Project Report

ELEC 292

Electrical and Computer Engineering Discipline

Faculty of Engineering and Applied Science

Queen's University

Prepared by Group 56

Morgan Haviland, 20412721, 22KFn4@queensu.ca

Andrew Heraldo, 20413223, 22wkm1@queensu.ca

Daniel Ifergan , 20413076, 22lhw@queensu.ca

04/04/2025

.

Contents

Table Of Figures

Executive Summery

This project focuses on developing a supervised machine learning pipeline to classify human motion using smartphone-based accelerometer data. The goal is to distinguish between two types of physical activity: walking and jumping.

The Phyphox mobile application was selected as the data collection tool due to its ability to capture linear acceleration data across three axes (X, Y, Z) while excluding the gravitational component. The absence of gravity from the signal ensures that only dynamic, user-generated motion is recorded. Each team member contributed by collecting a standardized set of walking and jumping trials across a variety of phone placements and orientations.

The collected accelerometer data was stored in a structured format using the Hierarchical Data Format (HDF5). To support further processing, each dataset was parsed to extract key metadata, preprocessed using a moving average filter to reduce noise, and segmented into 5-second intervals. These segments were then used to extract features for a binary classification model.

Visualization was used to identify patterns in raw data, evaluate preprocessing results, and compare characteristics between walking and jumping. Time-series plots, histograms, and scatter plots were generated to compare movement intensity and directional variation. These visualizations helped guide feature selection and assess how sensor placement affected signal quality.

The machine learning model was trained using logistic regression and tested against held-out data to assess classification accuracy. Evaluation metrics such as confusion matrices and ROC curves were used to validate performance. The final model was integrated into a graphical user interface (GUI), allowing users to classify new data by uploading CSV file.

1. Data Collection

To support accurate classification of 'walking' and 'jumping,' accelerometer data was collected using the Phyphox app, available on both Android and iOS devices. This app enabled the team to record raw sensor acceleration without gravity, providing linear acceleration data only. Linear acceleration was chosen because gravity is a constant (approximately 9.81 m/s^2) and does not contribute to distinguishing between different types of movement. Removing gravity ensures that the system captures only motion-induced forces, improving the accuracy of classification models.

1.1 Collection Procedure and Labeling

Each team member (M1: Morgan, M2: Daniel, M3: Andrew) conducted 12 trials—6 for walking and 6 for jumping—resulting in a total of 36 trials across the group. The time distribution for the trials can be seen in Table 1. Each trial lasted 50 seconds and was conducted based on a standardized data sampling table that detailed:

- **Walking Phase or Jumping Test**
- **Phone Placement**: e.g., front pocket, back pocket, jacket pocket, hand-held, backpack, belt
- **Phone Orientation**: upright, sideways, upside-down
- **Walking or Jumping Variation**:
    - Walking: slow (straight line), normal (around corners), brisk (straight line)
    - Jumping: light hops, medium hops, high jumps

This sampling plan ensured balanced coverage across different placements, orientations, and movement intensities.

Table 2 outlines the full data collection matrix used by the team, including timing and combinations assigned to each member. To maintain consistency, each trial was labeled using the following naming convention:

[MemberName]_Walking_[Placement]_[Orientation]_[Variation].csv

Table 1: Data Collection Time.

Task	Duration
Walking Data Collection (6 placements)	5 Minutes
Jumping Data Collection (6 placements)	5 Minutes
Total Collection Time Per Person	10 Minutes
Total Collection Time for Team (3 members)	30 Minutes

Table 2: Data Collection Matrix.

Phase	Phone Placement	Phone Orientation			Walking Variation/ Jumping Variation			T(S)
Walking Phase		M1	M2	M3	M1	M2	M3	
	Front pocket	Upright	Sideways	Upside-down	Slow (straight line)	Brisk (around corners)	Normal (straight line)	50
	Back pocket	Upside-down	Upright	Sideway	Normal (around corners)	Slow (straight line)	Brisk (straight line)	50
	Jacket pocket	Sideway	Upside-down	Upright	Brisk (straight line)	Normal (straight line)	Slow (around corners)	50
	Hand-held	Upright	Sideway	Upside-down	Slow (straight line)	Brisk (around corners)	Normal (straight line)	50
	Backpack	Upside-down	Upright	Sideway	Normal (around corners)	Slow (straight line)	Brisk (straight line)	50
	Belt	Sideway	Upside-down	Upright	Brisk (straight line)	Normal (straight line)	Slow (around corners)	50
Jumping Phase	Front pocket	Upright	Sideway	Upside-down	Light hops	Medium hops	High jumps	50
	Back pocket	Upside-down	Upright	Sideway	High jumps	Light hops	Medium hops	50
	Jacket pocket	Sideway	Upside-down	Upright	Medium hops	High jumps	Light hops	50
	Hand-held	Upright	Sideway	Upside-down	Light hops	Medium hops	High jumps	50
	Backpack	Upside-down	Upright	Sideway	High jumps	Light hops	Medium hops	50
	Belt	Sideway	Upside-down	Upright	Medium hops	High jumps	Light hops	50

1.2 Data Transfer and Storage Format

After data collection, CSV files (comma-separated, decimal point) were exported from the Phyphox app via email and saved to a PC. Each member ensured proper file naming and organization before uploading their files to a shared Microsoft Teams workspace. Separate CSVs were maintained for walking and jumping to simplify downstream labeling and analysis.

1.3 Hardware and Software Used

- **Smartphone Accelerometers** (one per team member)
- **Phyphox App** (for collecting linear acceleration data without gravity)
- **Email Clients** (used for file transfer)
- **Personal Computers** (for data storage and analysis)
- **Microsoft Teams** (centralized data sharing and organization)
- **CSV Format** (for compatibility and standardization)

1.4 Challenges and Solutions

- **Sampling Consistency**: To avoid duplication or omission, the team followed the predefined sampling table (Table 1), which laid out all combinations of motion types, placements, and orientations per member.
- **Environmental Variability**: Some trials were completed indoors and others outdoors, leading to slight inconsistencies in terrain. These variations were kept to promote a more diverse dataset for generalization.
- **Participant Differences**: Variability in height, movement speed, and jumping style was noted across team members. These were recorded as possible margins of error but also added realism to the dataset.
- **Formatting Issues**: There were minor issues with CSV export settings (e.g., decimal point format), which were resolved by standardizing settings across all devices and verifying file formats before transfer.

