Motion Tracking Report

Víctor Benito Segura K12416072

December 2025

1 Task Description

This assignment investigates human motion tracking during walking using tri-dimensional acceleration data obtained via a smartphone. The main goal is to analyze the movement patterns of the user and use them to predict the location of the phone on the body. Specifically, the task focuses on classifying the phone position among four possible locations:

- Left hand
- Right hand
- Left trouser pocket
- Right trouser pocket

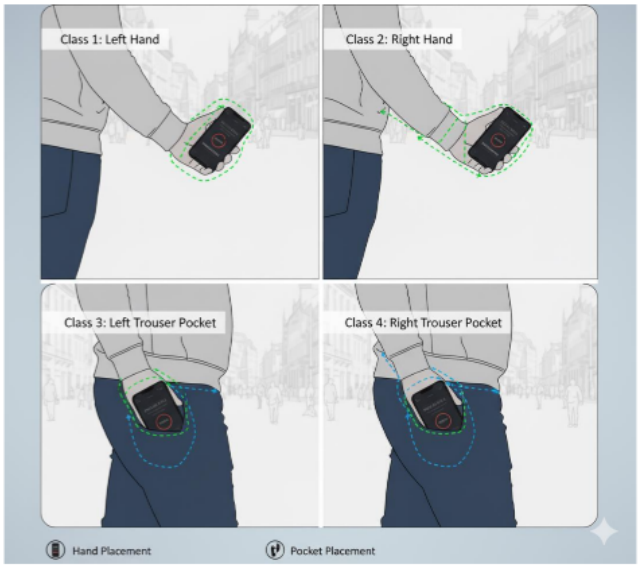

Figure 1: Task Description Image [1]

Unlike most of the Master's projects, this is an end-to-end machine learning project, as it covers all stages of a typical data-driven pipeline. This includes data collection, preprocessing, feature extraction, model training and evaluation, visualization of results, and the reporting of insights. By following the complete workflow, the project provides a comprehensive view of how raw sensor data can be transformed into actionable predictions.

2 Data Collection

The data was collected using the *phyphox* [**?**] app with an iPhone 13 in front of the Learning Center at Johannes Kepler University. Following teaching recommendations, I chose a flat outdoor area in order not to make the task unnecessarily complex. The walking track is approximately 350 m long, and the recording time is close to 3 minutes per class.

All data were recorded under walking conditions on the same outdoor track, with the subject walking at a natural pace. To ensure consistency across classes, recordings were repeated four times on the same path, sometimes in the direction from point A to point B and sometimes from point B to point A. During data collection, the subject's hands were allowed to move freely, reflecting realistic walking behavior.

A sampling rate of 100 Hz was selected for accelerometer data acquisition. Since human walking mainly contains frequency components below 20 Hz, the Nyquist theorem is satisfied with a Nyquist frequency of 50 Hz. This sampling rate ensures accurate signal representation while avoiding unnecessary oversampling and excessive data size.

The data were extracted into four separate CSV files, one per class, all sharing the same format. Each file contains five columns: time (s), acceleration in the x-, y-, and z-axes (m/s^2), and the absolute acceleration (m/s^2). Each recording consists of approximately 20,000 samples.

This project was my first time using the phyphox app, which is why I manually started and ended the recordings. This introduces minor artifacts at the beginning and end of the signals, which are handled in the preprocessing steps.

3 Preprocessing

The raw accelerometer data was preprocessed to prepare it for feature extraction and classification. The preprocessing pipeline consisted of the following steps.

3.1 Filtering

Accelerometer signals contain noise and irrelevant frequency components. To remove these, we applied Butterworth filters:

- Low-pass filter (20 Hz cutoff)
- High-pass filter (0.5 Hz cutoff)
- Band-pass filter (0.5–20 Hz cutoff)

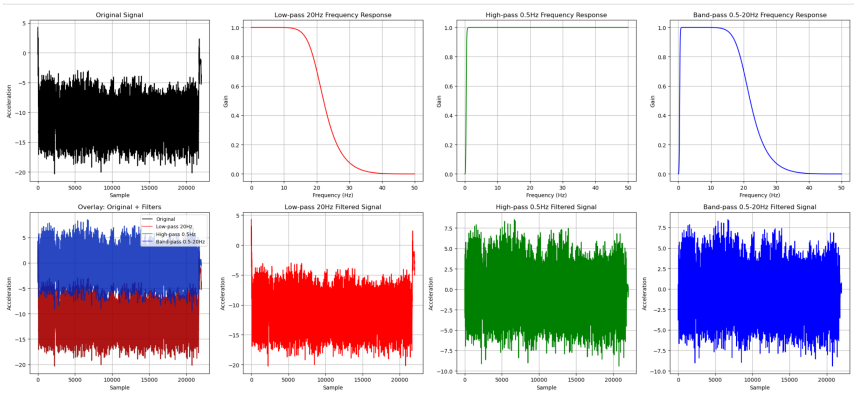

Figure 2: Original and filtered x acceleration signals. Top row: original signal and frequency responses of each filter; bottom row: time-domain signals after filtering.

