Project Report

EdgeHAR: Efficient Human Activity Recognition Using IMU Sensors

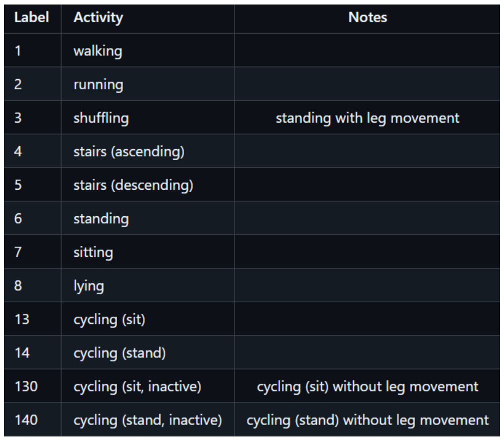
Author:

Abdulrehman Bin Yasin

1. Introduction

This project addresses the challenge of deploying Human Activity Recognition (HAR) on edge devices with limited processing power. We processed 50Hz accelerometer data from 2 three-axial Axivity AX3 accelerometers to classify distinct activities, including static postures (Sitting, Standing) and dynamic movements (Running, Cycling, Stairs). We used the harth dataset for this project (https://github.com/ntnu-ai-lab/harth-ml-experiments/tree/main/harth)

The core challenge addressed in this study is distinguishing between kinematically similar activities (e.g., Walking vs. Stair Climbing) using low complexity models suitable for deployment on edge devices.



Label	Activity	Notes
1	walking	
2	running	
3	shuffling	standing with leg movement
4	stairs (ascending)	
5	stairs (descending)	
6	standing	
7	sitting	
8	lying	
13	cycling (sit)	
14	cycling (stand)	
130	cycling (sit, inactive)	cycling (sit) without leg movement
140	cycling (stand, inactive)	cycling (stand) without leg movement

The 11 distinct physical activities to be predicted

(Notice we did not take label 140 into account as we had little to no data on it in any of our datasets)

2. Data Acquisition and Preprocessing

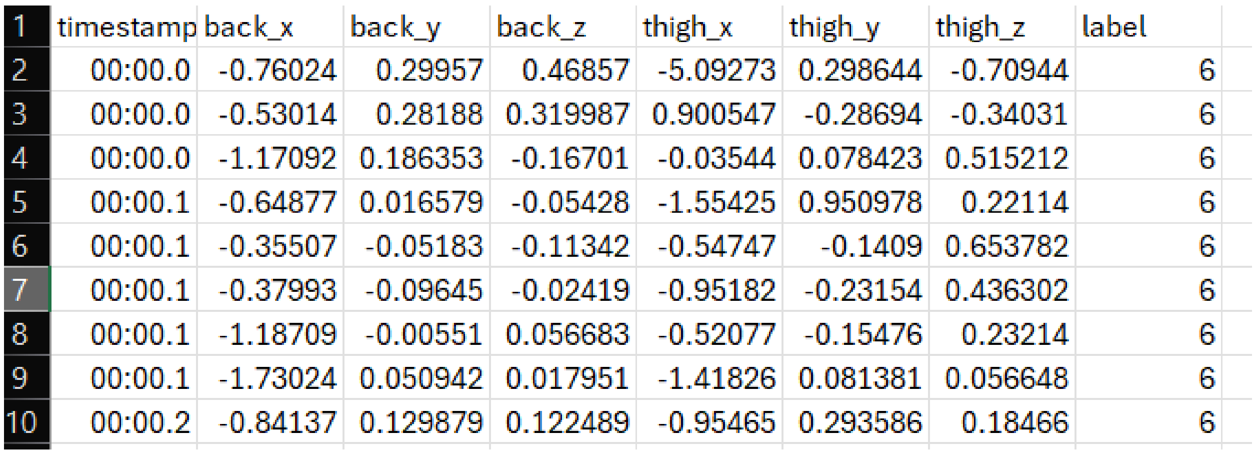
2.1 Dataset Construction

The dataset was aggregated from multiple subject files (S006.csv through S038.csv). To ensure data quality, we implemented a rigorous cleaning protocol:

- **Data Aggregation:** Raw sensor data (50 Hz) was compiled from individual CSV files.
- **Garbage Removal:** The first 250 rows of every file were discarded to remove noise associated with sensor initialization and subject preparation.
- **Balancing:** To prevent model bias towards dominant classes (like Walking), we implemented a down sampling strategy. Common classes were capped at **19,974 rows** and classes with less data had data appended from other datasets, ensuring the model did not prioritize majority classes at the expense of minority ones and so everything was balanced.

