ELEC 292 Final Project

Project Report

ELEC 292

Computer Engineering

Faculty of Engineering and Applied Science

Queen's University

Group 33

Erik Sachik, 20342364, 21es82@queensu.ca

Simon John, 20348233, 21snj4@queensu.ca

Emre Karaman, 20360265, 22aek@queensu.ca

[2024-04-07]

1. Data Collection

The team decided to collect data by alternating between walking and jumping for three minutes each, which exceeded the initial five-minute time constraint. They used their personal smartphones, including models like the iPhone 13, iPhone 13 Pro, and iPhone 14, and utilized the Phyphox app as instructed. Each team member recorded their movements while holding the phone in their right hand for one minute, then in their left pocket for another minute, and finally in their back right pocket for the last minute, totaling six minutes. The goal was to capture variations in movement patterns. The data collected included accelerometer readings in 'g' and was exported as a CSV file for further analysis.

Initially, the team encountered issues such as failing to record data for the required duration and obtaining poor-quality data due to the positioning of the phone. After trying various methods, they found that keeping the phone in the right pocket, left hand, and right back pocket led to the collection of high-quality data. One team member, Emre, was unable to participate in the jumping activity due to an injury. To address this, the team used another member's jumping data for Emre's portion, ensuring the integrity of their dataset with the professor's approval.

The team transferred the data from their phones to their computers by emailing it and then shared it on Microsoft Teams so all members could access it. Each member then placed the CSV files in the project directory within PyCharm for use in their analysis.

2. Data Storing

After data collection there were a total of six different csv files, each containing jumping or walking data from one of three people. The Code designed in the Data Storing part of this project should take the six csv files from data collection and create an HDF5 file formatted as shown below in Figure 1.

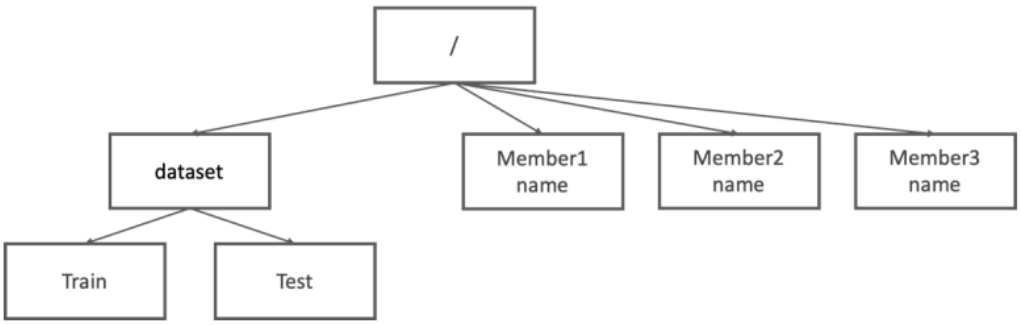

Figure 1: Desired organization of final HDF5 file

Before the HDF5 file is created the data must be manipulated for the Train and Test parts of the file shown on the left side of Figure 1. The steps are as follows: labeling, concatenation, shuffling, and splitting.

Labeling the data is a straightforward process. First a column titled “label”, is added to all the csv files. If a csv file contained walking data the label column contained all zeroes, if it contained jumping data then the label column contained only ones. This step is implemented with the basic of data frames in python.

Concatenating the data frames is another simple process. The paths of all the labeled csv files are loaded into 6 separate data frames and then with the use of the “concat” function from the pandas library the 6 data frames are concatenated.

Next the single big csv file containing all the labeled data must be shuffled. First the data is transferred to a data frame, then a column is added which numbers each unique 5 second window in the data, and an array is created with the same data held by this column. The new column in the data frame is mapped using the shuffled array to shuffle the data in 5 second chunks while maintaining the order of the data within each 5 second interval.

Like labeling and concatenation, splitting the shuffled data into a training and testing sets is a trivial task. After loading the shuffled data into a data frame, the “train_test_split” function from sklearn is used to split the data, a random state of 4 was used, and the data was split as 90% training and 10% testing. After the split data is saved all the files needed to create the desired HDF5 file now exist.

The files in the HDF5 file are the training and testing files, and all six labeled raw data files. Once the data is loaded into data frames and a new HDF5 file is opened, the groups can be

created. The first group is created at the root of the file and has the name of the first team member, datasets are created to hold their respective labeled walking and jumping data. This process is repeated for the other two members of the team. Next a training group is created with a parent group “data” as shown in the provided Figure 1. The shuffled training dataset from earlier is added to this group then the process is repeated for the testing data which is a separate group under the same parent group “data” at the root of the HDF5 file.

3. Visualization

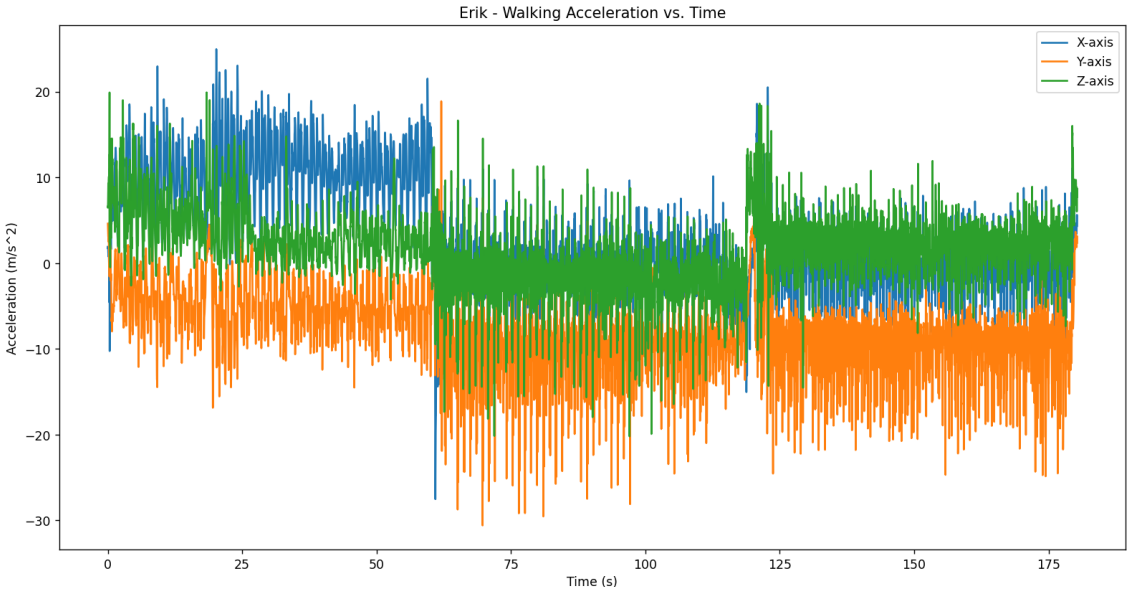

Figure 2: Erik's Walking acceleration vs. time graph, can clearly see 3 different segments of the graph where the position of the phone has switched. First segment being in right hand, then in left pocket, then in right back pocket. Can also see the switching points.

