A Hierarchical Three-Class Knee Extension Classifier Using EMG and IMU Signals

Gabriel Lin* William Hsu, PhD†

Abstract

In this project, a machine learning classifier was developed to evaluate knee extension exercises using **EMG (electromyography)** and **IMU (inertial measurement unit)** signals. The goal was to support future wearable-device systems that can help patients perform physical therapy exercises outside the clinic while automatically checking movement quality. Each trial was classified into one of three categories: Correct Extension, Wrong Extension–Low Lift (LL), and Wrong Extension–Not Fully Extended (NF).

A hierarchical two-stage LightGBM model with probability calibration and threshold gating was used, and the data were split by subject before preprocessing to ensure fair evaluation. By combining EMG-based muscle activation patterns with IMU-based movement dynamics, the model achieved strong performance and showed potential for accurate, on-device rehabilitation monitoring and automated feedback during unsupervised physical therapy.

Keywords: knee rehabilitation; knee extension classification; electromyography (EMG); inertial measurement unit (IMU); wearable sensors; LightGBM; hierarchical classification; probability calibration; subject-held-out evaluation; physical therapy monitoring

1 Introduction

Correct exercise form is critical for injury prevention and effective rehabilitation. Traditionally, physical therapists assess movement quality through visual observation, which is limited to supervised clinical environments. Wearable sensor technology enables automated and objective evaluation of movement patterns outside the clinic, allowing patients to perform physical therapy exercises at home with greater convenience and accessibility.

This project uses synchronized electromyography (EMG) and inertial measurement unit (IMU) signals to classify knee extension movements. EMG captures muscle activation patterns, while IMU sensors measure acceleration and rotational motion. Together, these signals provide complementary information about both neuromuscular activity and limb dynamics during exercise.

By integrating EMG and IMU data, a two-stage classification model was developed to distinguish correct from incorrect knee extension movements and to further differentiate types of incorrect forms (Extension_NF, Extension_LL).

*Faith Baptist Schools, Los Angeles, CA, USA

†Departments of Radiological Sciences, Bioinformatics, and Bioengineering; Institute of Quantitative and Computational Biosciences; Jonsson Comprehensive Cancer Center, University of California, Los Angeles, CA, USA

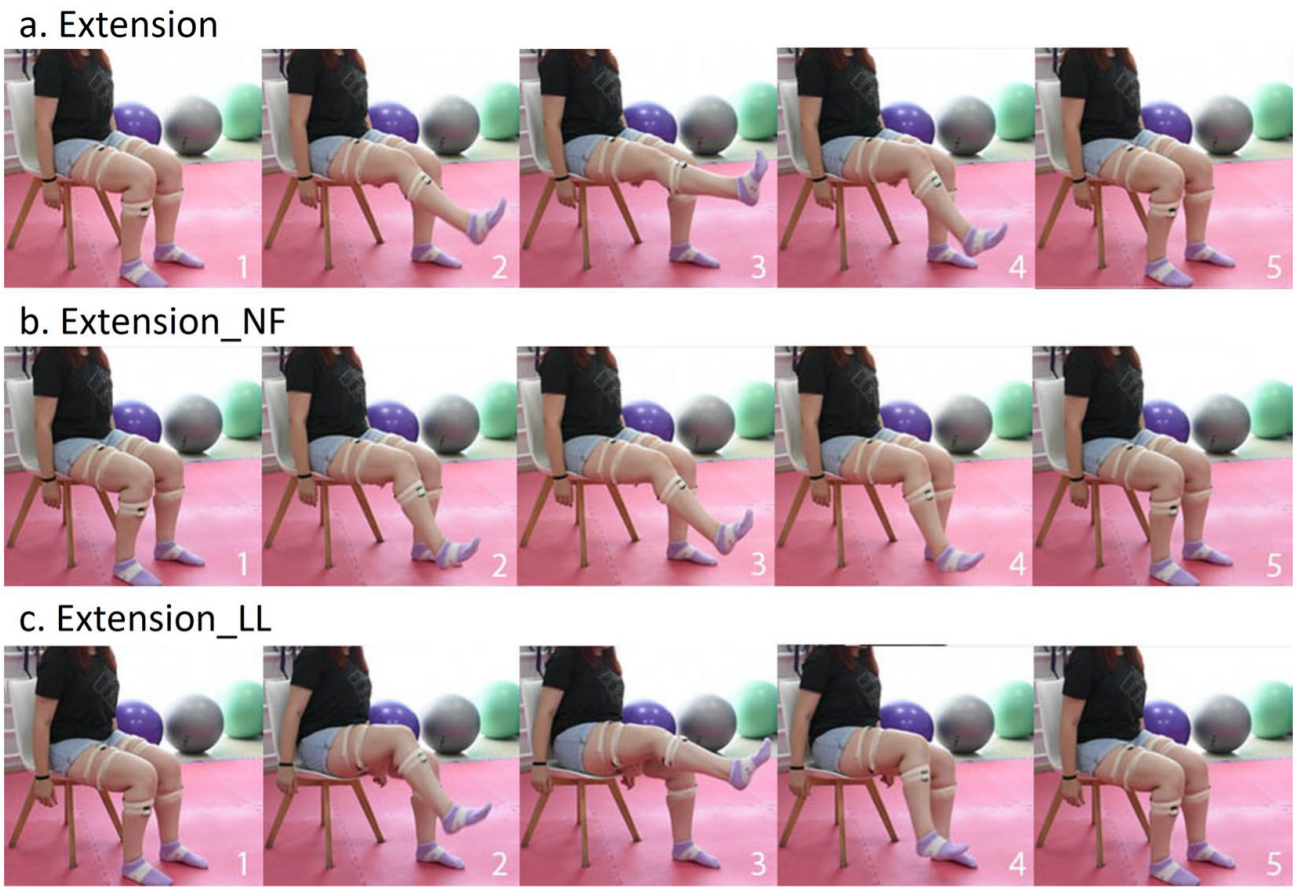

Figure 1: Visual example illustration of the performed seated leg extension variations: (a) Extension, (b) Extension_NF, and (c) Extension_LL.

2 Dataset Description

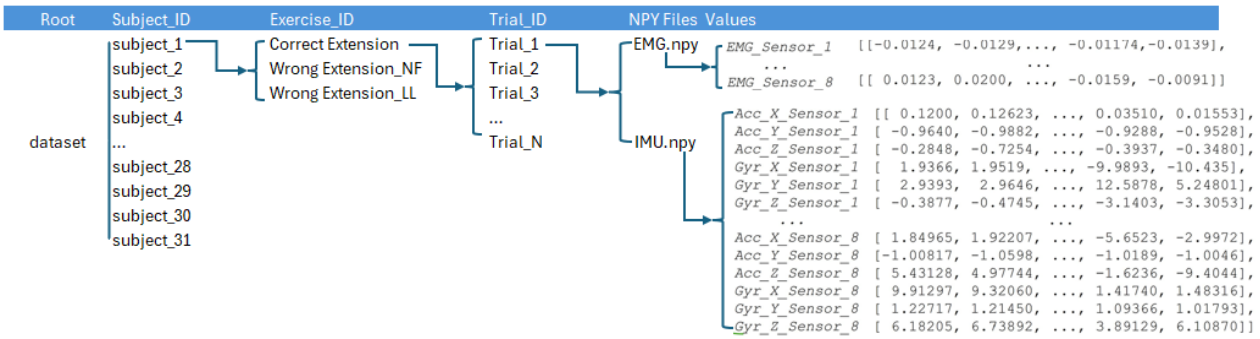

Figure 2: Data organization of KneE-PAD. The "Subject_ID" folders are included in the root directory, "dataset". The subject-dependent folders are split into activity classes ranging from 0-8, which are further divided into the executed repetitions ("Trial_ID"). Each contains the recorded sEMG samples (emg.npy) and IMU samples (imu.npy).

