Sensors, Noise and Walking: Human Activity Analysis
Final Project Report

CMPT 353: Computational Data Science
Section D100

Group: *Pydiots*
Sidak Singh Aneja
Diego Buencamino
(Matt) Chang-Hsiu Tsai

Project Scope

Smartphones are becoming an essential aspect of everybody's lives. The capabilities that these devices have are only improving from day to day. Many modern smartphones hold a fantastic array of sensors that record and collect data about them and their user's surroundings. This project explores human activity prediction by using Data Science and Machine Learning tools to analyze and predict a person's current activity. We use our smartphones to record simple human movements, such as walking, running, and climbing stairs, and build a Machine Learning model that accurately predicts these activities. This project is inspired by Guillaume Chevalier's LSTM Human Activity Recognition Project (*https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition*).

Data Gathering

The data in this project was collected on three different smartphones by each member. Two of them were Apple devices: one is an iPhone 13 Pro on iOS 15.1.1, and the other is an iPhone X on iOS 15.0.1. The third device was a Google Pixel 3 XL on Android 12. Cross-platform recording (iOS vs. Android) was not an issue for us as the application we used for all three devices was made by the same developer. Thus the underlying calculations and the resulting noise (if any) were similar across all three devices. The application we used to record our accelerometer data is the "Sensor Logger" app by Kelvin Choi Tsz-hei, which was available on both the iOS App Store (*Sensor Logger on the App Store*) and the Google Play Store (*Apps on Google Play*). The application allowed us to set a sampling rate for the collection; all of our datasets were recorded at 50 Hz. We standardized our data collection by placing our smartphones in our right pockets, with the phones pointing upwards and the screen towards us. This way guarantees that any variation in data collection due to the phone's orientation is significantly minimized. We also stood still for a short period of time at both the beginning and end of the recording in order to simplify cutting out the unnecessary sensor readings of us inserting and removing the phones from our pockets.

Recording with smartphones comes with its issues. Despite these devices having a multitude of sensors built-in, recording from these sensors is not their primary function. Aside from the noise, one should reasonably expect from a cheap smartphone sensor, the battery-saving feature of a phone

could also affect the data being recorded. One of our group members, Matt, faced an issue wherein his device went into a "sleep" state while recording his data. When plotted, his recording produced a graph with regularly occurring plateaus beginning about 25 seconds into the recording, as seen below. It is safe to assume that this occurred due to his device, in the "sleep" state, taking less frequent readings to preserve its battery life. In order to fix this, he increased the time-out period of this smartphone to prevent it from going into a "sleep" state during collection.

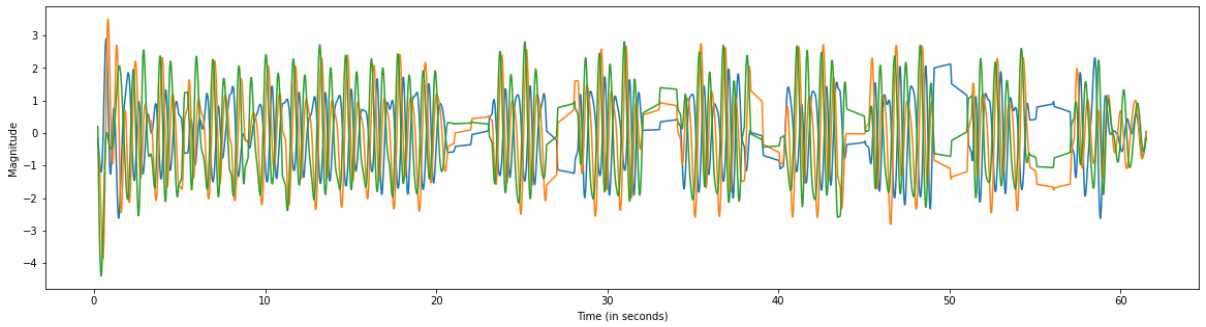

Figure: Matt's x-, y-, and z-axis data plotted.

Data Cleaning

All of our accelerometer data was exported into a comma-separated format (.csv) and was read into a Jupyter notebook using the imported Pandas' built-in `read_csv()` function. Each dataset contained segments at the beginning and end where the sensors were still, as mentioned in our "Data Gathering" segment. This short period, as well as sensor recordings of us inserting and removing the phones from our pockets, were unneeded data that may affect our results and were therefore removed from all datasets. The dataframe produced contained five columns: a `time` column containing a timestamp of the current index in terms of seconds from Unix epoch, a `seconds` column, which includes the number of seconds since the recording began, and columns for the `z`, `y`, and `x` readings. For the purposes of this project, we will be mainly focusing on the `seconds` column as the measure of time, as well as the `z`, `y`, and `x` columns, as these columns contain the data we will be operating on. Plotted below is our raw data, with the segments mentioned earlier highlighted.