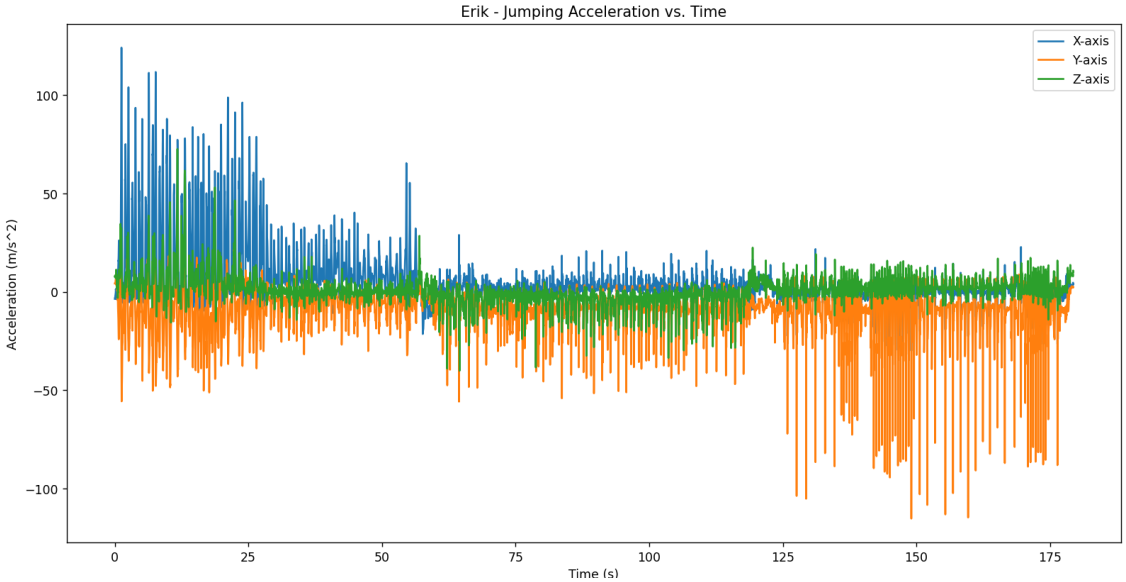

Figure 3: Erik's jumping acceleration vs time graph, can clearly see 3 different segments of the graph where the position of the phone has switched. First segment being in right hand, then in left pocket, then in right back pocket. Can also see the switching points.

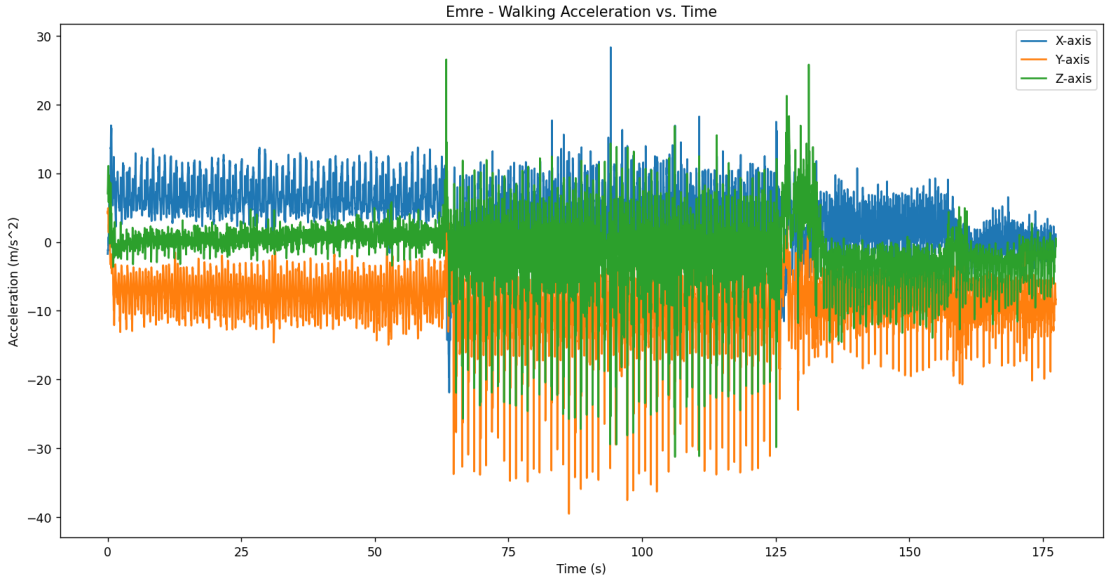

Figure 4: Emre walking acceleration vs time graph, can clearly see 3 different segments of the graph where the position of the phone has switched. First segment being in right hand, then in left pocket, then in right back pocket. Can also see the switching points.

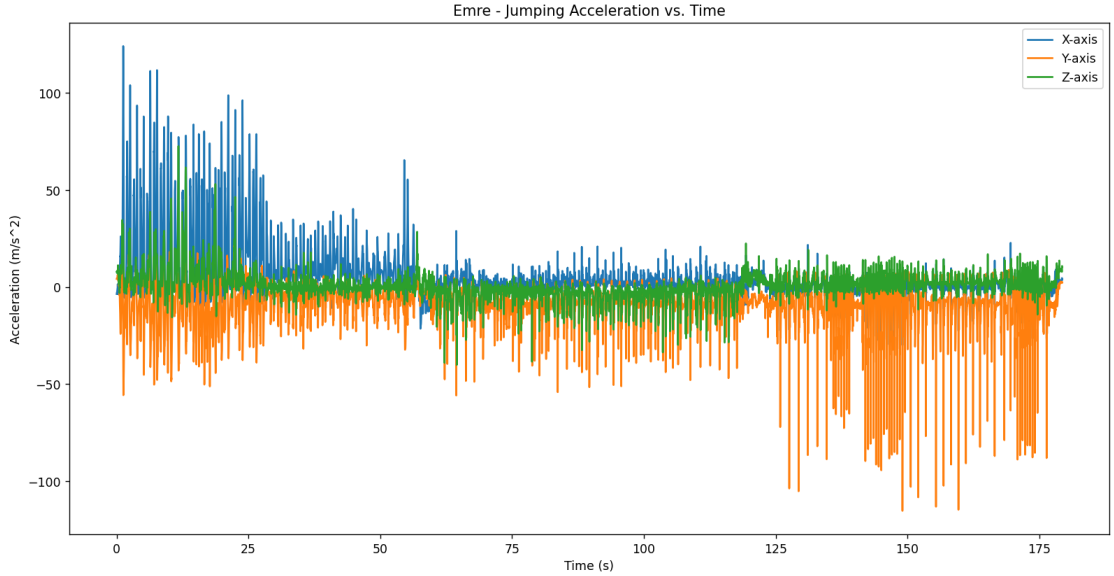

Figure 5: Emre's jumping acceleration vs time graph, , can clearly see 3 different segments of the graph where the position of the phone has switched. First segment being in right hand, then in left pocket, then in right back pocket. Can also see the switching points.

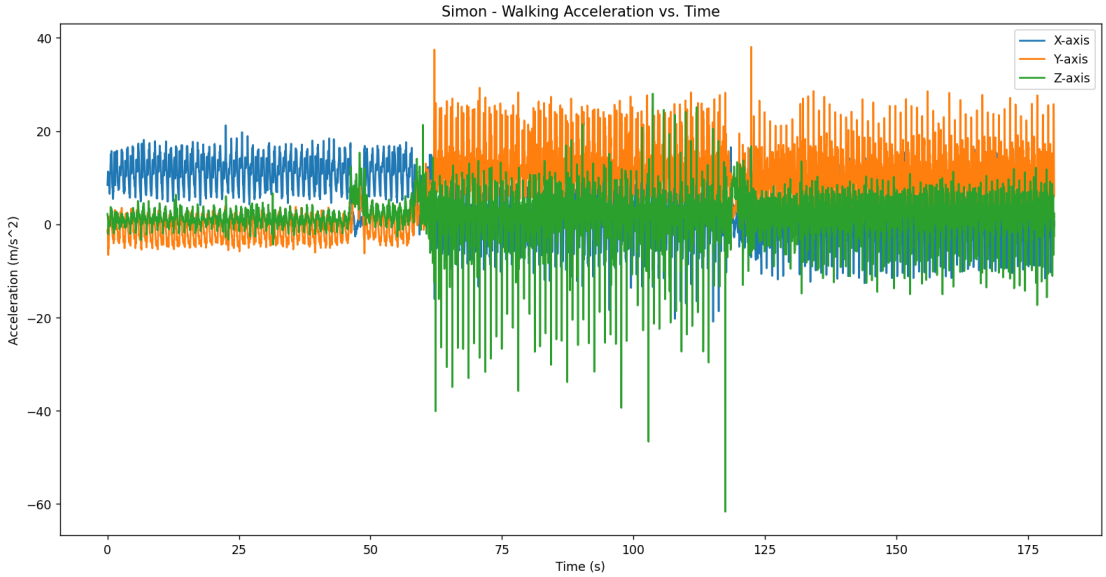

Figure 6: Simon walking acceleration vs time graph, can clearly see 3 different segments of the graph where the position of the phone has switched. First segment being in right hand, then in left pocket, then in right back pocket. Can also see the switching points.

