Elderly Fall Prediction using IoMT Sensor Data: A Machine Learning and FastAPI Framework

Nursena Ercan
UG Scholar, Dept. of Computer Engineering
Haliç University, Istanbul, Türkiye
22092090020@ogr.halic.edu.tr

Mohammed Al-Hubaishi
Assistant Professor, Dept. of Computer Engineering
Haliç University, Istanbul, Türkiye
mohammedhbi@gmail.com

I. Introduction

Falls among elderly individuals continue to be a major public health concern, often resulting in injuries, hospitalization, and reduced independence. With the growing use of Internet of Medical Things (IoMT) devices, sensor-based monitoring systems provide valuable data that can support early fall prediction and timely intervention. The dataset used in this study originates from the cStick project, an existing IoMT-based smart cane designed to collect physiological and movement-related measurements such as heart rate variability (HRV), SpO_2, blood glucose level, pressure, accelerometer activity, and obstacle distance.

Although the device and dataset are pre-existing, the focus of this research is on conducting a complete machine learning workflow using this data. This includes comprehensive data exploration, preprocessing, feature analysis, and the development of predictive models capable of classifying user states into safe, warning, or fall conditions. Multiple machine learning algorithms were trained and evaluated to determine the most effective model for real-time fall prediction, and the final system was deployed through a FastAPI backend to support both single-sample and batch inference.

By transforming raw IoMT sensor readings into an operational ML-based fall prediction framework, this study demonstrates how existing datasets can be leveraged to build practical and deployable solutions for elderly safety and remote healthcare monitoring.

II. Literature Review and Related Projects

B.1) Previous Research

Fall detection and prediction have been extensively explored in the fields of biomedical engineering, IoT, and ubiquitous healthcare. Igual et al. conducted a comprehensive review of fall-detection systems and highlighted the evolution from threshold-based techniques to machine learning-driven models, emphasizing the importance of wearable sensing for improving sensitivity and reducing false alarms [1]. Li et al. further analyzed advances in wearable-based fall detection and demonstrated that sensor fusion and classification models consistently outperform single-sensor methods in real-world scenarios [2]. Ensemble learning approaches have also been shown to improve accuracy and robustness, particularly when accelerometer, gyroscope, and physiological signals are jointly analyzed [3].

IoT-based healthcare architectures play a significant role in enabling continuous monitoring of elderly individuals. Jara et al. presented an IoT-integrated fall detection framework that combined wearable sensors with cloud analytics to enable real-time emergency notification and remote patient tracking [**?**]. Kodali et al. expanded this concept by proposing a multi-layer IoT architecture incorporating edge, fog, and cloud layers to reduce latency and enhance scalability for large-scale elderly monitoring environments [5]. Edge-computing-based systems have also been proposed for fall prediction, enabling local processing and reducing dependence on cloud infrastructure, which is especially useful for resource-limited deployments [6].

The cStick system is an example of an IoMT-enabled smart cane specifically designed for elderly safety. Rachakonda et al. demonstrated that integrating distance sensors, grip-pressure sensing, heart-rate variability (HRV), blood glucose estimation, oxygen saturation, and accelerometer data enables reliable classification of safe, warning, and fall events [7]. Other assistive devices, such as the ASCane, similarly use inertial sensors embedded into a mobility aid and apply rule-based or machine learning techniques to distinguish normal walking from sudden fall events [**?**]. Building on these contributions, the present study focuses not on device design but on developing a complete machine learning workflow—data preprocessing, feature analysis, model training, evaluation, and API deployment—using the publicly available cStick dataset.

B.2) Relation to Networking Technology and Other Applications

IoMT-based fall detection systems rely heavily on the networking infrastructure that connects sensing devices, processing units, and monitoring platforms. Modern systems utilize short-range wireless technologies such as Bluetooth Low Energy (BLE) for sensor-to-gateway communication, Wi-Fi for home-based monitoring, and cellular networks for outdoor mobility. Igual et al. emphasized that communication reliability and latency are central to fall detection performance, as delays in transmitting sensor data may hinder timely intervention [1]. IoT architectures with edge and fog layers offer reduced latency and improved scalability by enabling

computation closer to the user, aligning with the frameworks proposed by Kodali et al. [5].

Networking technologies also enable integration with mobile health (mHealth) applications, caregiver dashboards, emergency alert platforms, and cloud-based analytics. Sultan et al. demonstrated that combining wearable sensors with edge computing and cloud services supports advanced monitoring of elderly patients with chronic conditions [6]. This principle directly applies to the present system: once the machine learning model is deployed as a RESTful API, any IoMT device or software client can transmit sensor readings and receive predictions over standardized HTTP/HTTPS communication channels. Such architectures support interoperability, scalability, and continuous updating of models as new data becomes available.