2.2 Feature Engineering

We had 6 features in total that were raw accelerometer data Back X/Y/Z, Thigh X/Y/Z and it is noisy and difficult for linear models to interpret directly since it was at 50HZ(50 rows contribute to one second of data).



1	timestamp	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
2	00:00.0	-0.76024	0.29957	0.46857	-5.09273	0.298644	-0.70944	6
3	00:00.0	-0.53014	0.28188	0.319987	0.900547	-0.28694	-0.34031	6
4	00:00.0	-1.17092	0.186353	-0.16701	-0.03544	0.078423	0.515212	6
5	00:00.1	-0.64877	0.016579	-0.05428	-1.55425	0.950978	0.22114	6
6	00:00.1	-0.35507	-0.05183	-0.11342	-0.54747	-0.1409	0.653782	6
7	00:00.1	-0.37993	-0.09645	-0.02419	-0.95182	-0.23154	0.436302	6
8	00:00.1	-1.18709	-0.00551	0.056683	-0.52077	-0.15476	0.23214	6
9	00:00.1	-1.73024	0.050942	0.017951	-1.41826	0.081381	0.056648	6
10	00:00.2	-0.84137	0.129879	0.122489	-0.95465	0.293586	0.18466	6

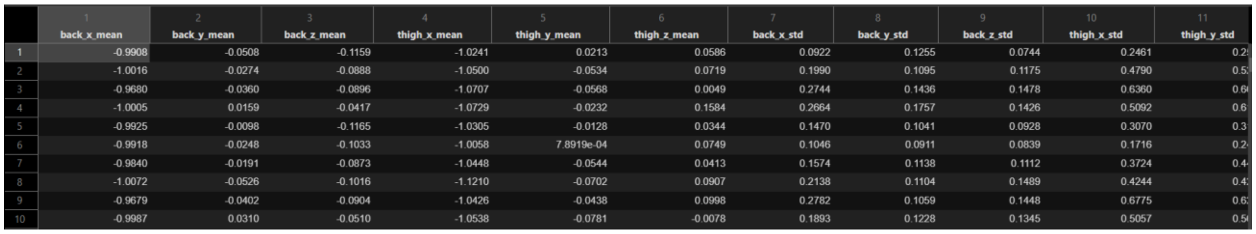
A look at how our unprocessed raw dataset looks

We applied a sliding window approach to extract temporal features:

- **Windowing:** Data was segmented into **2.5 second windows** (125 samples at 50Hz).
- **Features:** For each window, we calculated two key metrics:
    1. **Mean :** Captures the static orientation of the limb (e.g., Thigh vertical vs. horizontal).
    2. **Standard Deviation:** Captures the intensity and energy of the movement (e.g., Running vs. Standing).

