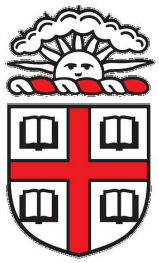

Human Activity Recognition using Accelerometer Data

Devraj Raghuvanshi
Data Science Institute, Brown University

[GitHub]

Contents

1 Introduction

1.1 Motivation

Human Activity Recognition (HAR) plays a crucial role in various applications, including healthcare monitoring, smart homes, fitness tracking, and human-computer interaction. The ability to automatically detect and classify physical activities from sensor data can significantly enhance systems designed to assist individuals with mobility impairments, improve personalized fitness training, or optimize health interventions.

Traditional approaches in HAR often rely on subject-specific models, which are tuned to individual users. These subject-specific models are typically optimized to improve subject-dependent performance, where the model is trained on data from the same user whose activities will be recognized by the system. However, these models struggle to generalize to new, unseen individuals, posing challenges for scalability [1]. Subject-independent models, on the other hand, aim to recognize activities without retraining for each individual, which makes them more applicable in real-world scenarios. This project aims to develop a subject-independent HAR model that generalizes well across different users, eliminating the need for retraining on individual-specific data.

1.2 Dataset

[2, 3], sourced from the UCI ML Repository, which contains recordings from 22 participants who wore two 3-axial Axivity AX3 accelerometers for approximately 2 hours. One sensor was placed on the right front thigh and the other on the lower back, recording data at a sampling rate of 50 Hz. The dataset consists of 6,461,328 samples, with each sample including the subject ID, accelerometer readings from both sensors across the x, y, and z axes (a total of 6 values), and a label indicating the activity performed.

The original dataset contains a variety of fine-grained activity classes. However, the class distribution was highly imbalanced across these activities, and similar activities were fragmented into separate classes. To address these issues, I merged related activity classes into broader categories as shown in Table 1, ensuring a more balanced class distribution and reducing fragmentation for model training.

Fine-Grained Classes	**Merged Class**
Standing, Sitting, Lying	Inactive
Walking, Shuffling	Walking
Stairs (ascending), Stairs (descending)	Stairs
Running	Running
Cycling (sit), Cycling (stand), Cycling (sit, inactive), Cycling (stand, inactive)	Cycling

Table 1: Mapping of Fine-Grained Activity Classes to Merged Classes

1.3 Prior work

In their work, Logacjov et al. [2] trained and evaluated seven baseline models on the HARTH dataset using nine activity classes, including support vector machine (SVM), k-nearest neighbor (KNN), random forest (RF), extreme gradient boosting (XGBoost), convolutional neural networks (CNN), bidirectional long short-term memory (BiLSTM), and a CNN with multi-resolution blocks. The SVM model outperformed others with an F1-score of 0.81 ± 0.18. Based on this, it is expected that the performance of my best model will be in a similar range to their reported SVM performance, with potential improvements due to the class merging.

1.4 Window Segmentation

Window segmentation involves dividing continuous time-series data into fixed-length segments (windows) to facilitate feature engineering. The sliding window technique is applied to the accelerometer data, segmenting it into 2-second windows with a 50% overlap. Studies have shown that window sizes in the range of 1-3 seconds are effective for human activity recognition tasks [4, 5].

1.5 Feature Engineering

For each window, 13 time-domain features are extracted from both accelerometers across all three axes: mean, standard deviation, skewness, kurtosis, range, interquartile range, maximum, minimum, sum, root mean square, energy, zero-crossings, and peak-to-peak. This results in a total of 78 features per window ($13 \times 2 \times 3$). The majority activity label within each window is then assigned as the target label for classification.

2 Exploratory Data Analysis

The overall class distribution of the dataset, shown in Figure 1, reveals a significant class imbalance. The majority class, *inactive*, comprises 63.08% of the samples, while the minority class, *stairs*, accounts for only 2.30%. This imbalance can lead to biased model predictions favoring the majority class, reducing the overall performance on minority classes. To mitigate this, two decisions are made. First, F1-macro is chosen as the evaluation metric because it evaluates the performance of the model on each class independently and gives equal weight to all classes, making it suitable for imbalanced datasets. Second, the `class_weight` parameter is utilized in the machine learning models to address the imbalance. This parameter assigns higher weights to underrepresented classes, forcing the model to focus more on correctly predicting these classes during training.