The KneE-PAD dataset [2] includes 31 patients with clinically diagnosed knee pathologies, ranging in age from 18 to 68 years, with diverse physical characteristics and injury types such as osteoarthritis, meniscus injuries, and ligament damage. All participants performed rehabilitation exercises in an unsupervised setting, designed to reflect realistic conditions where patients carry out prescribed exercises outside clinical supervision, such as at home.

The dataset contains three types of rehabilitation exercises: squat, seated leg extension, and gait. In this project, only the **seated leg extension** data were utilized, enabling a focused investigation of knee extension movement quality.

Each exercise trial was subsequently carefully curated and annotated, and categorized into three clinically meaningful classes:

- Correct execution
- Wrong_LL (low lift / compensatory lifting)
- Wrong_NF (not fully extended)

Across the 31 patients, each subject contributed multiple trials per class, typically around:

- ~10 trials for Correct execution
- ~10–14 trials for Wrong_LL
- ~9–12 trials for Wrong_NF

This results in a well-replicated and relatively balanced dataset at the subject level, ensuring that:

- Each patient contributes sufficient examples for all movement categories
- The model learns within-subject variability
- The evaluation (especially subject-held-out) is more reliable and less biased by uneven sampling

Although minor variations exist (e.g., a few subjects having fewer trials in certain categories), the overall dataset maintains strong consistency in trial counts across subjects and classes, which is critical for robust supervised learning.

Importantly, this combination of:

- Unsupervised, real-world data collection
- High-quality post-hoc labeling
- Consistent multi-trial coverage per subject

makes the dataset especially suitable for developing machine learning models that can automatically assess rehabilitation performance anywhere, including home-based settings without direct physical therapist supervision. Such systems can serve as virtual coaching tools, providing real-time feedback and improving rehabilitation adherence and outcomes.

The data was collected using eight Delsys Trigno Avanti sEMG sensors, each integrated with a 6-axis IMU (3-axis accelerometer and 3-axis gyroscope). As reflected in the dataset structure shown above, each subject's trial contains synchronized EMG and IMU recordings from all eight sensors. The EMG signals were sampled at 1259.259 Hz with a bandwidth of 20–450 Hz and an 11 mV range, while the IMU signals were sampled at 148.148 Hz, with an accelerometer range of ± 2 g and a gyroscope range of ± 250 deg/s.

Due to the high sampling frequencies and multi-channel sensor configuration, each trial contains thousands of time-series samples per channel. When aggregated across all sensors, subjects, trials, and exercise categories, this results in millions of raw data points, reaching the order of tens of millions overall. This high-resolution data capture preserves detailed temporal and biomechanical information, which is essential for distinguishing subtle differences in movement quality.

The sensors were placed bilaterally on four lower-limb muscles: rectus femoris, hamstrings, tibialis anterior, and gastrocnemius (right side: sensors 1–4; left side: sensors 5–8). This multi-sensor configuration provides detailed muscle activation and motion dynamics data for each trial, which serve as the input features for the proposed two-stage knee extension movement classification model.

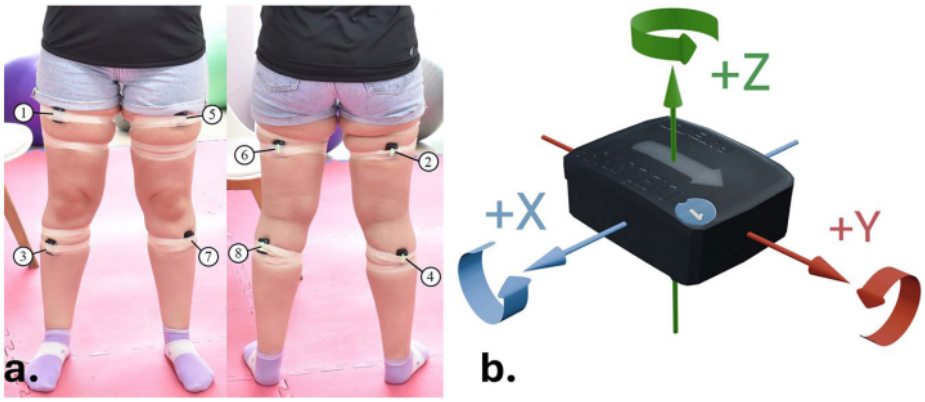

Figure 3: Illustration of the sensor placement including sensor ID (a) and X, Y, Z orientation (b).

Table 1: Sensor placements on muscles.

Sensor ID	Muscle
1	right rectus femoris
2	right hamstrings
3	right tibialis anterior
4	right gastrocnemius
5	left rectus femoris
6	left hamstrings
7	left tibialis anterior
8	left gastrocnemius

3 Data Preprocessing and Feature Engineering

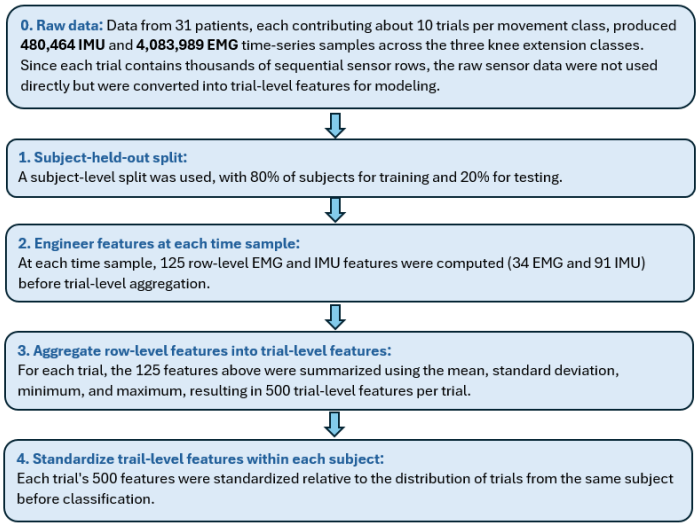

Figure 4: Overview of the preprocessing and feature-engineering workflow for knee extension movement classification.

3.1 Subject-Held-Out Split

The dataset was split at the subject level into 80% training subjects and 20% held-out test subjects. Entire subjects appear only in train or test set. There will be no data leakage between subjects.

