**Assignment 4**

**Assess the architecture of an AI-based system**

**The objective of this assignment**

1. Understand the trade-off between speed and accuracy of the ML model.
2. Understand the challenges of deploying real-time AI systems.

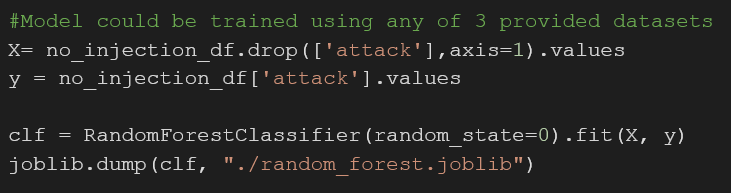
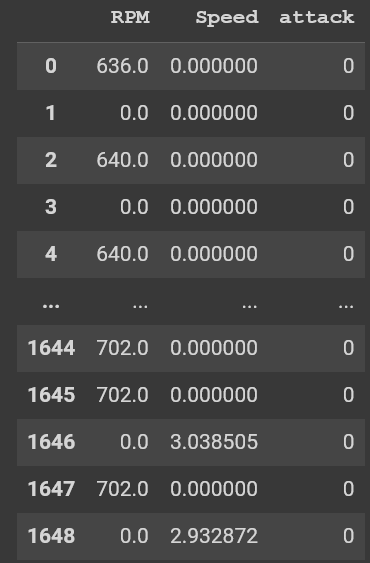
**Things need for this assignment**

1. Virtual application file
2. Virtual machine manager (preferably Oracle VM VirtualBox. It is assumed that you already downloaded and installed Oracle VM VirtualBox. If not, please download and install Oracle VM VirtualBox from <https://www.virtualbox.org/wiki/DownloadsLinks> to an external site..

**Step 1:** Train and create the ML object

First, train a supervised ML model that takes input data frame of three columns RPM, Speed and Attack, and index as shown on the right figure–as in your assignment 1.

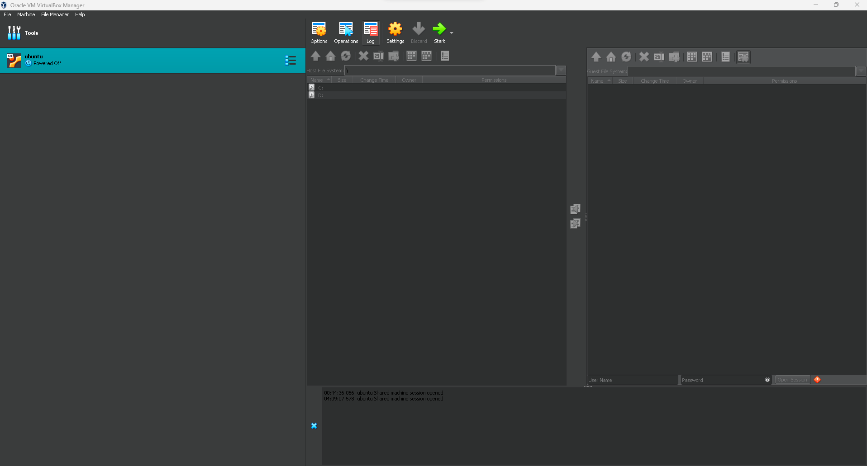
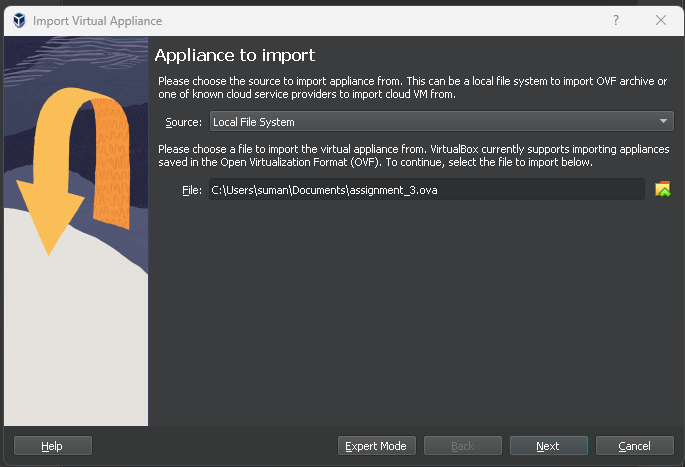
Second, use Joblib python library to save the trained ML model for future use, as in the code snippet below.