Figure 2 shows the frequency responses and the effect of each filter on the x-axis acceleration.

The signals were then standardized to zero mean and unit variance.

3.2 Segmentation

After filtering, the accelerometer signals are divided into fixed-length windows to prepare the data for feature extraction. For this project, we used a window size of 2.5 seconds (corresponding to 250 samples at a 100 Hz sampling rate) with a 50% overlap between consecutive windows.
The choice of a 2.5-second window balances two main factors:

- **Capturing sufficient motion dynamics:** Each window is long enough to include at least one complete walking cycle, allowing the extracted features to represent the characteristic patterns of human gait and phone location.
- **Maintaining temporal resolution:** Overlapping windows (50% overlap) ensure that subtle transitions in movement are not missed and provide more training samples for the classifiers without introducing too much redundancy.

3.3 Feature Extraction

For each segmented window, a set of features was computed for all accelerometer axes (x, y, z) and the absolute acceleration:

- **Time-domain features:** mean, standard deviation, variance, maximum, minimum, range, and root mean square (RMS). These features capture the overall magnitude, variability, and signal intensity within each window.
- **Frequency-domain features:** dominant frequency, spectral energy, and spectral entropy. These features characterize the periodicity, overall power, and complexity of the movement in the frequency domain.

By combining time-domain and frequency-domain features for all axes, each window is represented by a feature vector that summarizes the key motion characteristics. These vectors serve as the input for machine learning models, enabling classification of the smartphone location during walking.

3.4 Clean Data

After filtering, segmentation, and feature extraction, the feature vectors from all recordings were concatenated into a single dataset. Each instance contains features for all axes and the corresponding class label (phone location). The final dataset was saved as `motion_features.csv`.

As mentioned, there are artifacts at the start and end of each recording due to manual recording and initial movements. For the final dataset used for classification, the first 3 seconds and last 3 seconds of each recording were discarded to remove this noise. The plots shown in the figure correspond to the original recordings including these noisy segments.

We obtained a total of 630 windows, which are almost balanced among the four classes

3.5 Filter Comparison

We compared the three filters using kNN ($k = 5$, 10-fold cross-validation) to determine which filter preserves the relevant motion patterns:

- Low-pass: 0.990 ± 0.011
- High-pass: 0.944 ± 0.022
- Band-pass: 0.954 ± 0.015

The low-pass filter achieved the highest accuracy, which justifies its selection for the final preprocessing pipeline.

4 Model Comparison and Hyperparameter Tuning

In this study, four different classifiers were trained and evaluated to predict smartphone location based on the extracted motion features: k-Nearest Neigh-

bors (kNN), Decision Tree, Naive Bayes, and Multi-Layer Perceptron (MLP). For each model, hyperparameter tuning was performed using 10-fold stratified cross-validation. The approach was to find the best hyperparameters for each model individually and then compare the best-performing models across all classifiers.

4.1 k-Nearest Neighbors (kNN)

The k-Nearest Neighbors algorithm classifies a sample based on the majority class among its k nearest neighbors in the feature space. Hyperparameter tuning was performed for k using the set $[1, 3, 5, 7, 9, 11, 13, 15]$, with 10-fold stratified cross-validation.

The highest cross-validated accuracy is 0.990 for $k = 1$,$k = 3$ and $k = 5$.

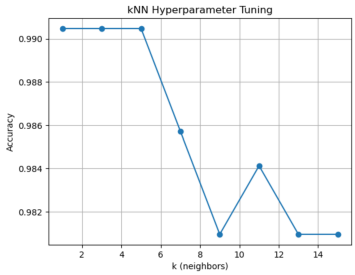

Figure 3: kNN accuracy vs k (neighbors)

$k = 1$ was selected for further model comparisons.

4.2 Decision Tree (J48)

A Decision Tree classifier was trained to predict phone location. Hyperparameter tuning was performed on the maximum depth to find the value that yields the best cross-validated accuracy.

The following depths were evaluated: 2, 4, 6, 8, 10, and unlimited (`None`).

The best maximum depth was 6, giving a cross-validated accuracy of 0.971.

