ELEC 390 PROJECT REPORT

Group 6

APRIL 14, 2023

MILE STOSIC (20233349), JEEVAN THIRUCHITTAMPALAM (20199456), & CHRISTOPHER SEGUIN BIANCHI (20217241)
EMAILS: 20mcs@queenu.ca, 19jt20@queensu.ca, and 19cjsb@queensu.ca

Abstract

The purpose of this project is to create a desktop application that takes accelerometer data from a mobile application (Phyphox) and can distinguish the difference between when a user is walking or jumping. The application outputs a new .csv file that contains the new output data that labels specific windows of space with ‘walking’ or ‘jumping’.

A seven-step process was followed in creating this application. The first step was *Data Collection*. For accurate training, each member of group 6 jumped and walked for three minutes each; with each member holding their phone in different locations (back pocket, front pocket, in hand). Once data was collected, an HDF5 file was created to sort the data in an organized matter. The data signals from Step 1 were divided into 5-second windows and were shuffled before being put into two data sets (Train and Test). The training data included 90% of the data, and the remaining 10% went to the Test dataset.

This organized information was then visualized into multiple plots. The first set of plots shows a linear acceleration on each of the axis’s x, y, and z, for each point of time; one for walking and one for jumping. These graphs were difficult to interpret, so another set of 2d graphs using absolute acceleration was created for each member’s data. The general trend was that the absolute acceleration had a higher range for jumping than walking (as expected).

The next three steps, *Pre Processing, Feature Extraction,* and *Training the Classifier* were all connected. Since all datasets have ‘noise’ or outliers in their data, a moving average filter was used to eliminate as much noise as possible. Z-score Normalizations were also used to get rid of outliers. *Feature Extraction* involved retrieving features such as max/min values, range, mean, medians, etc. for each ten-second window from Step 2 and Step 4. To train the Classifier in step 6, a logistic regression model was used with the Preprocessed features. The two sets, train, and test were combined with the help of the required libraries to create this linear regression model. The model was then fitted to the standardized training data using a pipeline. The model was trained using 100,000 iterations as the max number of iterations, allowing the model to converge to an accurate solution. Learning curves and model accuracy were obtained using plots for test and training set loss. The performance of the classifier was calculated using F1 scores, accuracy, recall, and AUC values.

The model deployment and GUI can be found in the *Model Deployment* section of the report.

Overall, the desktop application calculated whether the user was walking or jumping with an accuracy of 93%. The classifier training was a success. The main challenges that were faced included user stamina for jumping for 3 minutes straight and time management, as each member had multiple assignments and tests throughout the semester, making it difficult for the group to meet up and give progress updates.

Table of Contents

Table of Figures

Project

The goal of the project is to build a desktop app that can distinguish between ‘walking’ and ‘jumping’ with reasonable accuracy, using the data collected from the accelerometers of a smartphone.

Description

The project involves building a small and simple desktop application that accepts an accelerometer
data (x, y, and z axes) in CSV format, and writes the outputs into a separate CSV file. The output CSV file contains the labels (‘walking’ or ‘jumping’) for the corresponding input data. For classification purposes, the system will use a simple classifier, i.e., logistic regression.
To accomplish the goal of the final project and complete the report, the following 7 steps are required:
1. Data collection
2. Data storing
3. Visualization
4. Pre-processing
5. Feature extraction
6. Training the model
7. Creating a simple desktop application with a simple UI that shows the output.

Data Collection

The Data used for this project was collected using Phyphox, a measurement software that uses sensors in a phone to gather data for experiments. The team decided that everyone would gather three minutes of data for both jumping and walking. To maximize diversity in the data, Chris would place his phone in his back pocket during collection, while Jeevan would place his phone in his hand and Mile would put his phone in his front pocket. The data was transferred to a computer via email, as Phyphox can export data as CSV files as attachments.

One of the challenges faced with data collection was subject stamina. It was difficult to jump straight for three minutes, and the jump rate likely decreased from the start through to the end of the trial. We overcame this by taking multiple collection attempts. This challenge was quite minor and should have little to no effect on the product.

Data Storing

The data was stored in an HDF5 file via Python script. The file was organized by storing each team member's data under a group with their respective name after the root, and a separate group called dataset. The dataset itself contains two datasets, Train, and Test. Each signal from the data was divided into 5-second windows and shuffled and inserted into the Train and Tests datasets. The Train dataset contained 90% of the shuffled data, while the Test dataset contained the remaining 10%.