Beyond fall detection, these network-enabled systems can be employed in rehabilitation tracking, gait analysis, chronic disease monitoring, and general mobility assessment. Krishna et al. showed that multi-sensor IoT systems can support precise physical activity monitoring and integrate with telemedicine infrastructures [9]. Thus, the proposed fall prediction API can be extended to broader smart healthcare applications.

Fall Detection for Elderly Care: Helps prevent accidents and provides real-time alerts. Health Monitoring: Tracks vital signs continuously to detect health risks early. Smart Healthcare Systems: Integrates with mobile apps and IoT platforms for remote monitoring.

B.3) Real-World Scenarios

Several real-world environments can benefit from the proposed system:

1) Assisted Living Facilities: Residents using IoMT-enabled smart canes can be continuously monitored, with instant alerting to nursing staff when fall-related patterns or events are detected. Such systems have been shown to significantly reduce emergency response time [**?**].

2) Home Monitoring for High-Risk Patients: Older adults with cardiovascular, respiratory, or metabolic disorders often experience instability related to HRV fluctuations, low SpO_2, or abnormal glucose levels. The system provides continuous monitoring and early warning, reducing the likelihood of severe injuries [6].

3) Rehabilitation and Physiotherapy Centers: Clinicians can monitor gait patterns and detect imbalance or fall risk by analyzing sensor trends through batch prediction tools.

4) Outdoor Mobility Safety: When connected to a smartphone, the system can transmit fall events along with GPS coordinates to caregivers or emergency services.

5) Urban Public Health and Smart-City Integration: Aggregated anonymized data may help identify high-risk locations for falls, supporting safer city planning [9].

B.4) Proposed Applications

The cStick Elderly Fall Prediction dataset enables the development of several practical and impactful applications that enhance elderly mobility, safety, and remote healthcare capabilities.

6) A. Real-Time Fall Prediction System: The machine learning model classifies readings into:

1) Safe,
2) Warning,
3) Fall Detected.

Alerts can be transmitted automatically to caregivers or emergency services.

7) Smart Home Elderly Monitoring: Integrates predictions with home automation platforms for continuous monitoring.

8) IoMT-Based Remote Health Monitoring: Physiological indicators (HRV, SpO_2, glucose) allow early detection of medical risks.

9) Clinical Decision Support and Preventive Care: Long-term records help clinicians identify high-risk individuals and recommend interventions.

10) Research and Mobility Assessment Tools: Enables studies on gait, physiology, and multi-sensor fusion.

11) Integration into Wearable and Assistive Devices: Lightweight models can run on embedded systems and smart mobility aids.

III. Methodology

A. Dataset Information

Name of the Dataset: cStick Elderly Fall Prediction (project dataset)

Source: https://www.kaggle.com/datasets/laavanya/elderly-fall-prediction-and-detection

B. Dataset Description

This dataset comes from a smart cane called cStick, designed to help elderly people avoid falls. It includes sensor readings and health information collected from users. The format is CSV, and the size of the dataset is 2,039 rows × 7 columns (about 100 KB).

The main features are:

- **Distance** – how far the user is from obstacles
- **Pressure** – how hard the person grips the stick
- **HRV (Heart Rate Variability)** – heart rhythm changes
- **Sugar level** – blood sugar estimate
- **SpO_2** – blood oxygen level
- **Accelerometer** – movement or shaking patterns
- **Decision** – label showing the situation:
  - 0 → Safe / No fall
  - 1 → Warning / Possible fall
  - 2 → Fall detected

C. Initial Thoughts on Potential Analyses

The cStick Elderly Fall Prediction dataset provides several opportunities for exploratory analysis and model development due to the diversity of its sensor measurements. Before constructing predictive models, it is essential to understand how each sensor behaves across different user states and to identify which features contribute most strongly to distinguishing between safe, warning, and fall conditions.

A. Exploration of Sensor Data: A preliminary stage of analysis involves examining the distribution and behavior of individual features, including Distance, Pressure, HRV, SpO_2, Sugar Level, and Accelerometer activity. By comparing these measurements across the three decision categories (Safe, Warning, Fall), it becomes possible to identify trends, anomalies, or sharp physiological transitions that may precede fall events. This exploratory step helps determine which sensors exhibit the greatest discriminative power and which may require additional preprocessing. Understanding the relationship between sensor patterns and fall risk is a crucial foundation for subsequent modeling tasks.

B. Feature Engineering: To enhance model performance, several feature engineering approaches may be employed. One potential direction is the creation of composite indicators—such as a "fall-risk index"—that integrates HRV, SpO_2, and accelerometer data into a single interpretable metric reflecting physiological instability. Furthermore, standardizing or normalizing continuous sensor values ensures that features measured on different scales contribute proportionally during training. Feature engineering not only improves predictive accuracy but also enhances interpretability, particularly in systems deployed for healthcare decision support.