Both Stage-1 and Stage-2 models were trained exclusively on the training portion. Stage-2 training was further restricted to incorrect trials within the training set. Within the training set, an internal 15% split was reserved for isotonic probability calibration, while the remaining 85% was used to fit the LightGBM classifiers.

Isotonic probability calibration is a post-processing method that adjusts model-predicted probabilities so they better reflect true outcome likelihoods. Using a validation dataset, isotonic regression learns a monotonic mapping between predicted probabilities and observed labels, which is then applied to future predictions. This improves the reliability of probability estimates and supports more accurate threshold-based decisions in the classification.

3.2 Train-Only Preprocessing

All preprocessing statistics (1st/99th percentiles) were computed **only on training data**, then applied to test data.

3.3 EMG Row-Level Engineered Features (Before Trial-Level Aggregation)

Using the 8 raw EMG channels, 26 additional row-level features were engineered. A total of 34 EMG features (8 raw and 26 derived) were subsequently aggregated to the trial level using summary statistics (mean, standard deviation, minimum, and maximum) and used as part of the inputs to the LightGBM classifier.

Let R_m= right EMG channel for muscle m, L_m= left channel, and $\varepsilon = 10^{-6}$.

A) Left–right paired features (per muscle m)

- **Difference (4 derived features, one per paired muscle)**

$$d_m = R_m - L_m$$

- **Abs ratio (4 derived features, one per paired muscle)**

$$\text{ratio}_m = \frac{|\, R_m \,|}{|\, L_m \,| + \varepsilon}$$

- **Normalized asymmetry (bounded in $[-1, 1]$ 4 derived features, one per paired muscle)**

$$\text{asym}_m = \frac{|\, R_m \,| - |\, L_m \,|}{|\, R_m \,| + |\, L_m \,| + \varepsilon}$$

B) Composition / fractional contribution across all EMG channels

Let EMG channels be $\{x_1, \ldots, x_8\}$.

i. Define **total activity (1 derived feature)**:

$$\mathrm{TOT} = \sum_{k=1}^{8} \mid x_k \mid + \varepsilon$$

ii. Then **each channel's fraction (8 derived features, one per EMG channel)**:

$$frac_k = \frac{\mid x_k \mid}{TOT}$$

C) Geometry-Based Summary Features (5 derived features)

Let the matched right and left activation vectors per row be:

$$\mathbf{R}, \mathbf{L} \in \mathbb{R}^4$$

where 4 denotes the number of muscles per side.

i. **L2 norms (2 derived features)**

$$\parallel \mathbf{R} \parallel_2 = \sqrt{\sum_{j=1}^{4} R_j^2} \qquad \parallel \mathbf{L} \parallel_2 = \sqrt{\sum_{j=1}^{4} L_j^2}$$

ii. **Euclidean between right and left vectors (1 derived feature)**

$$D = \parallel \mathbf{R} - \mathbf{L} \parallel_2 = \sqrt{\sum_{j=1}^{4} (R_j - L_j)^2}$$

iii. **Cosine similarity (1 derived feature)**

$$\cos(\mathbf{R}, \mathbf{L}) = \frac{\mathbf{R} \cdot \mathbf{L}}{\parallel \mathbf{R} \parallel_2 \parallel \mathbf{L} \parallel_2 + \varepsilon}$$

The value is clipped to the interval $[-1, 1]$.

iv. **Normalized difference (1 derived feature)**

$$\mathrm{ndiff} = \frac{D}{\parallel \mathbf{R} \parallel_2 + \parallel \mathbf{L} \parallel_2 + \varepsilon}$$

These features try to capture "how aligned" left/right EMG patterns are.

3.4 IMU Row-Level Engineered Features (Before Trial-Level Aggregation)

From the 48 raw IMU channels (3 axes × 2 signal types × 2 sides × 4 muscles), 43 additional row-level features were derived, including magnitude-based and left–right symmetry measures. This resulted in 91 IMU features prior to aggregation. These features were then summarized at the trial level using mean, standard deviation, minimum, and maximum, and supplied as part of the inputs to the LightGBM classifier.

Assume each side–muscle pair has 3-axis acceleration (a_x, a_y, a_z) and 3-axis gyroscope (g_x, g_y, g_z), and let $\varepsilon = 10^{-6}$.

A) Vector magnitudes (2 derived features per side–muscle; 16 derived magnitude features in total)

For each side $s \in \{R, L\}$and muscle m:

Acceleration magnitude

$$A_{s,m} = \sqrt{a_{x,s,m}^2 + a_{y,s,m}^2 + a_{z,s,m}^2}$$

Gyroscope magnitude

$$G_{s,m} = \sqrt{g_{x,s,m}^2 + g_{y,s,m}^2 + g_{z,s,m}^2}$$

B) Left–right features computed on magnitudes (6 derived features per muscle; 24 derived features in total)

Let $A_{R,m},\ A_{L,m}$be right/left acceleration magnitudes and $G_{R,m},\ G_{L,m}$be right/left gyroscope magnitudes.

Difference

$$d_m^{acc} = A_{R,m} - A_{L,m}, \qquad d_m^{gyr} = G_{R,m} - G_{L,m}$$

Ratio

$$\text{ratio}_m^{acc} = \frac{A_{R,m}}{A_{L,m} + \varepsilon},\ \text{ratio}_m^{gyr} = \frac{G_{R,m}}{G_{L,m} + \varepsilon}$$

Asymmetry

$$\text{asym}_m^{acc} = \frac{A_{R,m} - A_{L,m}}{A_{R,m} + A_{L,m} + \varepsilon}, \qquad \text{asym}_m^{gyr} = \frac{G_{R,m} - G_{L,m}}{G_{R,m} + G_{L,m} + \varepsilon}$$

C) Global IMU magnitude summaries (3 derived features)

Let magnitudes $\{z_1, \ldots, z_{16}\}$ denote all 16 IMU magnitude features from Part A) corresponding to 4 muscles, 2 sides (right, left), 2 signal types (acceleration and gyroscope)

$$\mu_{\text{mag}} = \frac{1}{16}\sum_{i=1}^{16} z_i, \qquad \sigma_{\text{mag}} = std(z_1, \ldots, z_{16}), \qquad {}_{\text{mag}}max = \max_i z_i$$

3.5 Row-to-Trial Aggregation (Mean/Std/Min/Max)

Row-level features were aggregated to the trial level by grouping observations according to (subject_id, exercise_id, trial_id).

At the row level, a total of **125 features** were computed per time sample:

- 34 EMG features (8 raw + 26 derived)
- 91 IMU features (48 raw + 43 derived)

For each row-level feature $f$within a trial t, with samples $\{f_{t1}, \ldots, f_{tn_t}\}$, the following summary statistics were computed:

$$\text{mean}_t(f) = \frac{1}{n_t}\sum_{i=1}^{n_t} f_{ti} \text{std}_t(f) = \sqrt{\frac{1}{n_t - 1}\sum_{i=1}^{n_t}(f_{ti} - \text{mean}_t(f))^2} \min_t(f) = \min_i f_{ti}, \max_t(f) = \max_i f_{ti}$$

Thus, each row-level feature produced four aggregated statistics.
Since 125 features were aggregated, each trial was ultimately represented by:

$$125 \times 4 = \mathbf{500}$$

trial-level features.

