DORCHIES, KIEFFER
LIGNAC, MBOCK

ANNEE 2025/2026

Machine Learning introduction

Written by Dorchies Florian, Kieffer Nathan, Lignac Simon, Mbock Steeve

Human Gate Classification using Machine Learning

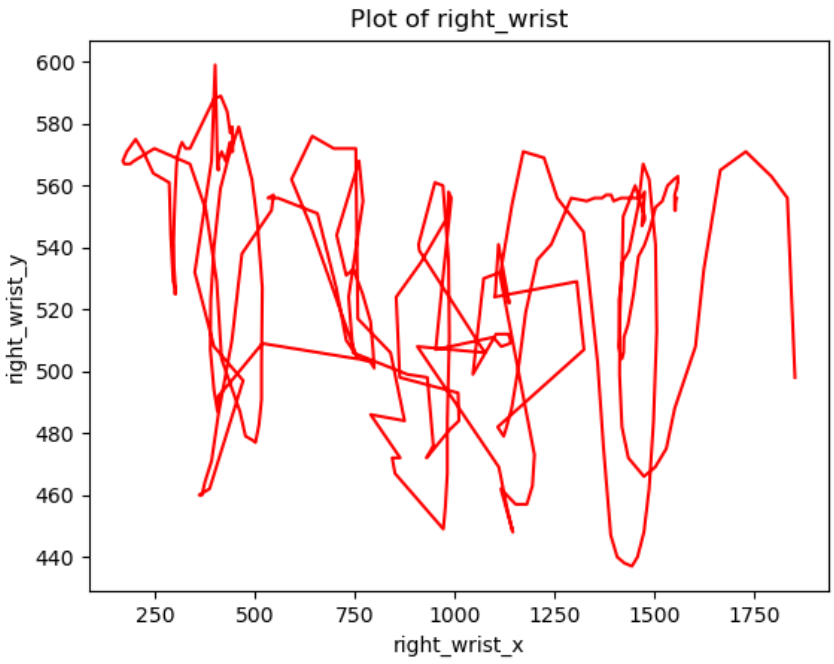


Figure 1

2025/2026

IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DES MOBILITÉS

TABLE OF CONTENTS

Introduction

This project is part of the machine learning course and aims to implement classification algorithms studied in lectures and practical sessions. The goal is to automatically distinguish CSV files representing normal gait from those representing abnormal gait, based on simple statistical descriptors extracted from each recording.

The main challenge is to compare the performance of several classifiers — from the simplest (least squares) to the most complex (neural networks, kernel SVM) — on the same dataset, with and without prior dimensionality reduction (PCA, LDA). Particular attention is paid to methodological rigor, notably the absence of data leakage in all validation steps.

I Data Description and Feature Extraction

I.1 Data Source and Collection

The dataset comprises several hundred records of body movements captured via motion capture.

The data includes different types of walking, namely:

- Normal walks: regular and undisturbed walking movements
- Abnormal walks: walks with anomalies, disturbances or limitations

Each record contains temporal data from 17 markers placed at key points on the human body:

Wrists, elbows, shoulders, hips, knees, ankles, toes, etc.

For each marker, two spatial coordinates are recorded: x and y.

Dataset Composition

The complete dataset comprises 410 CSV files:

- **321 files**: normal walks (label = 0)
- **89 files**: abnormal walks (label = 1)

Each CSV file has the following structure:

- One line per time instant (no explicit time column)
- 34 columns (17 markers $\times$ 2 coordinates)
- Variable temporal lengths depending on the recording

