Hi this is samohitha

Now I am going to explain about my midterm project that is

Cloud-based PE Malware Detection API

In this project we must complete three tasks

Task 1 is

Deploying the Model on AWS SageMaker: This involves setting up the model we trained in Lab 5.4 on a cloud service called AWS SageMaker.

Task 2 is

Developing a Python Client: We create a Python program. This program takes an executable file as input, extracts relevant features from it, and sends this information to the SageMaker endpoint. It then receives the classification results from the endpoint

Task -3

After that we Test our client and endpoint with one malware PE file and one benign PE file from the test dataset (created during Lab 5.4) and demonstrate it in your demo video.

Now i will explain about how i implemented this project using AWS sage Maker

First, we must create one notebook instance in amazon sage Maker

Building A static malware detection file explanation

after the completion of instance, we must run the trained model file in this first we are downloading all the required libraries to run this trained model and this is completely

trained model and we are going to extract extract four data files from this code, the first one is

1)k one most common engrams list and the other three are going to be

2)trained model.

And the other two are are the other two are imports_featurizer and a section named

these four files we are going to use for the deployment and client side

Deployment code

we have developed a inference.py script this is going to be used utilized by the sage maker to process the input we gave to the API

model_fn(model_dir):

This function deserializes the fitted model. It loads the model from the specified directory (model_dir) using joblib and returns the loaded model.

input_fn(request_body, request_content_type):

This function processes the input data sent to the model. If the content type of the request is JSON (application/json), it parses the JSON request body and returns the input data. Otherwise, it raises a ValueError indicating that the model only supports JSON input.

predict_fn(input_data, model):

This function makes predictions using the input data and the loaded model. It first processes the input data using the process input function, then uses the model to make predictions based on the processed data. The predictions are returned.

process_input(input_data):

This function preprocesses the input data before passing it to the model for prediction. It extracts features from the input data, transforms text features using featurizers loaded from files (imports_featurizer.pkl and section_names_featurizer.pkl), and concatenates all features into a single sparse matrix. The processed data is returned.

output_fn(prediction, content_type):

This function formats the prediction results before sending it back as the response. It takes the prediction result (which is typically a numerical value), converts it to an integer, creates a JSON response containing the output, and returns it.

we are mentioning the requirements required libraries here

After that, this code initializes connections to AWS SageMaker services, S3, and defines an IAM role for accessing resources. here we must give ARN Number from the created notebook instance

This code retrieves the Amazon SageMaker URI (Uniform Resource Identifier) for a specific version of the scikit-learn framework image. It specifies the framework, version, Python version, and instance type.

This code Imports the previously zipped model from S3 Bucket

This code creates a SageMaker model with specified configurations including the model's name, image URI, model artifacts URL, and execution role.

This code sets up an endpoint configuration for deploying our SageMaker model. It's like preparing the environment where our model will be hosted and made accessible. We give it a name and specify details like which model to use, what type of computing instance to run it on, and how many instances to start with. This configuration ensures that our model is deployed properly and ready to serve predictions.

Creating Endpoint on Sage maker

This code continuously checks the status of the endpoint creation process until it's completed. It retrieves the status of the endpoint and prints it, waiting for 15 seconds between each check. Once the endpoint status changes from "Creating", it stops and prints the final status along with other endpoint details.

After the endpoint is successfully created, we are checking here the end point is working or not.

This code sends input data to a SageMaker endpoint named 'sklearn-local-ep2024-04-04-22-38-27', expecting a JSON response, and prints the result.

Client file explanation

Now, let us configure the PE client. For the client, we will utilize the following libraries:

After installing the necessary libraries, we will retrieve an IP address from Google Colab. This IP address will serve as our endpoint for invoking the script on the client side. We will be processing executable files (exe) as input and sending them to the API. This client-side script will be the backbone of our web application.

To streamline the process, we are leveraging a pre-trained data set for efficiency. This ensures that we do not need to train the model each time we use it, saving time and resources.

Next, let us outline the Python script for invocation. With this script, we can interact with the API, passing our processed exe files for classification.

Finally, we will initiate a web application using Streamlit, providing an interactive interface for users to interact with the model effortlessly.