Note: For easier file transfer to the virtual machine. You can upload the files to a cloud driver (google drive, one drive, etc.) and then download them to the virtual machine.

**Step 2: Loading .ova file into Virtual Machine**

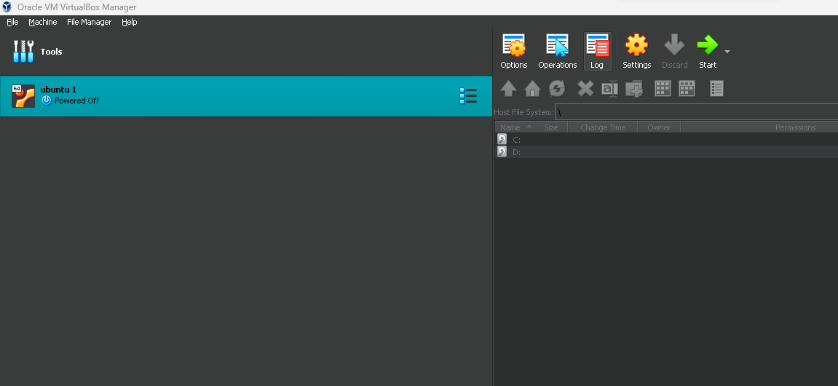
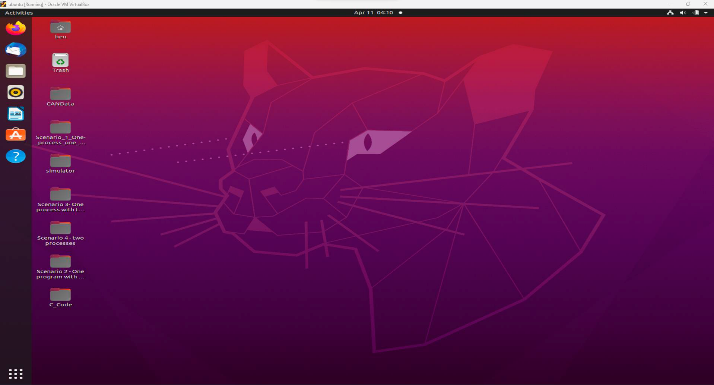
Download file assignment3.ova and load it to your VM manager by clicking File -> Import Appliance ->

Importing the ova file might take a long time. You must wait for the import to be complete before opening the VM.

Once the ova file is fully loaded, click the Start button to start the assignment VM. You will see following home page once the VM is running.

Note: The VM uses Ubuntu operating system

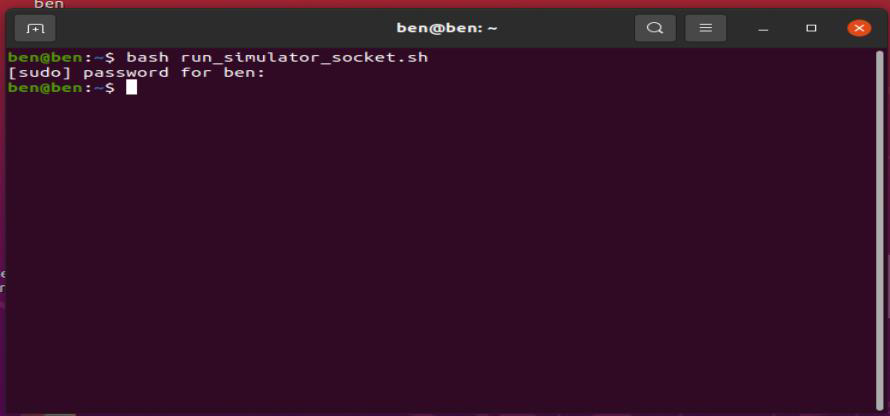
 

**Step 3: Setting up the vcanbus interface**

First, open a terminal. Then, run the script: bash run \_simulator\_socket.sh as in the figure below.

This will initialize the CAN bus ECU simulator socket.