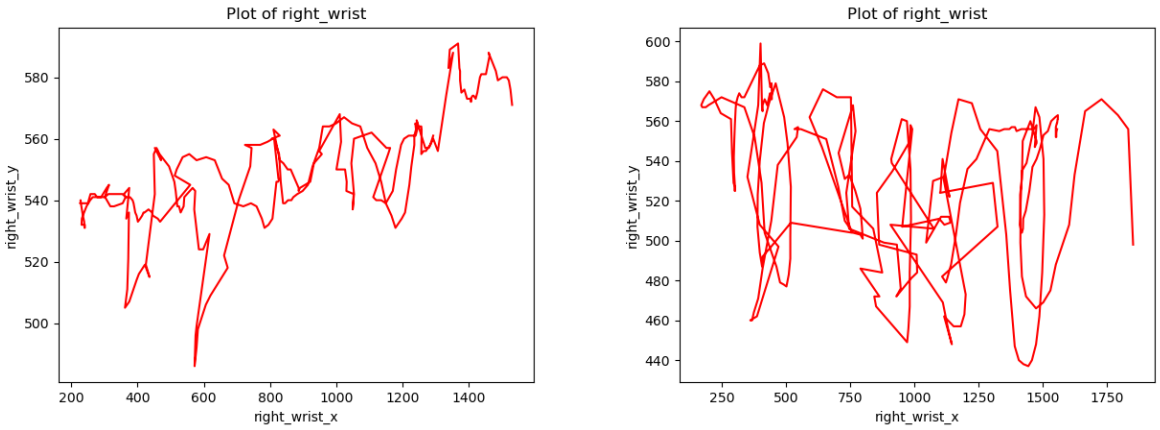
I.2 Raw Data Preprocessing

I.2.1 Robust File Loading

Before any analysis, the data underwent a preprocessing phase to ensure their suitability with the mathematical models and algorithms to be applied.

The initial step consisted of converting CSV files into an exploitable format. The `load_csv_robust` function was developed to automatically handle:

- **Separator Detection**: automatic identification of the delimiter used (comma, semicolon, etc.)
- **Encoding Management**: UTF-8 encoding by default, with fallback to latin-1 in case of failure
- **DataFrame Conversion**: using the Pandas library for easy manipulation



(a) Y coordinate in function of X - Normal walk - Wrist marker

(b) Y coordinate in function of X - Abnormal walk - Wrist marker

Figure I.1: Comparison of wrist marker trajectories, X and Y coordinates, for normal and abnormal walks

I.2.2 Trajectory Visualization

A crucial step in preprocessing was to develop visualization functions allowing exploration of each marker's trajectories. Although there is no explicit time variable, coordinates were plotted in the form $y = f(x)$ to visually examine walking patterns.

- **Normal walks**: marker paths (wrists, ankles) regular and fluid
- **Abnormal walks**: paths presenting irregularities, deviations and discontinuities

I.3 Data Structure

The raw data is organized in two folders: `normal_folder` and `abnormal_folder`, each containing CSV files. Each file corresponds to a recording of a walk (normal or abnormal). Files are loaded robustly thanks to the `load_csv_robust` function, which automatically handles separator detection and UTF-8 encoding fallback to latin-1.

I.4 Extraction of Statistical Features

Although raw data is in temporal form, direct use of time series would have led to variable length per recording and excessive computational complexity.

Therefore, three simple and understandable statistical features were extracted from each numerical coordinate: Mean, Variance and Range (max − min)

These descriptors were chosen for their ability to capture the essential characteristics of movement patterns while minimizing noise.

For each CSV file, a feature vector is built by the `extract_features_from_df` function. For each numerical column j of the signal, three statistics are calculated:

Proposition 1: Features extracted per column

For a column j containing the values $(s_1, \ldots, s_n)$, the extracted features are:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} s_i, \qquad \sigma_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(s_i - \mu_j)^2, \qquad \delta_j = \max_i s_i - \min_i s_i$$

The complete dataset $(X, y) \in \mathbb{R}^{410\times 102} \times \{0,1\}^{410}$ is built only once, then cached in RAM via `get_or_build_dataset` to avoid unnecessary recalculations during experiments.

II

Preprocessing and Dimensionality Reduction

II.1 Z-score Standardization

Before any training, the data is centered and scaled column by column. Standardization is fitted only on the training data, then applied to the test set:

When $\sigma_j = 0$ for a column j, the denominator is replaced by 1 to avoid any division by zero.

II.2 Principal Component Analysis (PCA)

II.2.1 Principle

PCA is implemented manually in `fit_pca`.

We chose to create a Python program that allows us to determine how many components are needed to explain a certain proportion of the total variance (typically 95%).

To do this, we used the cumulative explained variance formula, which is given by the sum of the k first eigenvalues divided by the sum of all eigenvalues.

Proposition 2: Cumulative explained variance

The proportion of variance explained by the k first components is:

$$\text{VarExp}(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

The function `n_components_for_target_variance` returns the minimal number of components k^* such that $\text{VarExp}(k^*) \geq \tau$.

