

AI&Cybersecurity

Midterm Project Report

DSCI6672 – spring 2020

-Afshar Mohammed

**Overview and Requirements**

This Document is all about the Midterm project of Artificial Intelligence and Cybersecurity. The requirement of this project is 1. to train a deep neural network, 2.Deploy the model through Amazon Sage Maker 3.creating a python script that takes in a PE file as a parameter and classify in to two classes such as Benign or Malicious. To implement this project it is comprised of three subtasks on the complications and approach to creating a solution.

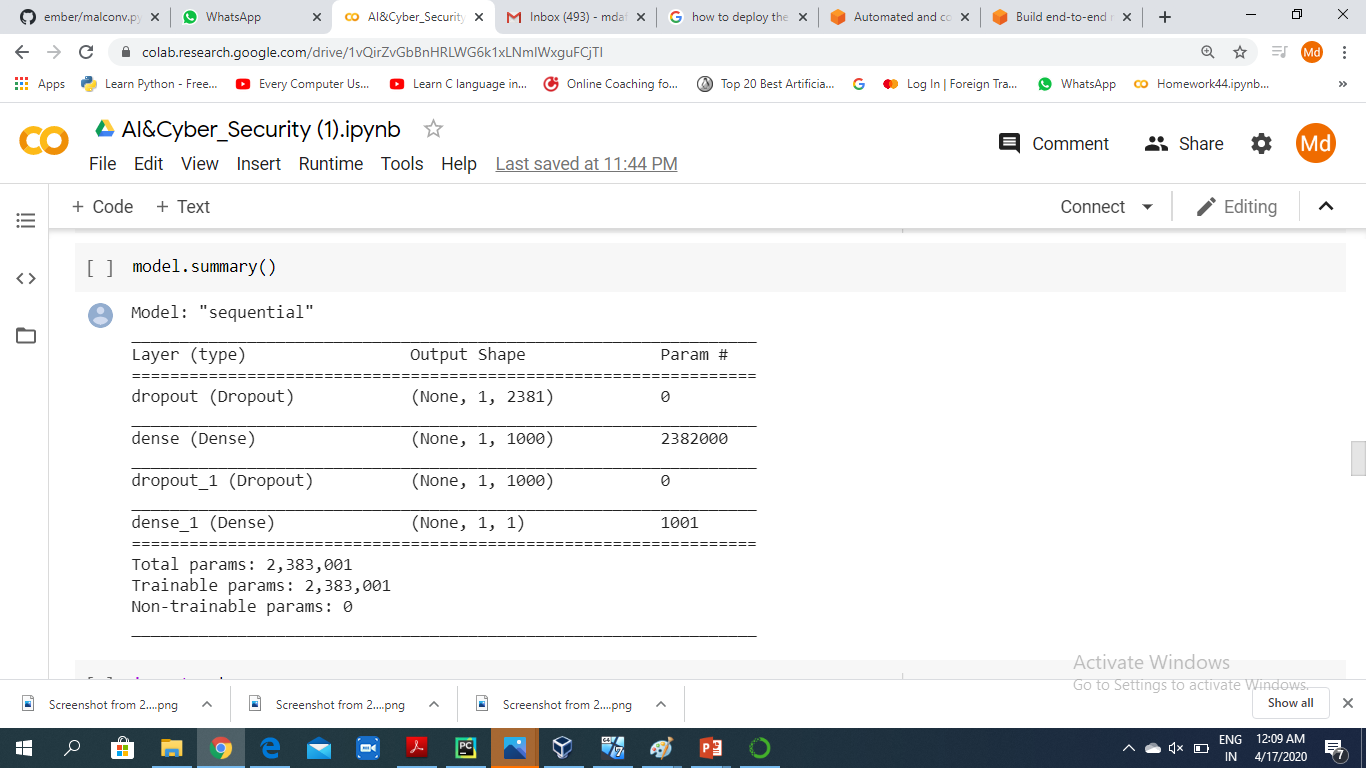
**Task 1: Training**

This is the initial task in implementation of this project. Initially, I was trying to load the dataset on my Local Machine. However, there was a storage issue then I decided to move on to Google Collaboratory service which led to a major constraint such as: Overloading RAM that caused session crashes.

Secondly, the note book was created and loaded the dataset on collaboratory while exploring the dataset which was published by arxiv paper. I got to know that it consists of Meta data, Started, visualizing the data, Converted the data in to vectorized format it took me around 5-7 hours in preprocessing the data

After the data was cleaned, as we know that in Machine learning. We usually split the data in to two sets one is training and other

is testing. So eventually, I normalized the data by using standard scaler technique, this was imported from Sk-Learn library when the data was fitted to scaler instance then I transform the data by using .fit\_ transform method. Finally, data was ready to be given to a neural network.



The above picture is all about the Model Summary for the neural network In this I used Input layer and a dense layers So I passed (1,feature\_size) as parameter for the input layer whereas, Dense layers add an interesting non-linearity property, thus they can model any mathematical function. for the dense layer I also used the drop out layer,  dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random.Eventually, after evaluating the model the results were accuracy: 0.9646, val\_accuracy: 0.9303, f1\_score:0.9367284, precision: 0.963842 respectively.