Below is the trimmed dataset that will be further analyzed.

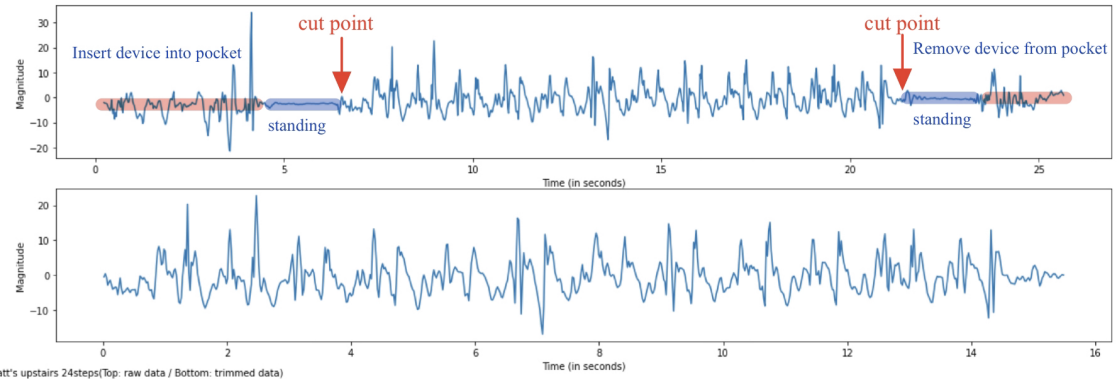

Figure: Matt's upstairs 24steps(Top: raw data / Bottom: trimmed data)

In an effort to reduce as much noise as possible, we passed all of our datasets through our implementation of a band-pass Butterworth filter. A Butterworth filter is a type of signal processing filter designed to produce a maximally flat frequency response (*Butterworth Filter*, n.d.). Alongside this filter, we also determined a specific range of frequencies that captured as little noise as possible from both the lower and higher frequencies. We chose a bound of 0.5 Hz for the high band-pass filter as none of our recorded movements were slower than this value. For the bounds of the low-pass filter, we chose a value of 2.2 Hz as none of our recorded activities exceeded this frequency. Any signal produced above or below these frequencies was interpreted as noise.

In terms of walking data, we performed a Fast Fourier Transform of the Butterworth filtered data to get an idea of the frequency of oscillation of our right legs. This was done by taking the absolute value of `numpy.fft()` on combined x, y, and z values.