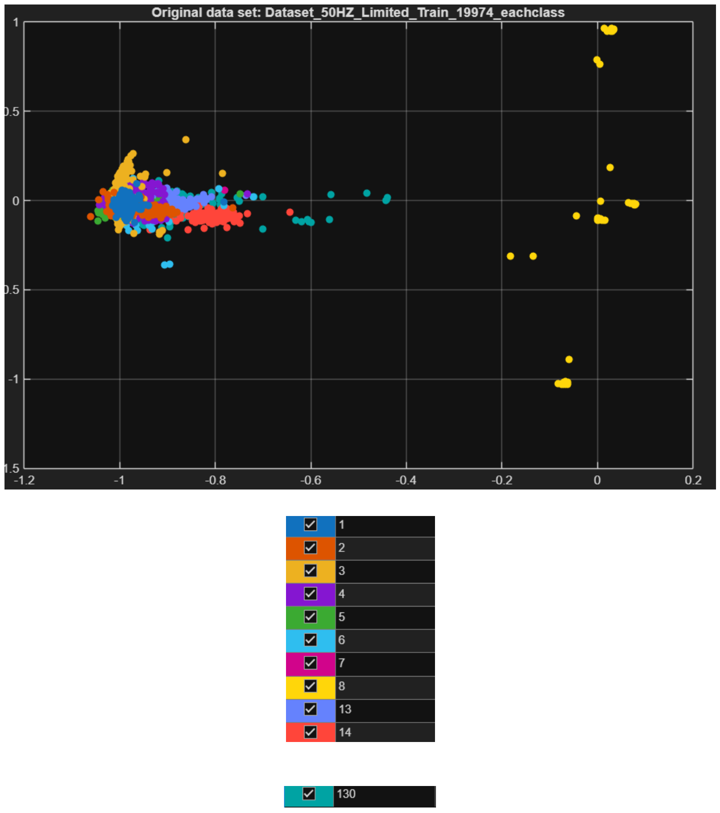
This resulted in a final feature vector of 12 dimensions per sample. This is the structure of Feature Extraction Logic we used in matlab:

- features = [mean(buffer_data), std(buffer_data)];



	1 back_x_mean	2 back_y_mean	3 back_z_mean	4 thigh_x_mean	5 thigh_y_mean	6 thigh_z_mean	7 back_x_std	8 back_y_std	9 back_z_std	10 thigh_x_std	11 thigh_y_std
1	-0.9908	-0.0508	-0.1159	-1.0241	0.0213	0.0586	0.0922	0.1255	0.0744	0.2461	0.2
2	-1.0016	-0.0274	-0.0888	-1.0500	-0.0534	0.0719	0.1990	0.1095	0.1175	0.4790	0.5
3	-0.9680	-0.0360	-0.0896	-1.0707	-0.0568	0.0049	0.2744	0.1436	0.1478	0.6360	0.6
4	-1.0005	0.0159	-0.0417	-1.0729	-0.0232	0.1584	0.2664	0.1757	0.1426	0.5092	0.6
5	-0.9925	-0.0098	-0.1165	-1.0305	-0.0128	0.0344	0.1470	0.1041	0.0928	0.3070	0.3
6	-0.9918	-0.0248	-0.1033	-1.0058	7.8919e-04	0.0749	0.1046	0.0911	0.0839	0.1716	0.2
7	-0.9840	-0.0191	-0.0873	-1.0448	-0.0544	0.0413	0.1574	0.1138	0.1112	0.3724	0.4
8	-1.0072	-0.0526	-0.1016	-1.1210	-0.0702	0.0907	0.2138	0.1104	0.1489	0.4244	0.4
9	-0.9679	-0.0402	-0.0904	-1.0426	-0.0438	0.0998	0.2782	0.1059	0.1448	0.6775	0.6
10	-0.9987	0.0310	-0.0510	-1.0538	-0.0781	-0.0078	0.1893	0.1228	0.1345	0.5057	0.5

Our processed dataset at 2.5 seconds worth of data per row with 12 features



Scatter plot of Our dataset

3. Model Selection

We evaluated multiple classifier families using the MATLAB Classification Learner App, including Support Vector Machines (SVM), Decision Trees, and K-Nearest Neighbors (KNN). Based on preliminary performance, two top candidates were shortlisted for final evaluation:

1. **Medium KNN (K-Nearest Neighbors):** A non linear model capable of capturing complex decision boundaries.
2. **Efficient Logistic Regression:** A linear, highly interpretable model optimized for computational efficiency.

Final Selection: Efficient Logistic Regression While the **Medium KNN** model achieved the highest **Training Accuracy**, we selected **Efficient Logistic Regression** as the final model for evaluation.