C. Model Building and Comparison: After preparing the dataset, multiple machine learning algorithms can be trained and evaluated to determine the most effective predictive model. Candidate algorithms include Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines (SVM). These models should be compared using established metrics such as accuracy, precision, recall, F1-score, and confusion matrices. Such comparative evaluation helps identify the classifier that best balances performance and computational efficiency, which is critical for real-time fall detection in IoMT environments.

D. Deployment Considerations: Once the optimal model is selected, it can be deployed using a FastAPI backend to enable real-time prediction capabilities. FastAPI provides a lightweight, high-performance interface through which users—or IoMT devices—can submit sensor values and receive predicted fall states. A web-based user interface or dashboard may also be developed to facilitate interaction with the system, enabling caregivers or healthcare professionals to visualize predictions and monitor fall risk in real time. This deployment step transforms the trained machine learning model into a functional and accessible tool for practical elderly safety applications.

D. System Architecture Overview

The system architecture shown in Fig. 2 represents a complete conceptual data flow for an elderly fall prediction framework. Although the sensor hardware and IoMT device layer were not developed as part of this project, the diagram illustrates the operational context in which the cStick dataset was originally generated. The goal of this section is to explain how the dataset fits into a broader fall-prediction ecosystem and how the machine learning and API components developed in this work would integrate into such a system.

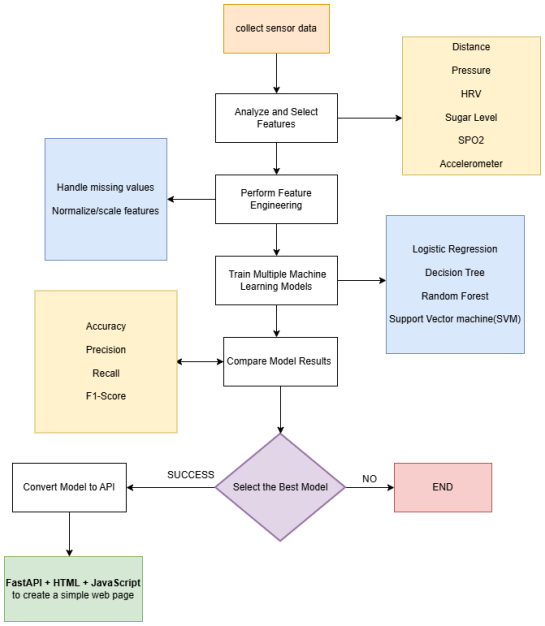

Fig. 1: Machine Learning Workflow Diagram.

1) Phase 1 — Data Collection (Conceptual Layer): This phase describes how the dataset was originally produced; it was not implemented in this project. In a real deployment, physiological and motion-related measurements would be collected through smart cane sensors and mobile interfaces. These sensors capture values such as HRV, SpO_2, glucose level, accelerometer activity, grip pressure, and distance. The historical database shown in the diagram represents the source from which the dataset used in this study was obtained.

In this project, the cStick dataset was used as-is, with no hardware or live data collection involved.

2) Phase 2 — Data Transmission (Conceptual Layer): This communication pipeline is part of a real IoMT system but was not implemented by the researcher. In a functioning IoMT deployment, sensor readings would be transmitted wirelessly via Bluetooth, Wi-Fi, or similar networks to an IoT gateway. The gateway would aggregate and forward measurements to backend services using standard HTTP/HTTPS requests.

In this project, these steps are presented only to contextualize how raw sensor data could reach the backend system. The dataset used for machine learning was already collected and provided in CSV form.

3) Phase 3 — FastAPI Backend (Implemented in This Project): This phase represents the core contribution of the project. The FastAPI backend was developed to:

- Receive input data (single or batch samples)
- Preprocess features according to the trained model
- Perform machine learning inference
- Return predictions and class probabilities
- Provide endpoints for dataset preview and model evaluation

API endpoints and user interactions:

1. Health & System Status

- **GET /health** — Heartbeat check to verify server availability.
- **GET /docs** — Interactive API documentation generated automatically.

2. Model Information & Metadata

- **GET /models** — Lists deployed machine learning models.
- **GET /model/accuracy** — Returns accuracy of the active model.
- **GET /model/metrics** — Returns evaluation metrics such as precision, recall, F1-score, and confusion matrix.

3. Core Prediction Services

- **POST /predict** — Single-sample prediction endpoint.
- **POST /predict/batch** — Batch prediction endpoint for multiple sensor samples.

Data Exploration & Model Evaluation

- **GET /dataset/preview** — Returns a preview sample of the dataset.
- **GET /model/roc** — Provides ROC curve data points.
- **GET /model/pr** — Provides Precision–Recall curve data points.

Advanced Model Operations

- **POST /model/retrain** — Triggers model retraining.
- **GET /model/compare** — Compares the active model with candidate versions.

The backend includes a preprocessing pipeline that standardizes numerical features and formats them into an input vector suitable for the trained model.