Each trial was therefore encoded as a 500-dimensional feature vector combining aggregated EMG and IMU statistics. These combined trial-level features served as inputs to the hierarchical 2-stage LightGBM classifier.

3.6 Trial-Level Per-Subject Z-Score Normalization

After row-to-trial aggregation, each trial was represented by a 500-dimensional feature vector, consisting of:

- 136 aggregated EMG features (34 $\times$ 4 statistics), and
- 364 aggregated IMU features (91 $\times$ 4 statistics).

To reduce inter-subject variability, each trial's 500-dimensional feature vector was standardized relative to the distribution of trials from the same subject, reducing inter-subject variability while preserving within-subject movement patterns.

3.6.1 Definition

For each subject s,

- Let $T_s = \{t_1, t_2, \ldots, t_{n_s}\}$ denote the set of all trials belonging to subject s, where n_s is the number of trials for that subject.
- Let $x_{t,j}$ denote the value of the j-th trial-level feature (out of 500 total features) for trial t.

3.6.2 Subject-Specific Mean and Standard Deviation

For each subject $s$and each feature $j \in \{1, \ldots, 500\}$, compute:

$$\mu_{s,j} = \frac{1}{n_s} \sum_{t \in T_s} x_{t,j} \sigma_{s,j} = \sqrt{\frac{1}{n_s - 1} \sum_{t \in T_s} (x_{t,j} - \mu_{s,j})^2}$$

These statistics are computed using only trials from the **same** subject.

3.6.3 Within-Subject Standardization

Each trial feature is then standardized as:

$$z_{t,j} = \frac{x_{t,j} - \mu_{s,j}}{\max(\sigma_{s,j}, 1)}$$

where:

- The denominator is lower-bounded by 1 to prevent division by very small values.
- This ensures numerical stability and avoids amplifying noise when variance is near zero.

4 Machine Learning Pipeline

A two-stage hierarchical classification approach [4] was developed to distinguish between correct knee extension and two types of incorrect execution (LL and NF). Instead of training a single flat three-class model, the problem is decomposed into two sequential binary decision stages.

In **Stage-1**, the model performs a coarse classification to determine whether a movement is **Correct or Incorrect**. This stage acts as a screening step, focusing on the most clinically important distinction—identifying movements that deviate from proper rehabilitation form.

In **Stage-2**, only those trials predicted as incorrect are further classified into the two specific error types: **Wrong_LL (low lift)** and **Wrong_NF (not fully extended)**. This second stage focuses on distinguishing between subtle differences within incorrect movements, which are more similar to each other than to correct execution.

This hierarchical design simplifies the learning task at each stage, reduces class confusion, and allows independent threshold control for each decision. In particular, Stage-1 can be tuned to minimize false "Correct" predictions, which is critical in rehabilitation settings. As a result, the proposed approach provides more stable, interpretable, and practically meaningful predictions compared to a flat three-class classifier.

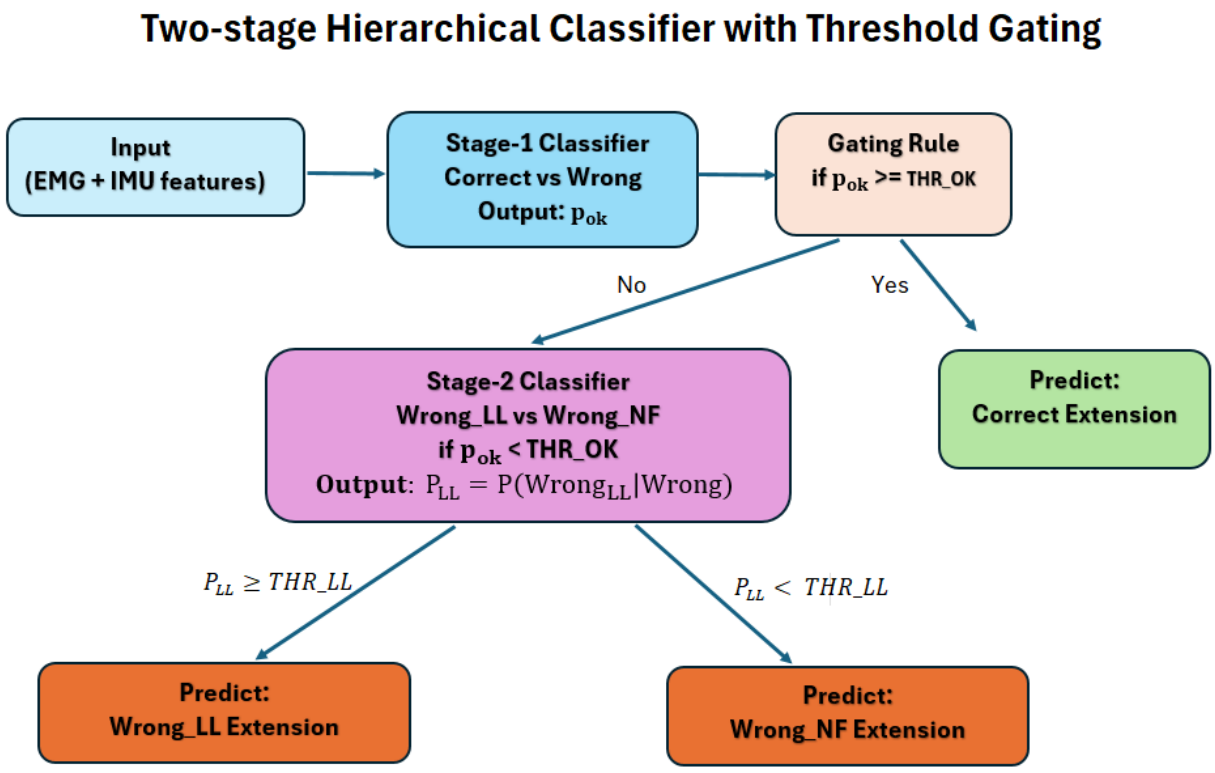

Figure 5: Two-stage hierarchical classifier with threshold gating for extension movement classification.

4.1 Stage-1: Correct vs. Wrong (Gating Layer)

After the subject-held-out split, 80% of the data was used for training. Within this training set, 85% was used to train the Stage-1 LightGBM model and 15% was reserved for isotonic calibration, resulting in approximately 68% of the total dataset used for Stage-1 model fitting and 12% for calibration.

The purpose of Stage-1 is to determine whether a knee extension is performed correctly or incorrectly.

The Stage-1 classifier outputs a calibrated probability:

$$p_{OK} = P(\text{Correct})$$

This value represents how confident the model is that a trial is correct.

A minimum decision threshold was applied:

$$THR_{OK} = 0.56$$

Decision rule:

- If $p_{OK} \geq 0.56$, the trial is classified as **Correct**
- If $p_{OK} < 0.56$, the trial is routed to Stage-2 for further analysis

The threshold of 0.56 was selected using validation data to achieve a favorable balance between precision and recall, reducing the number of incorrect trials mistakenly labeled as correct.