This decision was driven by a distinct "performance inversion" observed during validation:

- **Overfitting in KNN:** The Medium KNN model exhibited signs of overfitting, where its superior ability to memorize training data did not fully translate to unseen data (lower Test Accuracy).
- **Generalization in Logistic Regression:** The Logistic Regression model achieved a higher **Test Accuracy**, outperforming the KNN on the independent test set despite having a slightly lower training score.

We prioritized **Test Accuracy** as the primary metric for success, as it reflects the model's ability to generalize to new subjects in real-world scenarios. The superior performance of the linear model confirms that our feature engineering (Mean and Standard Deviation) created distinct enough clusters that simple linear boundaries were more robust than the complex, non-linear boundaries of KNN.

4. Experimental Results

4.1 Overall Performance

To validate the model selection, we compared the accuracy metrics of the top non-linear candidate (Medium KNN) against the linear candidate (Efficient Logistic Regression).

Medium KNN: This model achieved the highest fidelity during training but exhibited signs of minor overfitting when applied to unseen data.

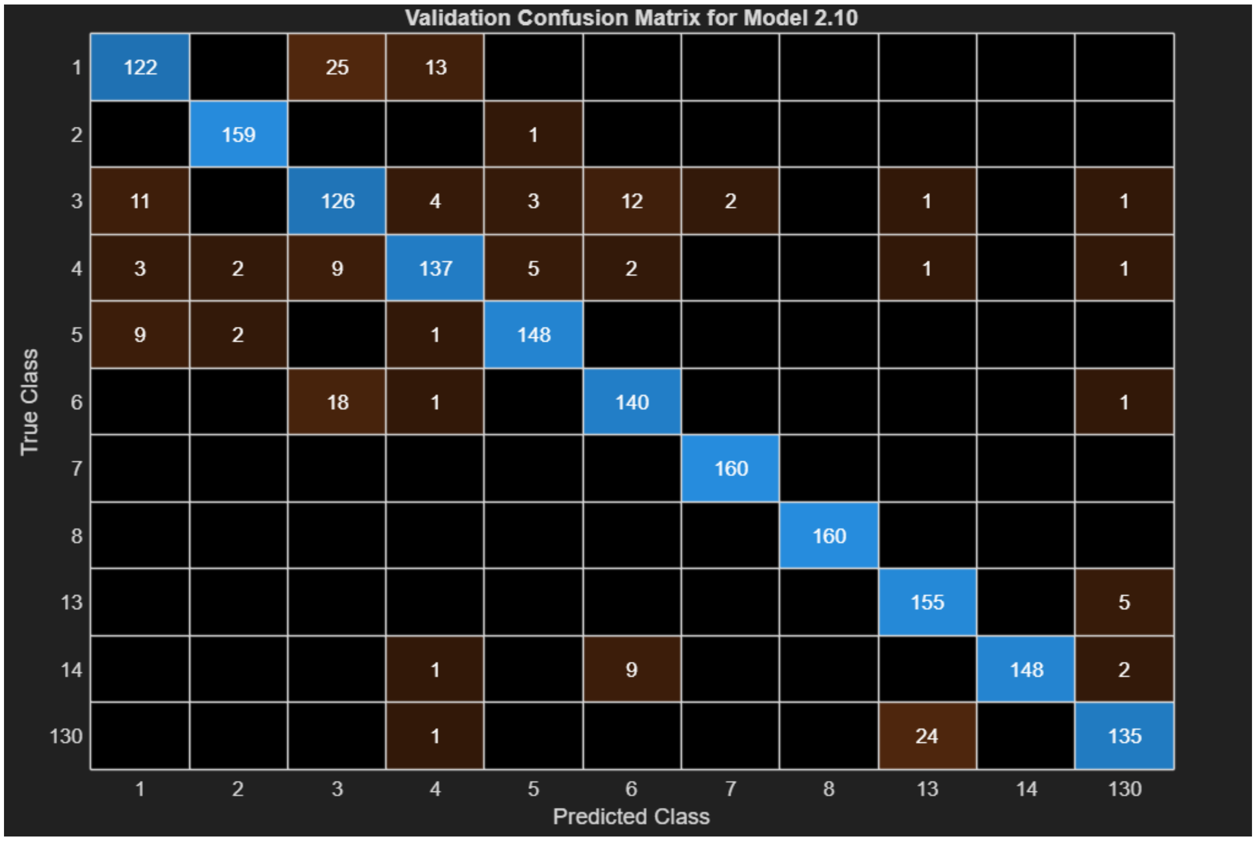
• Training Accuracy: **92.6%**

• Test Accuracy: **~91.2%**

Efficient Logistic Regression (Selected): The linear model demonstrated stronger generalization to new subjects, outperforming the complex model on the Test set despite a lower training score. •