II.2.2 Data leakage prevention principle

In the function `evaluate_classification_pca`, PCA is learned on the training fold at each iteration, then applied to the test fold. This ensures that no test information contaminates the transformation:

$$\texttt{mean\_pca, W\_pca} = \texttt{fit\_pca}(X_{\text{train}}), \qquad X_{\text{train}}^{\text{pca}} = \texttt{transform\_pca}(X_{\text{train}}), \qquad X_{\text{test}}^{\text{pca}} = \texttt{transform\_pca}(X_{\text{test}})$$

II.3 Linear Discriminant Analysis (LDA)

LDA seeks the projection axis that maximizes inter-class separability relative to intra-class dispersion (Fisher criterion). The projection matrix is obtained as the principal eigenvector of $S_W^{-1} S_B$:

Proposition 3: Fisher Criterion

Let S_W be the intra-class dispersion matrix and S_B be the inter-class dispersion matrix. The LDA projection is given by:

$$W_{\text{LDA}} = \text{argmax}_W \frac{W^\top S_B W}{W^\top S_W W}$$

For binary classification, only one discriminant component is available ($n_{\text{comp}} = \min(C - 1, d) = 1$). After projection, the LDA data are re-standardized to stabilize classifiers.

III Implemented Classifiers

III.1 Classifier selection and parameter justification

We selected a set of six classifiers spanning a spectrum of complexity, from simplest to most sophisticated. This strategy allows us to evaluate the impact of model complexity on performance and to demonstrate the importance of methodological validation in model selection.

III.1.1 Least Squares Regression

The least squares classifier constitutes the baseline. Its advantage lies in its *simplicity* and *computational speed* (algebraic closure). It is a good reference point to evaluate whether more complex models truly offer improvement.

Parameters: No hyperparameters to adjust; prediction is performed by thresholding: $\hat{y} = X\theta \geq 0{,}5$.

III.1.2 Logistic Regression

Logistic regression is chosen for its interpretability capacity. It offers a good balance between simplicity and expressiveness for well-separated binary problems.

Selected parameters:

- Learning rate $\alpha = 0{,}1$: small enough for stable convergence
- Number of iterations $n\_\text{iter} = 2000$: ensuring complete convergence
- L2 regularization coefficient $\lambda = 10^{-2}$: mild, to prevent overfitting without underfitting

III.1.3 Support Vector Machines (SVM)

SVMs offer good non-linear separation and robustness to dimensionality. We implemented two variants:

Linear SVM: To capture simple linear boundaries of the problem. Parameter $C = 1{,}0$ (default).

RBF SVM (Gaussian kernel): To model more complex boundaries. The hyperparameters (C, σ) were optimized via Bayesian optimization by maximizing the F1-score on validation folds:

- Initial values: $C = 1{,}0$, $\sigma = 0{,}5$
- Search range: $C \in [0{,}1, 100]$, $\sigma \in [0{,}01, 5]$
- Optimized results: $C = 68{,}27$, $\sigma = 0{,}36$ (see Section **??**)

III.1.4 Random Forest

The random forest is selected for its resistance to overfitting. It also provides natural feature importances. The model is scale-invariant, which justifies the absence of standardization.

Parameters: $n\_\text{estimators} = 200$ decision trees, a sufficient number to converge without computational burden.

III.1.5 Neural Network

The neural network represents maximum complexity with non-linear learning capacity. The architecture remains *simple and reproducible* for interpretability.

Architecture: One hidden layer with 25 neurons ($d \to 25 \to 1$) with sigmoid activation.

Learning parameters:

- Learning rate $\alpha = 0{,}2$: moderate for stability
- Number of iterations $n\_\text{iter} = 250$: avoids excessive overfitting
- L2 regularization $\lambda = 10^{-3}$: mild, to control network complexity

IV Evaluation Protocol and Results

IV.1 Experimental Protocol

To obtain robust performance estimates, each classifier is evaluated according to the following protocol:

1. **30 random train/test splits** (70%/30%) with `test_size=0.3`
2. Z-score standardization fitted on training data only
3. Dimensionality reduction PCA or LDA, learned on training data only
4. Classifier training on train set, evaluation on test set
5. Computation of metrics: accuracy, recall, precision, F1-score, balanced accuracy

IV.2 Metrics Used

To maximize our analysis and model comparison, we chose to use multiple evaluation metrics: accuracy, recall, precision, F1-score and balanced accuracy. Precision is not displayed in the summary table but is necessary to compute the F1-score, which is the primary metric selected for RBF SVM Bayesian optimization.