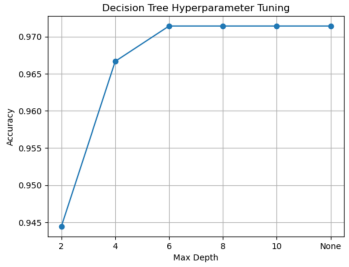

Figure 4: Decision Tree hyperparameter tuning: accuracy vs maximum depth.

This tree is equivalent to the J48 algorithm used in WEKA.

4.3 Naive Bayes

A Gaussian Naive Bayes classifier was used. No hyperparameter tuning was applied because the model has no parameters that significantly affect performance. It provides a quick baseline for comparison with other classifiers.

The model achieves an accuracy of 0.971.

4.4 Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) is a feedforward neural network capable of capturing nonlinear relationships in the data. To identify the best configuration, we performed hyperparameter tuning over different hidden layer sizes and regularization values (α). Cross-validation (10-fold) was used to evaluate each configuration as always. The following configurations were tested:

- (32,), $\alpha = 0.0001$
- (64,), $\alpha = 0.0001$
- (32,64), $\alpha = 0.001$
- (64,32), $\alpha = 0.001$
- (32,64,128), $\alpha = 0.001$
- (128,64,32), $\alpha = 0.001$

The best performing configuration was a single hidden layer with 32 neurons and $\alpha = 0.0001$, achieving a cross-validated accuracy of 0.995.

4.5 Models Comparison

In this subsection, we compare the performance of the four tuned classifiers: k-Nearest Neighbors (kNN), Decision Tree (J48), Naive Bayes and Multilayer Perceptron (MLP). The comparison is based on standard evaluation metrics including Accuracy, Precision, Recall, F1-score, ROC-AUC, and computation time.

Table 1: Model Performance Summary

Model	Accuracy	Precision	Recall	F1	ROC-AUC	Time (s)
kNN	0.990	0.991	0.990	0.990	0.994	1.673
Decision Tree	0.971	0.971	0.971	0.971	0.981	0.317
MLP	0.995	0.995	0.995	0.995	0.998	12.173
Naive Bayes	0.971	0.973	0.972	0.972	0.995	0.087

- All models achieve high accuracy and overall performance, indicating that the extracted features are highly informative for the task.
- MLP achieves the highest accuracy and ROC-AUC but is significantly more computationally expensive than the other models.
- kNN provides a good balance between accuracy and computational cost, making it a suitable choice for simplicity and real-time applications.

To further analyze class separation and model performance, we performed dimensionality reduction using both PCA and t-SNE.

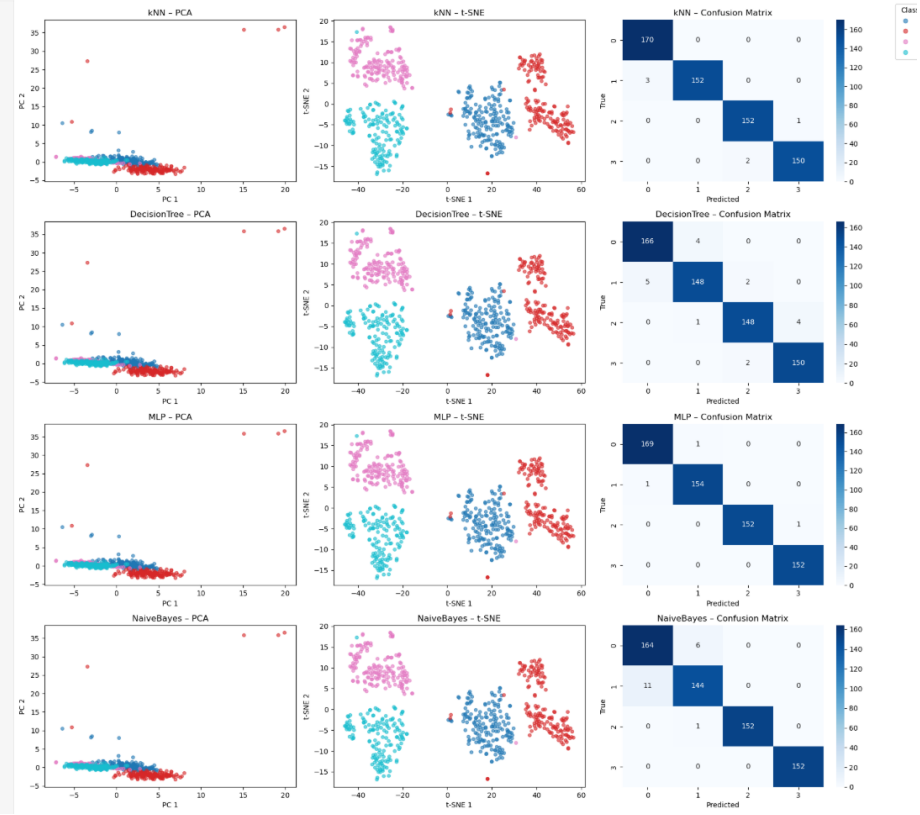

Figure 5: Comparison of dimensionality reduction and classifier performance. Each row corresponds to a model (kNN, Decision Tree, MLP, Naive Bayes). The first column shows PCA scatter plots, the second column shows t-SNE scatter plots, and the third column shows the confusion matrices.