Training Accuracy: **~90.3%**

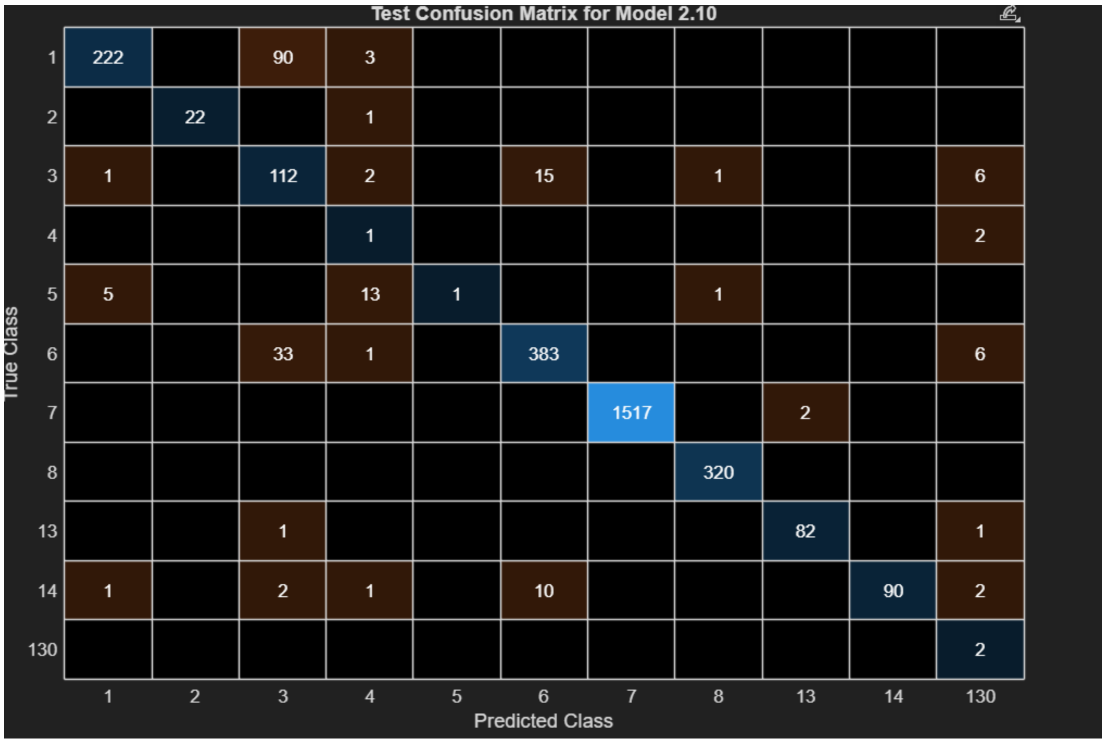
• Test Accuracy: **~93.2%**



Validation Confusion Matrix for Model 2.10

True Class \ Predicted Class	1	2	3	4	5	6	7	8	13	14	130
1	122		25	13							
2		159			1						
3	11		126	4	3	12	2		1		1
4	3	2	9	137	5	2			1		1
5	9	2		1	148						
6			18	1		140					1
7							160				
8								160			
13									155		5
14				1		9				148	2
130				1					24		135

Training Confusion Matrix showing high diagonal accuracy.



Test Confusion Matrix for Model 2.10

True Class \ Predicted Class	1	2	3	4	5	6	7	8	13	14	130
1	222		90	3							
2		22		1							
3	1		112	2		15		1			6
4				1							2
5	5			13	1			1			
6			33	1		383					6
7							1517		2		
8								320			
13			1						82		1
14	1		2	1		10				90	2
130											2

Testing Confusion Matrix showing generalization performance.

