ADAMMA Challenge 2025

Prediction of MET Classes from Smartphone Accelerometer Data

Author: Konstantinos Kalaitzidis

Abstract

In this report, we present ADAMMA, a health application that classifies smartphone accelerometer data into metabolic equivalent of task (MET) categories in near real time. The system integrates a machine learning pipeline with an Android mobile application. A Random Forest classifier has been employed in this approach, which achieved an F1-score of 0.98 on the WISDM dataset [1], outperforming the alternative models that were tested. The final app provides a responsive user interface that displays cumulative time spent in the different MET categories. Limitations, scientific rationale, and opportunities for future extensions are further discussed in this report.

1. Introduction

Physical inactivity is a major risk factor for chronic diseases and premature mortality. The metabolic equivalent of task (MET) is a standardised measure of energy expenditure, and cumulative time spent in specific MET categories is predictive of long-term health outcomes [2]. The ADAMMA challenge required building a system that continuously classifies MET categories from smartphone accelerometer data in real time. We designed an end-to-end solution combining accelerometer data preprocessing, feature engineering, supervised learning, and a mobile application.

2. System Architecture and Design

The architecture comprises of three components: (1) a machine learning (ML) pipeline for training and evaluation, (2) a lightweight backend for inference and calibration built with FastAPI [5], and (3) a React Native (Expo) mobile app for real-time data collection and visualisation [6]. In more detail:

• **Backend**: Implemented with FastAPI, providing health check (/ping) and inference (/predict) endpoints with a calibration shim.

• **Frontend**: Built in React Native (Expo), streams accelerometer data and queries the backend.

• **App UI:** Displays the current activity (colour-coded), cumulative timers, and allows backend URL configuration

In the Frontend, signals were segmented into 5-second windows (100 samples) with 50% overlap, and labels were assigned by majority vote. Development was conducted on a primary iOS device (via Expo Go), and installation was tested on a secondary Android device.

3. Data

We used the WISDM v1.1 accelerometer dataset [1], sampled at 20 Hz. It includes activities such as *sitting, standing, walking, upstairs, downstairs,* and *jogging*. These were mapped to MET categories as per the provided information in the assignment description and attributed as:

• **Sedentary**: Sitting, Standing

• **Light**: Walking

• **Moderate**: Upstairs, Downstairs

• **Vigorous**: Jogging

Signals were segmented into 5s windows (100 samples) with 50% overlap, and labels were assigned by majority vote. Importantly, the authors of the dataset note that the data were collected under controlled laboratory conditions, which may limit generalisability to real-world free-living scenarios.

Moreover, in the **exploration.ipynb** notebook, the interested reader can navigate our data analysis process, as well as the mappings of user activity to MET classes. To visually validate data homogeneity across different activity labels, we conducted trace visualisation as seen in ***Figure 1***.

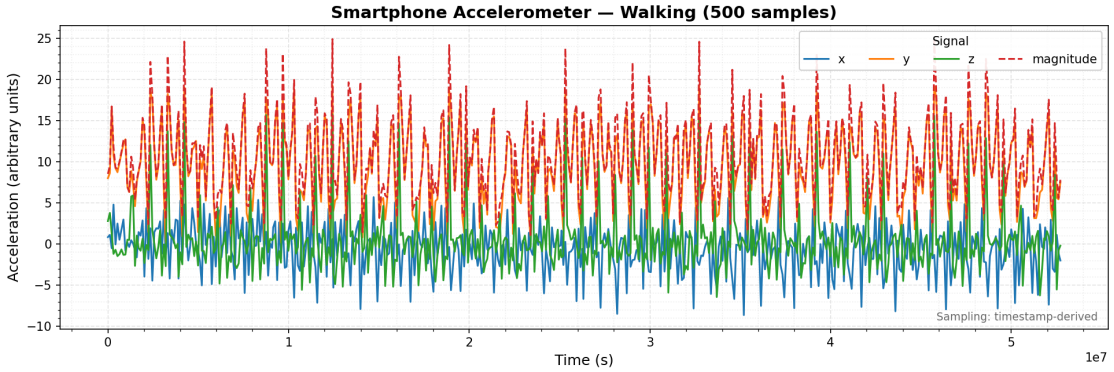

Figure 1. Visualisation of 500 samples from the WISDM dataset depicting walking activity. Periodic and structured activity resembles a reflecting footstep cadence.

4. Feature Engineering and Model Selection

From each window, 20 statistical features were extracted: *per-axis mean*, *std*, *min*, *max*, *median*, *interquartile range,* and *magnitude mean*. Models compared included *Logistic Regression, SVM (RBF), MLP (64–32 hidden units),* and *Random Forest* [3], implemented using scikit-learn [4].

Model comparison showed that Random Forest achieved the highest accuracy and F1, while Logistic Regression performed the worst. Random Forest with **n_estimators=300** was selected due to its strong performance and low-latency inference.

