**INSTRUCTIONS FOR RUNNING THE SOURCE CODE:**

🡺**Install pandas, numpy, scipy, and pynput**

🡪pip install pandas

🡪pip install numpy

🡪pip install scipy

🡪pip install pynput

🡺**Keep all the code files and data file in the same folder**

🡺**Open command prompt and run the code by the following commands**:

🡪**python anomaly\_detector.py**

In the anomaly\_detector.py code, we are taking first 10 samples for every user which adds up to 510 and taking the 11th user sample as the test data which adds up to 51. The 10 login samples we have taken are the training data and the 1 data from each user set is the test data.

We are then calculating the mean vector and the standard deviation for our distance algorithms. Here we have used Manhattan filtered distance here to scale down the score values to 0-1. We are then using the test data to login to the trained model and checking if the score is above 0.85. If the score is above 0.85, it is a match and if it is below 0.85 it is a poor match and if the score is exactly 1, it is a perfect match. The output should look like this:

![A screenshot of a cell phone

Description automatically generated]()![A close up of a white background

Description automatically generated]()

**🡪python user\_classification.py**

In the user\_classification.py code, we are calculating the user score out of 50(each user is trying to imitate the other 50 users present in the subject set) and then we divide the score by 50 and get the value between 0-1.

According to our understanding we have categorized the users as follows:

If score>0.8 🡺 wolves

If score<0.8 and >0.6 🡺 lamb

If score<0.6 and >0.45 🡺 goat

And the remaining are sheep.

The output should look something like this:

![A close up of a logo

Description automatically generated]()

**🡪python intelligent\_keystroke.py**

In our intelligent\_keystroke.py code, we tried to implement an intelligent system where we ourselves will be able to enter the data and the timings will be noted for the same. We are taking 2 samples for each user. Then we tried to do a single point crossover with the subject data user in the anomaly\_detector.py code.

The output should look something like this: A close up of text on a black background

Description automatically generated

![A screenshot of a cell phone

Description automatically generated]()