Visualization Samples

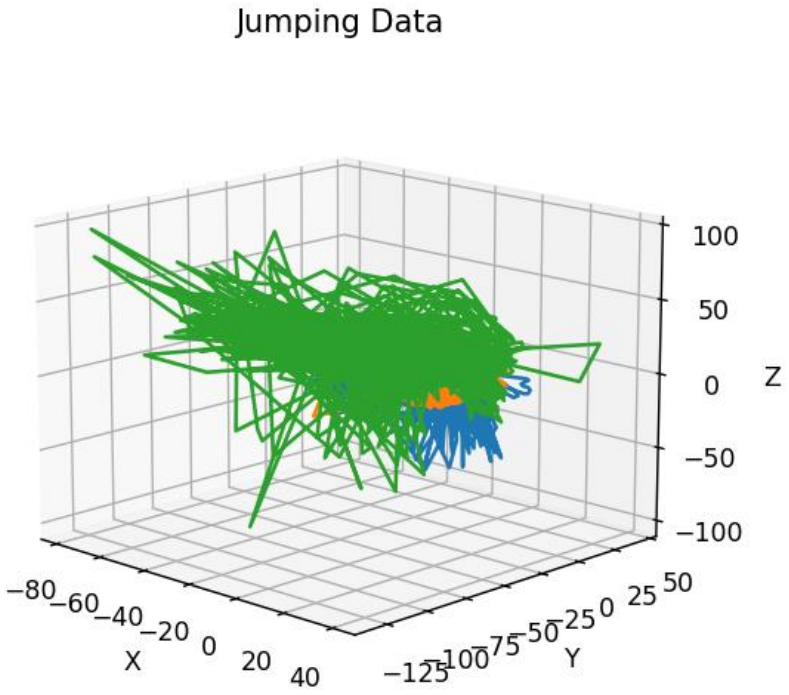

Figure 1: Jumping Data of all group members plotted on a 3D plot.

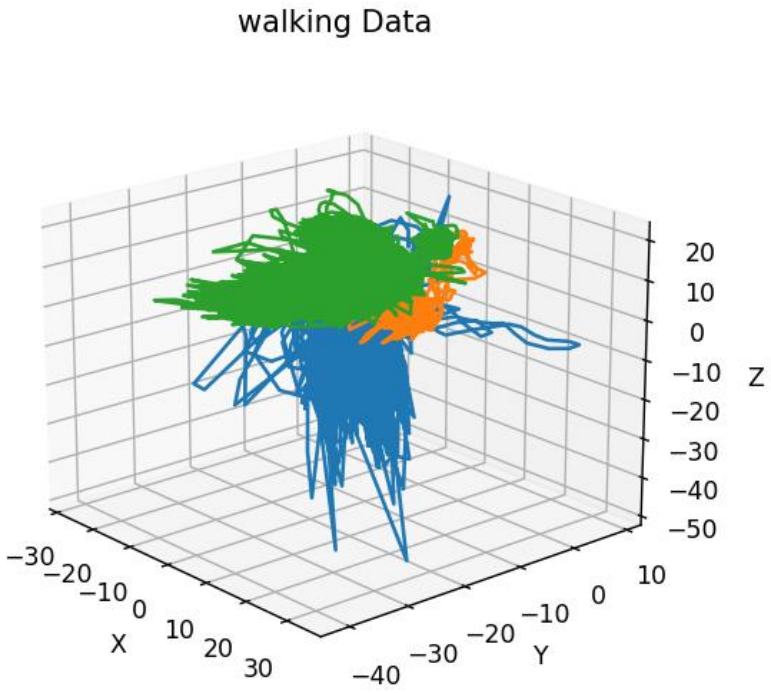

Figure 2: Walking Data of all group members plotted on a 3D plot.

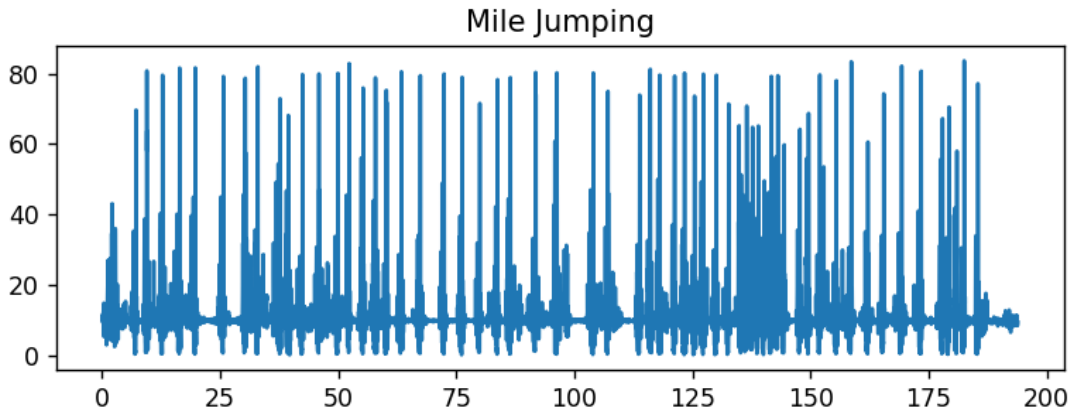

Figure 3: Accelerometer vs Jumping data points for Mile jumping.

