**Sleep Tracking using Tri-Axial Accelerometer Sensor**

**Background**

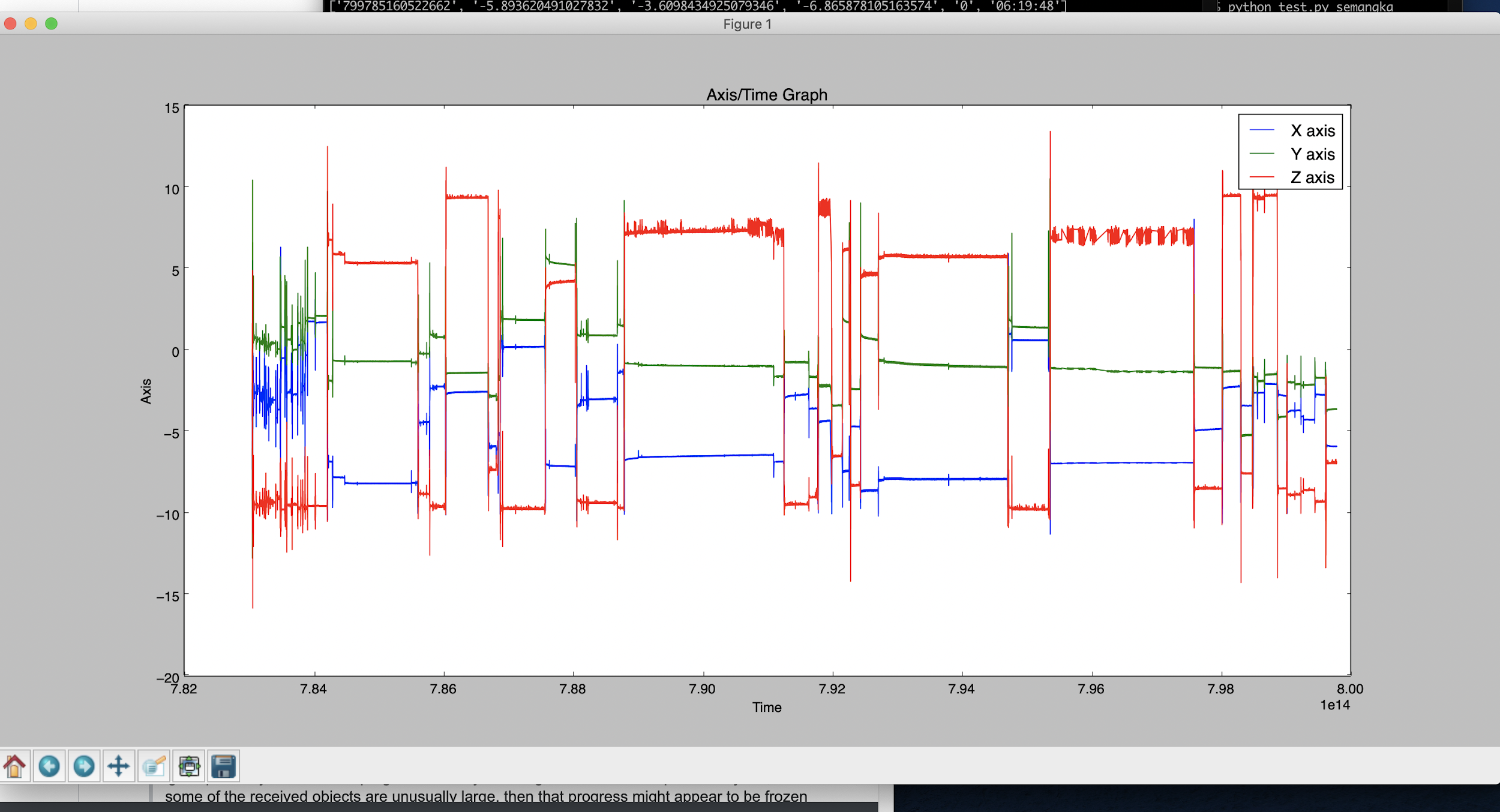
We wanted to build a live sleeping detector that can be used for the general public. This would involve the user putting their phone in their pocket while they sleep. The raw accelerometer data would be analysed using a Python script that has been trained using a Random Forest decision tree, using both simulated and real-life sleep data.