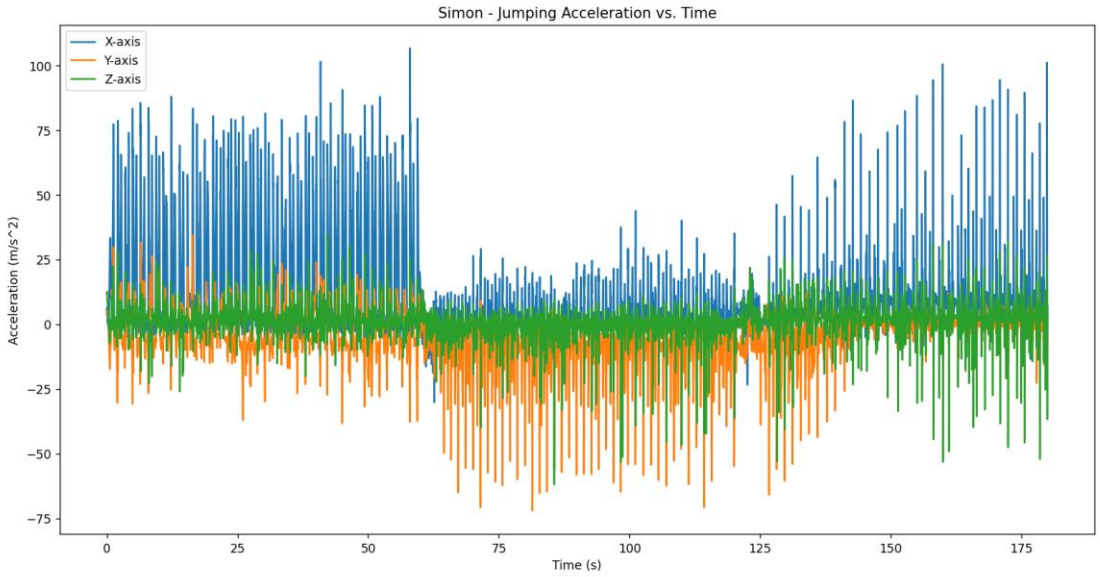

Figure 7: Simon jumping acceleration vs time graph, can clearly see 3 different segments of the graph where the position of the phone has switched. First segment being in right hand, then in left pocket, then in right back pocket. Can also see the switching points.

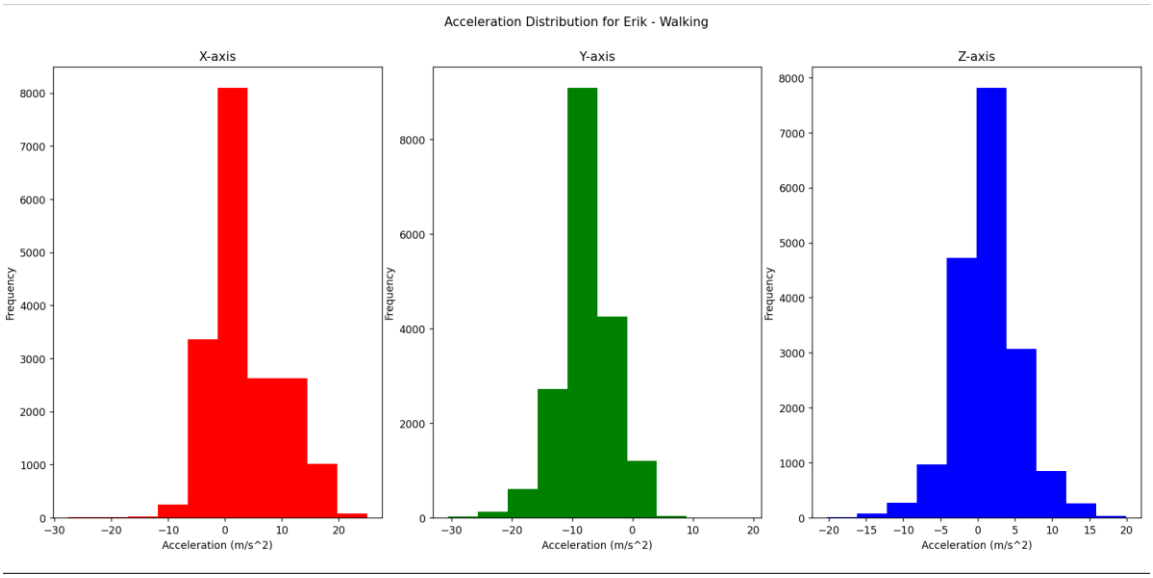

Figure 8: Erik walking acceleration distribution histogram showing the spread of acceleration values amongst different axis, seeing the values that occur most frequently.

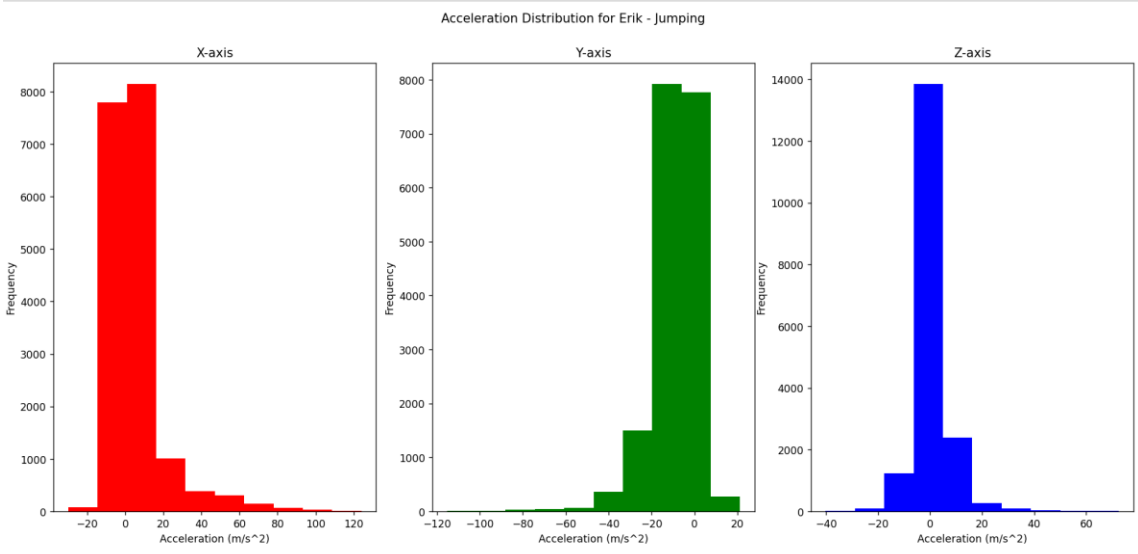

Figure 9: Erik jumping acceleration distribution histogram showing the spread of acceleration values amongst different axis, seeing the values that occur most frequently.

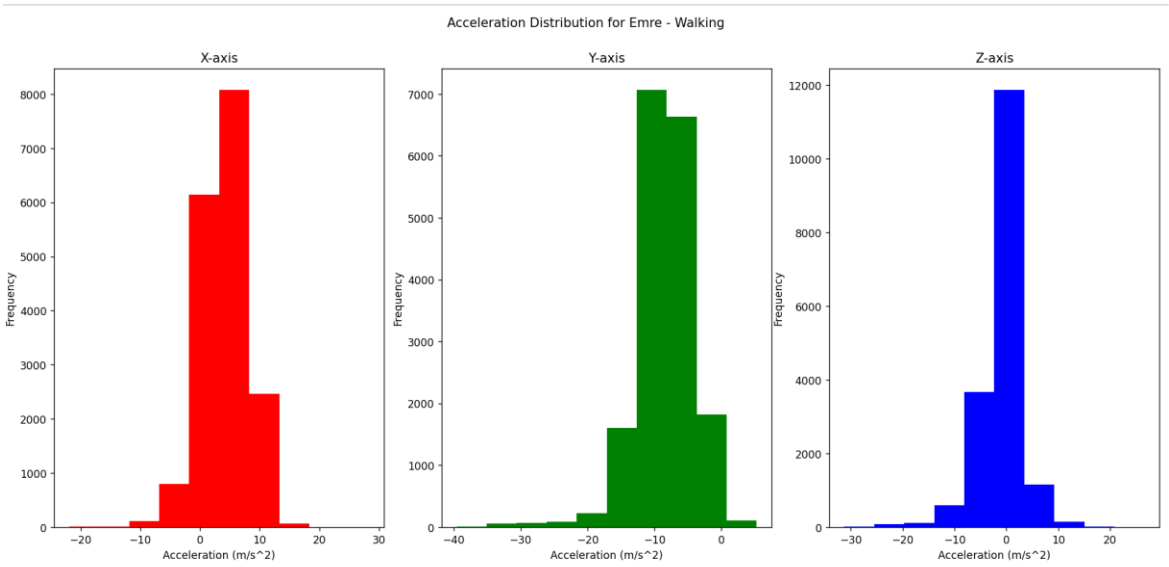

Figure 10: Emre walking acceleration distribution histogram showing the spread of acceleration values amongst different axis, seeing the values that occur most frequently.

