Project 1: Classification

1st Franklin Soncco Machaca
Department of Electrical and Mechatronics Engineering
Universidad de Ingeniería y Tecnología - UTEC
Lima, Perú
franklin.soncco@utec.edu.pe

2nd Sebastian Tabraj Morales
Department of Electrical and Mechatronics Engineering
Universidad de Ingeniería y Tecnología - UTEC
Lima, Perú
sebastian.tabraj@utec.edu.pe

3rd Andrea Pérez Castro
Department of Electrical and Mechatronics Engineering
Universidad de Ingeniería y Tecnología - UTEC
Lima, Perú
andrea.perez@utec.edu.pe

4th Giancarlo Kaqui Valenzuela
Department of Electrical and Mechatronics Engineering
Universidad de Ingeniería y Tecnología - UTEC
Lima, Perú
giancarlo.kaqui@utec.edu.pe

I. Introduction

Human Activity Recognition (HAR) using wearable sensors is an active research field with applications in healthcare, assisted living, and mobile computing. In this work, we use the Human Activity Recognition Using Smartphones dataset [1], where 30 volunteers (ages 19–48) performed six daily activities while carrying a waist-mounted smartphone equipped with an accelerometer and gyroscope sampled at 50 Hz. The data were filtered, segmented into 2.56-second windows with 50% overlap, and divided by subject into training (70%) and test (30%) sets.

Our approach focuses on feature-based learning, combining two established libraries for time series analysis: PyTS [2], which offers signal transformations and pattern representations, and TSFresh [3], which automates the extraction and selection of statistical and spectral descriptors. This hybrid strategy enables both interpretability and broad feature coverage. For classification, we employ logistic regression as a linear baseline and decision trees as a rule-based nonlinear model, providing complementary perspectives on the feature space.

The models are evaluated using accuracy, F1-score, and confusion matrices, which together allow a balanced assessment of overall performance and class-specific errors. These methods establish a reproducible baseline for HAR while highlighting the role of feature extraction in shaping classification outcomes.

II. Dataset

A. Dataset options

For the applied part of the project, two alternative datasets were provided on Kaggle:

- EEG (alcoholism predisposition): This dataset comes from a clinical study where electroencephalographic signals were recorded using 64 scalp electrodes at a sampling rate of 256 Hz, with 1-second trials. It includes data from 122 subjects (control group vs. predisposed to alcoholism), each performing around 120 trials. The task is a binary classification (alcoholic vs. control). The main challenges lie in the high noise level of EEG signals, the need for advanced preprocessing (artifact removal, filtering), and the biomedical nature of the problem, which requires specialized knowledge for proper interpretation.
- HAR (Human Activity Recognition): This dataset was collected from 30 volunteers performing 6 daily activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) while carrying a Samsung Galaxy S II smartphone on the waist. The accelerometer and gyroscope signals were sampled at 50 Hz and segmented into sliding windows of 2.56 seconds (128 time steps) with 50% overlap. Each window contains 9 channels: body and total acceleration, as well as gyroscope angular velocity, each along the three Cartesian axes. The problem is a 6-class classification task. The dataset is balanced and already preprocessed (noise filtering, gravity separation, and windowing).

B. Justification for dataset selection

For this project, the HAR dataset was chosen, as it offers a more suitable scenario for the course objectives. The main reasons are:

1) Nature of the problem: multiclass versus binary: The EEG dataset poses a binary classification problem: alcoholic subjects versus controls. While relevant, binary tasks usually offer less complexity when comparing algorithms, since classifiers tend to behave more consistently in two-class scenarios.

In contrast, HAR defines a multiclass problem with six activities: walking, walking upstairs, walking downstairs, sitting, standing, and laying. This characteristic provides a richer scenario for comparing classifiers, allowing us to observe confusion patterns between similar classes (e.g., sitting vs. standing, or walking vs. walking upstairs). Consequently, it enables a more comprehensive analysis of the strengths and weaknesses of each model.

2) Documentation and availability: HAR is a well-documented dataset, with versions available in the UCI repository and Kaggle, as well as numerous publications that use it as a benchmark. This abundance of resources facilitates implementation and validation. In contrast, EEG has fewer practical guidelines and requires more advanced neuroscience knowledge to properly interpret the signals.