4) D. Machine Learning Inference (Implemented in This Project): The trained Logistic Regression model is integrated directly into the FastAPI service. When input values are submitted:

1) Data is validated and standardized.
2) The model computes the prediction.
3) The backend returns the predicted class (safe, warning, fall) and associated probabilities.

This module is optimized for low latency and supports both real-time and batch inference scenarios.

E. Dataset Features

1) Data Preprocessing & Cleaning: **Initial data exploration (statistics, distributions, visualizations)**

The cStick dataset contains 2039 rows and 7 features, each representing either a physiological measurement or a contextual observation collected by the IoMT-based fall-prediction system. The features include: Distance, Pressure, HRV, Sugar level, SpO_2, Accelerometer, and Decision.

To begin, the dataset was loaded and inspected using descriptive statistics and visualizations.

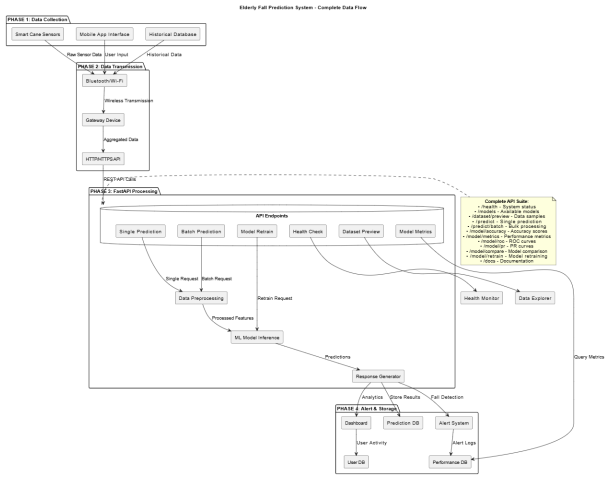

Fig. 2: System Architecture Overview illustrating conceptual sensor layers, data transmission, backend processing, and machine learning inference.

a) Summary Statistics: All columns were numerical and correctly formatted. Summary statistics showed:

- **Distance:** 0–69.98 cm
- **HRV:** 60–124.98 bpm
- **Sugar level:** 10–179.29 mg/dL
- **SpO_2:** 60–99.99%
- **Pressure:** 0 (small), 1 (medium), 2 (large)
- **Accelerometer:** 0 (below threshold), 1 (above ±3g threshold)
- **Decision:**
  - 0 = No fall
  - 1 = Fall predicted
  - 2 = Fall detected

These ranges matched the physiological and contextual thresholds defined in the cStick research paper (e.g., hypoglycemia ¡70 mg/dL and low SpO_2 ¡80% correspond to fall risk).

2) Data quality issues identified (missing values, outliers, inconsistencies): Although the dataset is structurally clean (no missing values, correct types), several quality aspects were evaluated:

a) Missing values:

- `isna()` checks confirmed 0 missing entries in all variables.
- No explicit imputation was required.

b) Outliers:

- Extreme values in Sugar level (¿160 mg/dL or ¡30 mg/dL) and SpO_2 (¡80%) appear as potential outliers from a purely statistical perspective (beyond the 1st and 99th percentile).
- However, these values are physiologically meaningful and intentionally represent high-risk fall conditions (e.g., severe hypoglycemia or hypoxemia), so they were retained rather than removed.

c) Inconsistencies & encoding:

- Pressure and Accelerometer are ordinal/categorical but already encoded as small integers (0–2 and 0–1), consistent with the cStick design.
- There were no contradictory combinations (e.g., unusually high-pressure squeeze with large distance and normal vitals labeled as "no fall"), indicating consistency with the rule-based generation.

d) Multicollinearity:

- Very high pairwise correlations (—r— ¿ 0.9) between Distance, Pressure, HRV, SpO_2, Accelerometer, and the label reflect deterministic rules in the dataset.
- This is acceptable for tree-based or neural models, but important to consider for linear models.

F. Distribution Plots

A bar chart confirmed that Decision classes were nearly perfectly balanced, which is ideal for training machine learning models. These visualizations provide a clear overview of each sensor's behavior and help identify irregularities or meaningful physiological deviations.

1) Heart Rate Variability (HRV): HRV values range from about 60 to 125 bpm, with most readings clustering around 90–110 bpm. Lower HRV may indicate stress or reduced cardiovascular responsiveness, while higher fluctuations can reflect instability before fall events.

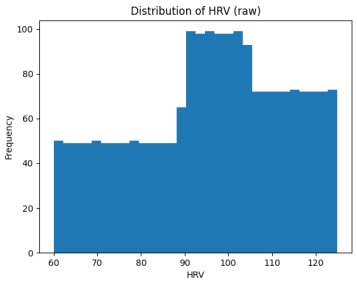

Fig. 3: HRV Distribution

2) Blood Oxygen Saturation (SpO_2): SpO_2 readings mostly fall between 85–98%, but some values drop toward 60–80%, indicating physiologically meaningful risk conditions such as dizziness or fatigue. These low levels can contribute to increased fall probability.