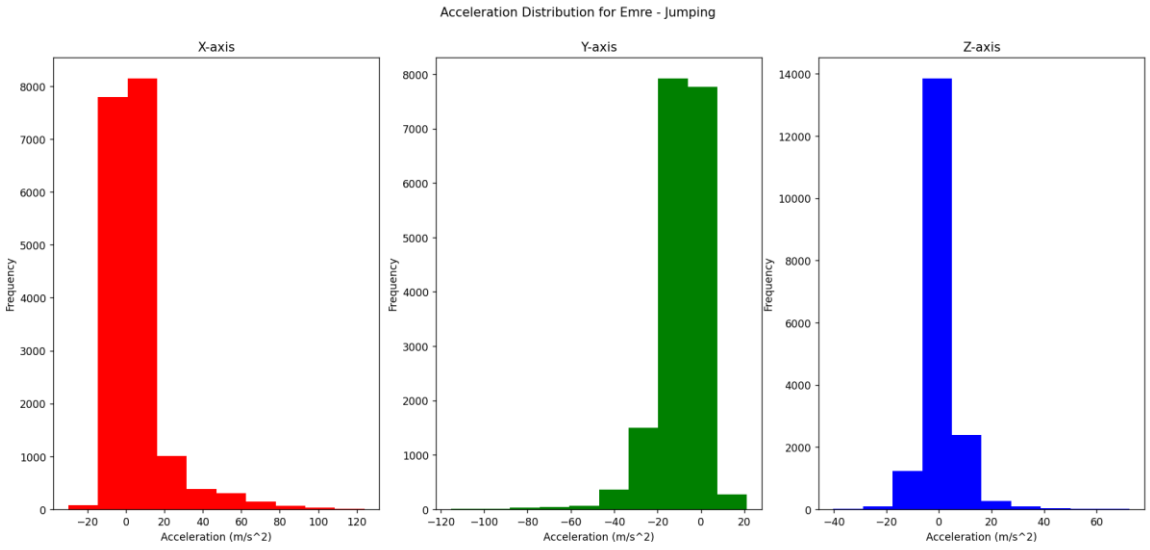

Figure 11: Emre jumping acceleration distribution histogram showing the spread of acceleration values amongst different axis, seeing the values that occur most frequently.

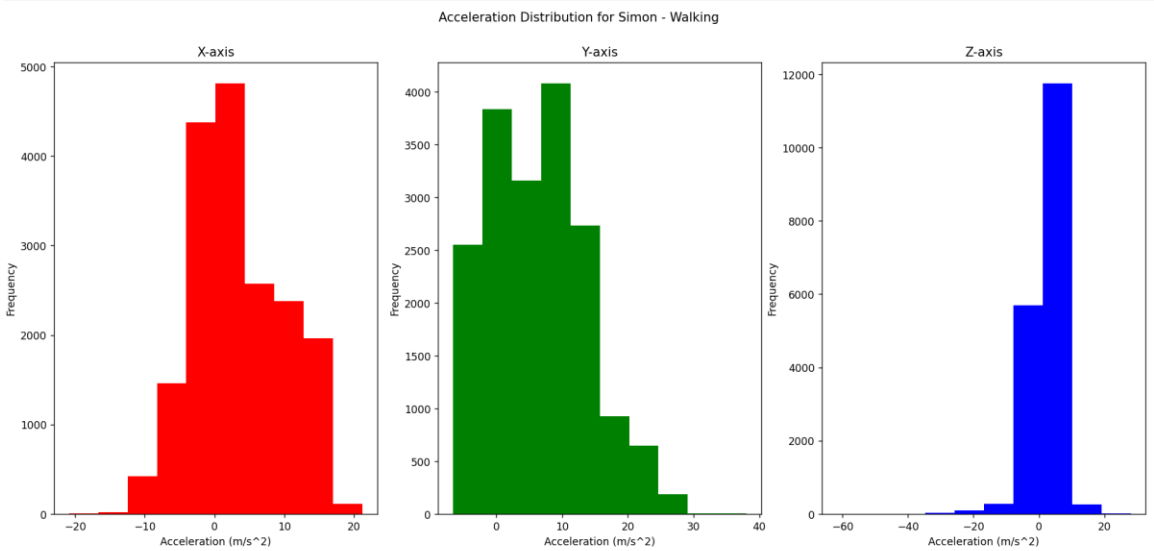

Figure 12: Simon walking acceleration distribution histogram showing the spread of acceleration values amongst different axis, seeing the values that occur most frequently

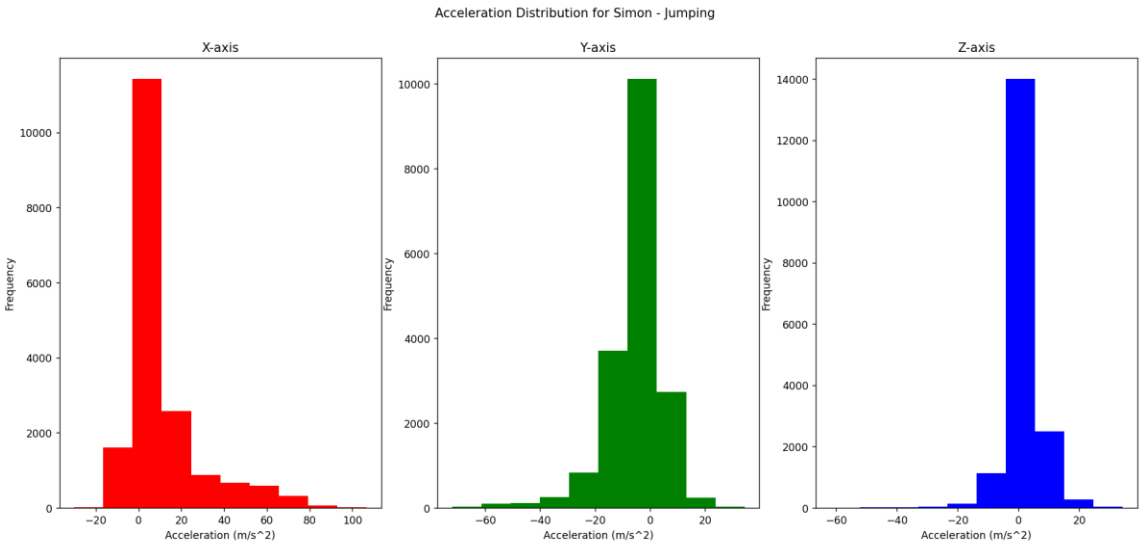

Figure 13: Simon jumping acceleration distribution histogram showing the spread of acceleration values amongst different axis, seeing the values that occur most frequently