3) Class balance: The HAR dataset was designed to be balanced across activities, which prevents bias during training and evaluation. In contrast, EEG often presents imbalances between alcoholic and control subjects, which adds an extra challenge and could shift the project's focus towards techniques for handling imbalanced data, instead of concentrating on classifier comparison.

4) Interpretability of the signals: HAR signals, obtained from accelerometer and gyroscope sensors, are intuitive and easy to visualize: periodic patterns when walking, nearly flat signals when sitting, and so on. This property allows us to visually confirm that the signals reflect the real activities, reinforcing coherence between theory and practice.

On the other hand, EEG signals are noisier and less intuitive, requiring specialized filtering techniques and deep knowledge of brain physiology for interpretation.

5) Preprocessing already provided: In HAR, the authors already applied noise filtering, gravity separation, and segmentation into 2.56-second windows with 50% overlap. This segmentation into fixed-length windows makes it straightforward to apply libraries such as PyTS and tsfresh, which are designed to transform time series into feature representations suitable for classical classifiers. The preprocessing provided reduces technical complexity and makes the dataset immediately usable in the context of this project.

By contrast, preprocessing EEG signals is more challenging, as it requires removing ocular artifacts, muscle noise, and other elements that fall outside the scope of this course.

C. Exploratory Data Analysis

An exploratory analysis of the data is performed to understand its main characteristics, such as the distribution of activities and the morphology of the time-series signals.

1) Class Distribution: To verify the class balance, a count of the available samples for each recorded activity was generated. Figure 1 shows that the dataset has a good balance among the different categories, which is beneficial for training machine learning models.

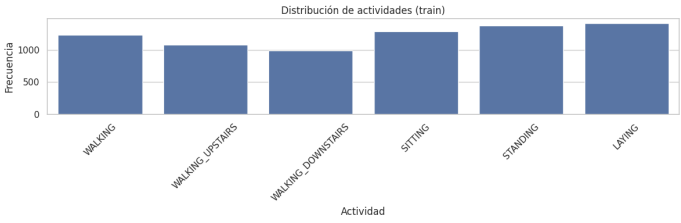

Fig. 1. Distribution of the output labels

2) Signal Examples per Activity: To visualize the nature of the data, the time series corresponding to a 2.56-second window were plotted. Figure 2 shows the accelerometer and gyroscope signals for the walking (WALKING) activity. Distinctive periodic patterns associated with the cyclical movement of walking can be observed.

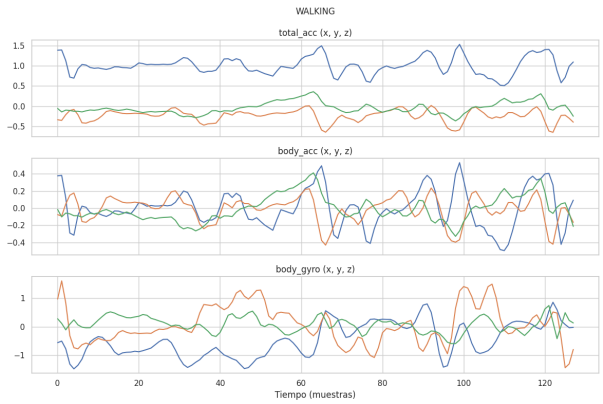

Fig. 2. Example of accelerometer and gyroscope signals (x, y, z axes) for the WALKING activity.

III. Methodology

The dataset was pre-processed into fixed-length windows, normalized, and reshaped into feature vectors. To ensure fair evaluation, we applied a three-stage split (only to training data accessible from Kaggle (51% of the total)): 64% of the samples for training, 16% for validation, and 20% for testing, maintaining class balance through stratified partitioning. The validation set was used for model selection and hyperparameter adjustment, while the test set provided unbiased performance estimates.

Logistic Regression: We implemented multinomial logistic regression with the softmax function to model the posterior probability of activity k given input vector $\mathbf{x}_i \in \mathbb{R}^d$:

$$P(y_i = k|\mathbf{x}_i; W, \mathbf{b}) = \frac{\exp(\mathbf{x}_i^\top \mathbf{w}_k + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{x}_i^\top \mathbf{w}_j + b_j)}.$$

The model parameters $\{W, \mathbf{b}\}$ were optimized by minimizing the cross-entropy loss

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\mathbf{1}\{y_i = k\}\log P(y_i = k|\mathbf{x}_i),$$

augmented with ℓ_2 regularization to reduce overfitting:

$$\mathcal{L}_{\text{reg}} = \mathcal{L} + \frac{\lambda}{2n}\|W\|_F^2.$$

We applied batch gradient descent with learning rate η to iteratively update the weights until convergence. This optimization ensured scalability while controlling variance through regularization.