By adhering strictly to the data sampling table and ensuring consistent protocols for labeling and transfer, the team successfully compiled a diverse, balanced, and well-organized dataset.

2. Data Storage

After collecting and labeling the accelerometer data in CSV format, the team structured and stored the dataset using the Hierarchical Data Format (HDF5) to ensure efficient organization, accessibility, and future scalability. The HDF5 file structure was designed to separate raw, pre-processed, and segmented data while preserving metadata embedded in the file names.

2.1 File Parsing and Metadata Extraction

Each CSV file followed a strict naming convention encoding key metadata components:
[MemberName]_[Action]_[Placement]_[Orientation]_[Variation].csv

A custom parsing function was developed to extract the following components:

- **Member Name**: Identifier for who collected the data.
- **Action**: 'walking' or 'jumping'.
- **Placement**: Position of the phone (e.g., backpack, front pocket).
- **Orientation**: Orientation of the phone (e.g., upright, sideways).
- **Variation**: Movement style or intensity (e.g., brisk, light hops).

This metadata was used to systematically organize the data within the HDF5 file structure.

2.2 HDF5 Structure

The file was named accelerometer_data.h5 and organized into three top-level groups:

- /Raw data: Contains raw sensor recordings.
- /Pre-processed data: Stores smoothed data using a moving average filter.
- /Segmented data: Contains data split into 5-second segments for training and testing.

Each data point was stored under a nested path that reflects its metadata as seen in Figure 1.

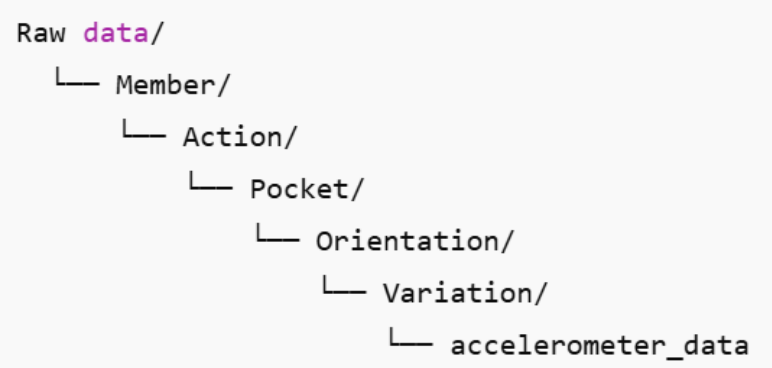

Figure 1: Nested Path Example.

2.3 Storage Process

1. **Raw CSV Upload**: All files were uploaded into a Python environment using the files.upload() function.
2. **Raw Data Storage**: Each file was read using pandas and stored in its corresponding hierarchical path using h5py.
3. **Pre-processing**: A simple moving average filter with a window size of 31 was applied to the x, y, and z acceleration channels. This smoothed version was saved in the /Pre-processed data branch under the same hierarchical structure.
4. **File Management**: Existing datasets were overwritten to prevent duplication. This was handled with a conditional check before dataset creation.

2.4 Sample Code Summary

The following Python functions were key to data storage:

- parse_filename(): Extracts structured metadata from file names.
- store_raw_data(): Reads and saves the original CSV data into the raw group.
- process_data(): Applies preprocessing and saves smoothed data into the corresponding group.

Finally, the complete HDF5 file was downloaded using files.download(hdf5_path) for submission and later use in visualization, feature extraction, and classification. This structured approach not only met the project's storage requirements but also enabled efficient data access for all downstream machine learning steps.

3. Visualization

Visualization was an essential step in understanding the patterns and variations present in the collected accelerometer data. It enabled the team to detect trends, distinguish between activities, and identify areas for improvement in both data collection and preprocessing.

3.1 Time-Series Acceleration Plots (X, Y, Z vs. Time)

To see how acceleration changes over time, time-series plots were created for each axis (X, Y, Z) using raw data from various combinations of participant, phone position, orientation, and activity. These plots were generated by reading raw accelerometer data from the HDF5 file and plotting the acceleration values against time for each axis.

Figure 2, Figure 3, and Figure 4 show examples of walking and jumping data for different users. Walking plots (e.g., Andrew - walking BackPocket Sideways Brisk and Daniel - walking BackPocket Upright Slow) show periodic, lower-amplitude oscillations, indicating more regular movement. In contrast, jumping plots (e.g., Morgan - jumping BackPocket UpsideDown HighJumps) reveal sharp, repetitive spikes—especially in the Y-axis—suggesting high vertical forces during lift-off and landing. These time-series plots helped visually confirm the differences in acceleration patterns between walking and jumping, and highlighted the influence of sensor placement and orientation on signal shape and amplitude. For instance, belt-mounted sensors produced different profiles compared to hand-held or pocket placements.