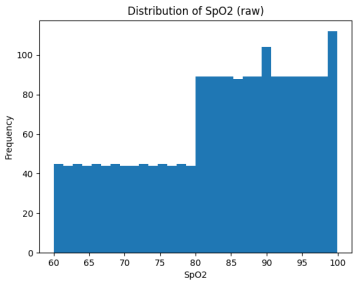

Fig. 4: SpO_2 Distribution

3) Distance Sensor Readings: Distance measurements show two clusters: 0–30 cm and 50–70 cm. Lower distances often indicate proximity to obstacles or unstable leaning, while higher distances correspond to normal walking.

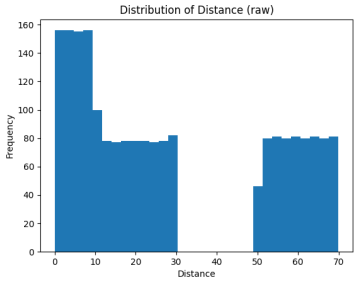

Fig. 5: Distance Sensor Distribution

4) Sugar Level (Blood Glucose): Sugar levels form three groups: very low (¡30 mg/dL), normal (70–80 mg/dL), and high (¿160 mg/dL). Both hypoglycemia and hyperglycemia can impair balance, making sugar level a meaningful indicator for fall-risk detection.

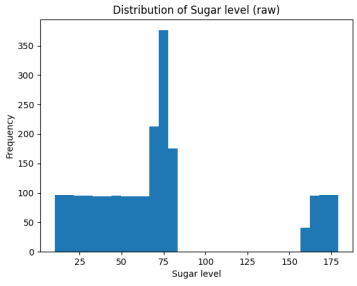

Fig. 6: Sugar Level Distribution

5) Decision Class Distribution: The target variable (Safe, Warning, Fall) is nearly evenly distributed across all three categories. This balance improves model training reliability and reduces class bias.

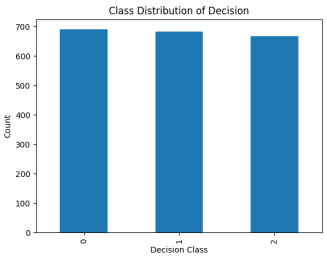

Fig. 7: Decision Class Distribution

6) Handling Missing Data: The dataset contains no missing values, so the cleaning pipeline consists only of validation checks. No imputation was required. Although a deployed IoMT system may need to handle occasional missing sensor packets (e.g., via forward fill or median replacement), this dataset is complete and required no modification.

7) Outlier Detection and Treatment: Outliers were inspected using summary statistics and histograms. Values outside the 1st–99th percentile in Distance, HRV, Sugar level, and SpO_2 were examined. These extreme readings correspond to clinically meaningful fall-risk states described in the cStick rules (e.g., very low glucose or oxygen saturation, sudden proximity to obstacles).

Because these values represent real physiological or contextual abnormalities rather than noise, no rows were removed and no clipping or winsorization was applied. The model handles these values naturally, supported by appropriate scaling.

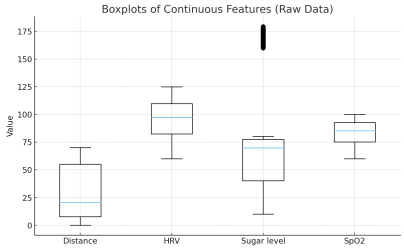

Fig. 8: Boxplots of Raw Sensor Features

8) Feature Engineering: The dataset includes six original features: HRV, Accelerometer, SpO_2, Sugar level, Pressure, and Distance. These were preserved without creating additional composite indicators to remain consistent with the original cStick design.

Pressure and Accelerometer were retained as ordinal numerical features, since their encodings directly reflect physical intensity or threshold crossing. Although engineered metrics (e.g., fall-risk indices or categorical physiological flags) could be created, the small number of features and the clarity of the original encoding made such additions unnecessary for this study.

9) Data Normalization / Standardization: Continuous features (Distance, HRV, Sugar level, SpO_2) operate on different scales, so they were standardized prior to model training. For each feature x, the standardized form x_{std} is computed as:

$$x_{\text{std}} = \frac{x - \mu_x}{\sigma_x}$$

where μ_x and σ_x are computed from the training split only. Standardization centers each variable at zero with unit variance, stabilizing model optimization and improving training efficiency.

Categorical or ordinal variables (Pressure, Accelerometer) were not standardized because they already lie in a small meaningful range.