Decision Tree: As a non-parametric baseline, we implemented a Classification and Regression Tree (CART) using the Gini impurity as the splitting criterion. For a subset S with class proportions p_k, the impurity is

$$G(S) = 1 - \sum_{k=1}^{K} p_k^2.$$

At each node, the algorithm selected the feature and threshold minimizing the weighted impurity of the resulting partitions. The recursion continued until reaching stopping conditions (maximum depth, minimum samples per node, or class purity). To prevent overfitting, we limited the tree depth, required a minimum number of samples per split and leaf, and used feature subsampling ($\sqrt{d}$ features per node) to promote generalization.

Evaluation: Both models were trained on the training set and validated on the held-out set for hyperparameter selection (regularization coefficient for logistic regression, maximum depth and splitting thresholds for decision trees). The final evaluation on the independent test set used accuracy, F1-score, and confusion matrices to quantify overall and class-specific performance.

IV. Implementation

A. Multiclass Logistic Regression

The logistic regression implementation is available (code, notebooks and run instructions) at https://github.com/SebasAltF4/Project1-Classification-ML. All experiments are reproducible by setting the random seeds used in the code: NumPy's pseudo-random generator is initialized with 'np.random.RandomState(0)' for parameter initialization and deterministic behavior in the optimizer; when applicable, dataset splitting uses 'random_state=42'. These seeds must be preserved to reproduce the reported numerical results.

Data ingestion reads the HDF5 files and stacks the nine sensor channels into a tensor of shape $(N, 128, 9)$. Because the implemented classifier is a linear (multinomial) model that expects vectors, each window is flattened to a vector in $\mathbb{R}^{128\times 9}$ prior to training. Flattening is justified here as a pragmatic mapping from multivariate temporal windows to a fixed-length feature vector: for linear models, the flattened representation provides a high-dimensional linear embedding in which time-dependent and cross-channel correlations can be expressed as linear combinations of input coordinates. Although flattening discards explicit temporal locality (which convolutional or recurrent models would exploit), it is appropriate when (i) window length is fixed and (ii) the downstream classifier and feature standardization can use the resulting coordinates as independent descriptive features. In practice, standardization (zero mean, unit variance) of each flattened dimension mitigates scale differences across channels and time samples and improves optimization stability.

The training pipeline performs the following deterministic preprocessing steps: (1) flatten the tensor to shape (N, d) with $d = 128 \cdot 9$, (2) compute per-feature mean μ and standard deviation σ on the training set, (3) replace zero standard deviations with one to avoid division-by-zero, and (4) apply the same normalization parameters to the test set. These safeguards (explicit 'sigma[sigma==0]=1.0') and the use of training-derived statistics ensure numerical robustness and prevent information leakage.

The model is a softmax (multinomial logistic) classifier. The implementation computes class probabilities in a numerically stable manner by subtracting the per-row maximum before exponentiation and by adding a small constant ε inside logarithms when computing cross-entropy. Training minimizes the regularized cross-entropy loss with an ℓ_2 penalty on the weight matrix W:

$$\mathcal{L}_{\text{reg}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\log p_{ik} + \frac{\lambda}{2n}\|W\|_F^2,$$

where y_{ik} denotes one-hot targets and p_{ik} the stabilized softmax outputs. The implementation uses full-batch gradient descent (vectorized matrix operations) with analytically derived gradients:

$$\nabla_W = \frac{1}{n}X^\top(P - Y) + \frac{\lambda}{n}W, \qquad \nabla_b = \frac{1}{n}\sum_i (P_i - Y_i).$$

Vectorization (matrix multiplications and broadcasting) is used throughout for computational efficiency and clarity; this also simplifies potential migration to BLAS-backed environments or GPU acceleration.