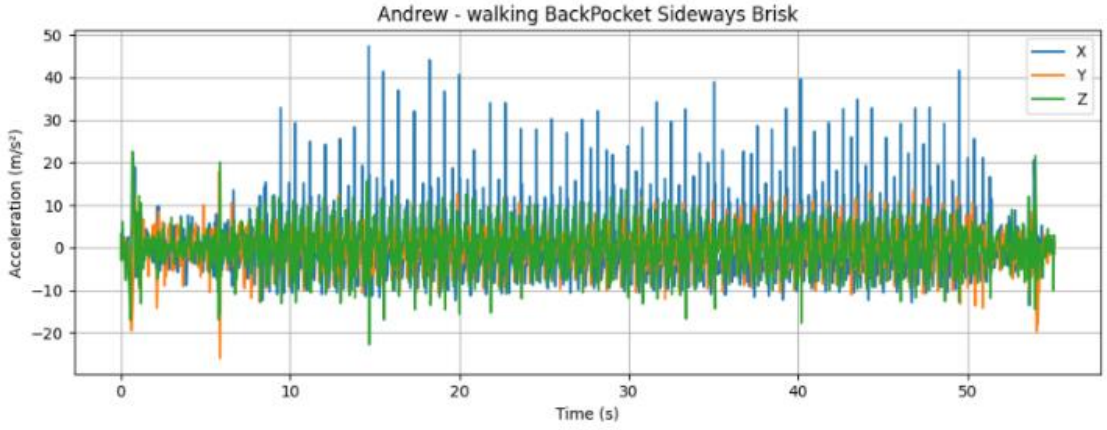

Figure 2: Andrew-Walking Back Pocket Sideways Brisk

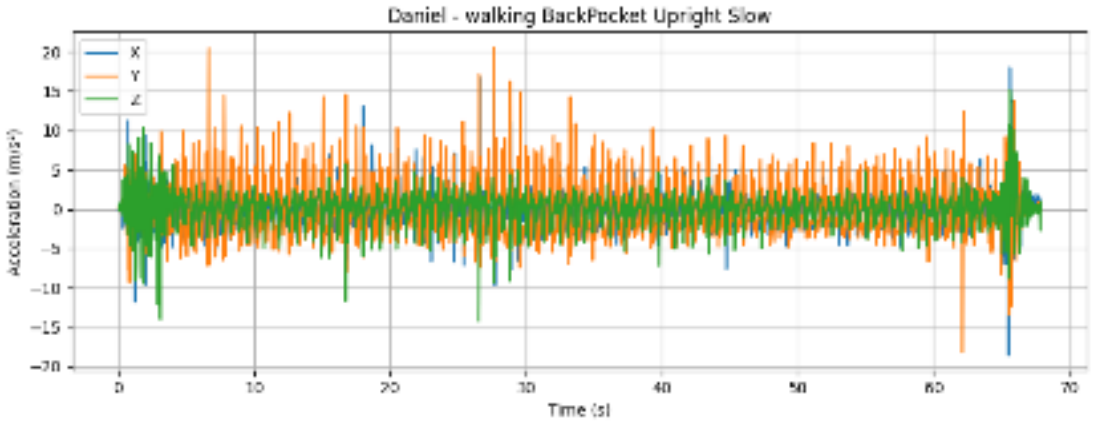

Figure 3: Daniel walking Back Pocket Upright Slow

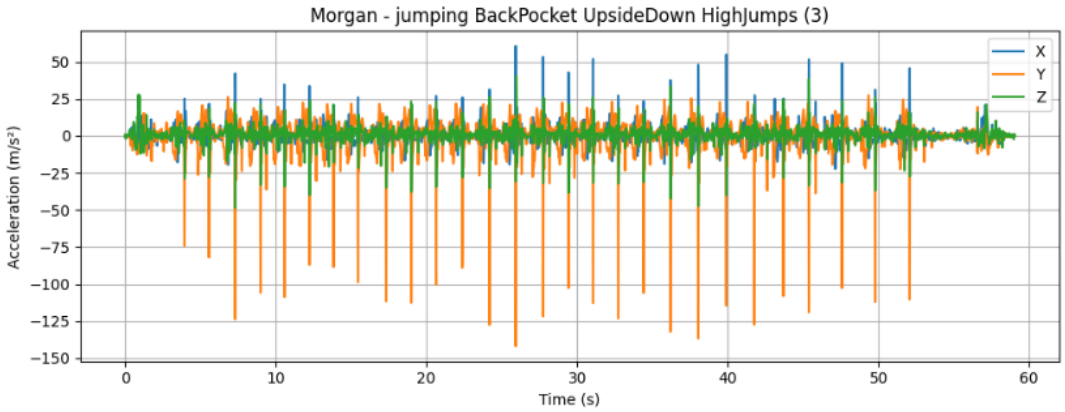

Figure 4: Morgan jumping Back Pocket Upside Down High Jumps

3.2 Histogram of Acceleration Magnitudes

To compare the overall force generated during each activity, a histogram of total acceleration magnitudes was created (Figure 5) The magnitude was computed using the Euclidean norm across the three axes, as seen in the equation below, for each time point.

$$a = \sqrt{x^2 + y^2 + z^2}$$

The histogram showed that walking data had a tighter, centralized distribution, typically peaking around 5–10 m/s². In contrast, jumping exhibited a wider, right-skewed distribution, with values in some samples reaching over 50 m/s². This analysis confirmed that jumping involves higher force magnitudes and variability, which is useful for separating the two classes during classification. It also reinforced the importance of this feature during feature extraction.

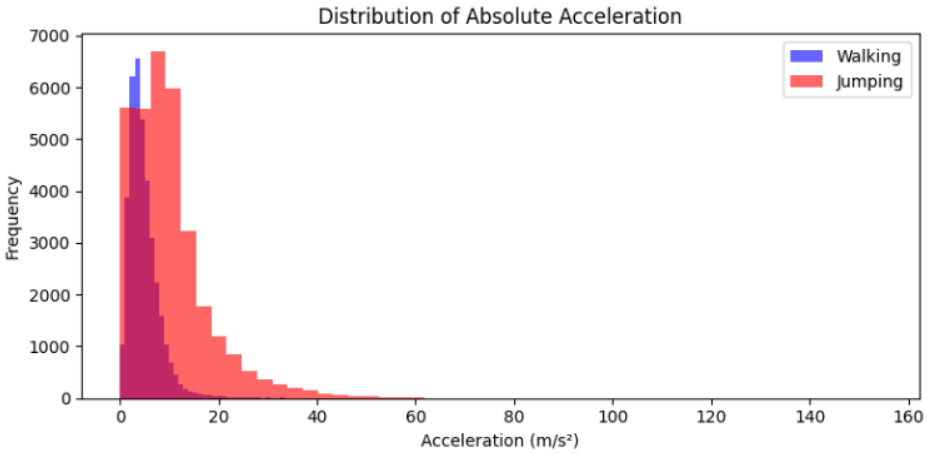

Figure 5: Distribution of Absolute Acceleration

3.3 Scatter Plot: Acc_X vs. Acc_Y

To analyze directional motion differences between walking and jumping, a 2D scatter plot was generated using X- and Y-axis accelerations (Figure 6) Walking data formed tight clusters, indicating consistent horizontal and vertical motion. Jumping data was more dispersed—especially along the Y-axis—reflecting greater variability in vertical acceleration.