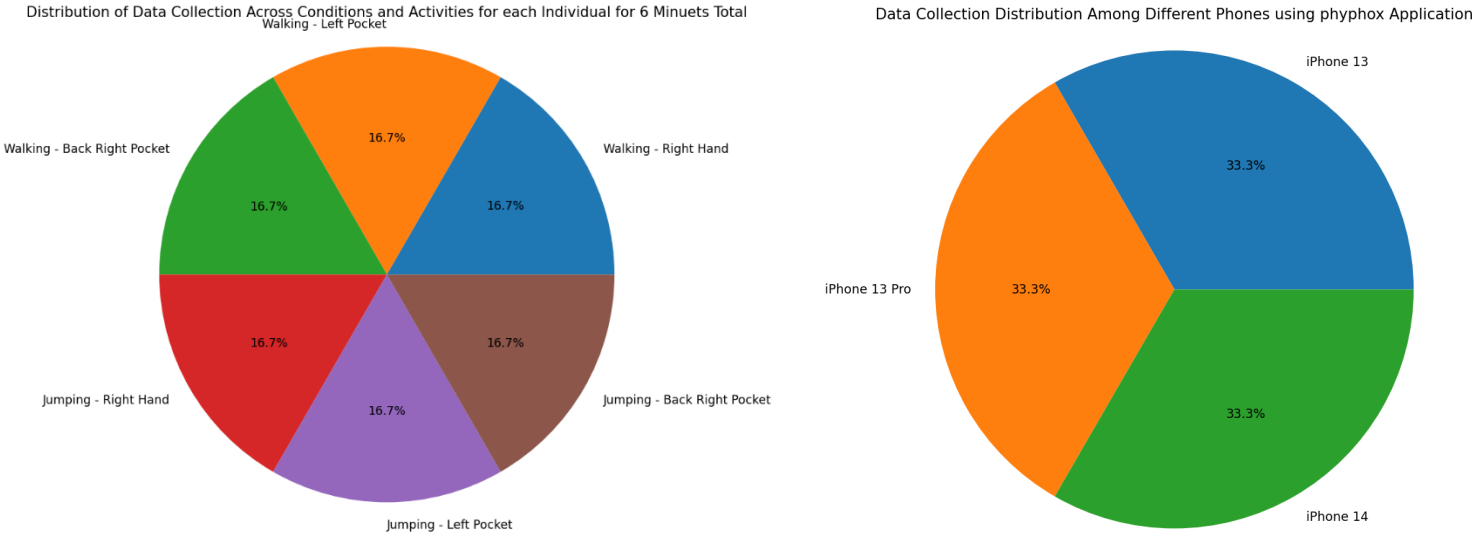

Figure 14: Two pie graphs showing meta data; more specifically, the pie graph on the left showing the distribution of data collection across conditions and activities for each individual for 6 minuets total, and the graph on the right showing data collection distribution among different phones using phyphox application

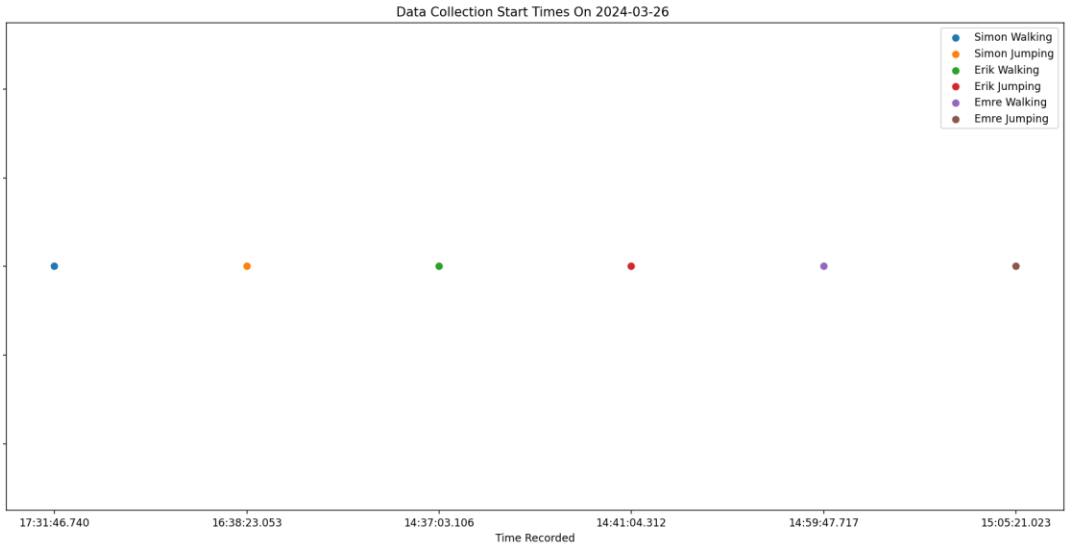

Figure 15: Another graph showing meta-data, more specifically a scatter plot showing the data collection recording times

The acceleration vs time graphs show the range of movements amongst different axis, identifying periods of higher activity within the different axis. Within those plots you can see the different periods of where the phone is placed for recording data. Since it is the raw data, you can also see the noise and artifacts on the plots, which is typical and could be due to the sensor's sensitivity.

The spikes in acceleration, suggest moments of high activity which would make sense to be jumping. It would make sense if the spikes for jumping were only in the Z-axis, but you have to take into account the orientation of the phone because it effects the accelerometers orientation aswell. The magnitude of the spikes shows the force of the jumps. The different patterns across the different axis can be analysed to show jumping and walking. For most cases you can see one of the axes has clear spikes indicating movement in the vertical direction (jumps), while the other two axis don't have as high spikes showing movement along the ground. Ideally this would be always spikes in the z axis, since that is typically the vertical axis. For the walking graphs, you don't see any spikes indicating little to no movement in the vertical direction and an overall lower absolute acceleration as seen by the scale of the graph. A big change for future data collection would be to ensure that the phones sensor is held in a specific way, oriented with x,y and z axes so it would be easier to analyze the patterns.

You can also directly see where the phone is being switched positions, next time if data collection could be redone, a better approach would be to have timed sections for each phone position where it automatically stops recording, that way you wouldn't be able to see the switching positions on the plot, and it would allow for a smoother plot reducing noise and artifacts.

The histograms do a good job of showing the skewness of distributions of acceleration values and showing predominant directions for movement. It also does a great job showing the range of acceleration values, the most frequent acceleration values, and acceleration outliers.

For the meta-data 3 total plots were created. The first plot showed the distribution of activities. As a group each individual completed the exact same pattern of data collection; 1 minuet of walking with phone in right hand, 1 minuet of walking with phone in left pocket, 1 minuet of walking with phone in back right pocket, 1 minuet of jumping with phone in right hand, 1 minuet of jumping with phone in left pocket, 1 minuet of jumping with phone in back right pocket. For future data collection, more variance in the phone position should have been used but the team did a good job of ensuring that each phone position was measured for the same amount of time, to not create false trends. Another graph was a pie graph showing the different phones being used to measure the acceleration data. All 3 people were using similar generations of iPhones which most likely all contain the same accelerometers. This could suggest a bias towards the iPhones as the data depends heavily on sensor quality and calibration and since the same devices were used this could affect the potential trends as different phones could output different acceleration readings.

4. Preprocessing

For preprocessing the data, two measures were used to help reduce the noise of the original data, and to normalize the data so that it becomes suitable for logistical regression.

Firstly, the noise was reduced so it wouldn't obscure or destroy important features in the signal. The noise was reduced using an average moving filter which is a simple filter that filters out high frequency noise which can help in identifying actual walking patterns as apposed to random noise. For this, a window size of 5 was used. This is because the larger the window size, the smoother the output will be, but it will also reduce the integrity of the original dataset losing some important information. During the testing of our models accuracy, we made small changes in the code to see what would output the highest accuracy and it was concluded that a window size of 15 helped output the highest accuracy score. Therefore, a window size of 15 does a good job of reducing the noise and keeping the datasets original integrity.

Next, the data was normalized, ensuring that the data had a zero mean. This helped create a uniform range across the dataset that encapsulated important features in the data, allowing the data to be easier to read and compare to different datasets. This was done using the preprocessing.StandardScalar() function from sklearn. The average moving filter alongside the normalization did a good job of reducing the noise and normalizing the datasets as seen in the following graphs on the next two pages.