Stage-1 acts as a "gate," allowing clearly correct trials to exit early while sending uncertain or incorrect trials to deeper evaluation.

4.2 Stage-2: Error Subtype Classification (LL vs. NF)

For Stage-2, the model was trained only on the subset of training samples with true "Wrong" labels (Wrong_LL and Wrong_NF).

Stage-2 determines the type of incorrect extension. Importantly, Stage-2 was trained using **all true incorrect extension trials** (LL and NF) in the training dataset. It was not trained only on samples rejected by Stage-1. This ensures that Stage-2 learns from the full distribution of incorrect movements and does not depend on Stage-1 errors.

However, during prediction, Stage-2 is only used for trials that fail the Stage-1 threshold. The Stage-2 classifier outputs:

$$p_{LL} = P(\text{LL} \mid \text{Wrong})$$

A threshold of 0.50 was applied. Unlike Stage-1, where a threshold of 0.56 was carefully chosen to control false "Correct" predictions, Stage-2 does not involve safety-critical decisions. Therefore, using 0.50 provides a neutral and statistically standard decision boundary.

$$THR_{LL} = 0.50$$

Decision rule:

- If $p_{LL} \geq 0.50$, predict **Wrong_LL**
- Otherwise, predict **Wrong_NF**

5 Overall Stage-1 and Stage-2 Probability Computation

Let:

- C= Correct
- W= Wrong
- LL= Wrong–LL
- NF= Wrong–NF

Stage-1 outputs:

$$\begin{aligned} p_{OK} &= P(C \mid x), \\ p_W &= P(W \mid x) = 1 - p_{OK}. \end{aligned}$$

Stage-2 outputs the conditional probability:

$$\begin{aligned} p_{LL|W} &= P(LL \mid W, x), \\ P(NF \mid W, x) &= 1 - p_{LL|W}. \end{aligned}$$

Applying the law of total probability:

$$P(LL \mid x) = P(LL \mid C, x)P(C \mid x) + P(LL \mid W, x)P(W \mid x).$$

Because $P(LL \mid C, x) = 0$ and $P(W \mid x) = 1 - p_{OK}$,

$$\boxed{P(LL \mid x) = (1 - p_{OK})\, p_{LL|W}}$$

Similarly,

$$\begin{aligned} P(NF \mid x) &= P(NF \mid C, x)P(C \mid x) + P(NF \mid W, x)P(W \mid x) \\ &= 0 + (1 - p_{OK})(1 - p_{LL|W}). \end{aligned}$$

Therefore,

$$\boxed{P(NF \mid x) = (1 - p_{OK})(1 - p_{LL|W})}$$

By the law of total probability, the final three class probabilities are:

$$\boxed{\begin{aligned} P(C \mid x) &= p_{OK}, \\ P(LL \mid x) &= (1 - p_{OK})\, p_{LL|W}, \\ P(NF \mid x) &= (1 - p_{OK})(1 - p_{LL|W}). \end{aligned}}$$

6 LightGBM for Binary Classification

The probabilities used in Stage-1 and Stage-2 are produced by machine learning classifiers trained on EMG and IMU features. In this study, we used Light Gradient Boosting Machine (LightGBM) [3], a tree-based ensemble learning algorithm, to model the relationship between sensor features and movement outcomes. Each stage of the hierarchical system uses a separate LightGBM binary classifier to estimate the likelihood of a particular class. These models first produce raw prediction scores, which are then converted into probabilities and calibrated before applying the decision thresholds described above.

LightGBM performs binary classification by estimating the probability:

$$p = P(y = 1 \mid x)$$

However, internally the model does not directly predict probabilities. Instead, it produces a **raw prediction score**:

$$s(x) \in (-\infty, +\infty)$$

This raw score is then transformed into a probability using the sigmoid (logistic) function:

$$p = \sigma(s) = \frac{1}{1 + e^{-s}}$$

6.1 Objective Function: Binary Cross-Entropy (Log Loss)

LightGBM is trained by minimizing the binary cross-entropy loss (log loss):

$$L(y, p) = -[y \log(p) + (1 - y) \log(1 - p)]$$

where:

- $y \in \{0, 1\}$ is the true label
- $p = \sigma(s) = \frac{1}{1+e^{-s}}$ is the predicted probability

The log loss measures how close the predicted probability p is to the true label y.

- If $y = 1$, the loss becomes $-\log(p)$, which is minimized when $p \to 1$.
- If $y = 0$, the loss becomes $-\log(1 - p)$, which is minimized when $p \to 0$.

Thus, the loss is small when the predicted probability matches the true label and increases rapidly when the model assigns high probability to the wrong class.

6.2 Initialization

Before building any trees, LightGBM starts with a constant prediction equal to the log-odds of the label mean:

$$\bar{y} = \text{mean of training labels}$$
$$s_0 = \log\left(\frac{\bar{y}}{1 - \bar{y}}\right)$$

All samples begin with this same raw score.

6.3 Boosting Mechanism: Iterative Error Correction

LightGBM builds trees sequentially to minimize the log loss. The overall raw prediction score after K trees is:

$$s(x) = s_0 + \sum_{k=1}^{K} \eta \, t_k(x)$$

where:

- $s_0$is the initial prediction
- $t_k(x)$ is the leaf value of the k-th tree corresponding to input x
- η is the learning rate (to prevent the model from changing too fast and overfitting)

After each tree is added, the model updates its raw prediction score:

$$s^{(k)}(x) = s^{(k-1)}(x) + \eta\, t_k(x)$$

The updated raw score is then converted into a predicted probability using the sigmoid function:

$$p^{(k)}(x) = \sigma(s^{(k)}(x))$$

This updated probability $p^{(k)}(x)$ becomes the starting point for constructing the next $(k+1)$-th tree. Therefore, each new tree does not start from scratch. Instead, it is built specifically to correct the remaining errors from all previous trees.

Over many iterations, this sequential error-correction process progressively reduces the overall binary cross-entropy loss.