**Task 2: Deploy of Model on Cloud**

The service used to deploy the model was Amazon’s Web Service’s sage Maker. AWS supports Tensor flow 2.0 for training and deploying Tensor Flow Serving models from its Estimators; however, to import an outside trained model, TensorFlowModel method is used to convert a Tensor Flow Model into an estimator... I decided to deploy a model through Sage Maker’s Endpoint, it would be wise to assure that tensor Flow version is compatible with the TensorFlowModel method. In the tutorial resource on the deployment of a model through sage Maker, tensor Flow 1.12 is compatible.

After getting the model to successfully import, the rest of the tutorial provided was sufficient. Depending the Amazon Web Service account; for example, an AWS educate account stores their keys and password before launching the AWS console; however, cannot be found elsewhere. A normal user AWS account can create their passwords and keys through the Identity Access Management.

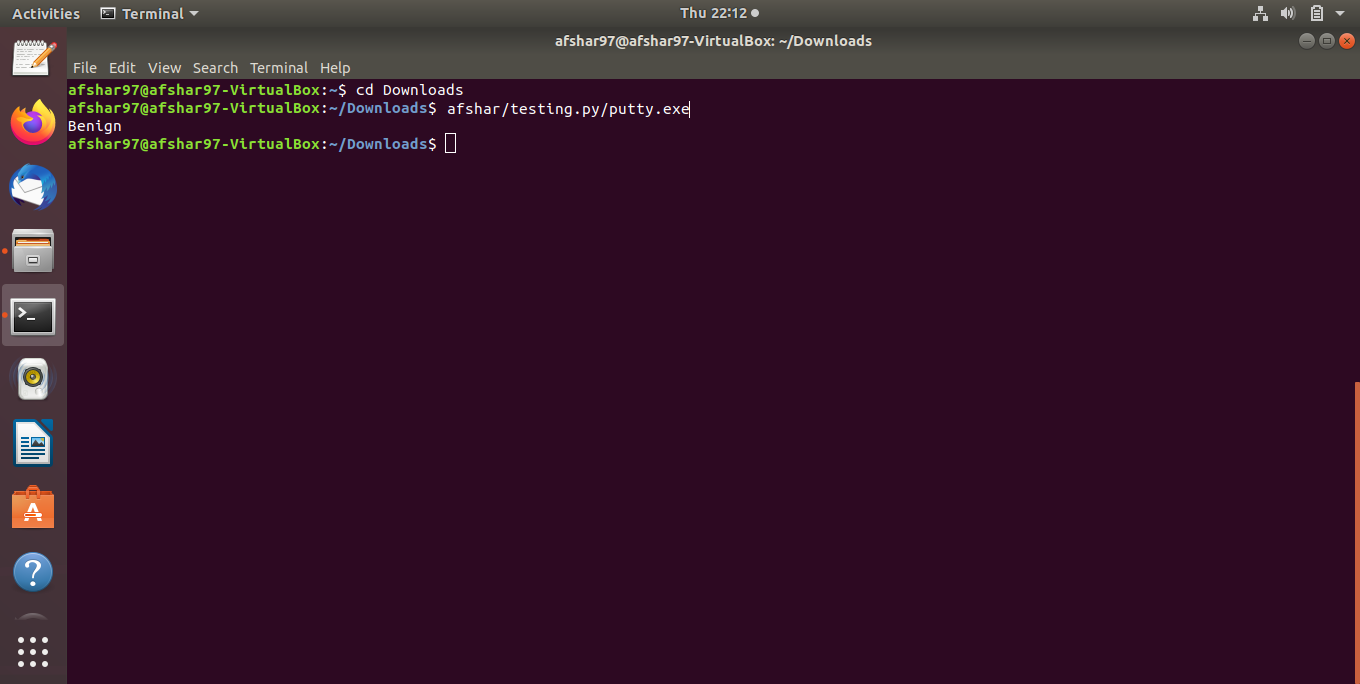
Initially, I created the Amazon S3 Bucket, and then notebook instance was done lastly, uploaded my jupyter notebook from google Collabaratory.The creation of the endpoint took around 51/2minutes. While invoking the endpoint from within the sage Maker notebook, response time was optimal

**Task 3: Create a Client**

In this final task I launched a virtual machine through Virtual Box, specifically the distribution Ubuntu 18.04 to create an environment to write the code. This was an optimal testing environment for packages. Only one necessary file was bundled and that was the pickle scaler file. This file should be hosted on the cloud as well with a Lambda function set to scale incoming data but said service was restricted. The python script was written through the text editor . The python script installs EMBER if it is not installed. **Putty.exe** was used as the deployment sample and returned benign.

The code included for the client side has the sensitive keys and passwords removed. The skeleton of the code was provided by the EMBER GitHub repository and adjusted for this model. The response time for invoking the endpoint and returning data was around 4 seconds. That is incredibly long for Malware detection and therefore should not be recommended for general use.

The picture below is the screen shot of the client response which I had done on the virtual machine terminal



This encourages the concept of having light models distributed on the client for more optimal speeds.

**Conclusion**

This project was comprises some of the complications in implementing it.In this, I got good experience in handling such big Datasets in terms of training and designing the model architecture to pass through the deep neural network in addition to that, learned what other services are provided by the Amazon Web Services like AWS Sagemaker,EC-2,S3 Bucket and also some machine learning services even I learned how to deploy a model on Cloud using Aws.while I was doing it, explored how do we create end point. In terms of creating client followed the steps how to install Virtual machine on our local machine using virtual box .loaded Portable executable file like putty.exe on Virtual machine eventually, I classified this file as Benign.

.