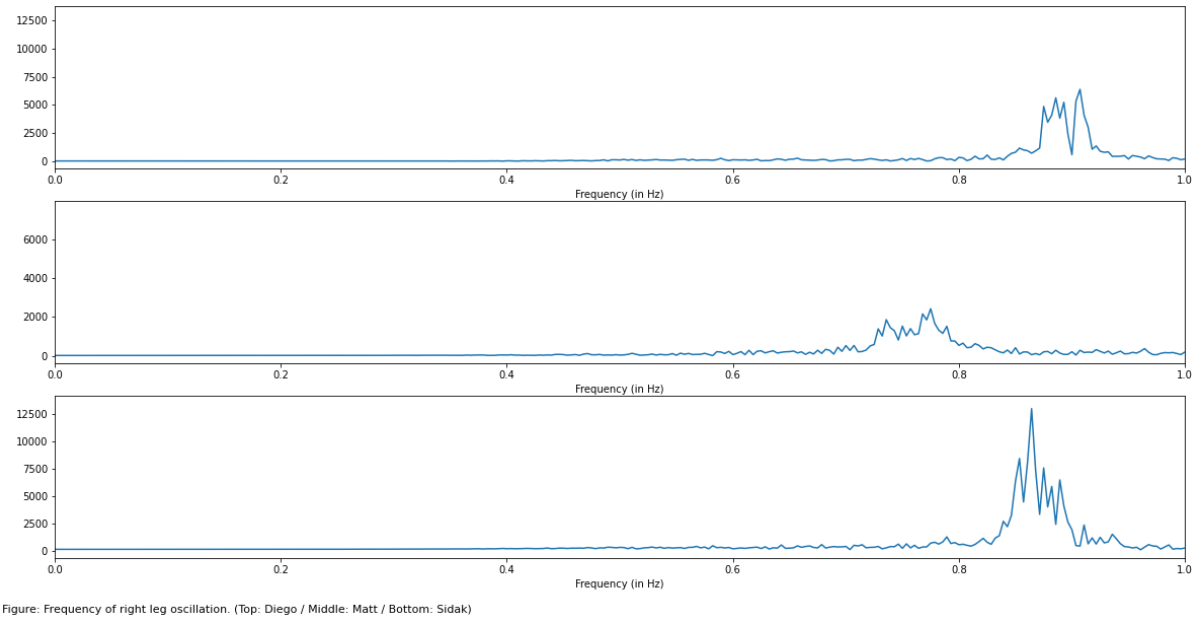

Figure: Frequency of right leg oscillation. (Top: Diego / Middle: Matt / Bottom: Sidak)

Findings

We built our training dataset by subdividing our walking, staircase, and running data into smaller data points. We added the columns of the corresponding axes of movement together and used that to build the indices of the training dataframe. This resulted in a dataset containing 69 indices of walking data, 63 indices of ascending staircase data, and 63 indices of descending staircase data. This training set was split into a training set and a validation set. We fitted our model on six different models: a naive Bayes model, a k-Nearest-Neighbours model, a Neural Network model, a Decision Tree model, a Random Forest model, and a `VotingClassifier()` that implements most of the models mentioned above. Below is a table with the validation scores of each model.

Model	**Validation Score**
Gaussian Naive Bayes Classifier	0.795
k-Nearest-Neighbours	0.897
Neural Network	0.571
Decision Tree Classifier	0.755
Random Forest Classifier	0.816
Ensemble Model	0.836

We also looked at the precision and recall scores of each model. These will be in the table below.

Model	Activity	Precision	Recall
Gaussian Naive Bayes	Descending	0.73	0.79
	Ascending	0.80	0.67
	Walking	0.84	0.94
k-Nearest-Neighbours	Descending	0.92	0.79
	Ascending	0.84	0.89
	Walking	0.94	1.00
Neural Network	Descending	0.44	0.50
	Ascending	0.62	0.56
	Walking	0.65	0.65
Decision Tree Classifier	Descending	0.62	0.71
	Ascending	0.71	0.67
	Walking	0.94	0.88
Random Forest Classifier	Descending	0.72	0.93
	Ascending	0.91	0.56
	Walking	0.85	1.00
Ensemble Model	Descending	0.70	1.00
	Ascending	1.00	0.56
	Walking	0.89	1.00

We then recorded a new set of data as the prediction datasets. The desired output is walking, ascending (upstairs), and descending (downstairs). Below is what our models predicted.

Model	Prediction
Gaussian Naive Bayes Classifier	`['downstairs' 'upstairs' 'walking']`
k-Nearest-Neighbours	`['upstairs' 'upstairs' 'downstairs']`
Neural Network	`['walking' 'upstairs' 'downstairs']`
Decision Tree Classifier	`['walking' 'upstairs' 'upstairs']`
Random Forest Classifier	`['downstairs' 'upstairs' 'walking']`
Ensemble Model	`['downstairs' 'upstairs' 'downstairs']`

Based on the tables above, despite the lower validation accuracy of the model, the Neural Network model was able to predict the desired results correctly. The precision and accuracy scores of the Neural Network model were also significantly lower than the rest of the models. The results above may be affected by the lack of more prediction data but, based on the above, it is safe to conclude that we can predict basic human activity using a Neural Network model on smartphone accelerometer data.

Challenges

One of the challenges we faced was deciding where to record our data from. Our initial set of walking data contained four recordings: Two of the datasets were recorded with a 10 Hz sampling rate, with one recording from our left pocket and the other recording from our right pocket. The other two were recorded similarly but with a 50 Hz sampling rate. Upon loading these datasets and processing them, we discovered that a sampling rate of 10 Hz was insufficient for our data, as the plots recorded with this sampling rate did not cooperate very well with the Butterworth filter and the Fast Fourier Transform. We also discovered that recording from either the left or right pocket gave us similar results. Based on these results, we decided that further data collection will be done with a 50 Hz sampling rate from the right pocket.

As mentioned in the “Data Gathering” section, one of our group members, Matt, produced a significantly different recording from the others. There were some notable stops in his plot, despite Matt walking continuously while recording. We initially assumed that this might be caused by the device going into a “sleep” state. We tested this assumption by having Matt record two new datasets, one with his device’s screen turned on and the other with his screen turned off (i.e. “sleep” mode). As shown below, the screen timeout affects the resulting data. Therefore, we ensured that all data must be collected with a screen-on device.