Table 1: Classification Reports Comparing Models

Model	Accuracy	F1_macro	TrainTime (s)
RandomForest	0.978147	0.978267	1.854569
MLP_64x32	0.969497	0.968920	0.805469
SVM_RBF	0.963123	0.964795	3.348416
LogisticRegression	0.881175	0.890001	0.119043

5. Evaluation

For our evaluation protocol, we conducted feature extraction, and the labels were encoded and features standardised using a *StandardScaler* fit on the training split. We performed a stratified 80/20 train-test split with **random_state=42**.

We compared the aforementioned models and reported accuracy and macro-F1 on the held-out test set as primary metrics. For completeness, confusion matrices are exported for each model. The best model is chosen by macro-F1 and saved along with the scaler and label encoder as a deployable artefact **(model.pkl).**

Results: The **Random Forest model** achieved **97.81% accuracy** and **0.9783 macro-F1**, outperforming MLP (96.95% / 0.9689), SVM-RBF (96.31% / 0.9648), and Logistic Regression (88.12% / 0.8900). Residual errors occurred mainly between the *Light* and *Moderate* classes.

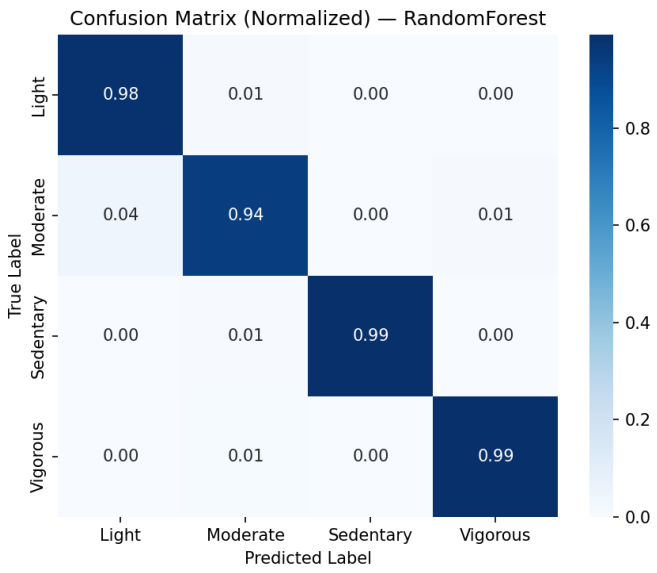

Figure 2: Confusion Matrix of the Random Forest model

6. App Implementation

The Android app was packaged as an APK with Expo Application Services (EAS). Users sideload the APK, configure the backend URL, and track live predictions. On iOS, development testing used Expo Go; however, distribution was not feasible due to the financial requirements bound to an Apple Developer account. The interface shows the current MET class, timers, and session summaries. It does not support background execution in Expo, e.g. use with the display off.

7. Scientific Rationale

Random Forest was chosen for its high empirical accuracy, interpretability, and efficiency [3]. While deep learning models like CNNs/LSTMs might offer marginal gains, they require heavier computation and training data, less suited to lightweight mobile scenarios such as in this case. Mapping activities (e.g., walking, stairs) to MET classes anchors outputs in a clinically meaningful scale widely used in epidemiology and guidelines. This enables interpretable summaries such as MVPA minutes and percentages directly relevant to behaviour change and risk stratification, rather than raw activity labels that are harder to compare across users and studies [2].

8. Conclusion, Limitations, and Future Work

ADAMMA demonstrates an end-to-end system for real-time MET classification with strong performance and a usable mobile interface. Our contributions include a reproducible ML pipeline, head-to-head model evaluation, backend deployment with FastAPI, and an Android

mobile application. However, several limitations have emerged on this 3-day development journey.

Limitations:

• App must remain in the foreground; Expo-managed workflow blocks background services.

• Apple distribution requires a paid developer account, limiting deployment to Android for now.

• Dataset bias: The WISDM dataset is controlled and may not fully generalise to diverse, real-world contexts.

Future work:

Future work includes implementing Android background services for continuous logging, conducting real-world validation with diverse users, adding cross-platform capabilities and exploring CNN/LSTM architectures for richer temporal modelling and potential accuracy gains.

9. Third-party Sources and AI Tools

This project used external resources: the *WISDM v1.1 dataset* [1] for training our ML model, open-source frameworks including *scikit-learn* [4], *FastAPI* [5], and *React Native* [6]. Plots were generated with *matplotlib* and *seaborn*. Additionally, *OpenAI's ChatGPT* was used as an assistant for structuring this report and for code-related tasks (e.g., training utilities, debugging errors, evaluation scripts, app integration, etc).

10. Acknowledgements

The author would like to thank the ETH Zurich Digital Biomarker group and the ADAMMA challenge organisers for the opportunity to partake in this candidacy evaluation for the PhD position in AI & Software Engineering for Digital Health!

11. References

[1] Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. ACM SIGKDD Explorations Newsletter, 12(2), 74–82.

[2] Ainsworth, B. E., et al. (2011). Compendium of physical activities: a second update of codes and MET values. Medicine & Science in Sports & Exercise, 43(8), 1575–1581.

[3] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

[4] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[5] Tiangolo, S. (2018). FastAPI: modern, fast web framework for building APIs with Python 3.6+. https://fastapi.tiangolo.com/

[6] Meta Platforms Inc. (2015). React Native: a framework for building native apps with React. https://reactnative.dev/