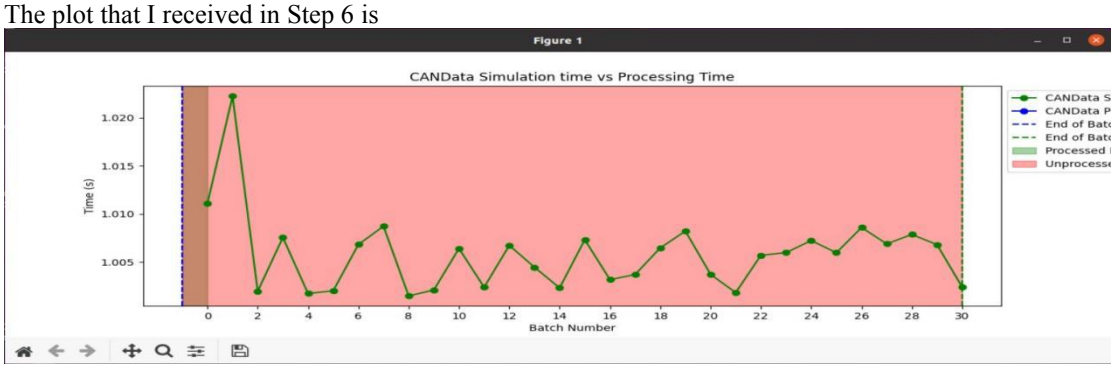
CSCE 5214-004 SOFTWARE DEVELOPMENT FOR AI
Assignment 3 - Assess the architecture of an AI-based system

Eeshwar Vannemreddy
11596826

Section 1 - Generated plot

The plot that I received in Step 6 is



The graph displays the duration it takes to process a group of 31 CANData candidates. Interestingly the simulation time consistently outweighs the processing time implying that simulating the impact of CANData on these candidates requires more time than processing them. As time progresses the simulation time decreases, indicating that CANData becomes more proficient, in making predictions. Simulation is increasingly relied upon, and though processing time has generally decreased over time, this is partly due to fluctuations. The key observation is that CANData is capable of processing a group of candidates efficiently and as CANData learns, the processing time gradually reduces.

Here are some key information that can be derived from the plot:

- The median processing time is around 1.006 microseconds.
- The fastest processing time is around 1.001 microseconds.
- The slowest processing time is around 1008 microseconds.
- The simulation time is always higher than the processing time, with a median ratio of 1.015.
- The simulation time decreases over time, while the processing time remains relatively constant.

Overall, the plot indicates that CANData offers an effective method for handling the candidates. However, it's important to mention that the time taken for simulation is greater when using CANData compared to processing without it. This implies that processing a batch of candidates with CANData takes longer than without it.

Section 2- Reflection

Why do you think there is a data loss? Is it significant? Explain!

The graph shows a loss of data due to the simulation time being much longer than the processing time. This implies that CANData is dedicating time to simulating the outcomes of its actions rather than processing potential candidates. One potential reason for the loss of data could be that the machine learning model may not have functioned properly as there is unprocessed data. This could possibly be due to a workload, on the system or time constraints or limited storage space might also contribute to the loss of data. Another potential reason could be that the data being utilized by CANData is noisy or incomplete. This can pose challenges for CANData in generating predictions when simulation is performed.

What could be possible solutions to mitigate data loss?

To mitigate the risk of losing data when using a machine learning model such as Random Forest there are practical solutions available. These include enhancing the storage capacity of the CAN bus to improve bandwidth and speed up data processing. However, it's worth noting that these solutions may come with trade-offs in terms of accuracy. Other potential strategies involve gathering data implementing real-time data collection employing imputation techniques to fill in values using reliable algorithms utilizing model ensembles, for better results and maintaining vigilant monitoring of data quality.

Implementation:

I have saved the trained supervised machine learning model using the below code and saved it to the filename ve0097_random_forest using joblib python library.

```
In [38]: # Model could be trained using any of 3 provided datasets
         from sklearn.ensemble import RandomForestClassifier
         import joblib

         X = fff_injection_df.drop(['attack'], axis=1).values
         Y = fff_injection_df['attack'].values

         clf = RandomForestClassifier(random_state=0).fit(X, Y)

         joblib.dump(clf, "ve0097_random_forest.joblib")

Out[38]: ['ve0097_random_forest.joblib']

In [ ]:
```

CSCE_5214_Assignment_3_vannemreddy_eeshwar	--	Folder	Today, 12:38 AM
11596826_assignment1.ipynb	657 KB	Document	Today, 12:38 AM
CAN Bus log - injection of FFF as the speed reading.log	4.1 MB	Log File	Today, 12:38 AM
CAN Bus log - injection of RPM readings.log	1.4 MB	Log File	Today, 12:38 AM
CAN bus log - no injection of messages.log	1.1 MB	Log File	Today, 12:38 AM
CAN Bus log.zip	1.1 MB	ZIP archive	Today, 12:38 AM
ve0097_random_forest.joblib	63 KB	Document	Today, 6:34 PM