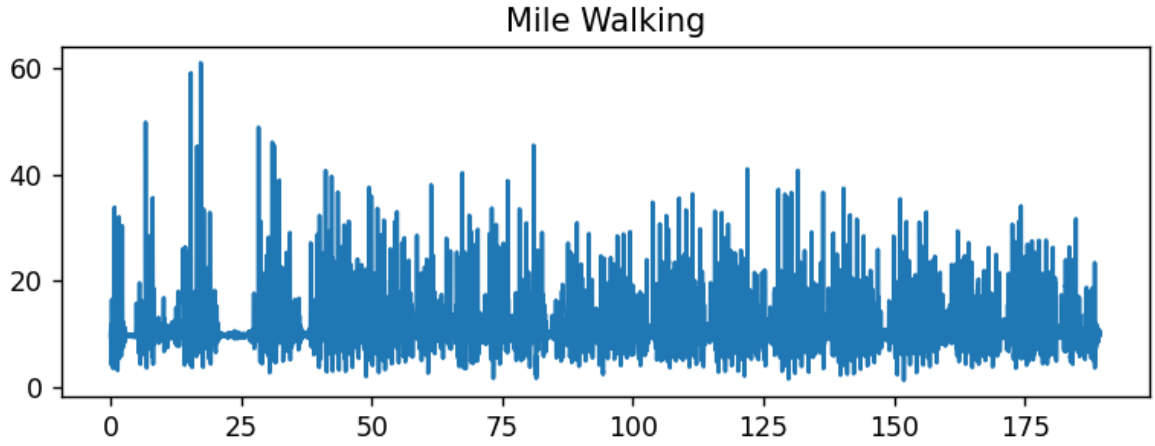

Figure 4: Accelerometer vs Jumping data points for Mile walking.

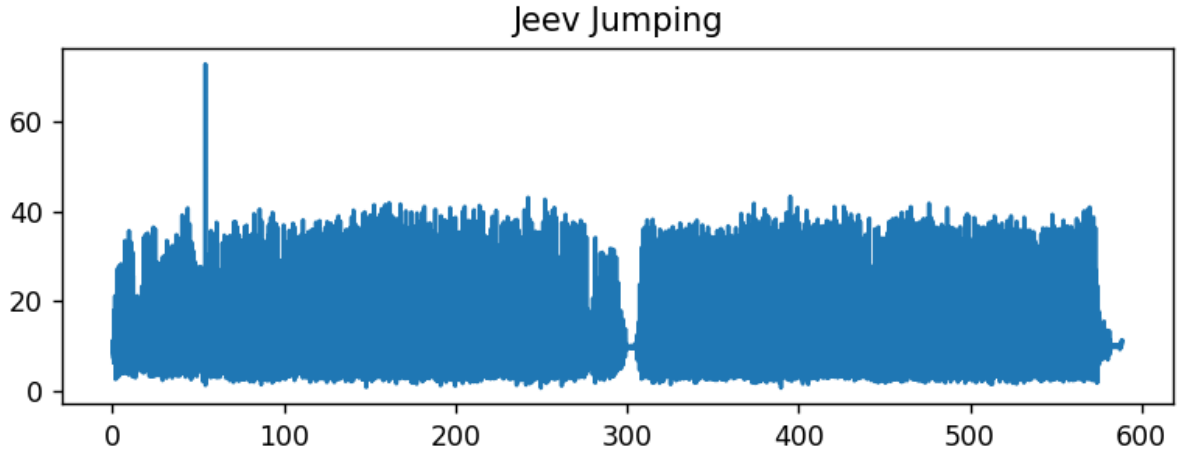

Figure 5: Accelerometer vs Jumping data points for Jeev jumping.

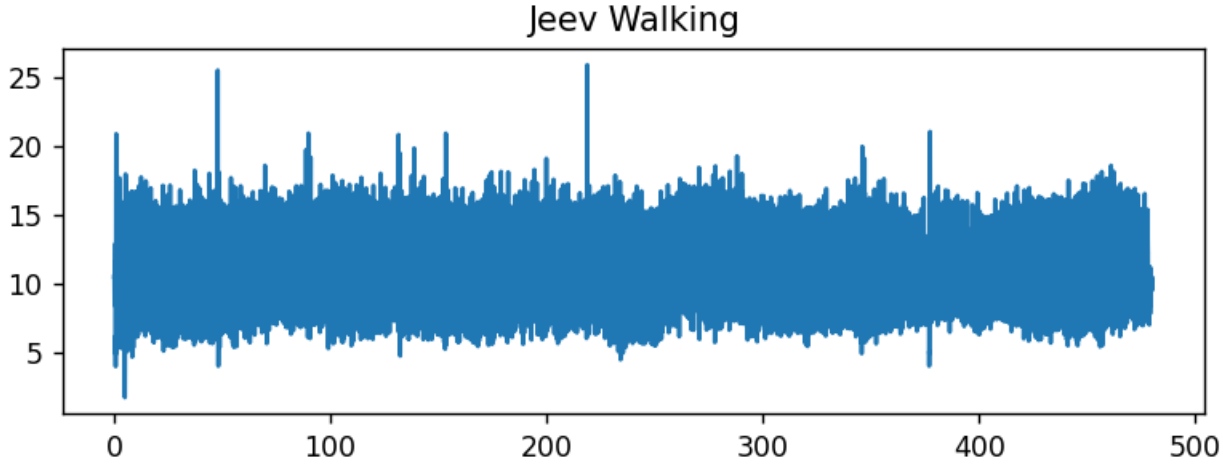

Figure 6: Accelerometer vs Jumping data points for Jeev walking.