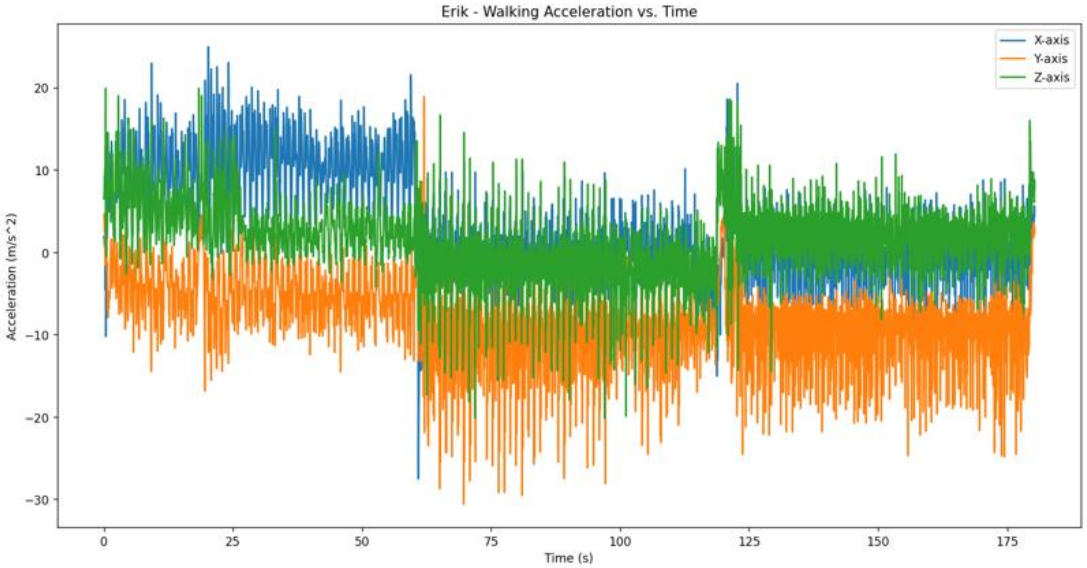

Figure 16: Erik's Original Walking acceleration vs. time graph. Can clearly see the noise in the data aswell as the scale of the acceleration. The original data shows a wide range of values with significant spikes and a lot of variability which could be due to noise in the data, or actual spikes in the data.

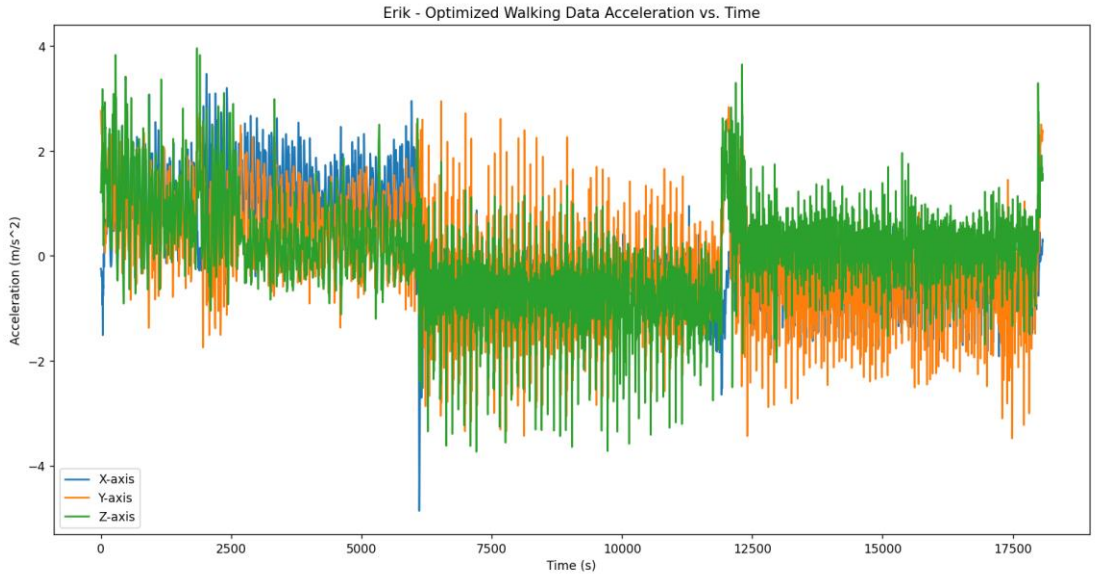

Figure 17: Erik's walking acceleration vs time graph after noise reduction and normalization. The data appears much smoother with reduced noise making it easier to detect patterns and trends and has a largely reduced range of values indicating the moving average filter worked as intended. Normalization has also scaled the data, so the acceleration values fall within a more constrained range.

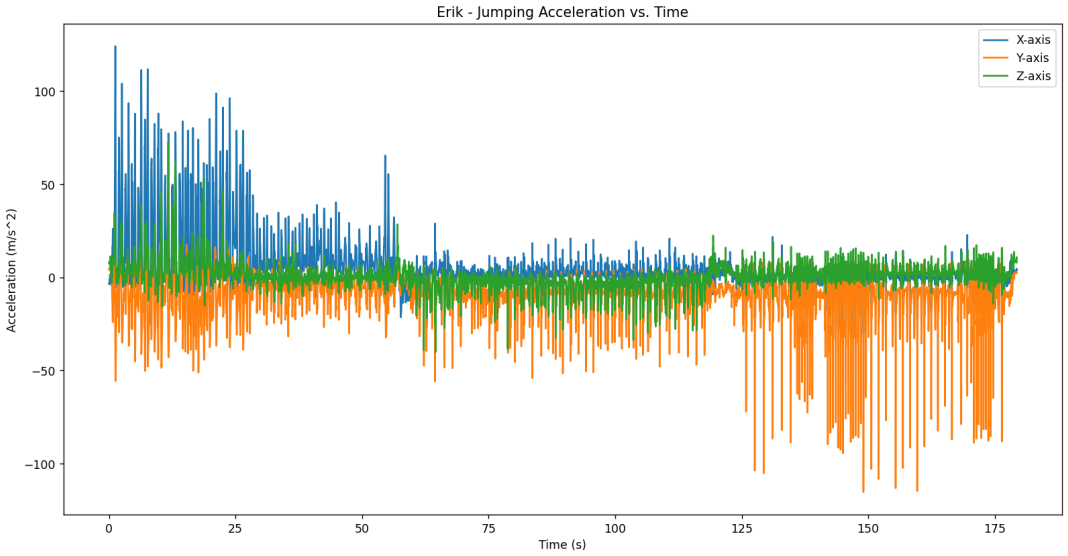

Figure 18: Erik's Original Jumping acceleration vs. time graph. Can clearly see the noise in the data aswell as the scale of the acceleration. The original data shows a wide range of values with significant spikes and a lot of variability which could be due to noise in the data, or actual data spikes such as jumps.

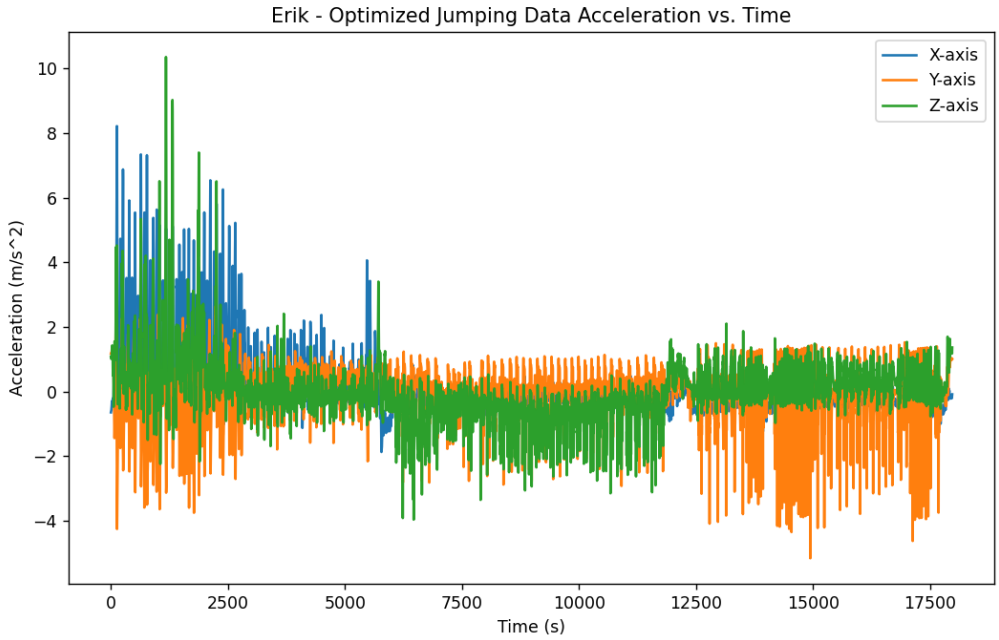

Figure 19: Erik's Jumping acceleration vs time graph after noise reduction and normalization. The data appears much smoother with reduced noise making it easier to detect patterns and trends and has a largely reduced range of values indicating the moving average filter worked as intended. Normalization has also scaled the data, so the acceleration values fall within a more constrained range.