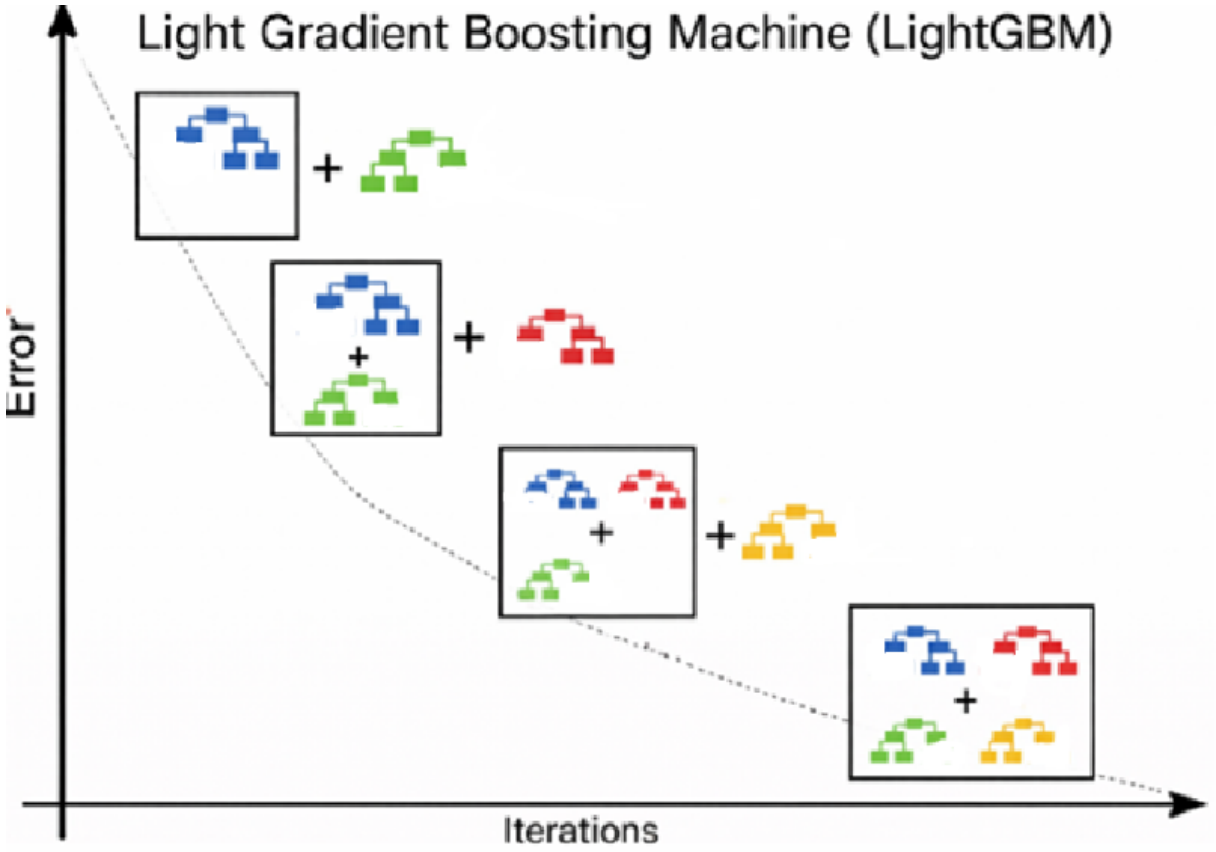

Figure 6: Conceptual illustration of the Light Gradient Boosting Machine (LightGBM) training process.

6.4 Tree Construction and Split Selection

At each boosting iteration, LightGBM constructs a new decision tree to reduce the current log loss.

6.4.1 Compute First- and Second-Order Derivatives

For each training sample i, let:

$$p_i = \sigma(s_i)$$

be the predicted probability, where s_i is the current raw prediction score produced by the ensemble of trees.

The derivatives of the log loss are:

- Gradient (first derivative): measures how wrong the prediction is.

$$g_i = p_i - y_i$$

- Hessian (second derivative): measures the confidence and stability of the current prediction.

$$h_i = p_i(1 - p_i)$$

These values guide how the tree should correct the model.

6.4.2 Evaluate Candidate Splits

At each node, LightGBM considers possible feature splits. For a candidate split dividing the data into left (L) and right (R) child nodes, define:

$$G_L = \sum_{i \in L} g_i, \quad G_R = \sum_{i \in R} g_i H_L = \sum_{i \in L} h_i, \quad H_R = \sum_{i \in R} h_i$$

where:

- $G_L,\ G_R$ are sums of gradients
- $H_L,\ H_R$ are sums of Hessians

The quality of the split is measured by the gain formula:

$$\text{Gain} = \frac{1}{2}\left(\frac{\widetilde{G}_L^2}{H_L + \lambda_2} + \frac{\widetilde{G}_R^2}{H_R + \lambda_2} - \frac{\left(\widetilde{G}_L + \widetilde{G}_R\right)^2}{H_L + H_R + \lambda_2}\right)$$

The adjusted gradient is computed using the threshold function:

$$\widetilde{G} = threshold(G, \lambda_1)$$

$$threshold(G, \lambda_1) = \begin{cases} G - \lambda_1, & G > \lambda_1 \\ 0, & \mid G \mid \leq \lambda_1 \\ G + \lambda_1, & G < -\lambda_1 \end{cases}$$

where:

- λ_1 is L1 regularization which applies a threshold to gradient sums, removing weak signals and preventing unnecessary splits.
- λ_2 is L2 regularization that appears in the denominator of the gain formula and shrinks leaf values, reducing the magnitude of model updates.
- λ_1 *and* λ_2 together help improve model stability and reduce overfitting.
- $\widetilde{G}_L, \widetilde{G}_R$ are adjusted gradient sums after applying L1 regularization

Gain measures how much the loss decreases if we split. If Gain > 0, it means splitting reduces loss, so the split is accepted. If Gain is less than or equal to zero, the node remains a leaf, because further splitting would not improve model performance.

During tree construction, LightGBM evaluates, for each current leaf node, all possible feature splits and identifies the split that produces the highest gain for that leaf. It then compares these best gains across all leaves and selects the single leaf whose split yields the largest positive gain. That leaf is split into two child nodes. This process repeats until no further positive-gain splits remain.

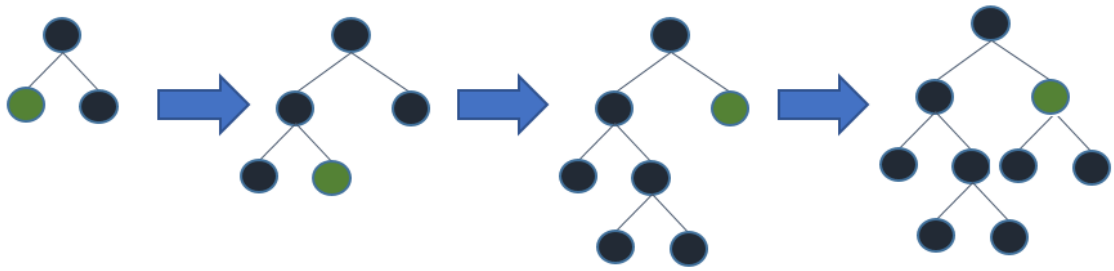

Figure 7: Leaf-wise growth for one tree. Illustration of leaf-wise growth for a single decision tree. At each step, the leaf with the largest gain is selected and split, resulting in an asymmetric tree structure that grows along the most informative branch until no remaining leaf produces a positive gain.

6.4.3 Compute Leaf Value

In LightGBM, trees are built sequentially. Before a new tree is created, each training sample already has a prediction score from all the previous trees combined. Based on how far these predictions are from the true labels, the algorithm calculates how wrong the model currently is.

The new tree is built to correct those mistakes. Inside the tree, each leaf is given a value that reduces the training error for the samples that fall into that region. This leaf value acts like a small adjustment that will be added to the model's existing prediction score.

