset is preprocessed to remove duplicates, extract relevant features, and label each URL as either malicious or benign.

2. Feature Engineering:

Feature extraction techniques are applied to transform raw URL data into meaningful feature representations suitable for machine learning.

Features such as URL length, domain age, presence of special characters, and lexical analysis are extracted to capture the characteristics of both malicious and benign URLs.

3. Model Training with XGBoost:

XGBoost, a powerful gradient boosting algorithm, is selected as the primary model for URL classification due to its efficiency and high performance.

The dataset is split into training and validation sets, and hyperparameters for XGBoost are optimized using techniques like grid search and cross-validation.

4. Model Evaluation and Validation:

The trained XGBoost model is evaluated on an independent test dataset to assess its performance in accurately classifying malicious and benign URLs.

Evaluation metrics such as accuracy, precision, recall, and F1 score are computed to quantify the model's performance and effectiveness in real-world scenarios.

5. Deployment as a Web Service:

The trained XGBoost model is deployed as a scalable web service using Amazon SageMaker, allowing seamless integration with existing systems and applications.

An API endpoint is created to handle incoming URL requests, classify them using the deployed model, and return the predicted labels in real-time.

Through the systematic execution of these steps, the project aims to develop a robust and scalable solution for the automated detection and classification of malicious URLs, thereby enhancing cybersecurity measures and mitigating potential risks associated with cyber threats.

Execution Steps:

1. Model Development and Training:

- Set up the development environment within AWS SageMaker, ensuring compatibility with scikit-learn version 1.2.1.
- Prepare the labeled dataset of URL samples, extracting relevant features and assigning binary labels for classification.
- Train an XGBoost binary classifier using the scikit-learn library, optimizing hyperparameters for improved performance.
- Evaluate the trained model's performance using cross-validation techniques or a separate validation dataset to ensure robustness.
- Serialize the trained XGBoost model into a file using joblib for efficient storage and future deployment.

2. Deployment of the Model as a Cloud API:

- Log in to the Amazon SageMaker console and navigate to the Model section.
- Create a new model by uploading the serialized joblib file containing the trained XGBoost model.
- Configure deployment settings, specifying the instance type and number of instances required for hosting the model.
- Deploy the model by creating an endpoint, which provides a scalable API for real-time predictions.
- Perform endpoint testing to validate the functionality and ensure accurate predictions.

3. Development of Client Application:

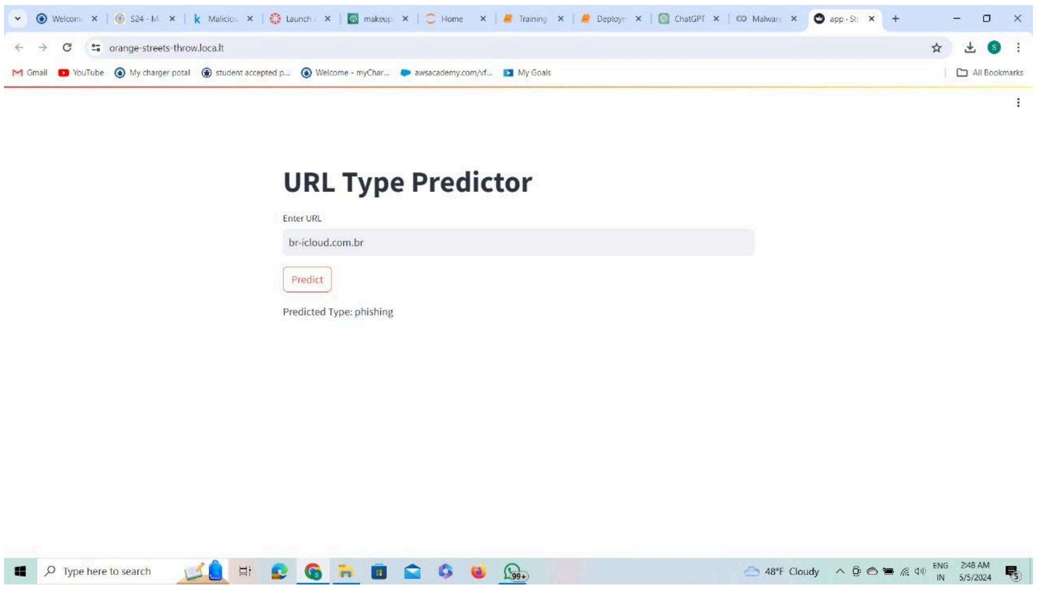
- Install Streamlit and other necessary dependencies to develop the client application within the AWS environment.
- Design a user-friendly interface using Streamlit for users to interact with the system.
- Implement functionality allowing users to input URLs for classification and visualization of results.
- Utilize the deployed model's API endpoint to send URL data for prediction and receive classification results.

- Present the classification results to users within the client application interface for easy interpretation.

4. Validation:

- Select a diverse set of URL samples, including both known malicious and benign URLs, for validation purposes.
- Develop a validation script or pipeline to submit URLs to the deployed API endpoint for classification.
- Capture the classification results returned by the endpoint for each URL and compare them against ground truth labels.
- Compute evaluation metrics such as accuracy, precision, recall, and F1 score to assess the model's performance.
- Iterate on model improvements based on validation results and refine the system to enhance classification accuracy and reliability.

Output :



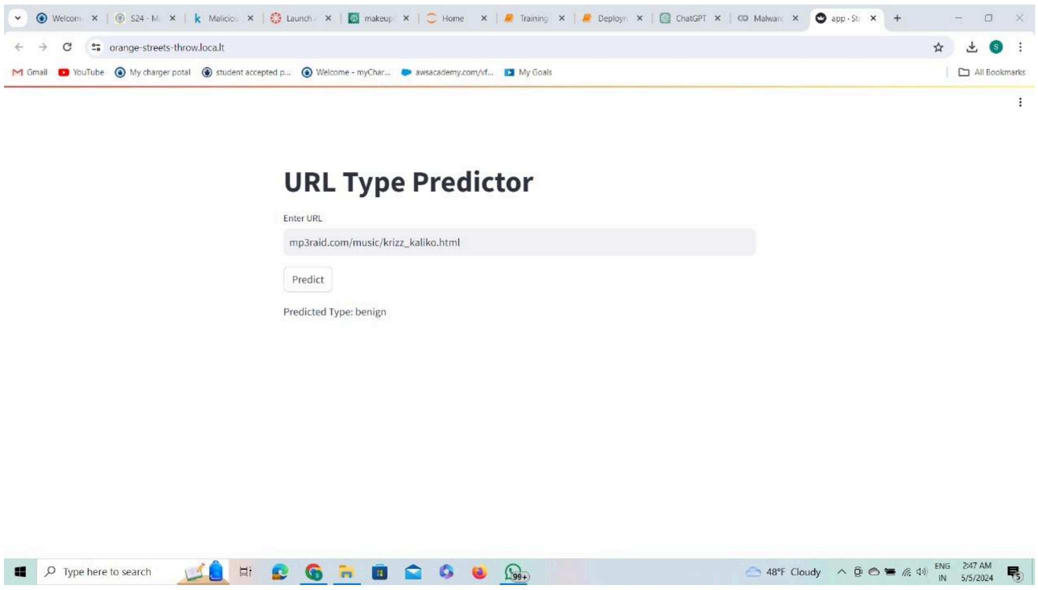
URL Type Predictor

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