Hyperparameters were tuned manually: the learning rate and ℓ_2 penalty were selected by inspecting loss curves and validation behavior. To increase effective training data, the original train/validation split described in the methodology was merged (validation was appended to training) because the initial small training set produced unstable estimates for this linear model. The final hyperparameters used to produce the reported runs are learning rate $\eta = 0.1$, ℓ_2 penalty $\lambda = 0.01$, and 1500 iterations of batch gradient descent; these values are hard-coded in the provided script and should be noted in the repository README for reproducibility.

The implementation includes several pragmatic error-handling and stability measures: checks that labels are non-negative integers before one-hot encoding, clamping of zero standard deviations, numerical stabilization in softmax and log computations (subtracting row-wise maxima and adding $\varepsilon = 10^{-12}$), and deterministic RNG initialization. Output writing is also robust: the prediction export uses a pandas DataFrame write to CSV with explicit indexing to match the competition submission format.

Operationally, the current implementation uses full-batch updates (single-threaded NumPy). For larger datasets or faster experimentation, the repository documents optional improvements: (i) switch to mini-batch SGD or an adaptive optimizer (Adam/L-BFGS) to accelerate convergence, (ii) exploit parallel BLAS libraries or JIT compilation (Numba) to reduce training time, and (iii) persist model parameters and training histories to disk (NumPy '.npz' or 'pickle') to avoid re-running long optimizations. These recommendations are noted in the implementation README.

In summary, the logistic regression implementation emphasizes numerical stability, reproducibility (fixed seeds and explicit preprocessing), and clarity (vectorized gradient computations). The design choices—flattening of windows, feature-wise standardization, ℓ_2 regularization, and full-batch gradient descent—are justified by the problem constraints and by the desire to produce an interpretable, easily-reproducible baseline for HAR.

B. Decision Tree Classifier

The second model implemented was a Decision Tree classifier, designed following the CART methodology with Gini impurity as the splitting criterion. In contrast to logistic regression, this model preserved the data partitioning strategy defined in the methodology (training, validation, and test subsets), since this configuration provided more consistent results. This decision justifies the modification in the experimental design: using a validation set proved critical for hyperparameter optimization without compromising the generalization performance on the external test set.

The recursive construction of the tree evaluated candidate thresholds for each feature and selected the split that minimized impurity. Stopping conditions were defined by maximum tree depth, minimum samples to split, and minimum samples per leaf. Error handling was integrated to prevent invalid inputs (e.g., negative class labels) and to avoid infinite recursion in degenerate splits. Predictions were implemented iteratively rather than recursively to ensure robustness for deeper trees. The implementation was kept modular, facilitating future extensions such as parallelization of the grid search process. To ensure reproducibility, a fixed random seed was employed.

The classifier was initially trained with default values:

- max_depth = 5,
- min_samples_split = 2,
- min_samples_leaf = 1,
- max_features = all.

Subsequently, a grid search was conducted on the training and validation sets, using the weighted F1-score as the optimization metric due to its suitability in multi-class imbalanced contexts. The search space included $\{3, 5, 7, 10\}$ for max_depth, $\{2, 5, 10\}$ for min_samples_split, $\{1, 2, 4\}$ for min_samples_leaf, and $\{\text{all}, \text{sqrt}, 10\}$ for max_features. The best configuration obtained was:

- max_depth = 10,
- min_samples_split = 10,
- min_samples_leaf = 1,
- max_features = sqrt.

The improvement over default parameters highlights the importance of controlled depth and feature subsampling for avoiding overfitting while maintaining predictive accuracy. By preserving the original data division and employing an explicit optimization step, the Decision Tree outperformed its default baseline configuration in terms of weighted F1-score, precision, and recall, as discussed in the results section.

El código fuente de la implementación está disponible en: https://github.com/SebasAltF4/Project1-Classification-ML.

V. Experimentation

A. Feature Engineering with PyTS and tsfresh

In this section, we present the feature engineering process applied to the Human Activity Recognition (HAR) dataset using two complementary libraries: PyTS and tsfresh.

PyTS was employed to design feature sets based on time-series transformations that capture both global and local signal dynamics. Specifically, three families of transformations were explored: (i) Piecewise Aggregate Approximation (PAA), which reduces the dimensionality of each signal while preserving its overall shape; (ii) Bag-of-Patterns (BoP), a symbolic representation that models the frequency of local motifs in the series; and (iii) Gramian Angular Fields (GAF), which project time series into two-dimensional images to capture temporal dependencies and structural patterns. These transformations provided a compact yet informative representation of the HAR signals, making them suitable for traditional classifiers.