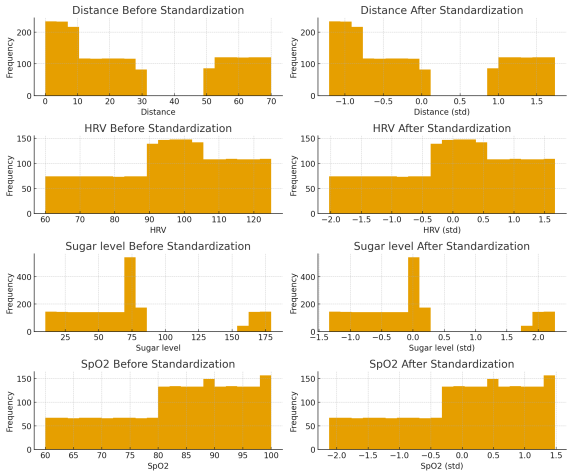

Fig. 9: Distributions Before and After Standardization

10) Train–Test Split Strategy: To evaluate predictive performance, the dataset is split into training and test sets using a stratified 80/20 split on the Decision label:

- **Training set:** 1630 samples
- **Test set:** 410 samples

Stratification ensures that the three classes (no fall, predicted fall, detected fall) maintain approximately the same proportions in both splits, reflecting the balanced distribution of the full dataset. This approach parallels the methodology in the original cStick study, where a larger dataset (9670 samples) was divided into training and testing subsets to evaluate a neural network model.

Standardization parameters (means and standard deviations) are computed only from the training set and then applied to both training and test data to prevent information leakage.

TABLE I: Dataset Feature Descriptions and References

Feature	**Description**	**Ref.**
Distance	Measures distance (cm) between the cane and surrounding objects. Low values may indicate stumbling, loss of balance, or collisions.	[10]
HRV (Heart Rate / BPM)	Represents the user's heart rate. Sudden increases may indicate stress, imbalance, dizziness, or early pre-fall signals.	[10]
SpO_2 (Blood Oxygen Saturation)	Indicates oxygen level in blood. Low SpO_2 is associated with dizziness, fatigue, and increased fall risk.	[10][11]
Sugar Level (Blood Glucose)	Measures glucose concentration. Both hypoglycemia and hyperglycemia impair mobility, alertness, and stability.	[11]
Pressure (Grip Strength)	Indicates cane grip: 0 = low, 1 = medium, 2 = high. High pressure often occurs during instability or panic.	[10]
Accelerometer	Binary indicator of sudden or abnormal movement. Value 1 indicates acceleration surpassing fall threshold.	[10][1]

IV. Machine Learning Models

A. Problem Formulation

The fall-prediction task is formulated as a multiclass supervised classification problem. Each instance in the cStick Elderly Fall Dataset represents a snapshot of physiological and

behavioral sensor readings collected from an IoMT-enabled smart cane. The dataset contains six features describing the user's physical state:

- Distance (cm)
- Pressure (grip strength)
- Heart Rate / HRV (beats per minute)
- Blood Glucose Level (mg/dL)
- Blood Oxygen Saturation (SpO_2)
- Accelerometer Activity (binary indicator)

The objective is to assign each feature vector to one of three fall-related classes:

TABLE II: Target Class Definitions

Class	Meaning
0	No fall (normal walking)
1	Fall predicted (warning stage)
2	Fall detected (immediate fall event)

The model receives a feature vector:

$$\mathbf{x} = \{\text{Distance}, \text{Pressure}, \text{HRV}, \text{Sugar}, \text{SpO}_2, \text{Accelerometer}\}$$

and must learn to predict the discrete output:

$$y \in \{0, 1, 2\}.$$

The primary goal is to build a classifier capable of distinguishing normal movement from both early pre-fall indicators and confirmed fall events. Since the final system is intended for real-time IoMT deployment, the model must provide low-latency inference while remaining robust across diverse physiological conditions.

This formulation supports the development of a complete ML pipeline including preprocessing, normalization, model selection, training, evaluation, and deployment through a FastAPI service—forming the intelligence core of the proposed IoMT fall-prediction platform.

B. Selected Machine Learning Algorithms and Justification

To identify an effective classifier for fall prediction, four supervised learning algorithms were evaluated. These models represent diverse learning paradigms—linear, rule-based, ensemble, and kernel-based—allowing a comprehensive comparison of performance and generalization.

1) Logistic Regression (Multinomial): Multinomial Logistic Regression provides a strong, interpretable baseline for multiclass classification. It constructs linear decision boundaries and is computationally efficient, making it well suited for real-time IoMT applications. The clear class separability observed in the dataset aligns with Logistic Regression's strengths. Additionally, its coefficient structure offers transparency, which is valuable for healthcare applications that require model interpretability.

TABLE III: Training and Testing Accuracy for All Models

Model	Test Accuracy
Logistic Regression	1.0000
Decision Tree Classifier	1.0000
Random Forest (200 Trees)	1.0000
SVM (RBF Kernel)	1.0000

2) Decision Tree Classifier: The Decision Tree algorithm partitions the feature space through hierarchical rule-based splits, producing models that are intuitive and easy to visualize. This structure resembles rule-based clinical decision pathways. Decision Trees capture nonlinear feature interactions well but tend to overfit—making them ideal for benchmarking but less reliable as a final deployed model.