**Process**

1. **Data Collection**

We used two methods of collecting data. The first method is by collecting real-time sleep data. We do this by putting our phones in our pockets and going to bed as normal. We composed a rough sleeping log in which we tried to write down at what time we would go to bed and when we would wake up.

We wanted to strictly use real-sleep data because we feel it would result in a better activity recognition. However, there are two main problems that came up during this process of collecting data. The first one is the phone stops collecting data after 4 hours and this could be seen when the final timestamp of the generated CSV file was hours before we actually woke up, making us lose half or more than half of our collected data because of this. Another problem that we encountered is that it becomes increasingly difficult to know the exact time we fall asleep. We knew it wouldn’t be precise but more often than not we just end up guessing to estimate the time when we were tossing and turning and when we were falling asleep. We did this process of data collection for 4 or 5 nights before moving on to another method, because of the problems that would require a lot of work to solve.



One of the real-time sleep data that we managed to extract.

We then came up with a solution, we figured it would save a lot of time and be easier if we just simulated the activities. We then performed two different activities, sleep and vigil. Simulating sleep was easier, as you can just lie down and make minimal movement. Simulating a vigilant state however, is more complex. Simply moving a lot would not work as that is not how people sleep, so you had to gauge how much movement we should make that would a) be realistic and b) would be different enough to the sleep data, so our machine can differentiate between the two.

The problem with this approach is obviously the fact that we are not getting ‘real’ data, but we figured it was close enough to the real data and the time it saved was very beneficial for us.

|  |  |
| --- | --- |
|  |  |
| Simulated sleep data | Simulated vigilant data |

1. **Feature Extraction**

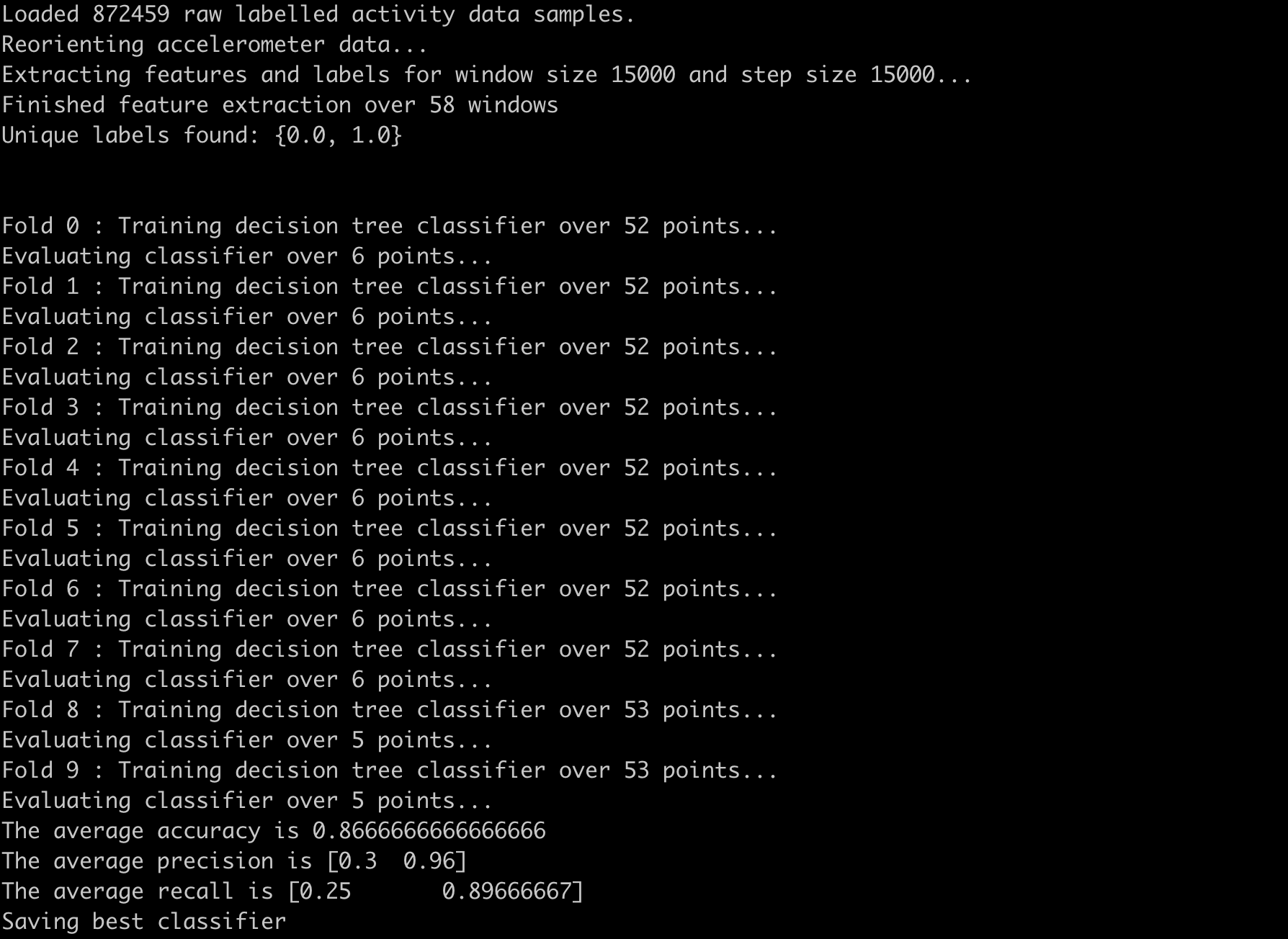
These are the features we extracted are the following:

1. Mean
2. Standard Deviation
3. Peak Length
4. Magnitude
5. Entropy

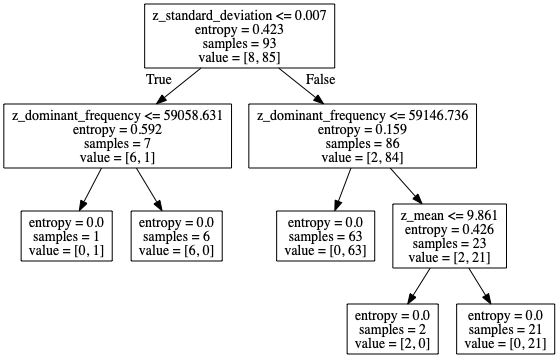
It is pretty clear why we would use these features since we are detecting the difference it accelerometer movement.

**3. Classification**

We created a random forrest decision tree using a Stratified KFold cross validation acrros 10 folds in the data. First we split up the data into 5 minutes intervals and randomly split them up into folds of 10 using the stratidiefKFold method from scikit-learn. We then used a Random Forrest classifier to generate a random forrest decision tree for our model. We found that this approach yields the best result rather than using the model\_selection.KFold method and the regular decision tree that we have used in previous projects.



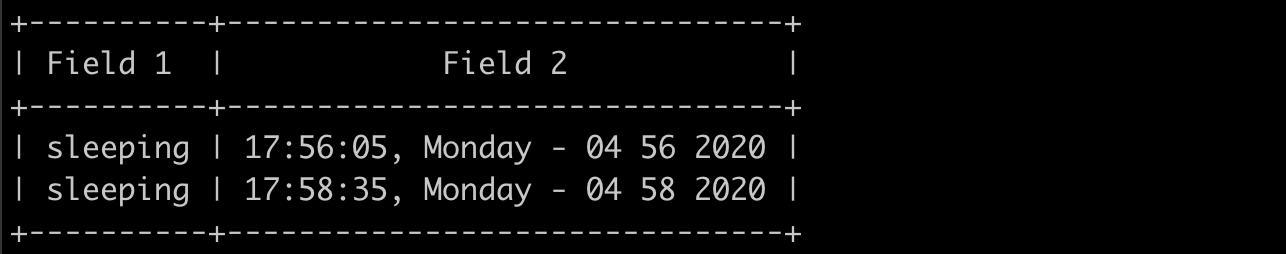
The output when we train the classifier.