When asked about the password, enter the following: MESO@123

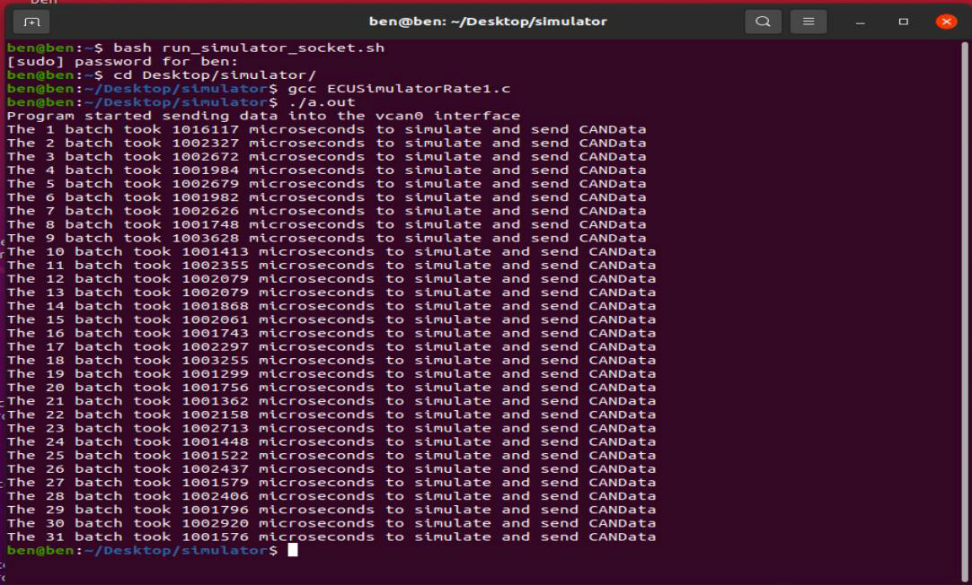


**Step 4: Setting up the ECUSimulator**

Now, change the current directory to the folder containing the simulation script using the command: cd Desktop/simulator

Compile simulation script using the following command: gcc ECUSimulatorRate1.c

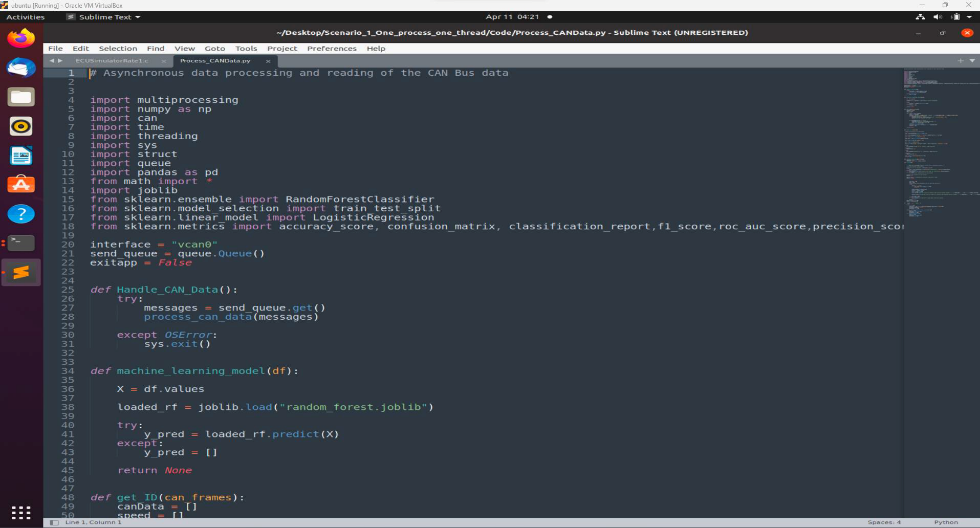
Run the following command to check if the simulator Works: ./a.out. The output should be similar to the one on the screenshot.



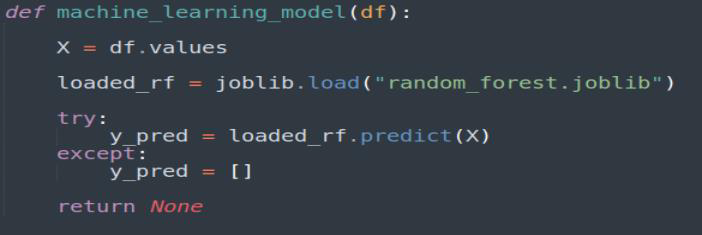
**Step 5: Python script to receive and process simulated CANData**

Without closing the simulation terminal, open a new terminal Window and enter the following command to change the current directory: cd Desktop/Scenario\_1\_One\_process\_one\_thread/Code/