This plot demonstrated that while there is some overlap in motion patterns (e.g., brisk walking vs. light hopping), the increased Y-axis spread in jumping can be exploited to improve classification performance. The scatter plot also suggested that adding a third axis (Z) or using 3D visualizations might better separate classes. Adding density contours or class-specific color gradients could further enhance interpretability.

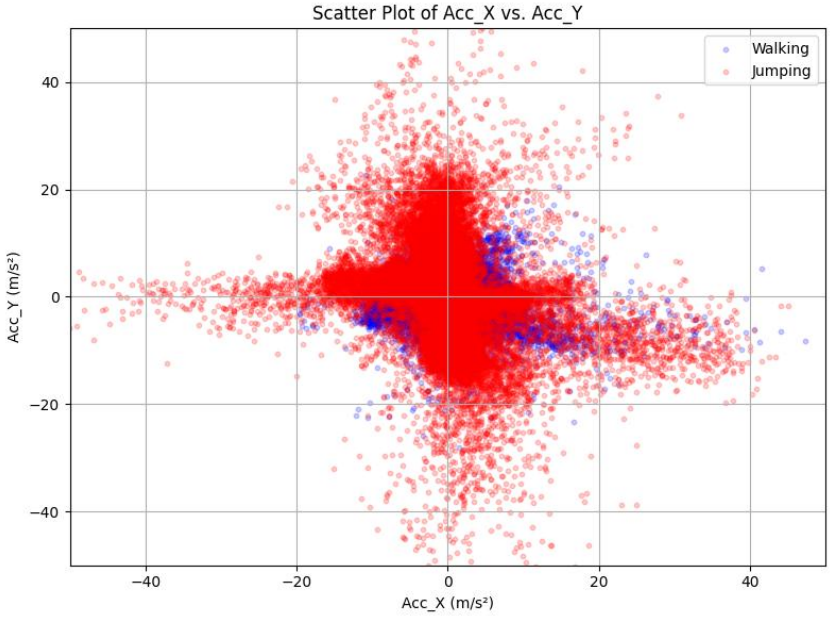

Figure 6: Scatter Plot ACC_X vs ACC_Y

3.4 Summary of Findings

From the visualizations, it became clear that jumping activities consistently produce higher-amplitude acceleration and more variation—especially in the vertical direction—than walking. Walking signals typically followed smoother, lower-frequency cycles. Sensor orientation and placement played a significant role in the observed signal characteristics, with sideways or loosely secured phones generating noisier data. Some anomalies—such as signal clipping or high-amplitude bursts—were likely caused by sudden impacts or instability during movement.

3.5 Reflections: Improvements for Future Data Collection

Several improvements were identified that could enhance the quality and reliability of data in future collection efforts. First, standardizing sensor mounting with straps or enclosures would reduce motion artifacts and help maintain consistent orientation. Logging orientation metadata automatically using onboard IMUs or gravity-based references would remove ambiguity when labeling data. Increasing the duration of each sample would provide more context and improve feature extraction reliability. Calibrating sensors before each session would help eliminate bias and ensure consistency across devices. Finally, balancing the number of samples across activities, users, and conditions would prevent class imbalance and help improve classification performance.

4. Preprocessing

4.1 Missing Value Handling

The first important step that was taken in preprocessing was addressing any missing values in the raw accelerometer data. A forward-fill function was applied which replaces any missing data points with the most recent valid value which ensures continuity. This method prevents any cuts/gaps in the signal that could create issues with future processing such as segmentation or feature extraction.

4.2 Signal Smoothing with Moving Average

One all the data was completed; the accelerometer signals were smoothed using a moving average filter. The filter was applied to all four axes (*Acc_X, Acc_Y, Acc_Z,* and *Absolute_Acc*). A window size of 25 samples was chose, as this created approximately 0.5 seconds of motion given the 50 Hz sampling rate. This is the best value as it reduces short-term noise without overly distorting the actual motion patterns in the signal. If smaller sizes like 5 or 10 were used, there would be very little visual impact. On the other hand, if too large a window size was used like 50, this would over-smooth the signal and obscure relevant trends.

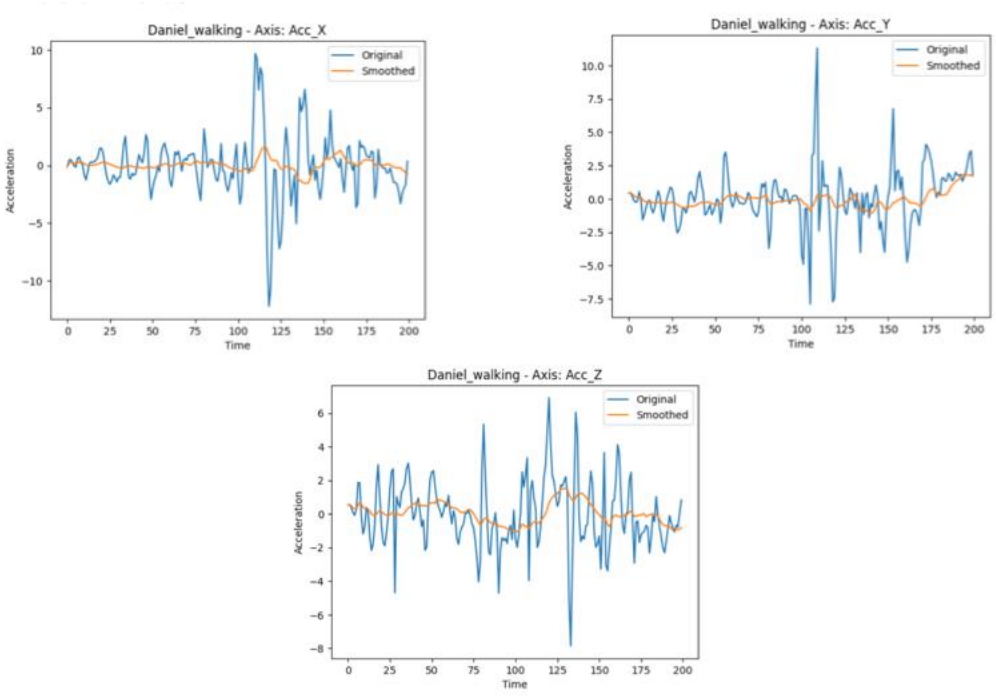

Figure 7: Comparison of Raw and Smoothed Accelerometer Signals for a Walking Sample

4.3 Impact of Processing

After preprocessing, the accelerometer signals were noticeably smoother and less noisy. Random spikes and fluctuations minimized which made the patterns in the data much more organized and easier to interpret. The smoothing helped the statistical features (mean, maximum, minimum, skewness, etc.) reflect the actual motion instead of any sudden fluctuations. As motion was required to start and pause the accelerometer, there are notable fluctuations that occur during the start and end of all datasets. In retrospect, preprocessing data could have been improved by clipping the first and last seconds of all test data. Nonetheless, the preprocessing step significantly improved the quality and reliability of the data used in later stages of the project.