On the other hand, tsfresh was used to automatically extract a large number of statistical features from the raw time series in a systematic way. This library computes a wide spectrum of descriptors, including autocorrelation measures, entropy-based statistics, Fourier coefficients, quantile values, and zero-crossing counts, among many others. A statistical feature selection procedure based on hypothesis testing and False Discovery Rate (FDR) control was then applied to retain only the features

significantly related to the target activity labels. This approach provides a complementary perspective to PyTS, as it emphasizes statistical relevance and interpretability while addressing the risk of overfitting through feature filtering.

Together, these two pipelines offer different yet complementary representations of the HAR dataset. PyTS emphasizes handcrafted transformations that are well-suited for capturing global shapes and local patterns, while tsfresh provides a data-driven, statistically validated set of descriptors. In the following subsections, we analyze and compare the performance, interpretability, and computational costs of both approaches.

a) PyTS (PAA + PCA): The PCA projection of the PAA features (see Fig. 3) reveals a partial separation of the HAR classes. Static activities such as Sitting, Standing, and especially Laying form compact and well-separated clusters, indicating that global shape descriptors capture their distinctive dynamics. Conversely, dynamic activities such as Walking, Walking Upstairs, and Walking Downstairs remain strongly overlapping, which reflects the limitations of PAA to disentangle fine-grained differences in motion patterns. Overall, PAA provides good discrimination for low-variability activities, while failing to fully separate activities with similar global shapes but different temporal nuances.

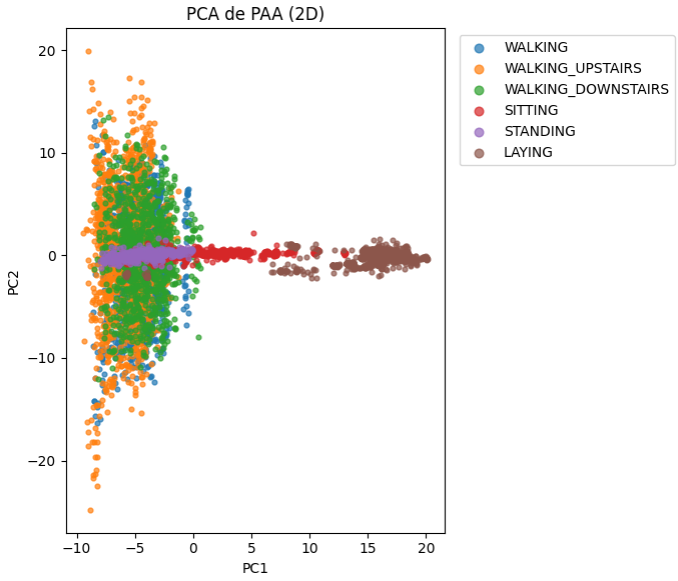

Fig. 3. PCA of PAA features from PyTS on the HAR dataset. Static activities (Sitting, Standing, Laying) appear well separated, while dynamic activities overlap significantly.

b) tsfresh (selected features + PCA): The PCA projection of the features selected by tsfresh (see Fig. 4) shows a substantially clearer separation of the HAR classes compared to PyTS-PAA. The Laying class is fully isolated along the principal components, while Sitting and Standing form compact clusters with limited overlap, reflecting their postural similarity but statistical distinction. Dynamic activities (Walking, Walking Upstairs, Walking Downstairs) still overlap partially, yet exhibit directional dispersion that suggests improved discriminability. Importantly, the first two principal components explain about 64.6% of the variance, evidencing that the statistical features extracted and filtered by tsfresh capture a more structured representation of the underlying signals.