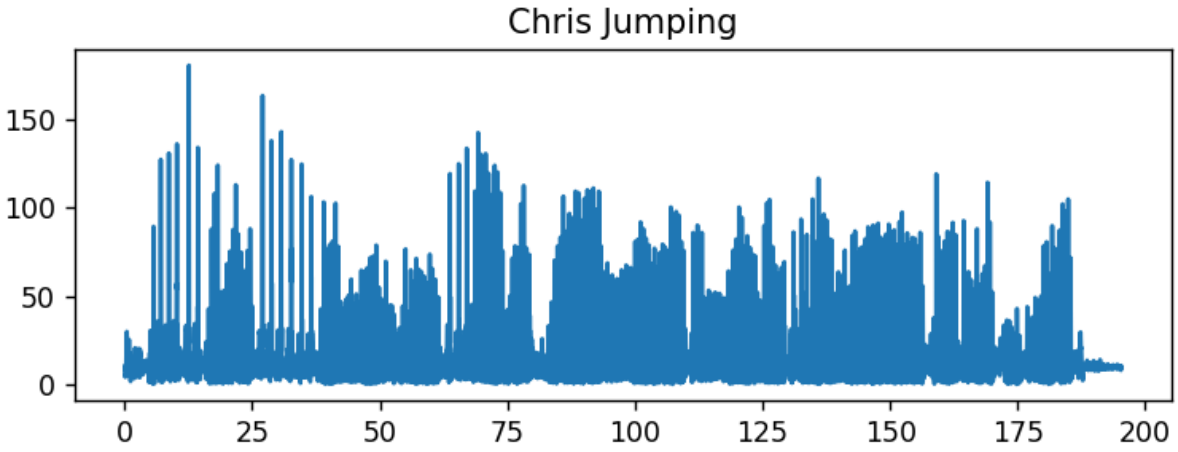

Figure 7:: Accelerometer vs Jumping data points for Chris Walking.

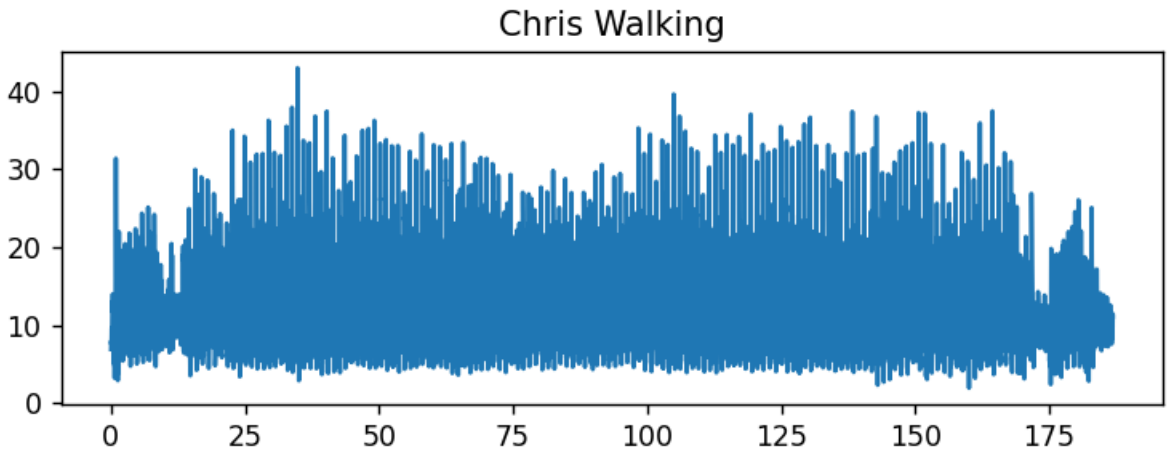

Figure 8: Accelerometer vs Jumping data points for Chris Walking.