If the model's predictions in that region are too low, the leaf value will be positive and increase the score. If the predictions are too high, the leaf value will be negative and decrease the score. In this way, each new tree gradually fixes the errors made by the previous trees.

Leaf Statistics

For a leaf containing samples i, define:

$$G = \sum g_i, \quad H = \sum h_i$$

The optimal leaf value is:

$$v = -\frac{threshold(G, \lambda_1)}{H + \lambda_2}$$

The threshold function used for L1 regularization is:

$$threshold(G, \lambda_1) = \begin{cases} G - \lambda_1, & G > \lambda_1 \\ 0, & |\, G \,| \leq \lambda_1 \\ G + \lambda_1, & G < -\lambda_1 \end{cases}$$

where:

- λ_1 is L1 regularization which applies a threshold to gradient sums, removing weak gradient signals before computing the leaf value.
- λ_2 is L2 regularization that appears in the denominator of the leaf value formula and shrinks the resulting leaf values.

- $\lambda_1$and λ_2 together prevent overly aggressive model updates and improve generalization.
- If the model underpredicts (many $g_i < 0$), $v > 0$
- If the model overpredicts (many $g_i > 0$), $v < 0$
- If predictions are accurate, $v \approx 0$

Tree Output

Consider a fully constructed k-th tree with multiple terminal leaves.
Each leaf ℓ has an assigned leaf value $v_{k,\ell}$, computed during training.

During prediction, given a new input x, the tree applies its sequence of split rules and directs x to one terminal leaf. Each leaf corresponds to a specific region of the feature space defined by the tree's decision boundaries.

The output of the k-th tree for input x is the leaf value stored in the terminal leaf reached by x:

$$t_k(x) = v_{k,\text{leaf}(x)}$$

6.4.4 Update the Raw Prediction Score

Each tree contributes a scaled correction:

$$\eta\, t_k(x)$$

where η is the learning rate to prevent the model from changing too fast and overfitting.

After K trees, the overall raw score becomes:

$$s(x) = s_0 + \sum_{k=1}^{K} \eta\, t_k(x)$$

where:

- $s_0$is the initial score
- Each tree gradually refines the prediction

6.4.5 From the Raw Prediction Score to Predicted Probability

The final LightGBM probability is:

$$p_{\text{GBM}}(x) = \sigma\left(s(x)\right) = \frac{1}{1 + e^{-s(x)}}$$

Positive raw scores indicate increasing confidence in class 1; negative scores indicate increasing confidence in class 0.

7 Probability Calibration with Isotonic Regression

Although LightGBM converts raw prediction scores into probabilities using a sigmoid function, these probabilities may not be perfectly calibrated.

LightGBM is a gradient boosting model designed to optimize classification performance, but it can produce irregular probability spacing. For example, predictions around 0.90 might show a lower observed positive rate than predictions around 0.80. This means the probability scale is not properly aligned with real-world outcomes.

To address this issue, an isotonic regression was applied, which learns a monotonic mapping:

$$P_{\text{cal}} = f(P_{\text{GBM}})$$

where $P_{\text{GBM}}$is the original LightGBM probability and $P_{\text{cal}}$is the calibrated probability.

Isotonic regression adjusts the probability values while preserving their ranking order. If neighboring probability regions violate monotonicity, they are automatically pooled and reassigned consistent calibrated values. This restores a smooth increasing relationship between predicted probability and observed positive rate.

Calibration was performed on validation data and applied to test predictions before thresholding. This step improved the reliability and interpretability of the probabilities used in our hierarchical decision process.

8 Results

8.1 Overall Three-Class Performance (Subject-Held-Out Test Set)

The hierarchical EMG+IMU classifier was evaluated on a subject-held-out test set (N = 189 trials). In this evaluation, the model was tested on data from participants who were completely excluded from the training process, ensuring that the system was evaluated on unseen individuals rather than memorized movement patterns.

The final system achieved:

- Accuracy: 92.6%
- Macro-F1: 0.926
- Macro ROC-AUC (One-vs-Rest): 0.978

These results indicate that the classifier performs consistently well across all three movement categories and has excellent ability to distinguish between classes.

The confusion matrix is shown below (rows = true class, columns = predicted class):

True \ Predicted	Wrong_NF	Wrong_LL	Correct
Wrong_NF (62)	52	8	2
Wrong_LL (66)	1	63	2
Correct (61)	0	1	60

Figure 8: Probability calibration with isotonic regression.

Per-class F1-scores:

- Wrong_NF: 0.904
- Wrong_LL: 0.913
- Correct: 0.960

These results show that the model performs especially well at identifying correct movements, while still maintaining high accuracy in detecting both types of incorrect knee extensions.

Overall, the model demonstrated strong and balanced performance across all three movement categories, indicating that combining EMG muscle signals with IMU motion signals provides useful information for detecting movement quality.

8.2 Stage-1: Correct vs. Wrong Gating

Stage-1 separates Correct trials from all Wrong trials (Wrong_LL + Wrong_NF).

Using a calibrated probability threshold of THR_OK = 0.56, the model achieved:

- Precision (Correct): 0.938
- Recall (Correct): 0.984
- Accuracy: 97.4%

These results indicate that the model is very reliable at identifying correctly performed knee extension, while also minimizing the chance of incorrectly labeling a wrong knee extension as correct.

Importantly:

- 99.2% of true Wrong trials were successfully passed to Stage-2
- Only 1 Correct trial was missed

This means Stage-1 acts as a high-recall safety gate, ensuring that almost all incorrect movements are forwarded for further analysis rather than being mistakenly labeled as correct.

Stage-1 forwarded 66.1% of trials to Stage-2 for detailed classification of the specific error type.

8.3 Stage-2: Wrong_LL vs. Wrong_NF

Stage-2 classifies only trials that Stage-1 identifies as incorrect knee extension, separating them into two specific error types:

- Wrong_LL: insufficient knee lift
- Wrong_NF: incomplete extension

Two evaluation scenarios were considered.

Evaluation on True Incorrect Cases Only

Under ideal conditions (evaluated on all true Wrong trials):

- Accuracy: 92.97%
- F1-score: 0.935
- ROC-AUC: 0.970
- PR-AUC: 0.959

For detecting Low-Load errors (LL):

- Precision: 0.890
- Recall: 0.985

These results show that the model has excellent ability to distinguish between the two types of incorrect knee extension movements, with particularly strong recall for detecting low-lift errors.

End-to-End Pipeline Evaluation (Using Stage-1 Predictions)

This reflects real-world deployment, where Stage-2 operates on the output of Stage-1 rather than ground-truth labels. Under realistic pipeline conditions (after Stage-1 filtering):

- Accuracy: 92.74%
- F1-score: 0.933
- ROC-AUC: 0.969
- PR-AUC: 0.958

For detecting Low-Lift errors (LL):

- Precision: 0.887
- Recall: 0.984

Performance decreased only slightly compared to the oracle setting, demonstrating that Stage-1 filtering process does not significantly distort Stage-2 classification results.

8.4 Summary of Findings

A hierarchical three-class classifier combining EMG muscle activation signals and IMU motion measurements was successfully developed to evaluate knee extension exercises.