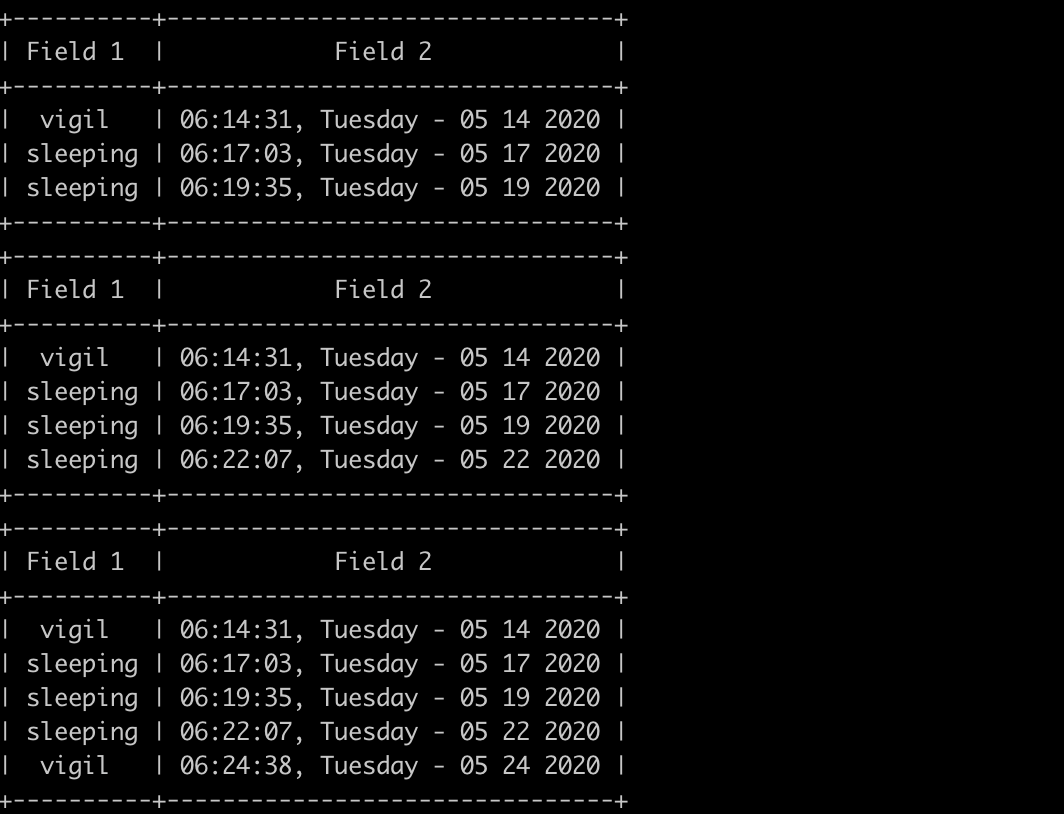


The random forrest tree generated

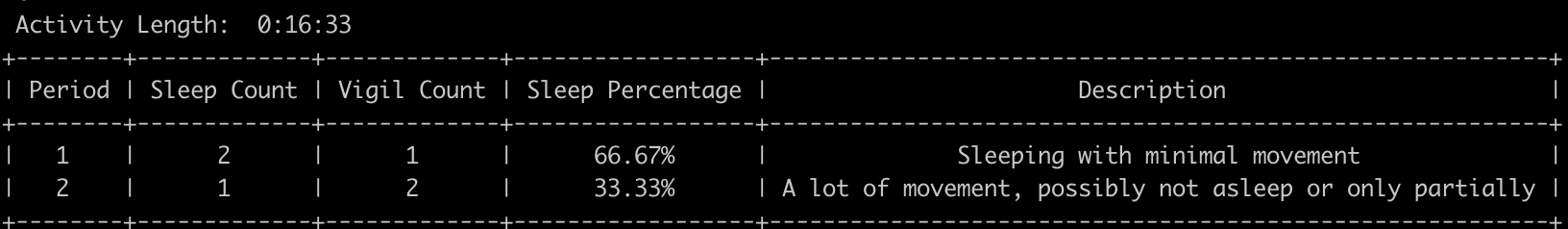
**4. Output**

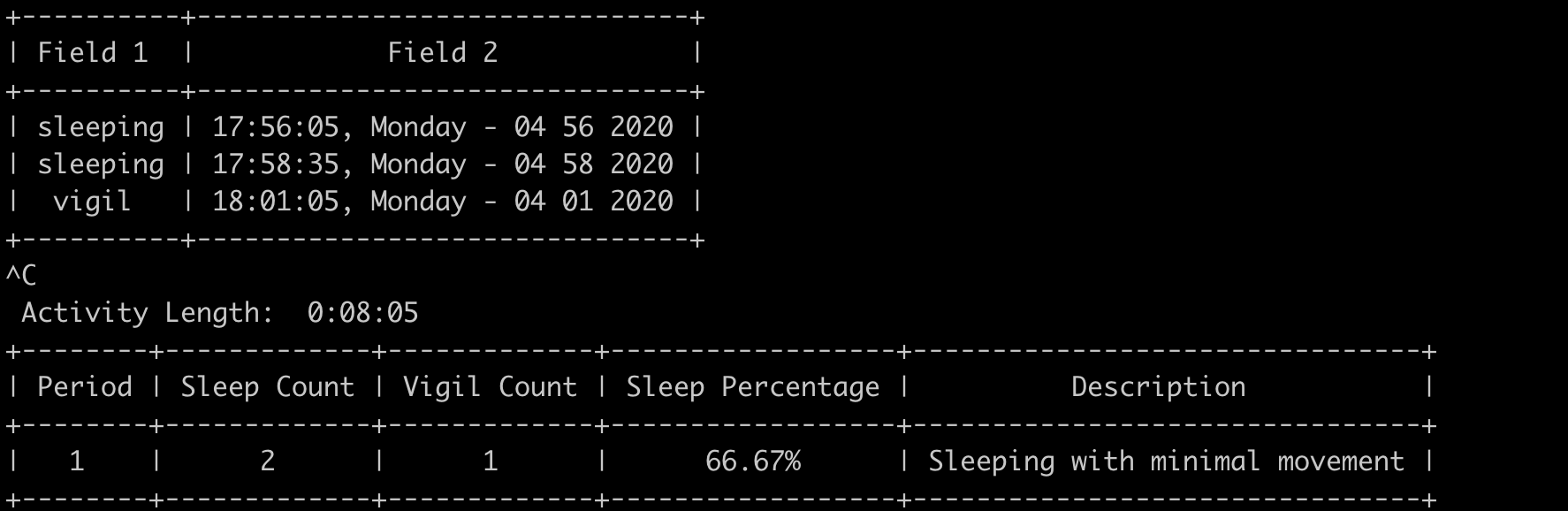
Here are some screenshots of the output that will be shown in the Terminal as the user is sleeping:





Here are a few examples of the output when the script is killed:





**5. Project Difficulties**

* Script would have to run for the whole night to collect sleeping data so the CSV file was very large which meant that it would took a long time to process and graph the data
* Data collection between two people is extremely difficult as a lot of people will have different sleeping postures, etc.
* There were several times where the final timestamp on the CSV file would be before we woke up, meaning the script ended in the middle of the night
* Occasionally, the CSV file itself would not generate after the script was ended, meaning there was no data to work off with for that night
* Since data collection are only done by the two of us (and comes with the problems mentioned above) , there is the case of not having enough data to train