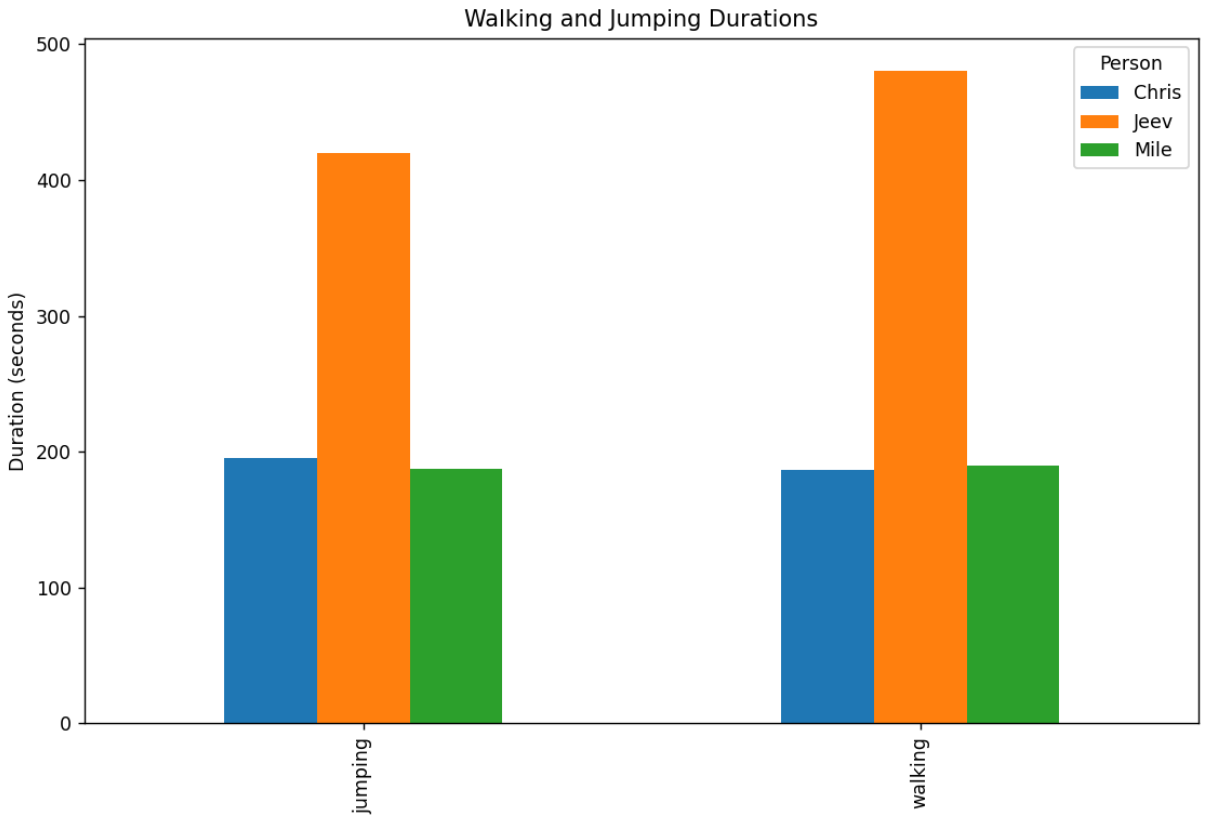

Figure 9: Duration of Experiment vs System Time Metadata plots.

Visualization Analysis

Figures 1 and 2, Jumping Data and Walking Data, show all the jumping and walking data points respectively. The axes show the acceleration in the X, Y, and Z directions. Each point represents a different instance of time, and so all points are connected. The team found these graphs were quite abstract and difficult to interpret. This led to the creation of a second set of graphs (Figures 3-8). These graphs were made from either walking or jumping (for each team member) data and plotted the absolute acceleration versus time. These graphs were much easier to understand, and we can clearly see distinctions between walking and jumping data. It seems that jumping data reaches far higher maximums and changes acceleration at a much quicker rate. It is also choppier, as there are greater breaks between maximums and minimums. Furthermore, there is more variance in the pattern.

Figure 9 shows the durations of each jumping and walking test for each team member. Jeev's times were much higher, as he collected a larger sample of data. Chris and Mile's times were the same. This actually led to later challenges when we were training the model, as the code would not run properly unless all the testing data was of the same length. As mentioned later in the report, we solved this issue by trimming all data samples to the smallest value.

Preprocessing

In the Pre-Processing step, the main goal was to reduce noise and remove outliers to improve the data for when the classifier was to be trained. First, a moving average filter was implemented to smooth the data and reduce noise the overall high-frequency noise. The chosen window size of 5 was used as a balance between keeping the signal's essential attributes and reducing the high-frequency noise. The filter was applied through a rolling window on each data frame's respective windows. The filtered data was then used to extract features from feature extraction step 5, which was arranged into separate data frames.

Z-score normalization was applied to the features obtained to remove outliers and normalize the data. For each feature, the mean was subtracted, and the returned result was divided by the standard deviation calculated before. Any data points with Z-scores greater than 3 or smaller than -3 were considered outliers and replaced with the average of the data points that remained. This step ensured that the model would be trained on conditions that had the most characteristics.

Finally, the features from each channel were concatenated into a single data frame for further analysis. To visualize the preprocessed data, two sets of plots were created, showing the first five and the next five features, respectively. This visualization helps provide analysis of the data after it was processed.

By applying these steps to our data, the data sets were washed, and the noise was reduced, making it optimal and efficient for the logistic regression. The chosen parameters aimed to balance the preservation of essential signal characteristics while mitigating the impact of noise and outliers on the model's performance.

Feature Extraction

The features extracted from the processed data from step 3, aimed to obtain useful characteristics that would help in differentiating between walking and jumping. Ten features were extracted for each of the three accelerometer axes and the total acceleration, they were the max value, min value, range, mean value, median value, variance, skewness, root mean square (RMS), kurtosis, and the standard deviation.

The features were calculated using the function we implemented into the code called 'compute_features', based on the given window data it calculated the corresponding ten features listed above. The 'extract_train_features' and 'extract_test_features' functions were then implemented to extract the features from the train and test obtained in the previous steps. These functions loop through each window of the data and calculate the corresponding features using the 'compute features' function from previously.

The features that were extracted were chosen based on their ability to provide a comprehensive understanding of the characteristics. Features like max, min, and range values can capture the amplitude information, while mean, median, and standard deviation can provide insights into central tendency and dispersion. Skewness, kurtosis, and variance provide additional information about the data's distribution shape, and RMS can indicate the signal's energy.

These features were chosen as they have been used in previous research studies based in activity recognition and have shown promising results in displaying relevant information from accelerometer data (Boa & Intille, 2004). Activity recognition from user-annotated acceleration data. In Pervasive Computing (pp. 1-17). Springer, Berlin, Heidelberg.). The feature extraction process was performed separately for each axis, allowing the model to capture activity-specific information in all three dimensions and provide a precise and accurate trained model that will be implemented in future steps to deploy our application.

Training the Classifier

In this step, a logistic regression model was trained to classify accelerometer data into 'walking' and 'jumping' classes using the preprocessed features obtained from previous steps. The process begins by importing the required libraries and preprocessed datasets obtained, separating them into train and test sets, and extracting the feature data and labels. Then, the train and test sets are combined, and a logistic regression model is created and fitted to the standardized training data using a pipeline.