5. Feature Extraction & Normalization

Mean: Average of the data given the window size which shows the average acceleration. The mean is a good indicator of the intensity and magnitude of activity. For the walking datasets, the mean would show the average walking pace, and for jumping datasets the mean would show the average jumping acceleration and typically the average acceleration for jumping would be higher than walking.

Standard Deviation: Measures the variability from the mean which can be very useful for the acceleration data. For walking datasets, the standard deviation is going to be much lower than the standard deviation for jumping datasets since a higher standard deviation indicates more intense movement, and the walking standard deviation will suggest a less intense walking movement.

Maximum & Minimum: Shows the extreme values for the datasets given the window size, which can be useful for acceleration data as it can show intense movements like jumps or sudden stops. A walking dataset is going to have much less extreme values then a jumping dataset as the acceleration should be steadier and within a lower range.

Range: The difference between the max and min values. Like the maximum and minimum values, the range will be quite different from walking and jumping datasets, as for walking datasets, the range will be much smaller indicating a less intense movement, and with the jumping values, the range will be much larger spanning from landing values to jumping values.

Kurtosis: Describes the shape of the distribution. High kurtosis means more of the variance is from extreme deviations that don't happen often, as opposed to medium sized deviation that happen often. It is useful to show if the data has outliers which could be jump values that are much higher or lower than the mean. Similar for walking, it can help show irregular walking patterns that deviate from the mean [1].

Skewness: Indicates the asymmetry of the dataset distribution. For acceleration data, skewness can show whether the data has a bias toward more positive or negative values, which could relate to movement patterns. The jumping datasets should have more skew as there is more distribution in the acceleration values, and a walking dataset would be more symmetrical and less skewed as the values are closer to the mean [2].

Variance: Variance is the squared deviation from the mean which provides very similar information to standard deviation but emphasises the extreme deviations more. Like standard deviation, the variance can help identify walking vs jumping as jumping will have a higher variance.

Median: Similar to the mean but it shows the middle value of the dataset. Can be also very useful and is typically less sensitive to outliers which can therefore show a more typical acceleration value for the datasets.

Sum: Shows the total acceleration value within the window size of the dataset. For walking datasets, the sum could be lower showering more consistent movement vs a higher sum for jumping as it represents higher acceleration values.

Standard Error: Shows the accuracy of the mean value, so as the standard error is lower, the more accurate the mean value is. This is very important to understand because it shows how accurate the mean value is which provides a big insight on the acceleration when walking vs jumping.

Exponential Moving Average: Shows the moving average with the use of noise reduction and smoothing. It helps show the mean trend much more effectively then a regular .mean() as it highlights trends better, and uses most recent data points by exponentially decreasing the weight of older data points making it extremely useful for acceleration data.

The features were extracted using a rolling window, with a window size of 5 to ensure the features were closely related to the exact data. They were then extracted using the different feature operations in pandas like .mean(), .max(), .skew(), etc [3]. These features show clear differences between walking and jumping datasets and will be used to train the AI model which can be used to distinguish between walking or jumping based on the features.

6. Training the classifier

The team put together the training and test datasets by combining the data for walking and jumping activities, using the 'pd.concat' function from the Pandas library. This is because it allows the model to learn from a wider set of examples that include different activities. Mixing these datasets helps the model grasp the complex patterns and behaviours linked to each activity, improving its prediction accuracy.

The next step is to make sure the dataset is ready and in good shape. A key part of getting the data ready involves dealing with any missing or infinite values in the features. The team handled this by using the 'replace' and 'fillna' methods to fill these gaps with zeros. This step is vital to ensure the dataset is clean and primed for effective training of the model.

Once the model is set up, the next move is to train it with the combined dataset of training features and labels. In this stage, the model adjusts its settings to get closer to the correct answers in the training data. This helps the model get better at making accurate predictions.

Following the training, the model undergoes testing by making predictions on the test dataset. Its performance is assessed by comparing these predictions to the actual labels of the test data, utilizing the 'accuracy_score' function. This step of evaluating the model's performance gives insight into how well the model can adapt to data it hasn't seen before.

After testing, the model's accuracy on new data is shown. This accuracy tells us how well the model is performing and if it needs improvement. The team carefully trains, tests, and improves the model to make sure it learns from data and makes accurate predictions in real situations.

7. Model deployment

This section can be broken down into a few parts, making the application, processing and predicting the data, and plotting the results.

tkinter was used to create the application. A few labels were defined to give the application a title and to be used later to show file paths and other information. A button was defined to give the user some more sense about how to use the application. Shown in Figure 20 below, when the user initially runs the app there is a title, some short instructions, and a button to press to select a file.

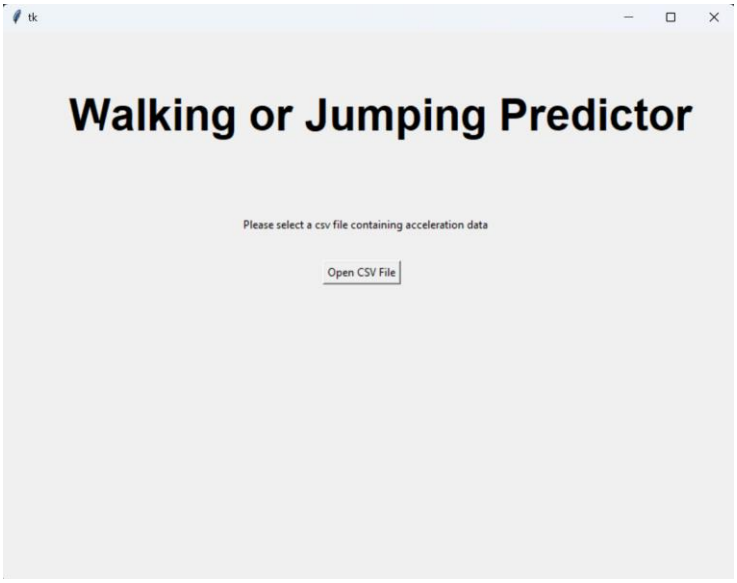

Figure 20: State of the application on start

When the user clicks the button, they get another pop-up of file explorer, shown in Figure 21 so that can select a csv file. After choosing an appropriate file, file explorer disappears and the file path they selected shows on the GUI of the application as seen in Figure 22.