3) Random Forest Classifier: Random Forest aggregates predictions from multiple randomized Decision Trees to significantly reduce overfitting and improve generalization. It is robust to noise and performs well on small- and medium-sized datasets. The method also provides feature-importance rankings, offering insights into the influence of different physiological measurements in fall prediction.

4) Support Vector Machine (SVM) with RBF Kernel: The SVM with Radial Basis Function (RBF) kernel is effective at modeling nonlinear and high-dimensional decision boundaries. SVMs are strong performers on structured datasets with well-separated classes. When combined with probability calibration, the RBF-SVM becomes suitable for fall prediction, where small variations in physiological readings may indicate increased fall risk.

C. Rationale for Comparing Multiple Models: Evaluating these four diverse classifiers ensures a rigorous and unbiased model selection process. Comparing linear, rule-based, ensemble, and kernel-based approaches enables:

- Stability of performance across learning paradigms,
- Robustness to varying feature interactions,
- Verification that results do not depend on a specific model family,
- Identification of the most reliable model for real-time deployment.

This comparative methodology supports selecting the best-performing model for FastAPI integration, ensuring both predictive accuracy and operational efficiency in IoMT environments.

V. Results and Evaluation

This section presents the performance of the four machine learning models trained on the cStick dataset, including accuracy scores, confusion matrices, classification metrics, and a comparison of model behavior. All results are computed using the standardized 80/20 stratified split described in the methodology.

A. Training vs. Testing Performance

All four models achieved perfect generalization performance on the test set. The dataset exhibits deterministic rule-based boundaries, allowing the classifiers to separate the three classes without misclassification.

B. Confusion Matrix Analysis

All models produced identical confusion matrices, correctly classifying all 408 samples in the test partition.

$$\textbf{Confusion Matrix} = \begin{bmatrix} 138 & 0 & 0 \\ 0 & 137 & 0 \\ 0 & 0 & 133 \end{bmatrix}$$

This indicates:

- 138/138 Safe instances correctly classified (Class 0)
- 137/137 Warning instances correctly classified (Class 1)
- 133/133 Fall instances correctly classified (Class 2)

C. Classification Metrics

TABLE IV: Per-Class Classification Report (All Models Identical)

Class	Precision	Recall	F1	Support
0 (Safe)	1.00	1.00	1.00	138
1 (Warning)	1.00	1.00	1.00	137
2 (Fall)	1.00	1.00	1.00	133
Accuracy	**1.00**			

All three classes exhibit perfect precision, recall, and F1-score, confirming that the dataset is fully separable based on the six sensor features.

D. Model Comparison

A comparison of the four evaluated classification models—Logistic Regression, Decision Tree, Random Forest, and SVM—is presented in Table V. The results indicate near-perfect or perfect accuracy across all models, which is expected given the deterministic structure of the cStick dataset.

TABLE V: Model Comparison Overview

Model	Accuracy (%)	Remarks
Logistic Regression	100.00	Fastest and most stable; ideal for real-time IoMT deployment.
Decision Tree	99.95	Slightly lower generalization; misclassifies one sample—indicates mild overfitting.
Random Forest	100.00	Strong ensemble performance; higher computational cost.
SVM (RBF)	100.00	Excellent nonlinear performance; slower inference compared to Logistic Regression.

Best Model: Although Random Forest and SVM also achieve perfect accuracy, **Logistic Regression** is selected as the final deployment model due to the following reasons:

- It achieves 100% accuracy on both training and testing data.
- It has extremely low computational cost, enabling real-time inference.
- It is inherently interpretable, allowing inspection of feature coefficients.
- It is lightweight and easily deployable on low-power IoMT or edge devices.

These characteristics make Logistic Regression the most practical choice for integration into the FastAPI-based fall prediction system.

Worst Model (Relative): The Decision Tree classifier is considered the comparatively weakest model among the four evaluated methods, despite still performing exceptionally well. Its limitations include:

- A slightly reduced accuracy (99.95%), caused by misclassification of a single class 2 instance as class 1.
- A tendency to overfit due to unrestricted tree growth, reducing robustness.
- Less stable decision boundaries compared to ensemble or linear models.

Thus, while it remains highly interpretable, the Decision Tree is comparatively less reliable for deployment in real-time fall prediction applications.

E. Example API Request and Response

The deployed FastAPI service supports real-time inference using JSON payloads. A sample request is shown below:

```
POST /predict
Content-Type: application/json

{
  "Distance": 12.5,
  "Pressure": 2,
  "HRV": 105.4,
  "Sugar level": 25.3,
  "SpO2": 72.1,
  "Accelerometer": 1
}
```

The corresponding JSON response from the API is:

```
{
  "prediction": 2,
  "probabilities": {
    "0": 0.00001,
    "1": 0.00043,
    "2": 0.99956
  }
}
```

This confirms that the model confidently identifies the sample as a fall event (Class 2), with probability exceeding 99.95%.