Then, edit the file Process\_CANData.py. To a chive that, execute the following command: subl Process\_CANData.py which will open the file



Now, change the file to add your own trained ML model. Change the joblib load file to the previously saved ML object (from step 1). Then, save the edited script

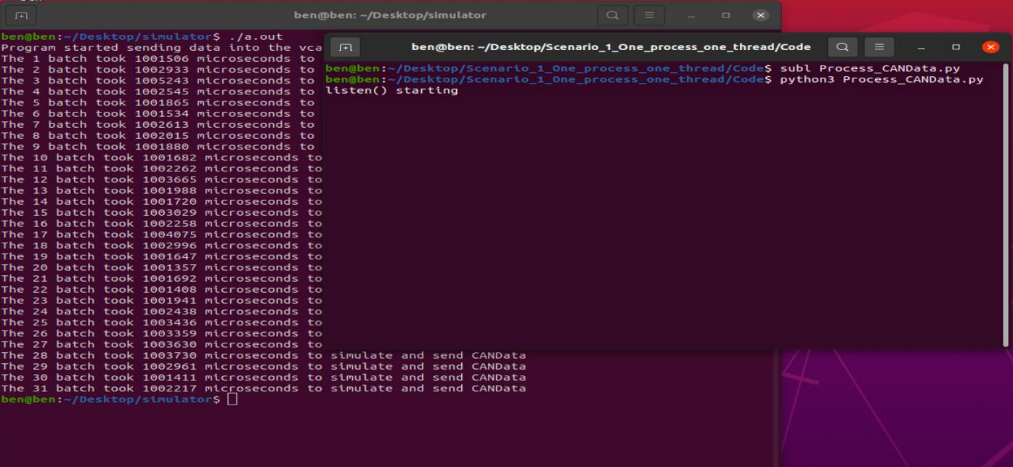


Now, run the edited Python script using the following command: python3 Process\_CANData.py.

A successful compilation of the Python script will generate a message, similar to the one I the figure on the right. Here, the Python script is running and listening for any CAN Data messages from connect socket

While the Python script is still listening for the message, run the following command on the simulator terminal: ./a.out

Now the Python script will receive messages generated by the simulator and start processing them.

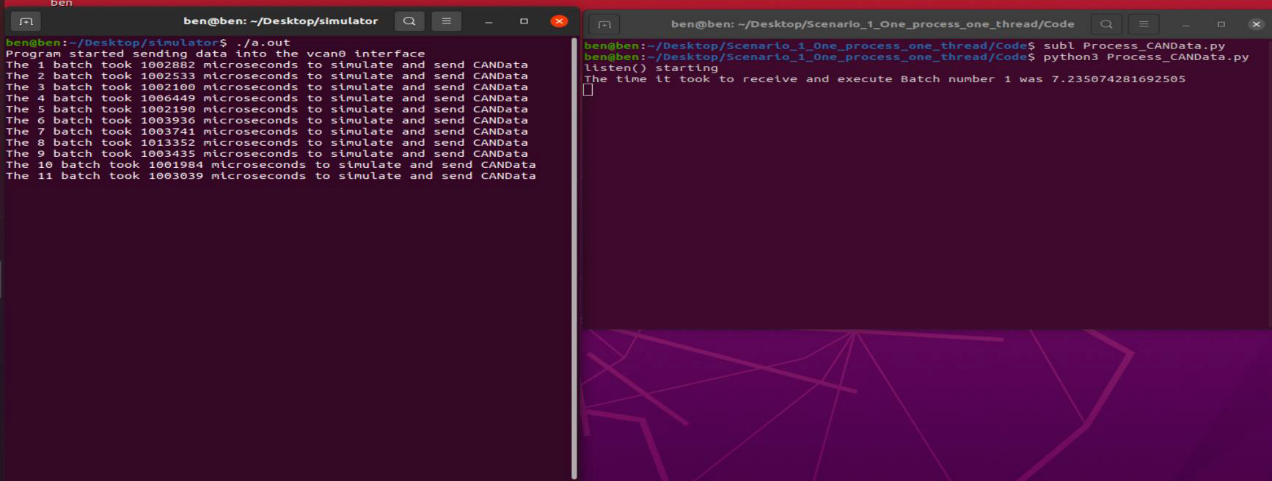


While the Python script is still listening for the message, run the following command on simulator terminal: ./a.out