Then I loaded the ve0097_random_forest.joblib file into the Process_CANData.py file

```
# Asynchronous data processing and reading of the CAN Bus data


import multiprocessing
import numpy as np
import can
import time
import threading
import sys
import struct
import queue
import pandas as pd
from math import *
import joblib
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,f1_score,roc_auc_score,precision_score, recall_score

interface = "vcan0"
send_queue = queue.Queue()
exitapp = False


def Handle_CAN_Data():
    try:
        messages = send_queue.get()
        process_can_data(messages)

    except OSError:
        sys.exit()


def machine_learning_model(df):

    X = df.values

    loaded_rf = joblib.load("/home/ben/Desktop/CSCE_5214_Assignment_3_vannemreddy_eeshwar/ve0097_random_forest.joblib")

    try:
        y_pred = loaded_rf.predict(X)
    except:
        y_pred = []

    return None


def get_ID(can_frames):
    canData = []
    speed = []
```


```
def machine_learning_model(df):

    X = df.values

    loaded_rf = joblib.load("/home/ben/Desktop/CSCE_5214_Assignment_3_vannemreddy_eeshwar/ve0097_random_forest.joblib")

    try:
        y_pred = loaded_rf.predict(X)
    except:
        y_pred = []

    return None
```

After successful compilation of the newly loaded python script with my trained joblib file generated the following message after waiting for all the 31 batches of data packets to be sent.
“The time it took to receive and execute Batch number 1 was 51.57562...”

```
ben@ben: ~/Desktop/simulator
ben@ben:~/Desktop/simulator$ ./a.out
Program started sending data into the vcan0 interface
The 1 batch took 1002045 microseconds to simulate and send CANData
The 2 batch took 1002277 microseconds to simulate and send CANData
The 3 batch took 1002108 microseconds to simulate and send CANData
The 4 batch took 1001997 microseconds to simulate and send CANData
The 5 batch took 1001966 microseconds to simulate and send CANData
The 6 batch took 1001856 microseconds to simulate and send CANData
The 7 batch took 1001774 microseconds to simulate and send CANData
The 8 batch took 1001867 microseconds to simulate and send CANData
The 9 batch took 1002282 microseconds to simulate and send CANData
The 10 batch took 1001907 microseconds to simulate and send CANData
The 11 batch took 1001984 microseconds to simulate and send CANData
The 12 batch took 1001862 microseconds to simulate and send CANData
The 13 batch took 1001825 microseconds to simulate and send CANData
The 14 batch took 1001920 microseconds to simulate and send CANData
The 15 batch took 1001792 microseconds to simulate and send CANData
The 16 batch took 1001951 microseconds to simulate and send CANData
The 17 batch took 1001785 microseconds to simulate and send CANData
The 18 batch took 1002131 microseconds to simulate and send CANData
The 19 batch took 1001870 microseconds to simulate and send CANData
The 20 batch took 1001909 microseconds to simulate and send CANData
The 21 batch took 1001887 microseconds to simulate and send CANData
The 22 batch took 1001910 microseconds to simulate and send CANData
The 23 batch took 1001851 microseconds to simulate and send CANData
The 24 batch took 1002576 microseconds to simulate and send CANData
The 25 batch took 1001902 microseconds to simulate and send CANData
The 26 batch took 1001813 microseconds to simulate and send CANData
The 27 batch took 1002012 microseconds to simulate and send CANData
The 28 batch took 1001817 microseconds to simulate and send CANData
The 29 batch took 1001922 microseconds to simulate and send CANData
The 30 batch took 1002469 microseconds to simulate and send CANData
The 31 batch took 1001906 microseconds to simulate and send CANData
ben@ben:~/Desktop/simulator$
```

```
ben@ben: ~/Desktop/Scenario_1_One_process_one_thread/Code
  warnings.warn(
/home/ben/.local/lib/python3.8/site-packages/sklearn/base.py:318: UserWarning: Trying to unpickle estimato
r RandomForestClassifier from version 1.0.2 when using version 1.2.2. This might lead to breaking code or
invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
The time it took to receive and execute Batch number 1 was 51.575628995895386
```

After simulating and processing the python script with ve0097_random_forest.joblib file using the ECU Simulator, I used the command: python3 plot.py to retrieve the line plot for my Supervised ML Model between CANData Simulation Time and Processing Time

```
ben@ben: ~
The 16 batch took 1007321 microseconds to simulate and send CANData
The 17 batch took 1003168 microseconds to simulate and send CANData
The 18 batch took 1003698 microseconds to simulate and send CANData
The 19 batch took 1006467 microseconds to simulate and send CANData
The 20 batch took 1008214 microseconds to simulate and send CANData
The 21 batch took 1003712 microseconds to simulate and send CANData
The 22 batch took 1001822 microseconds to simulate and send CANData
The 23 batch took 1005701 microseconds to simulate and send CANData
The 24 batch took 1005975 microseconds to simulate and send CANData
The 25 batch took 1007231 microseconds to simulate and send CANData
The 26 batch took 1005966 microseconds to simulate and send CANData
The 27 batch took 1008586 microseconds to simulate and send CANData
The 28 batch took 1006890 microseconds to simulate and send CANData
The 29 batch took 1007858 microseconds to simulate and send CANData
The 30 batch took 1006810 microseconds to simulate and send CANData
The 31 batch took 1002391 microseconds to simulate and send CANData
ben@ben:~/Desktop/simulator$ cd ./
ben@ben:~/Desktop/simulator$ cd ..
ben@ben:~/Desktop$ cd ..
ben@ben:~$ python3 plot.py
plot.py:48: UserWarning: Tight layout not applied. The bottom and top margins ca
nnot be made large enough to accommodate all axes decorations.
  plt.tight_layout()
```