The hierarchical EMG+IMU classifier demonstrated:

- Strong three-class discrimination (Accuracy $\approx$ 93%)
- Near-perfect Wrong coverage in Stage-1 (99.2%)
- Excellent separation between Low-Lift and Not-Fully-Extended errors in Stage-2 (ROC-AUC $\approx$ 97%)
- Minimal performance degradation in realistic pipeline operation

These results suggest that the two-stage architecture provides both high reliability and strong class separability, even when evaluated on participants not seen during training.

This makes the system promising for future wearable physical-therapy monitoring systems, where automated feedback could help patients perform rehabilitation exercises more accurately outside the clinic.

9 Discussion

This study presents a hierarchical two-stage EMG+IMU-based classifier for assessing knee extension quality, achieving strong performance on a subject-held-out test set (Accuracy $\approx$ 92.6%, Macro-F1 $\approx$ 0.926, ROC-AUC $\approx$ 0.978) . The model demonstrated particularly high reliability in distinguishing correct versus incorrect movements in Stage-1, with near-complete coverage of incorrect trials (99.2%), and strong discrimination between error subtypes (LL vs NF) in Stage-2. Importantly, performance under real pipeline conditions closely matched the idealized evaluation, indicating minimal degradation when the two stages are combined.

The observed performance can be attributed to both the hierarchical design and the multimodal feature representation. By decomposing the problem into two sequential decisions, the model simplifies the learning task at each stage. Stage-1 focuses on the clinically critical distinction between correct and incorrect execution, which is a relatively separable problem due to clear differences in both muscle activation and movement dynamics. In contrast, Stage-2 addresses a more subtle classification problem between two similar incorrect patterns. This separation reduces class confusion and allows each model to specialize. Additionally, the integration of EMG and IMU signals provides complementary information—EMG captures neuromuscular activation patterns while IMU reflects kinematic behavior—enabling the model to detect both physiological and biomechanical deviations in movement.

The system also demonstrates strong robustness and generalization. The subject-held-out evaluation ensures that the model is tested on entirely unseen individuals, reducing the likelihood of overfitting to subject-specific patterns. Furthermore, the close agreement between oracle evaluation (true wrong cases only) and real pipeline evaluation suggests low cascade error and stable probability calibration across stages. This indicates that Stage-1 filtering does not introduce significant distribution shift for Stage-2, and that the calibrated probabilities remain consistent across different operating conditions. However, potential failure modes still exist. Misclassifications may occur in borderline cases where movements are only slightly incorrect, or when EMG signals are noisy due to sensor placement variability. Additionally, extreme or atypical movement patterns not well represented in the training data could reduce performance.

From a practical and clinical perspective, the hierarchical design aligns well with real-world rehabilitation needs. Stage-1 acts as a safety gate, minimizing the risk of incorrectly labeling a faulty movement as correct, which is critical in preventing improper exercise reinforcement. Stage-2 provides more granular feedback on the type of error, which could support targeted coaching (e.g., improving range of motion vs. increasing lift height). This makes the system suitable for deployment in wearable or home-based rehabilitation settings, where automated, real-time feedback can enhance adherence and reduce the need for constant clinical supervision.

Despite these strengths, several limitations should be acknowledged. First, the dataset size is relatively modest (31 subjects), which may limit the model's ability to capture the full variability of patient populations. Second, although the dataset was collected in an unsupervised setting, it still represents a controlled experimental environment and may not fully reflect real-world home conditions, where sensor placement and movement variability are less consistent. Third, the current model focuses only on seated leg extension and does not generalize directly to other rehabilitation exercises. Finally, while probability calibration improves decision reliability, threshold selection (e.g., THR_OK = 0.56) may need to be re-tuned for different clinical priorities or populations.

Overall, the results demonstrate that a hierarchical EMG+IMU-based approach can provide accurate, stable, and clinically meaningful assessment of rehabilitation movements. Future work may focus on expanding the dataset, incorporating additional exercise types, and validating the system in real-world deployment scenarios to further improve generalization and clinical applicability.

Acknowledgement

I am very grateful to Professor William Hsu from UCLA for his invaluable mentorship and support for this project.

I used ChatGPT AI tool to brainstorm ideas, to draft and debug the Python codes, and to revise wording for clarity. I also used Perplexity AI to help identify and explore relevant research papers for this project.

References

[1] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[2] Panagiotis Kasnesis, Theodora Plavoukou, Anna Contiero Syropoulou, Loukas Toumanidis, and Georgios Georgoudis. A knee rehabilitation exercises dataset for postural assessment using wearable devices. *Scientific Data*, 12:610, 2025. doi:10.1038/s41597-025-04963-4.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154, 2017.

[4] J. Mao et al. Two-stage classification for robustness in machine learning. *arXiv preprint arXiv:2206.05323*, 2022. https://arxiv.org/abs/2206.05323.

A Brief Explanation of Evaluation Metrics

To evaluate classifier performance, several standard machine learning metrics were used. This appendix gives a brief explanation of the main metrics reported in this study.

A.1 Confusion Matrix

A confusion matrix compares predicted labels with true labels and shows where correct classifications and errors occur. It is useful for identifying which classes are most often confused.

In this project:

- **Stage-1** separates **Correct** from **Wrong**
- **Stage-2** separates **Wrong_LL** from **Wrong_NF**

A.2 Accuracy

Accuracy is the proportion of all predictions that were correct.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

It measures overall classification performance, but it can be misleading when class sizes are uneven.

A.3 Precision

Precision measures how often a predicted positive class is actually correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

High precision means the model makes few false positive errors.

A.4 Recall

Recall measures how many true positive cases are correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN}$$

High recall means the model misses very few true cases.

A.5 F1-Score

The F1-score is the harmonic mean of precision and recall.

$$\mathrm{F1} = \frac{2(\mathrm{Precision} \cdot \mathrm{Recall})}{\mathrm{Precision} + \mathrm{Recall}}$$

It is useful when both false positives and false negatives matter.
Macro-F1 gives equal weight to each class.

A.6 ROC-AUC

The ROC curve plots [1] recall against false positive rate across different thresholds.
The **ROC-AUC** summarizes this curve into a single value from 0 to 1, where a higher value indicates better class separation.

A ROC-AUC of 0.97 indicates excellent discrimination between the two classes.

A.7 PR-AUC

The Precision-Recall curve shows the tradeoff between precision and recall across thresholds.
PR-AUC is especially useful when class imbalance is present or when correct detection of one class is particularly important.

A.8 Threshold and Gating

A probability threshold determines how predicted probabilities are converted into class labels. In this hierarchical model, the Stage-1 threshold also controls whether a trial is classified as **Correct** or forwarded to Stage-2 for further classification.

A.9 Coverage

Coverage describes how many samples are passed from Stage-1 to Stage-2.

$$\mathrm{Coverage} = \frac{\text{Samples sent to Stage-2}}{\text{Total samples}}$$

High Wrong-case coverage means most true incorrect trials are successfully forwarded for subtype classification.