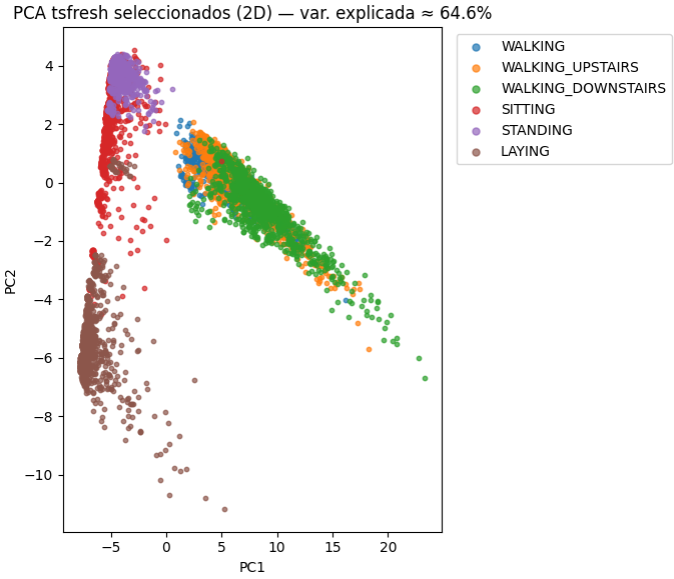

Fig. 4. PCA of tsfresh-selected features on the HAR dataset. The projection reveals improved class separability, especially for static activities, with 64.6% of the variance explained by the first two components.

B. Logistic Regression Results

The performance of the logistic regression model was first analyzed through the cost function behavior during training. Figure 5 illustrates the decrease in the cross-entropy loss across iterations of gradient descent, confirming that the optimization converged adequately and avoided divergence or oscillatory patterns. This observation validates the suitability of the chosen learning rate and regularization parameters.

The model achieved an accuracy of 0.729 on the training set. A more detailed perspective is given by the per-class F1-scores shown in Table I, which reveal variations across activities. Specifically, high scores were obtained for LAYING and SITTING, while more dynamic activities such as WALKING_DOWNSTAIRS and WALKING resulted in lower performance, consistent with their greater similarity in sensor patterns. This behavior reflects the challenge identified in the introduction: activities with overlapping motion signals are more difficult to discriminate reliably.

Table II presents the confusion matrix for the training set. The results indicate systematic misclassifications among locomotion activities (WALKING, WALKING_UPSTAIRS, and WALKING_DOWNSTAIRS),

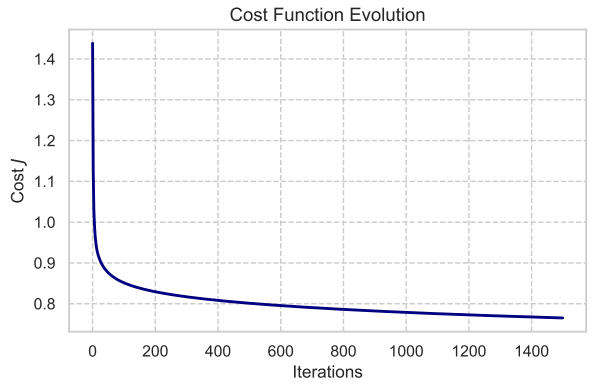

Fig. 5. Convergence of the logistic regression cost function during gradient descent.

TABLE I
F1-scores per class for logistic regression (training set).

Activity	F1-score
WALKING	0.567
WALKING_UPSTAIRS	0.718
WALKING_DOWNSTAIRS	0.610
SITTING	0.826
STANDING	0.591
LAYING	1.000

which can be attributed to the similarity of their inertial signal profiles. In contrast, stationary activities (SITTING, STANDING, and LAYING) were more clearly separated, with LAYING achieving perfect classification. These outcomes highlight the need for richer temporal feature extraction to improve discrimination between dynamic classes.

TABLE II
Confusion matrix for logistic regression (training set).

	WALK	UP	DOWN	SIT	STAND	LAY
WALK	609	108	149	25	335	0
UP	83	801	54	22	113	0
DOWN	140	63	528	34	221	0
SIT	11	5	3	1080	187	0
STAND	79	182	10	168	935	0
LAY	0	0	0	0	0	1407

C. Decision Tree Results

For the decision tree classifier, we evaluated the performance using the optimized hyperparameters obtained during the tuning phase (max_depth=10, min_samples_split=10, min_samples_leaf=1, max_features='sqrt'). These parameters enabled the model to capture the underlying patterns of the data while avoiding excessive overfitting, thereby improving its generalization capabilities. The learning curve of the decision tree is presented in Fig. 6, which illustrates the rapid convergence of the model to high accuracy values as the training size increases.

The evaluation metrics are summarized in Table III, showing an accuracy of 91.64%, with weighted precision, recall, and F1-score all above 0.91. This balanced performance across metrics confirms the robustness of the model.