Figure 8 : Clean Andrew Walking Back Pocket Sideways Brisk

Figure 9: Clean Daniel Walking Back Pocket Upright Slow

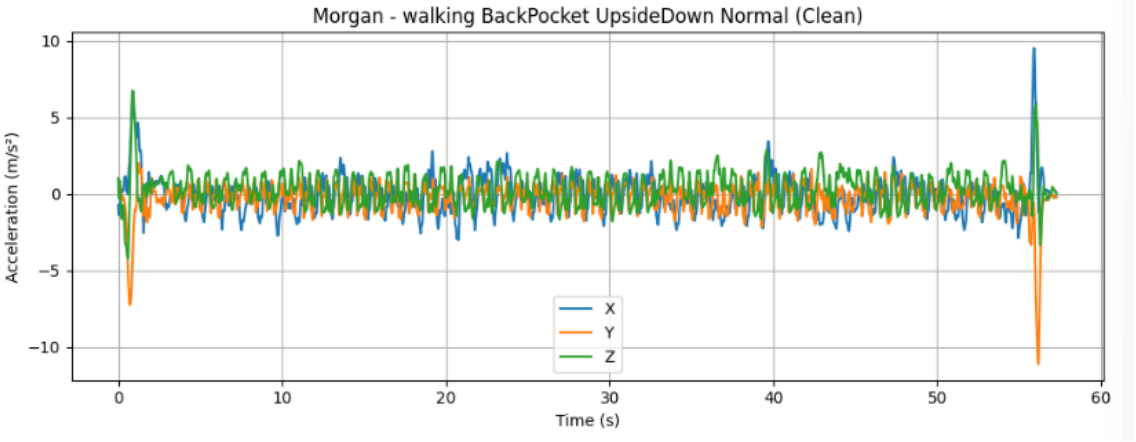

Figure 10: Clean Morgan walking back pocket upside down normal.

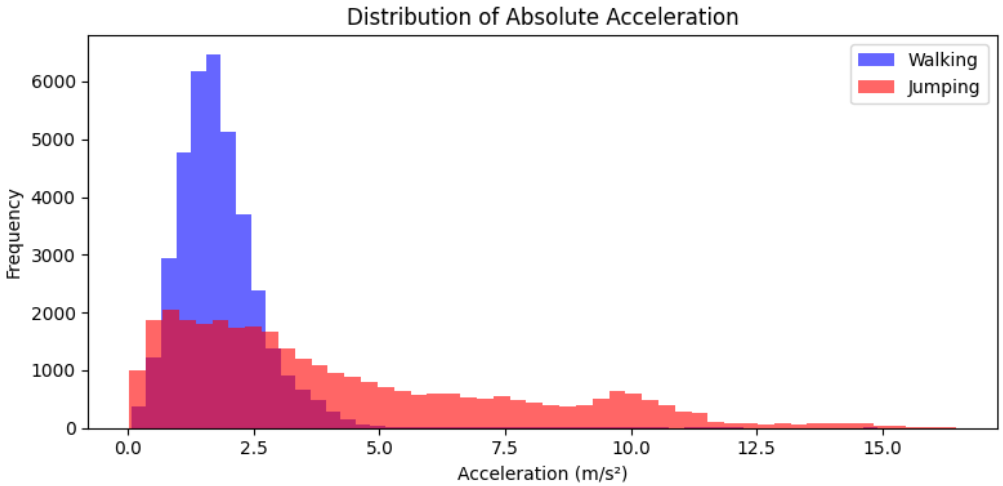

Figure 11: Clean Distribution of Absolute Acceleration.

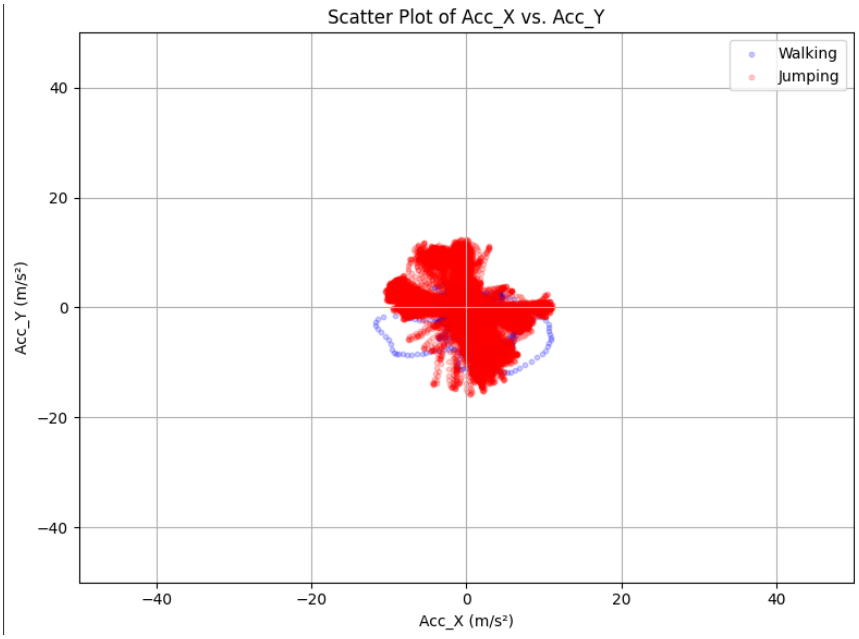

Figure 12: Clean Scatter Plot of ACC_X, vs. ACC_Y.