IV.3 Comparative Results

	Classifier	Accuracy	F1-Score	Recall	Bal. Acc.
Raw Features	Least Squares	0.935	0.846	0.766	0.877
	Logistic Regression	0.909	0.782	0.698	0.837
	Linear SVM	0.909	0.785	0.707	0.840
	RBF SVM	0.902	0.780	0.743	0.848
	Random Forest	0.907	0.770	0.675	0.827
	Neural Network	0.920	0.803	0.698	0.844
PCA	Least Squares	0.771	0.079	0.058	0.526
	Logistic Regression	0.813	0.423	0.302	0.638
	Linear SVM	0.790	0.255	0.165	0.575
	RBF SVM	0.747	0.486	0.510	0.665
	Random Forest	0.814	0.503	0.409	0.674
	Neural Network	0.824	0.479	0.354	0.663
LDA	Least Squares	0.935	0.846	0.766	0.877
	Logistic Regression	0.936	0.853	0.783	0.884
	Linear SVM	0.934	0.850	0.795	0.886
	RBF SVM	0.934	0.849	0.789	0.884
	Random Forest	0.921	0.836	0.835	0.892
	Neural Network	0.935	0.846	0.766	0.877

Table IV.1: Classifier comparison on raw features, PCA and LDA (averaged over 30 splits)

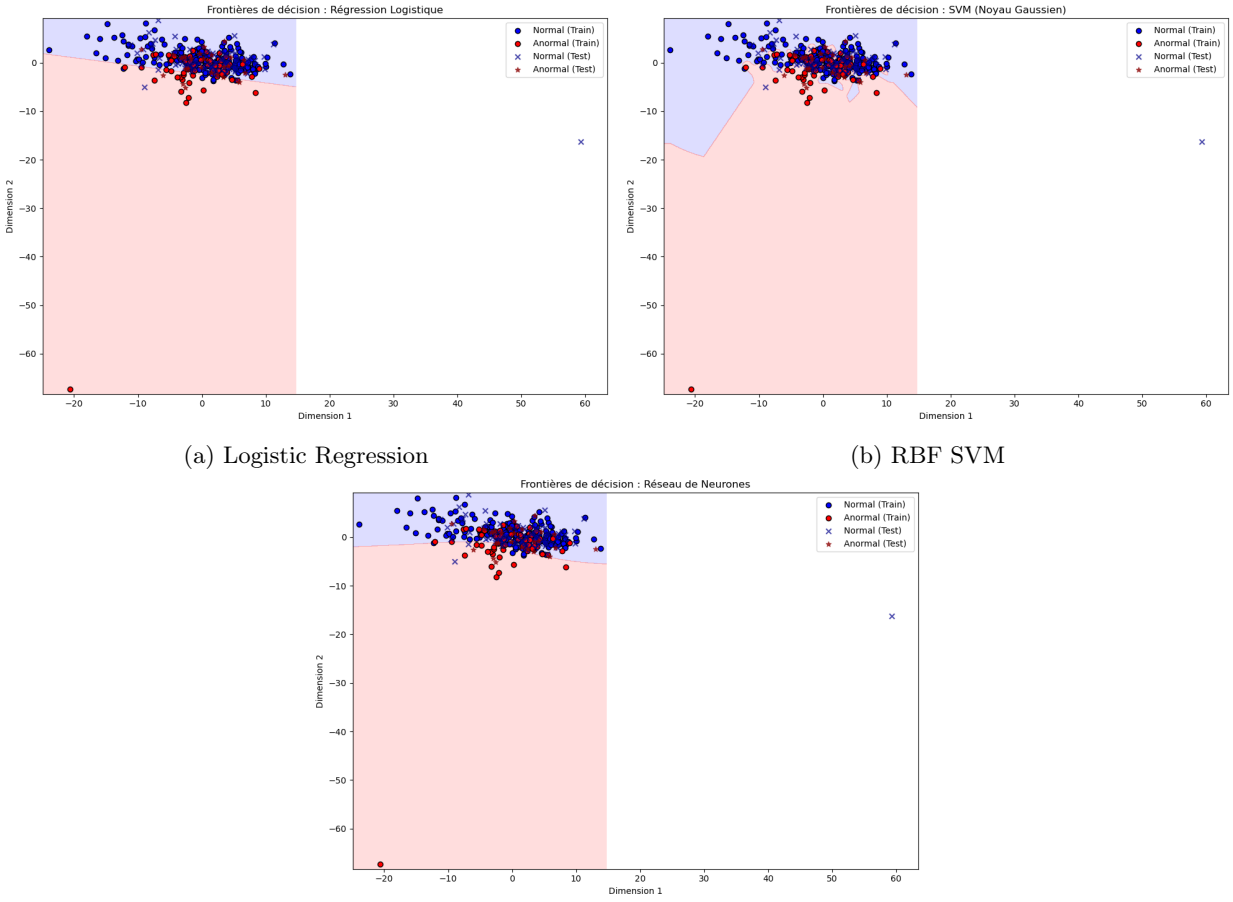
IV.4 Impact of Dimensionality Reduction

Despite 99% variance explained by the 3 PCA components, classifier performance drops significantly, suggesting that low-variance dimensions contain crucial discriminative information. We observe that PCA tends to reduce classification to essentially random data separation. Indeed, the dataset is split into two non-uniformly distributed classes (approximately 70% normal walks and 30% abnormal walks). Consequently, if classifiers fail to find linear separation in PCA space, they tend to classify all samples into the majority class (normal walk), explaining the very low F1 scores.