The model was trained using 10,000 iterations as the maximum number of iterations, which provides ample opportunity for the model to converge to a solution that is accurate. The learning curves and accuracy of the model on the training and test sets are not shown explicitly in the code but can be obtained using additional tools, such as plotting the training and test loss during the training process. The chosen parameters, including the maximum number of iterations, were set to balance computational complexity and model performance. The accuracy, recall, F1 score, and AUC provide a comprehensive evaluation of the classifier's performance.

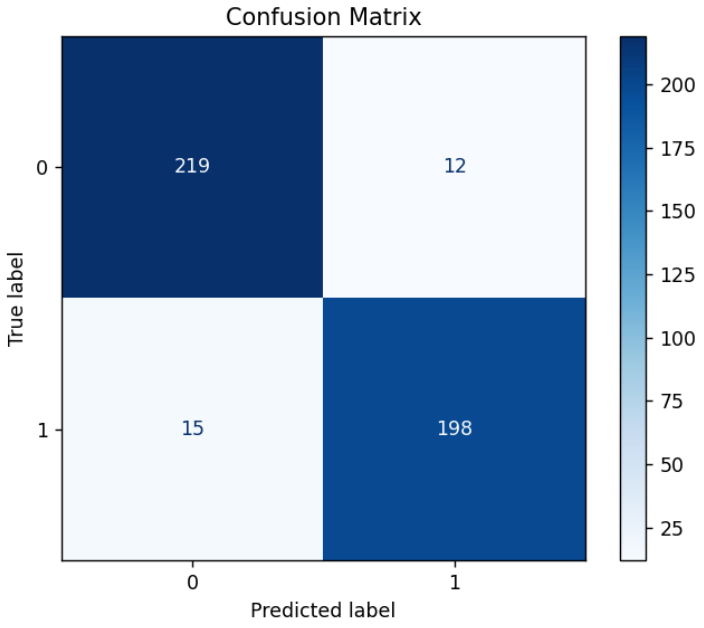

Figure 10: Confusion of accuracy.

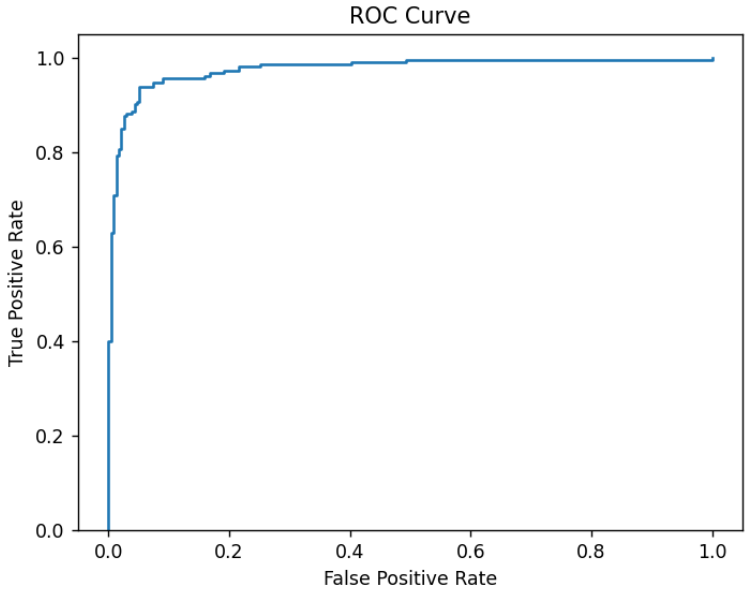

Figure 11: ROC curve of the trained classifier.

```
Accuracy of the model is: 0.9321891891891891.
Recall of the model is: 0.9295774647887324.
F1 Score: 0.9419354838709677
AUC: 0.9763632298843568
```

Model Deployment

We deployed our trained model into a desktop app using Python's Tkinter library to build a simple graphical user interface (GUI). The GUI design features a logo, a title, and several buttons for selecting input files, submitting out file names, showing the plots, and clearing the form. We chose Tkinter to deploy are application for its simple capabilities and its cross-platform possibilities, making it reach out to a larger audience. The process of deploying the model involved the following steps, importing the libraires, defining functions compute the features and extract data from the two datasets. Next, we created a function called processData() that processes the data from the input file and extracts the features using the previous function. Using the processed data, it was then run through the trained classifier model we created using the previous steps in the project to predict what activity was being performed. Next, we used sample data in CSV format and input the data into the desktop application. The application then processed the data as expected and generated an output CSV file with the correct labels (walking or jumping) for each window for 5 seconds. We then displayed the results in a plot and confirmed that the results matched our expectations of the activity being performed. This test demonstrates that the system works as intended and can successfully distinguish between individual walking and jumping activities. The GUI, output plot, and CSV output file of the tested CSV files can be seen below in the following figures.