5. Feature Extraction and Normalization

5.1 Feature Extraction from Accelerometer Segments

Ten statistical features were extracted on a per-axis basis after segmenting the 5 seconds of raw accelerometer data into windows- Axis Acc_X, Acc_Y, Acc_Z, and absolutue_Acc. The features are maximum, minimum, median, variance, skewness, standard deviation, range, energy, and absolute mean. These features were defined considering the dynamic perspective required to have a statistical summary of the behaviour of the signal in each segment, capturing both the average and variation in the signal. For instance, mean, and median reflect average acceleration, spread or dynamic features are displayed through range, standard deviation, and variance. Skewness provides a hint at the symmetry of the signal, while energy shows the "intensity" of motion, allowing classification as higher "impact" (jumping) versus lower (walking).

5.2 Normalization for Scale Consistency

After feature extraction, z-score normalization was applied. This normalizing feature produces a mean value corresponding to the different features as 0 and standard deviations equal to 1. It is necessary since these raw features operate in different scales; for example, energy can be on the scale of hundreds, while standard deviation or skewness might be fractions. Hence, without normalization, such features might dominate the learning algorithm and spoil classification accuracy. Therefore, z-score normalization aims at the comparability of features about their treatment by the classifier.

6. Training and Testing the Classifier

6.1 Shuffling Testing Data

Recall the segmentation practices performed in the previous section, where all walking and jumping data is segmented and labelled according to their origin. The train_test_split_data function shuffles and designates 90% of the samples into a training set and the remaining 10% into a testing set. After both sets are normalized via StandardScaler, methods from sklearn (LogisticRegression, make_pipeline, fit) are then used to train the logistic regression model and develop a pipeline. Afterwards, both datasets were stored into the HDF5 file as Segmented data/Train/x_train, Segmented data/Train/y_train, Segmented data/Test/x_test, and Segmented data/Train/y_test.

6.2 Applying Evaluation Metrics

Training from sklearn is performed via binary classification. The model analyzes patterns from the segmented training set to determine chance the probability of input data being discrete 'walking' (0, negative) or 'jumping' (1, positive) data. The discrete option with the highest probability is used to label the data.

A Confusion Matrix (CM) and Receiver Operating Characteristic (ROC) curve were used to analyze the classification model's accuracy. Individual data from the testing set was run through the model to be classified. The original label from the test data was then compared with the classification label provided by the model to output True Positive, True Negative, False Positive, and False Negative values. Table 3 displays each discrete result, what they translate to, and the correctness of the classification model.

Table 3:Discreat result and correctness of classification model

Output	Discrete Result	Translation	Outcome
True Positive	1, 1	Wanted: Jumping Classified as: Jumping	Correct
True Negative	1, 0	Wanted: Jumping Classified as: Walking	Incorrect
False Positive	0, 1	Wanted: Walking Classified as: Jumping	Incorrect
False Negative	0, 0	Wanted: Walking Classified as: Walking	Correct

The occurrences of each discrete result were plotted on the CM, with the rates of True Positive and False Positive results being used to design the ROC (Figure 15). As per procedures outlined in ELEC 292 course material, the Area Under the ROC Curve (AUC) was calculated to analyze the accuracy of the trained classification model.

With an AUC of 0.77, the classification model falls under an acceptable 0.7-0.8 range and thereby discriminates between both labels quite accurately. This result is further enhanced by the accuracy and recall results obtained by sklearn (which analyzes how the model fares against easy and unpredictable input data)

```
Classification Accuracy:  0.6931818181818182
Classification Recall:  0.5952380952380952
the AUC is:  0.7701863354037267
```

Figure 13: Classification performance metrics — overall accuracy of 69.3% and recall of 59.5% for the binary walking/jumping classifier.

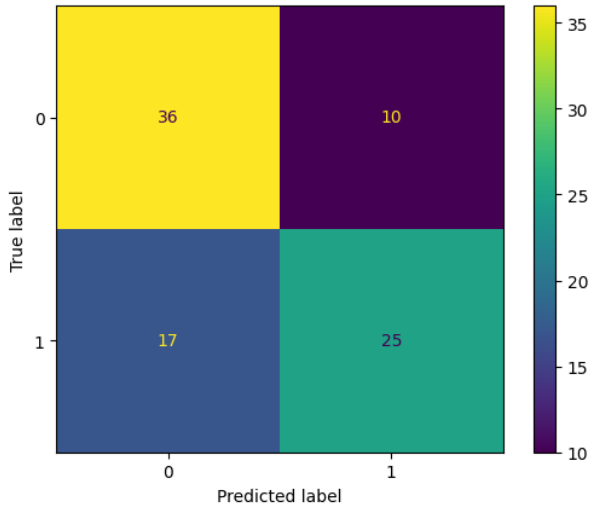

Figure 14: Confusion matrix showing true and predicted labels for walking (0) and jumping (1) activities. The model correctly classified 36 walking and 25 jumping instances, with 27 total misclassifications.

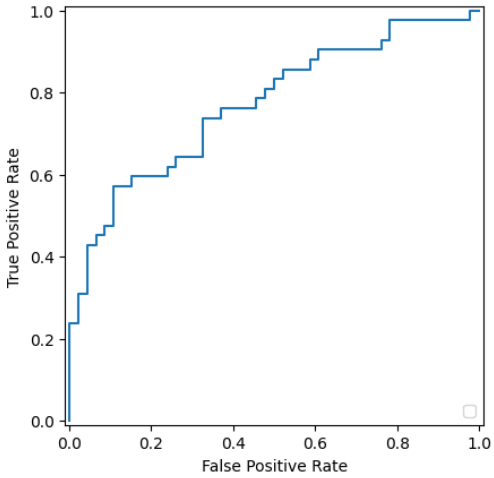

Figure 15:Receiver Operating Characteristic (ROC) curve for the classifier, illustrating the trade-off between true positive rate and false positive rate across different thresholds.

6.3 Classifying External Data

The following section underlines how external data is classified using the logistic regression mode. Data is to be pre-processed, normalized, and segmented. Much like the data from the testing set, labels for each divided segment are predicted using the model and compiled into a list. The program takes the more prominent label to determine the overall data.

```
[1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 1]
I think this is jumping data!
```

Figure 16: Output from the classification system displaying individual segment predictions (1 = jumping, 0 = walking) followed by the final decision based on majority vote.