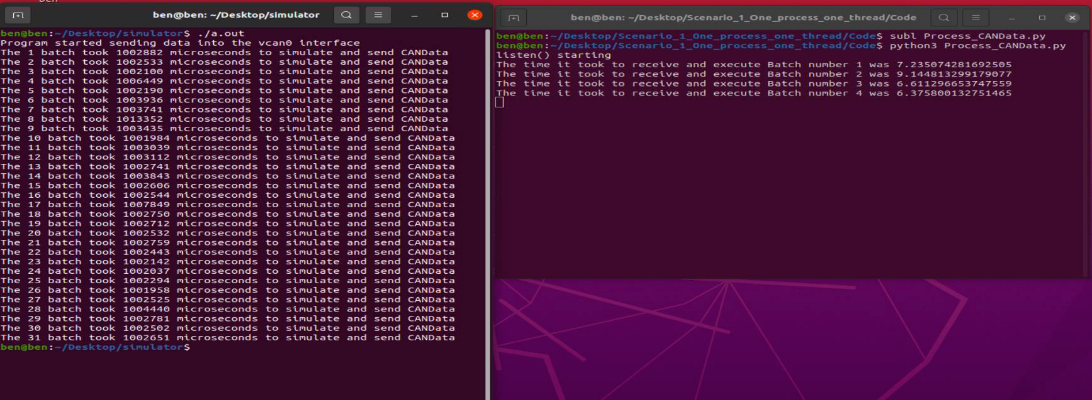
Now the Python script will receive messages generated by simulator and will start processing them.

This is an important step.

Wait for the simulator to send all 31 batches of data packets. When it finishes sending all data packets, go to the terminal that runs Process\_CANData and interrupt the script by

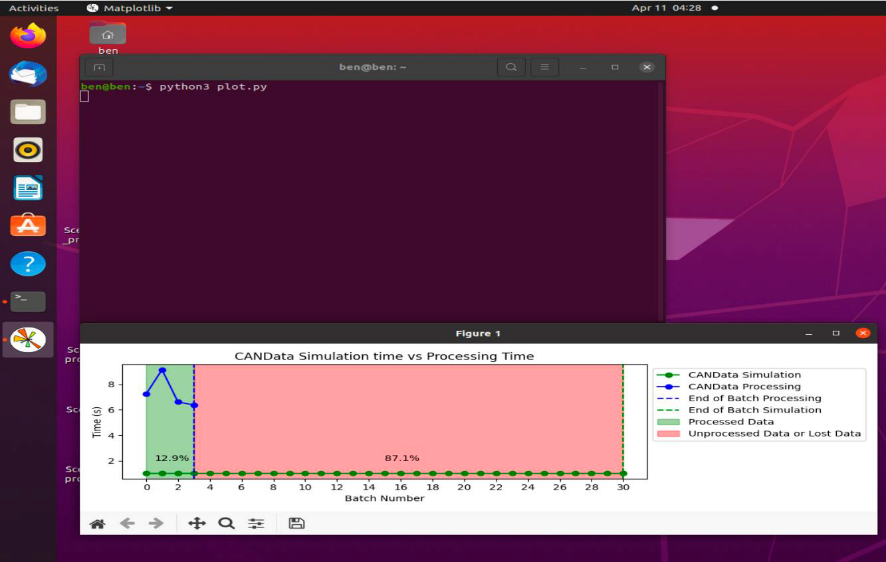


hitting the Control key + C. Process\_CANData has no time-out function and will listen to messages indefinitely unless interrupted..

****

**Step 6: Plotting Data loss and processing time difference plot**

Now you can close all terminals and open a new terminal. Now run the following command: Python3 plot.py. You will receive different plots depending on your ML model’s computational time and memory use.

****

**Submission:**

Submit the trained ML object file that you generated in Step 1–file with extension joblib and a short report that includes two sections. The model file and report should be in separate files that could be zipped together.

**Section 1 - Generated plot.**

Describe the plot that you received in Step 6.

Your answer should address the question:

1. What are key information that can be derived from the plot?

**Section 2- Reflection**

Reflect on the results you obtained in step 6 in two or three paragraphs. The reflection should address the following questions:

1. Why do you think there is a data loss? Is it significant? Explain!
2. What could be possible solutions to mitigate data loss?