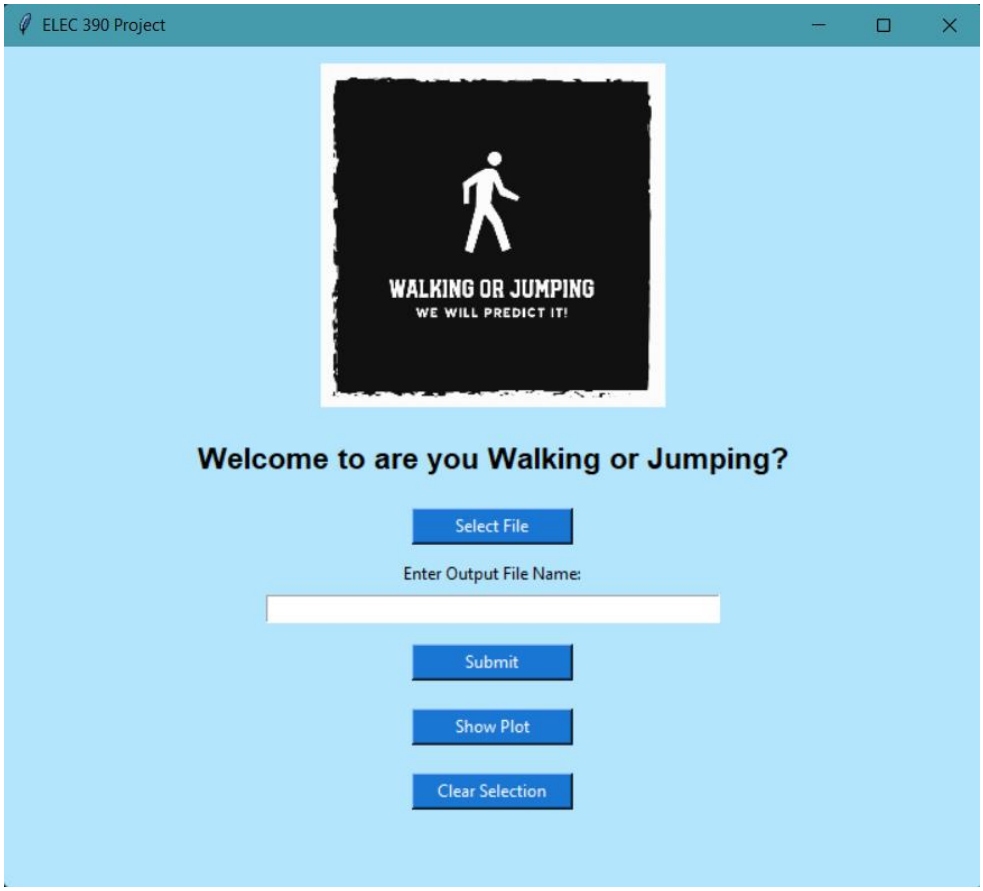

Figure 1212: Image of the design and visual aspect of the User Interface.

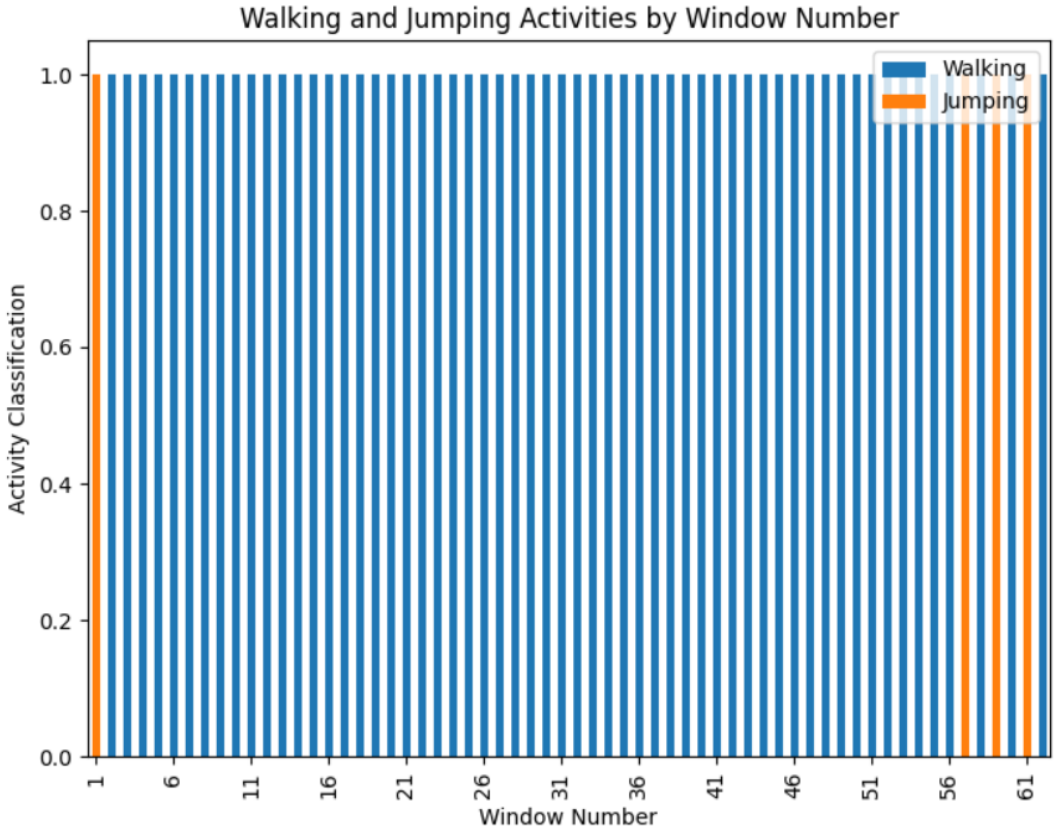

Figure 1313: Final graph showing the final predictions at the given window size.