7. Model deployment

7.1 Model Extraction via Pickle

As the Team's classification model was designed in Google Colab notebook, a form of Model extraction was required to design the Graphical User Interface (GUI). Pickle, an external library, was used to export the Logistic Regression Model as a pkl file. The file downloaded from the

Colab (accelerometer_model.pkl) was then inserted into the files of a separate PyCharm project that contained GUI. Other files (i.e. ROC data and the resultant HDF5 file) were also extracted for reference.

```
# save the model
import pickle
with open('accelerometer_model.pkl', 'wb') as f:
  pickle.dump(clf, f, protocol=pickle.HIGHEST_PROTOCOL)
```

Figure 17: Python snippet used to serialize and save the trained logistic regression model using the pickle module for future use in the GUI application

7.2 Resulting Graphical User Interface

An image of the Team's base GUI (designed with PyQt5, matplotlib, pandas, and numpy) is shown in the Figure below:

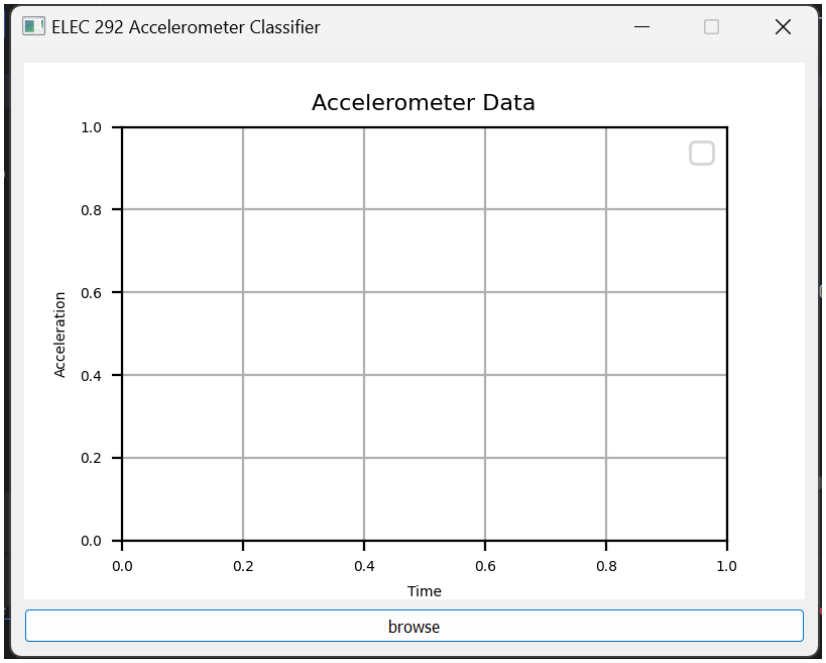

Figure 18: Initial state of the ELEC 292 accelerometer classifier GUI, awaiting user input to load and visualize CSV accelerometer data.

The design of the GUI simply consists of a matplotlib graph and a browse button. When the user clicks the browse button, they are prompted to provide a csv file to analyze:

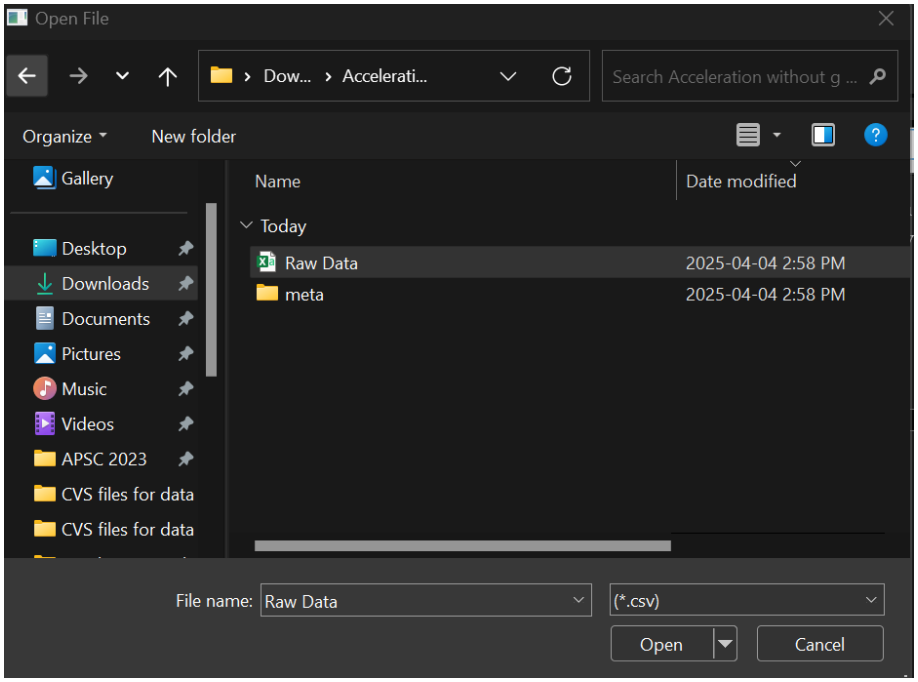

Figure 19: File dialog window prompting the user to select a CSV file containing raw accelerometer data to be visualized and classified within the GUI.

When a csv is selected, matplotlib is used to display the resultant accelerometer graph. Note that the QFileDialog.getOpenFileName method assists in retrieving the name and direction of the csv file. Common practices from numpy and pandas were then used to format data for matplotlib.

It is important to consider that all data from the csv file undergoes all preprocessing procedures demonstrated for model training (i.e. preprocessing and feature extraction) to ensure that the data is biased correctly. The resultant label for the classified data is shown the title for the graph. Figure AAA displays the results of walking and jumping csv files that were used in the demo video and not used in the original dataset.