Conversely, LDA, which maximizes class separability, preserves or even improves performance for certain models (notably SVM and random forest).

IV.5 Decision Boundary Visualization

Decision boundaries are visualized in 2D PCA space using the `visualize_any_decision_boundary` function. This function evaluates each classifier on a dense point grid to plot the learned separation.



(a) Logistic Regression

(b) RBF SVM

(c) Neural Network

Figure IV.1: Decision boundaries in 2D PCA space for Logistic Regression, RBF SVM and Neural Network classifiers.

IV.6 Bayesian Optimization of RBF SVM

The hyperparameters (C, σ) of RBF SVM are optimized via Bayesian optimization in LDA space. A Gaussian process models the F1-score surface as a function of (C, σ), and the **Expected Improvement** acquisition function guides the search:

Proposition 4: Expected Improvement (EI)

For a candidate point x with Gaussian prediction $(\mu(x), \sigma^2(x))$ and best observed value f^*:

$$\mathrm{EI}(x) = (\mu(x) - f^*)\,\Phi(Z) + \sigma(x)\,\phi(Z), \qquad Z = \frac{\mu(x) - f^*}{\sigma(x)}$$

where Φ and ϕ are the CDF and PDF of the standard normal distribution.

Version	C	σ	Accuracy	F1-Score	Bal. Acc.
RBF SVM (default)	1.0	0.5	0.959	0.894	0.927
RBF SVM (BO)	68.27	0.36	0.959	0.894	0.927

Table IV.2: RBF SVM comparison before/after Bayesian optimization in LDA space

Analysis: Performance Plateau and Saturation

Bayesian optimization explored a wide spectrum of hyperparameters (C, σ) over 8 additional iterations after 4 random initializations. However, no combination improved the initial F1-score of **0.8824**. This result illustrates a common phenomenon in machine learning:

- **Performance Plateau**: The default hyperparameters ($C = 1.0, \sigma = 0.5$) already lie in a near-optimal region of the search space. Despite varying C over 4 orders of magnitude (from 16.41 to 100) and σ between 0.30 and 5.00, the F1-score remains constant.
- **Separability Saturation**: In 1D LDA space, the two classes are sufficiently well separated that regularization and RBF kernel scale provide only marginal improvement. The Gaussian process thus detects no region of significant improvement.
- **Stable Cross-Validation**: Average scores over 30 cross-validation splits indicate very low variance (std $\approx$ 0.02 for balanced accuracy). This stability suggests the default model generalizes well, hence no gain from optimization.

In conclusion, Bayesian optimization confirms that **LDA space** offers natural separability favorable to RBF SVM with standard hyperparameters, rendering further adjustments unnecessary for this problem.

Conclusion

This project illustrates a complete supervised classification pipeline, from simple statistical feature extraction to rigorous comparison of multiple algorithms with and without dimensionality reduction.

1. **Statistical Features**: Descriptors (mean, variance, range) by column provide a compact and interpretable representation of gait recordings.
2. **Data Leakage Prevention**: All transformations (standardization, PCA, LDA) are systematically fitted on training data only, guaranteeing unbiased evaluation.
3. **Classifier Comparison**: The most expressive models (RBF SVM, random forest, neural network) tend to achieve the best performance on this type of data.
4. **Dimensionality Reduction**: Supervised LDA concentrates discriminative information into a single component, potentially simplifying the problem. Unsupervised PCA allows noise reduction but without guaranteeing class separability preservation.
5. **Bayesian Optimization**: Using a Gaussian process for SVM hyperparameter optimization efficiently explores the (C, σ) space with a limited number of costly evaluations.

We observed that, in this specific case, RBF SVM default hyperparameters already approach the optimum, illustrating a performance plateau phenomenon. This underscores the importance of understanding problem nature and data structure before investing in complex optimizations. Moreover, we demonstrated that PCA is not always an excellent solution for dimensionality reduction in classification problems, especially when low-variance dimensions contain crucial discriminative information.

In our specific human gait classification case for anomaly detection, our results remain highly satisfactory outside PCA, with F1-scores exceeding 0.8 for several classifiers—an excellent outcome given the simplicity of features used and reproducibility in hospital settings.