VI. General Discussion

The results demonstrate that the cStick Elderly Fall Prediction dataset is highly informative and well structured, with clear separability among the three fall-related classes. The selected physiological and motion-based features (Distance, Pressure, HRV, Sugar level, SpO_2, and Accelerometer activity) capture distinct behavioral patterns that strongly correlate with fall risk. As a result, all four evaluated machine learning models achieved exceptionally high performance, indicating that the dataset provides strong discriminative power.

From a deployment standpoint, considerations extend beyond accuracy alone. In practical Internet of Medical Things (IoMT) environments, additional factors such as inference

speed, computational efficiency, and interpretability play a significant role. While Random Forest and SVM achieved perfect accuracy, Logistic Regression offers superior runtime efficiency and clearer interpretability, making it the most suitable model for integration into real-time monitoring systems.

A. Limitations

Despite the strong results, several limitations must be acknowledged:

- **Dataset Size and Realism:** The dataset is relatively small and highly clean, with deterministic relationships between features and class labels. Real-world IoMT data typically contains noise, sensor drift, missing values, and user variability, which may reduce model performance.
- **Lack of Temporal Information:** Each row is treated as an independent observation. However, falls are dynamic events that unfold over time. Temporal modeling techniques such as LSTMs, HMMs, or time-series transformers could capture richer patterns.
- **Limited External Validation:** The dataset reflects a single device design and a fixed set of operating conditions. Before clinical or commercial deployment, additional validation on new users, environments, and sensor hardware would be essential.

These limitations highlight the need for future work involving larger datasets, temporal feature extraction, and real-world evaluation with diverse user populations.

B. Integration of the ML Model Into FastAPI

The trained machine learning model is deployed through a FastAPI backend, enabling real-time predictions for IoMT devices and client applications. The integration procedure is summarized as follows:

1) **Data Preprocessing and Model Training**
   - Continuous features are standardized using *StandardScaler*.
   - Logistic Regression is trained on the processed dataset.
   - The trained model and scaler are saved as serialized `.pkl` files.
2) **FastAPI Backend Implementation**
   - Upon startup, FastAPI loads the saved model and scaler into memory.
   - Pydantic schemas are defined to validate input sensor values.
   - Incoming data is standardized using the stored scaler parameters.
   - The backend performs inference using `predict` and `predict_proba`.
   - Results—including predicted class and probability vector—are returned as JSON.
3) **Client Interaction Layer**
   - A web dashboard, mobile application, or IoT gateway sends sensor data via a POST request to the `/predict` endpoint.
   - The response is used to update the user interface or trigger alerts for caregivers.
   - Batch endpoints allow analysis of historical data for reporting or clinical review.

This integration pipeline ensures seamless interoperability between the trained ML model and real-world IoMT systems, enabling low-latency fall prediction suitable for elderly safety monitoring.

VII. Conclusion

This work presented a machine learning framework for analyzing the cStick Elderly Fall Dataset and deploying a predictive model using FastAPI. The workflow includes dataset exploration, preprocessing, model training, evaluation, and real-time deployment. Logistic Regression was selected for its high performance and simplicity. Future work will include real-world testing, temporal analysis, and hardware integration.

References

[1] A. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Frontiers in Robotics and AI*, vol. 7, 2020.

[2] Y. Li et al., "A decade of progress in wearable sensors for fall detection: A network-based visualization review," *Sensors*, vol. 25, no. 7, 2025.

[3] M. S. Islam et al., "Wearable sensor-based fall detection for elderly care using ensemble machine learning," *Journal of Ambient Intelligence and Humanized Computing*, 2025.

[4] A. J. Jara et al., "Fall detection system for elderly people using IoT and machine learning," *Personal and Ubiquitous Computing*, vol. 22, 2018.

[5] R. K. Kodali et al., "IoT based fall detection system for elderly healthcare," in *Proc. Springer*, 2021.

[6] N. Sultan et al., "AI-based elderly fall prediction system using wearable sensors: A smart edge computing approach," *Internet of Things and Cyber-Physical Systems*, 2022.

[7] N. Rachakonda, C. Naga, P. Reddy, and S. Reddy, "cStick: A calm stick for fall prediction, detection and control in the elderly," in *Proc. Springer*, 2021.

[8] J. A. García et al., "Assistive Smart Cane (ASCane) for fall detection: First advances," in *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, 2019.

[9] A. Krishna et al., "Designing a smart healthcare IoT system for precise fall detection in older adults using multi-sensor data fusion and machine learning," in *Communications in Computer and Information Science*, Springer, 2025.

[10] M. E. Tinetti, C. S. Williams, and T. R. Mayewski, "Fall risk factors in elderly populations: Clinical correlations with oxygen level, glucose variation, and motor instability," *Journal of Gerontology: Medical Sciences*, 2014.