Test3

```
activity,start_time
jumping,0.001161499997
walking,5.02023025
walking,10.03928875
walking,15.05832612
walking,20.07735013
walking,25.096373
walking,30.11538238
walking,35.13439229
walking,40.153402
walking,45.17240675
walking,50.19141592
walking,55.21042925
```

Figure 1414: Sample out of deployment into a CSV file.

The design in Figure 12 was chosen for its simplicity and visual appeal. The black icon was created by Dall-E, an AI software that can turn text into images. The select file allows the user to access their computers file directory and select a CSV file to input. The submit button starts the program to use that data and create the output CSV file, which will have the output text box as its name. The show plot button shows a figure with the same format as Figure 13. This image shows whether each window was jumping or walking.

One issue we faced with our model was that it occasionally had difficulties interpreting the beginning of a file. This can be seen in Figure 13, as the initial solved value is of jumping when it should be of walking. Going forward, we may opt to cut the first 5-10 seconds of our data, as those values are when the operator is putting their phone from their hand into testing position, which may skew the data.

Figure 14 shows a sample output file, which a user would expect to receive. In this case, the input file was of all walking data.

Conclusion

In conclusion, this project successfully developed a desktop application capable of distinguishing between 'walking' and 'jumping' activities using accelerometer data collected from a mobile application, Phyphox. The project followed a systematic approach consisting of data collection, data storage, visualization, pre-processing, feature extraction, training the classifier, and deploying the model into a simple desktop application with a GUI.

The desktop application has an accuracy of 93%. The 7% error deficit comes from a lack of diverse testing data. Increasing the amount and diversity of the sample data size would have helped train the classifier more precisely, which would increase the accuracy rate of the application.

Future improvements to the project include gathering additional testing data sets separate from the training data to ensure the model's robustness and generalizability. Exploring alternative machine learning models, such as Support Vector Machines (SVM) or Deep Learning algorithms, could potentially improve the classification accuracy further. Moreover, incorporating additional activities and expanding the application's functionality could make it more versatile and useful for various applications in health and fitness monitoring.

If the project was continued to be worked on, the data sample size would be increased as described above. Another aspect that would be focused on is making the desktop application able to predict whether you are walking or jumping in real-time. This would be done by accessing a web page created by the Phyphox application when the “enable remote access” option is enabled. This would replace the current method that is used where an existing CSV file is submitted to the application. This would make the application a lot more user-friendly and efficient.

Overall, the project's successful completion highlights the potential of using accelerometer data to classify different human activities effectively. The developed desktop application has the potential to be used in various contexts, such as sports performance analysis or personal fitness monitoring, with further development and refinement.

Challenges Faced

The first phase of challenges faced came during the data collection phase. Each group member had to redo trials several times due to a variety of reasons. These included unfamiliarity with the collection software used (Phyphox), errors saving data, errors exporting data out of the phones, and differences in time of data samples. That final issue was discovered when, during the creation of the model, we had difficulty successfully running our code without each training sample being of the same size. We got around this issue by trimming all data samples so that they were the same size as the smallest initial sample. If this project were redone, these issues might not be as relevant due to newfound experience with the process.

Another challenge we faced was overfitting the data. Overfitting occurs when the model is trained to fit the training data too closely and captures the noise in the data instead of the underlying patterns or relationships. We solved this through several iterations of new code, and gradually our accuracy increased.

A third challenge we faced was with the preprocessing steps. It was quite difficult to get the data processed and normalized, and this was by far the most time-consuming step we faced due to the many errors we incurred. These issues were solved entirely through trial and error.

A final issue that was realized once the product was finalized was that our testing data came from the same sample as our training data. It might have been more telling to gather one or more new sets of testing data, as there is a chance our model was too specialized for the datasets we trained with.

Demo video

The demo video will provide a comprehensive overview of our project, highlighting the crucial aspects and steps involved in the process. Then the steps followed to execute the project will be briefly explained to give an overview of how this project was completed to give the viewers a clear understanding of the project’s purpose, methodology, and results. Finally, a conclusion statement to wrap up the presentation and the project.

Appendix I

The participation table below clearly shows which members have been present for and contributed to each question.

Participation Table

X indicates who participated in each section of the project.

	Group Member Name		
Project Aspect	Mile	Jeevan	Christopher
Data Collection	X	X	X
Data Storing			X
Visualization		X	
Preprocessing	X		
Feature Extraction	X	X	
Training The Model			X
Demo Video	X	X	X
Report	X	X	X

References

1. Boa, L., & Intille, S. S. (2004). Activity Recognition from User-Annotated. *Activity Recognition from User-Annotated*.

2. Kim, S. K., & Kirchner, E. A. (2016). Handling Few Training Data: Classifier Transfer Between Different Types of Error-Related Potentials. *Transactions on neural System and Rehabilitation Engineering*.

3. Migueles, J. H., Candenas-Sanches, C., Ekelund, U., Nystrom, C. D., Mora-Gonzalez, J., Lof, M., . . . Ortega, F. B. (2017). Accelerometer Data Collection and Processing Criteria to Assess Physical Activity and Other Outcomes: A Systematic Review and Practical Considerations. *Springer Link*.