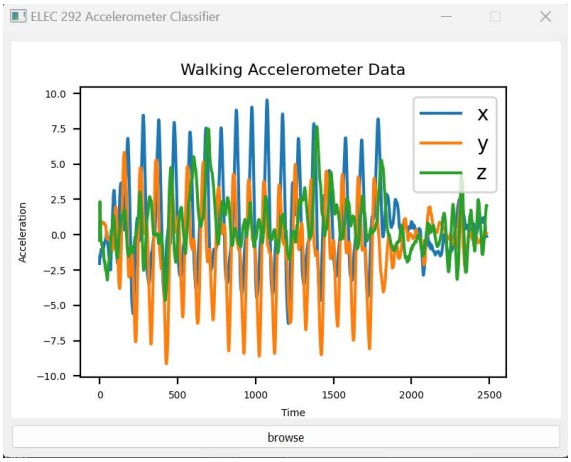

Figure 20: Example plots of raw X, Y, and Z accelerometer data as visualized in the GUI. Walking data with smoother, periodic signals.

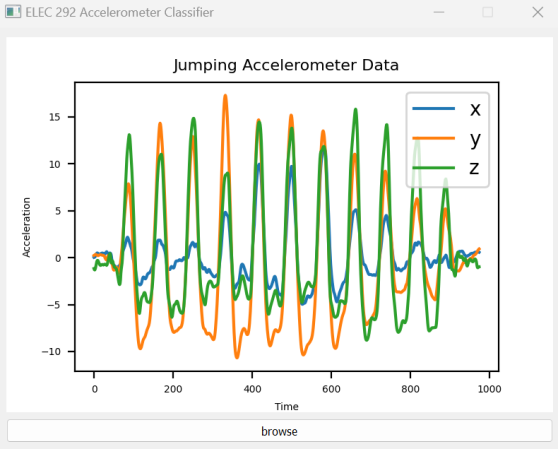

Figure 21: Example plots of raw X, Y, and Z accelerometer data as visualized in the GUI. Shows jumping data with larger, higher-amplitude spikes.

7.3 Future Design Considerations

With the simplicity of the GUI in mind, there remains a few ways to improve the classification app for the user. Extracted features from the original dataset could be displayed on a separate window and back-end classification results could also be shown to justify the outcome for labelling. To a similar extent, the GUI could have been redesigned to track and classify live PhyPhox data.

Participation Report

Task	Description of Activity	Team Member
Data Collection (M1)	Created a chart guide that walked the team through how to collect accelerometer data using Phyphox app. Made sure had consistent protocols for phone placement, orientation, and movement type. I also participated in the actual data collection by completing 12 trials- 6 for walking and 6 for jumping- based on the standardized sampling table. I followed all the assigned combinations of placement and orientation to help ensure we had a well-balanced dataset.	Morgan
Data Collection (M3)	Adhered to the outline for data collection and used Phyphox to collect walking and jumping csv files.	Andrew
Data Storage	After collecting my data, I made sure each CSV file was correctly labeled and formatted following the team's naming convention. I then uploaded everything to our shared Microsoft Teams folder. I also worked on parsing the filenames to extract metadata like phone placement and activity type, which helped with organizing the data in the HDF5 structure. This made it easier to manage and use later for preprocessing and segmentation.	Morgan
Data storage	I created the Python script that generated the HDF5 file structure used to store our accelerometer dataset. I started by writing a function to parse each file name and extract metadata like member name, action, phone placement, orientation, and movement variation. I used this information to organize the data hierarchically within the HDF5 file under /Raw data, /Pre-processed data, and /Segmented data.	Morgan

	For raw data storage, I used pandas to read each CSV file and h5py to store the acceleration data in the appropriate path, making sure each dataset had properly labeled attributes like column names. I included checks to avoid duplicate entries and ensure the data was complete. I also implemented segmentation logic, dividing the raw acceleration data into 5-second windows to prepare it for machine learning. I then used a train_test_split function to split the dataset into training and testing groups, which were saved into the /Segmented data/Train and /Segmented data/Test branches of the HDF5 file. Throughout the process, I made sure that all file uploads, processing, and saving steps were automated and verified. By handling both the code and execution for this task, I helped ensure the data was cleanly structured and ready for analysis.	
Pre-Processing (debugging)	Assisted in revising the code for Part 5, utilizing pre-processing techniques that were introduced for the lab.	Andrew
Data Visualization	I generated and analyzed time-series plots, histograms, and scatter plots based on my collected data. I looked at how different placements and orientations affected the signal and identified patterns that helped distinguish walking from jumping. I also reflected on ways we could improve future data collection—like using straps for consistent phone placement and balancing the number of samples across all categories.	Morgan
Demo Video Production	I created the demo video for our project. I recorded the footage, edited	Morgan

	the video, and uploaded the final version. The video walked through how we collected the data and showed the experimental setup, helping explain our process in a clear and engaging way.	
Written Documentation and Report Editing	Introduction, Data Collection, Data Storage, and Visualization parts. For each section, I focused on writing clearly and technically, making sure that the methodology, reasoning, and results were easy to follow and well-supported by evidence. In addition to writing these sections, I also revised and edited the full document to improve flow, correct grammar, and maintain a consistent voice across all contributors.	Morgan
Work on Part 6	Used segmented test/train data from part 5 to develop a classification model via sklearn. Assisted in importing the classifier into an external program on PyCharm with Pickle. Tested the model with walking and jumping csv files and evaluated the success of the model with a CM and ROC AUC. Documented procedure and results in report.	Andrew
Work on Part 7	Imported pickle model from Google Colab. Designed a simple GUI using PyQt5 that pre-processes, graphs, and classifies accelerometer data. Provided a brief description of how the GUI functions and what could be improved for future reference.	Andrew
Work on Part 4 (Code and Report)	Preprocessed the raw accelerometer data to make it cleaner and more consistent. Filled any missing values using forward fill to ensure continuity in the signal. Then applied a moving average filter to smooth out short-term noise without losing the overall motion trend. Wrote the summary and explanation of the process and a outcome of this part on the report.	Daniel

Work on Part 5 (Code and report)	Extracted 10 statistical features from each 5-second segment of preprocessed data for four axes: Acc_x, Acc_Y, Acc_Z, and Absolute_Acc. These features included mean, standard deviation, minimum, maximum, median, range, variance, skewness, energy, and absolute mean resulting in 40 features per segment. After extracting these features, z-score normalization was applied to scale them to a mean of 0 and standard deviation of 1. Wrote the summary and explanation of the process and a outcome of this part on the report.	Daniel