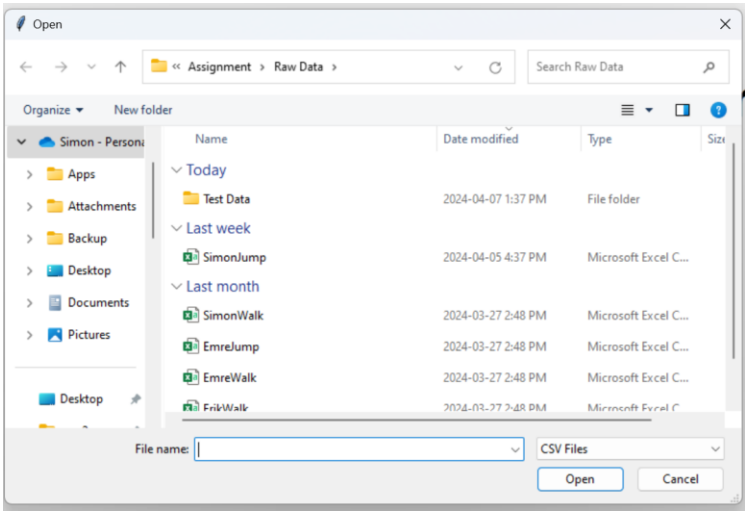

Figure 21: File Explorer pop-up

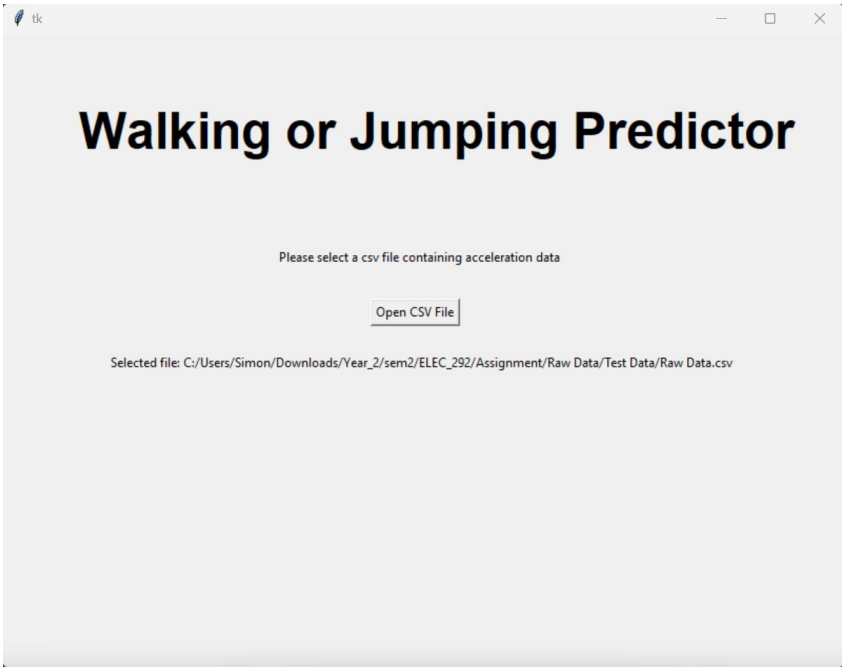

Figure 22: State of application after selecting a file

Earlier some unused labels were mentioned, the line showing the selected file path makes use of one of the unused labels. To center the text a combination of pack and place from the tkinter library was used.

Before the model is used to make a prediction the chosen csv file must be processed so that it has all the same data that the model was trained on. The model was trained on a file which holds features from a shuffled csv containing acceleration data. The chosen file must undergo, noise reduction, feature extraction, and normalization before it can be given to the model.

Because noise reduction, feature extraction, and normalization (normalization is included in the feature extraction function and so will no longer be mentioned as it can be though of as a part of the feature extraction function) were all done earlier in the project their definitions can be reused for the most part. The noise reduction and feature extraction functions were changed to accept and return a data frame rather than accepting input and output paths and returning a .csv file, the feature extraction function was also changed to work with the slightly different shape of the chosen csv file compared to the ones used to train the model, this is mostly due to the lack of a label column in raw data.

After extracting the features from the chosen file, the model is used to make a prediction about which parts of the code correspond to walking vs jumping. After making the prediction the file is converted to a data frame then is concatenated to the original file in data frame form.

With the prediction data a plot is made showing absolute acceleration vs time data which is colour coded as blue if the data point was predicted to be walking or red for jumping. Figure 23 shows the plot created by the application when given jumping data.

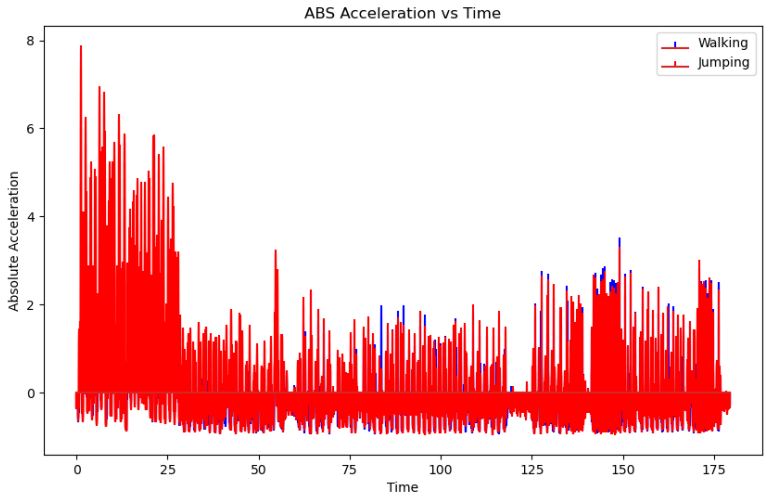

Figure 23: Acceleration vs Time for jumping data

The graphs like Figure 23 are created with stem graphs from matplotlib with the marker removed to improve readability.

After the user closes the graph, the final predicted data is converted to .csv file and is stored at the root directory of the program. The last unused label from earlier is used to provide the user with some direction, as seen in Figure 24, about the name of the saved file and where they can find it.

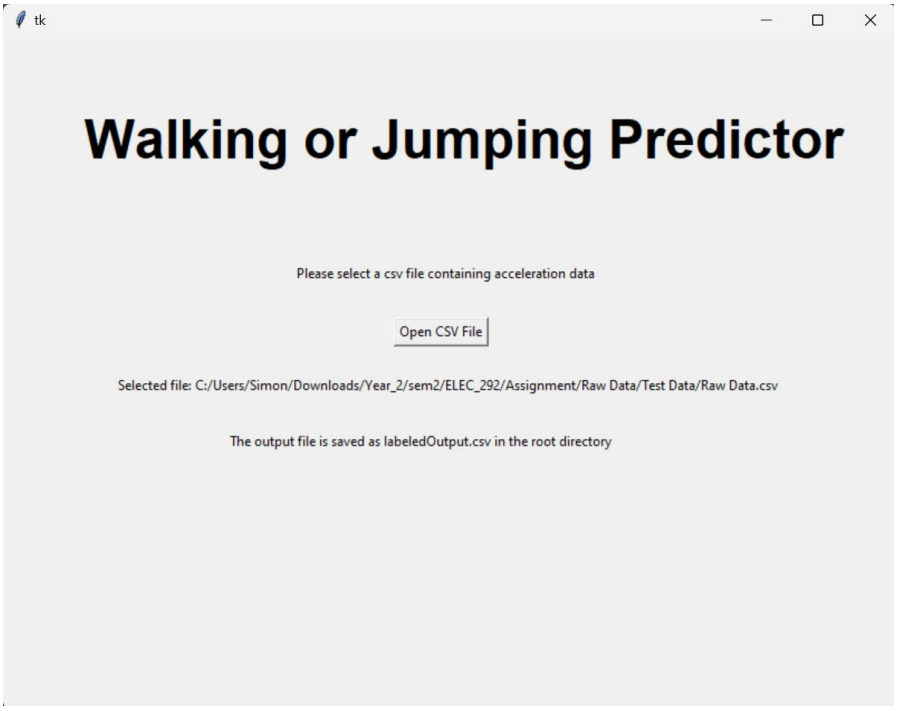

Figure 24: GUI after final instruction to user

After the file is saved the user is free to select a different file to plot and predict.

8. Participation Report

Project Tasks:	**Completed By:**
Data Collection	Everyone (Emre for report)
Data Storing	Simon (Report and Code)
Visualization	Erik (Report and Code)
Pre-processing	Erik (Report and Code)
Feature extraction and normalization	Erik (Report and Code)
Creating a classifier	Emre (Report and Code)
Deploying the trained classifier in a desktop app	Simon (Report and Code)
Demo Video	Everyone
Report	Everyone

9. References

[1] "Datacamp," [Online]. Available: https://www.datacamp.com/tutorial/understanding-skewness-and-kurtosis. [Accessed 30 March 2024].

[2] "Geeks for geeks," [Online]. Available: https://www.geeksforgeeks.org/how-to-calculate-skewness-and-kurtosis-in-python/. [Accessed 30 March 2024].

[3] "Analyist Soft," [Online]. Available: https://www.analystsoft.com/en/products/statplus/content/help/pdf/analysis_basic_statistics_descriptive_statistics.pdf. [Accessed 30 March 2024].

[4] tutorials point, "tutorialspoint," Tutorials Point India Private Limited, 2014. [Online]. Available: https://www.tutorialspoint.com/python/python_gui_programming.htm. [Accessed 7 4 2024].