As observed in the figure:

- PCA provides a general view of the feature space but does not completely separate the classes.

- t-SNE, being a non-linear dimensionality reduction technique, better separates the classes, which helps in visually understanding classifier performance.
- Confusion matrices show that all models make very few errors, confirming the high effectiveness of the selected features. However, we have also observed some key misclassifications insights: those are (specially for Naive Bayes) between left and right hand (classes 0 and 1).

In conclusion, although MLP performs slightly better in terms of accuracy, kNN is chosen for simplicity due to its lower computational cost and still very high performance.

4.6 Feature-Group Importance

To understand which types of features contribute most to classification, we grouped the extracted features by axis and domain (time vs frequency) as follows: x_time, y_time, z_time, abs_time, x_freq, y_freq, z_freq, abs_freq. We then computed the mean feature importance per group for each classifier, using either the built-in feature importances for Decision Tree or permutation importance for kNN, MLP, and Naive Bayes.

The figure shows the resulting heatmap of feature importance per model and feature group.

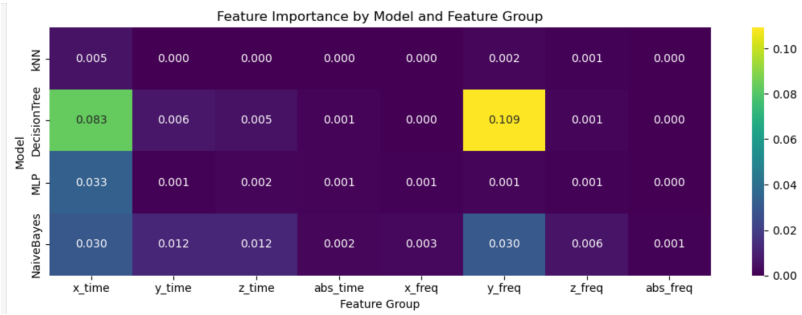

Figure 6: Feature importance by model and feature group. Each row corresponds to a model, and each column corresponds to a group of features extracted from a specific axis and domain (time or frequency). The color intensity represents the relative importance.

From the figure and the computed importance values, we observe:

- In general, the most important feature group is x_time, which is reasonable since the data was recorded along a mostly flat path where the x-axis (forward motion) dominates.
- The y_time and z_time features also contribute noticeably for Decision Tree and Naive Bayes, reflecting small lateral and vertical movements.

- Interestingly, y_freq has the highest importance for Decision Tree and a significant but slower contribution for Naive Bayes, likely capturing periodic oscillations or subtle hand motions.
- The better-performing classifiers (kNN and MLP) show lower overall feature importance values per group. This is because these models leverage the full feature space without heavily relying on any single feature or group, distributing importance more evenly.

These observations help in interpreting which motion components and feature types drive the classification, and confirm that both time-domain and frequency-domain features are relevant for the walking activity recognition task.

5 Conclusion

Through this project, I have experienced firsthand what it takes to execute a complete machine learning workflow for motion tracking, starting from data collection, through preprocessing and feature extraction, to model training, evaluation, and visualization.

I have learned how to use the Phyphox app to record 3-axis acceleration data from a smartphone, which gave me practical experience in setting up sensors, choosing sampling rates, and labeling motion data for supervised learning. I have also seen how important proper filtering and preprocessing are to create a clean and reliable dataset, which directly affects the quality of the features and the performance of the classifiers.

By comparing different models, tuning their hyperparameters, and analyzing feature importance, I could understand which models work best and which aspects of the data they rely on most. Interestingly, even relatively simple models like kNN or Decision Trees achieved very good performance, and more complex models such as MLP only marginally improved accuracy while being more computationally expensive. This shows that simple models can be effective even for complex scenarios.
It will be interesting to test if these simple models are also sufficient for more complex datasets with non-flat surfaces and more realistic “real walking tracking”.

References

[1] Task Description Image: Pervasive Computing JKU Lectures, image created with Gemini 2.5.

[2] Staudenmaier, G., et al. *Phyphox: Smartphone Experiments for Physics Education.* Available at: https://phyphox.org

[3] Code and Data Repository: Available at: https://github.com/Victorbs18/Motion-Tracking