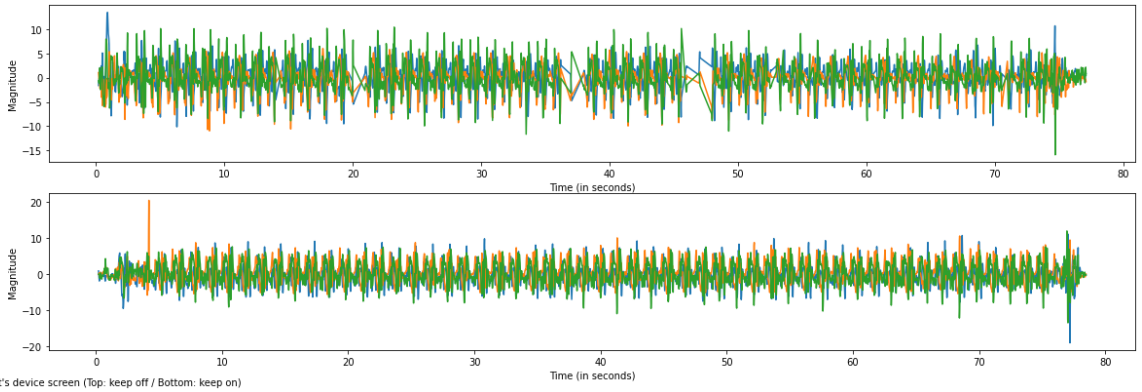

Figure: Matt’s device screen (Top: keep off / Bottom: keep on)

With our data cleaned, filtered, and transformed, we were faced with the challenge of manipulating our data such that it can be used to train machine learning models. As the `sklearn` models only took one-dimensional data, we had to figure out what form the data should be in in order for us to be able to pass it into the models. We initially tried training the models with the y-values produced by our Fourier transformed datasets but were unsuccessful as the `sklearn` models did not work well with complex number data. Some research revealed that we could simply add the columns corresponding to the axes of movement together (i.e. y- and z-axis movement for walking) to produce a “combined” frequency wave. We were then able to use the values of this “combined” frequency wave to train our machine learning models for this project.

One final challenge we faced was the inaccuracies we encountered when testing the model. When testing the models with our initial dataset, we experienced issues where our models would underfit the validation set. We assumed that this resulted from the disproportionate amount of walking data present in the training set. Initially, we had around 360 instances of walking in our training set and only 33 instances each for ascending and descending staircase data. This resulted in extremely

high precision scores for walking and poor precision scores for both staircase datasets. In order to fix this, we decreased the number of walking instances in our training dataset to equal the staircase sets, which, in turn, improved the precision for all three classifiers.

Accomplishment Statements

Diego Buencamino:

- Collected and organized smartphone accelerometer data using data manipulation tools available in Python's Pandas library.
- Used tools offered by Python's Scikit-learn library to train machine learning models to accurately predict basic human movement from accelerometer data.
- Documented, in detail, the process of acquiring and analyzing accelerometer data, and training machine learning models to predict human activity.

(Matt) Chang-Hsiu, Tsai:

- Acquired and organized smartphone accelerometer data, manually trimmed the useless data by Pandas DataFrame.
- Wrote and managed meeting records to list the task to finish, summarized the findings in a notebook demonstrating the central concept the project focuses on, and documented part of detail in the report.
- Inspected screen on/off for the gathering data on the smartphone with Python's Matplotlib library and analyzed machine learning models for human activity recognition.
- Separated Butterworth filter, Fourier Transform, initial Machine Learning into separate files, then writing the CSV files out, makes other files read the processed file directly and make the code more understandable.

Sidak Singh Aneja:

- Collected accelerometer data for walking and using stairs and pre-processed it to get relevant data using Pandas.
- Created a pre-processing file that reads all raw accelerometer datasets and trims them down to necessary intervals.
- Used the Scikit-learn library to write a signal processing function that used the Butterworth Bandpass filter to filter out the noise from the accelerometer data.
- Wrote the Fast Fourier Transform function to get the frequency of steps for each dataset and plot the results using the Matplotlib library.
- Trained the human activity machine learning model using tools provided in the Scikit-learn library.

References

Butterworth filter. (n.d.). Wikipedia. Retrieved December 7, 2021, from https://en.wikipedia.org/wiki/Butterworth_filter