4.2 Test Performance Analysis: Strengths & Weaknesses

Strengths:

The model demonstrated high reliability for distinct activities where signal patterns were unique.

- **Sitting (Class 7):** Achieved near-perfect precision due to the distinct static thigh orientation. (99.9% Accuracy)
- **Lying (Class 8):** Robustly identified due to the unique horizontal sensor axis. (100% Accuracy)
- **Cycling Sit (Class 13):** Accurately identified active cycling while seated. (97.6% Accuracy)
- **Running (Class 2):** The high standard deviation allowed for easy recognition of this high-intensity state. (95.7% Accuracy)
- **Standing (Class 6):** Showed strong stability with minimal confusion against other static classes. (90.5% Accuracy)
- **Cycling Stand (Class 14):** Successfully distinguished active standing cycling from standard standing. (84.9% Accuracy)

- **Walking (Class 1):** Effectively captured the standard gait pattern, though frequently confused with Shuffling. (70.5% Accuracy)

Weaknesses:

Performance degraded significantly for ambiguous activities or those with insufficient test data.

- **Shuffling (Class 3):** Frequently misclassified as Standing because the movement intensity was too subtle. (81.7% Accuracy)
- **Cycling Inactive (Class 130):** Sample size was too low to determine reliability, with only 2 instances available in the test set. (2 Correct - Insufficient Data)
- **Stairs Ascending (Class 4):** Indistinguishable from Walking in the time-domain without an altimeter. (~33% Accuracy)
- **Stairs Descending (Class 5):** Misclassified as Walking due to nearly identical kinematic signatures. (~5.0% Accuracy)

5. Performance Limitations & Mitigation Attempts

Despite the high overall accuracy, four specific classes showed significant performance degradation during testing: **Shuffling (3)**, **Stairs (4 & 5)**, and **Inactive Cycling (130)**.

5.1 The "Stairs vs. Walking" Problem (Classes 4 & 5)

- **Issue:** The model frequently misclassified Stair Climbing as Walking (Class 1).
- **Root Cause:** Using an accelerometer alone, the kinematic signature of climbing stairs (rhythmic leg movement, vertical thigh orientation) is nearly identical to walking on flat ground. Without an altimeter (barometer) or frequency-domain features (FFT) to detect cadence differences, the signals are mathematically indistinguishable in the time domain.

5.2 The "Inactive Cycling" Problem (Class 130)

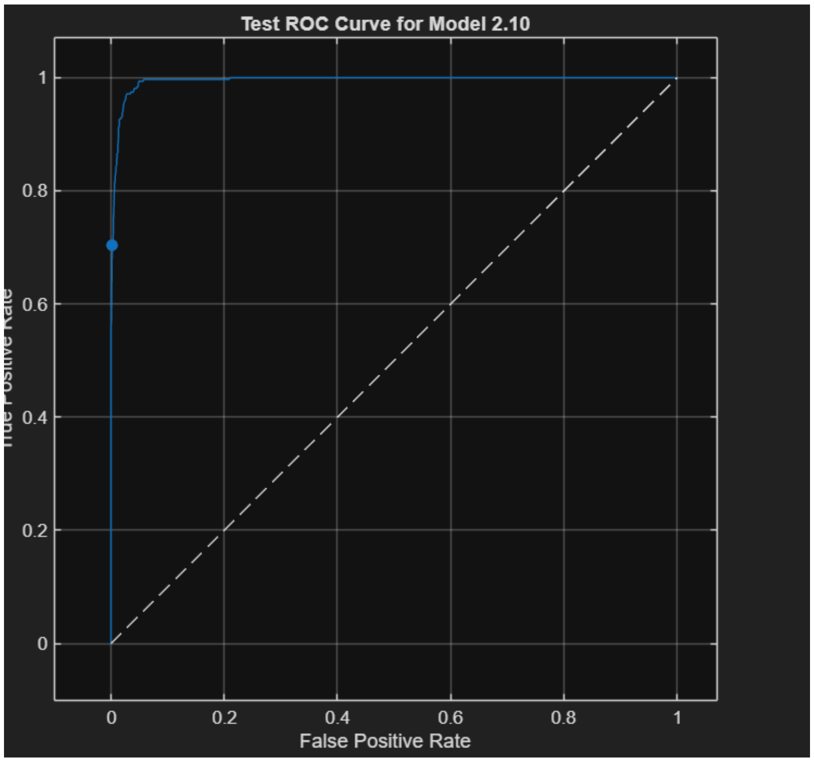
- **Issue:** Class 130 was confused with Class 14 (Cycling Stand) or Class 7 (Sitting).
- **Root Cause:** "Inactive Cycling" is a static posture. If the seat height or sensor placement varies slightly between subjects, the "Mean Thigh Angle" changes, crossing the linear decision boundary into a different class.