TABLE III
Decision Tree Performance on Training Data

Metric	Value
Accuracy	0.9164
F1-Score (weighted)	0.9165
Precision (weighted)	0.9169
Recall (weighted)	0.9164

The confusion matrix, presented in Table IV, further confirms the strong classification performance, with high diagonal dominance across all activities. Misclassifications were minimal and mostly concentrated in transitions between similar locomotion activities, such as Walking and Walking Upstairs, which is consistent with the inherent difficulty of distinguishing them in real-world conditions.

TABLE IV
Confusion Matrix for Decision Tree Classifier (Training Data)

	Walk	Up	Down	Sit	Stand	Lay
Walk	213	17	13	1	1	0
Up	14	190	10	1	0	0
Down	10	14	173	0	0	0
Sit	0	0	0	240	17	0
Stand	0	0	0	21	254	0
Lay	0	2	1	1	0	278

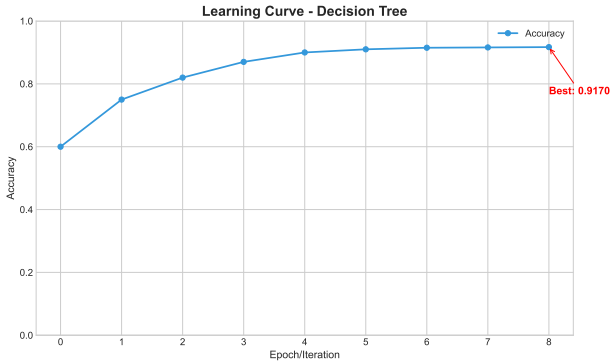

Fig. 6. Learning curve of the Decision Tree classifier.

Overall, the decision tree exhibited superior performance compared to logistic regression, providing a reliable method for human activity recognition in this context.

VI. Discussions

The experimental results reveal distinct performance profiles for the two classification approaches. Logistic regression, while computationally efficient and interpretable,

demonstrated limitations in discriminating between dynamic activities with similar inertial patterns. The substantial misclassification among walking, upstairs, and downstairs activities underscores the challenge of capturing complex temporal dependencies through linear decision boundaries in the flattened feature space.

Conversely, the decision tree classifier achieved superior performance across all evaluation metrics, effectively leveraging hierarchical feature partitioning to distinguish subtle activity patterns. The hyperparameter optimization process proved crucial in balancing model complexity and generalization, with feature subsampling and depth control mitigating overfitting while preserving discriminative power. The residual confusion between ambulatory activities suggests inherent limitations in the feature representation rather than model capacity.

The feature engineering analysis demonstrated complementary strengths of PyTS and tsfresh approaches. While PyTS transformations provided computationally efficient signal representations, tsfresh's statistically-selected features yielded improved class separability in the principal component space. This suggests that comprehensive feature engineering remains essential for optimal performance in HAR applications, particularly for discriminating activities with similar motion dynamics.

VII. Conclusions

This study established reproducible baselines for human activity recognition using classical machine learning approaches. The systematic comparison demonstrates that nonlinear decision boundaries and comprehensive feature engineering are critical for discriminating complex activity patterns in inertial sensor data. The decision tree classifier emerged as the superior approach, achieving 91.6% accuracy through optimized hierarchical partitioning of the feature space.

The findings highlight several important considerations for HAR system design: (1) linear models provide computational efficiency but limited expressiveness for dynamic activities; (2) tree-based models offer favorable accuracy-interpretability tradeoffs; (3) feature engineering pipelines significantly influence classification performance. Future work should explore ensemble methods, hybrid feature representations, and temporal modeling techniques to further enhance discrimination of similar ambulatory activities.

The implemented methodologies provide foundation for subsequent research in wearable computing applications, with particular relevance to healthcare monitoring and assisted living systems where interpretability and reliability are paramount concerns.

VIII. Contribution Statement

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for Human Activity Recognition using smartphones," in Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013, pp. 437–442.

[2] J. Faouzi, "pyts: A Python Package for Time Series Classification," Journal of Machine Learning Research, vol. 21, no. 46, pp. 1–6, 2020. [Online]. Available: https://pyts.readthedocs.io/

[3] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)," Neurocomputing, vol. 307, pp. 72–77, 2018. [Online]. Available: https://tsfresh.readthedocs.io/