5.3 Attempted Mitigation: Strategic Data Augmentation

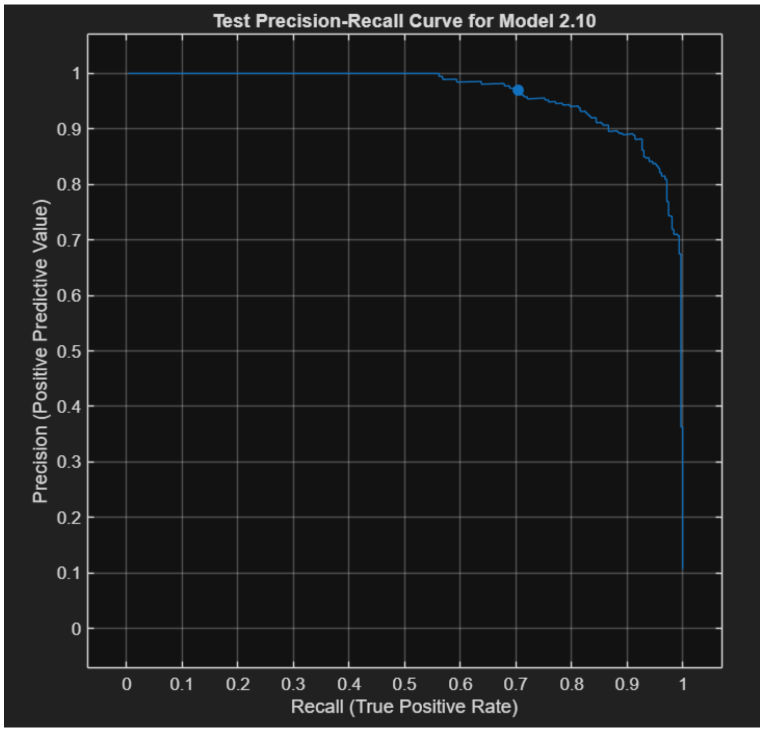
To address these failures, we attempted a "Strategic Imbalance" experiment.

- **Method:** We increased the training data for the weak classes (3, 4, 5, 130) well beyond the 19,974 limit, effectively flooding the model with more "Stairs" examples to force it to learn better.
- **Result: The performance did not significantly improve.**
- **Scientific Conclusion:** This confirms that the issue is not a lack of data quantity, but **Covariate Shift** and **Sensor Limitation**. The training subjects climbed stairs with a specific intensity that differed from the test subjects. Adding more data of the *same* style did not help the model generalize to the *new* style found in the test set.

6. More Evaluation Metrics:



Test ROC Curve



Test Precision Recall Curve

7. Conclusion

The Efficient Logistic Regression model successfully classifies the majority of human daily activities with **93.2% Test accuracy**, proving that simple time domain features (Mean/Std) are sufficient for general HAR tasks.

The limitations observed in classifying Stairs and Shuffling highlight the inherent boundaries of accelerometer-only systems. Future improvements would require:

1. **Sensor Fusion:** Adding a Barometer to detect vertical displacement for stairs.
2. **Domain Adaptation:** Using techniques to calibrate the model to new users' specific movement styles.