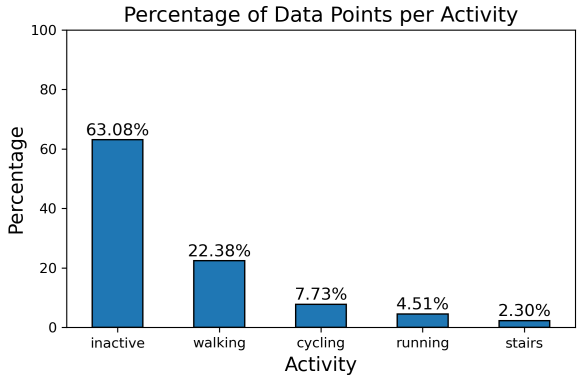

Figure 1: Class distribution of the dataset.

A more granular examination of the class distribution at the subject level is provided in Figure 2. This figure illustrates the proportion of activities for each of the 22 subjects in the dataset.

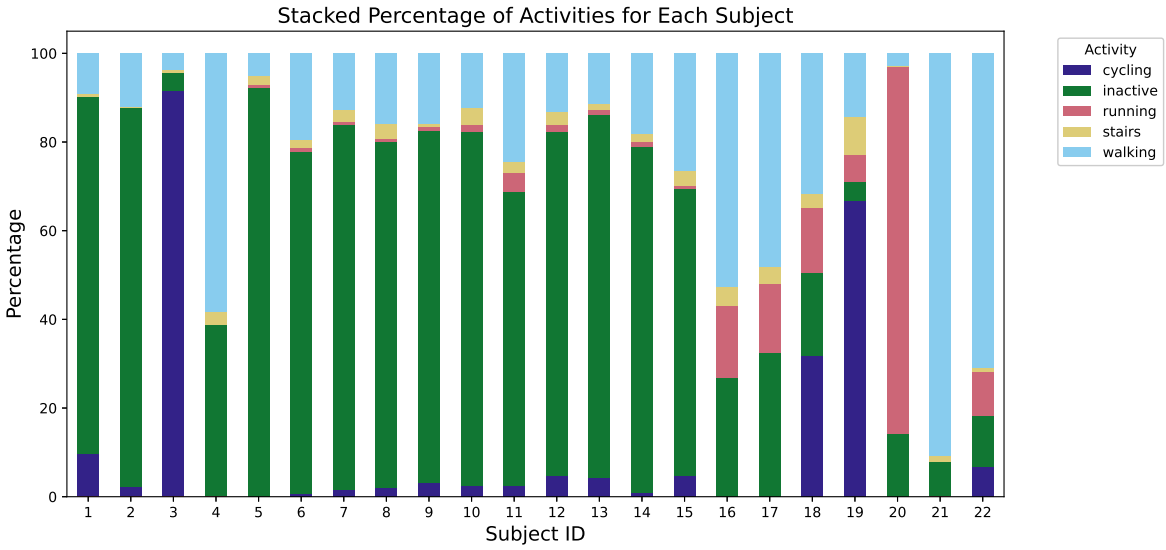

Figure 2: Class distribution at the subject level.

Feature dynamics are analyzed to identify patterns and relationships that enhance model predictive power. For instance, the back sensor's x-axis acceleration (`back_x`) over time is depicted in Figure 3. The analysis shows distinct activity-specific behaviors. Intense activities like cycling exhibit rapid fluctuations in `back_x` within short time windows, indicating higher variability. In contrast, less intense activities like walking or *inactive* maintain relatively stable values, suggesting lower variability. This observation underscores the predictive potential of standard deviation as a derived feature. Consequently, features representing the standard deviation for each of the six acceler-

ation components over time windows are engineered to enhance the model's ability to differentiate between activities.

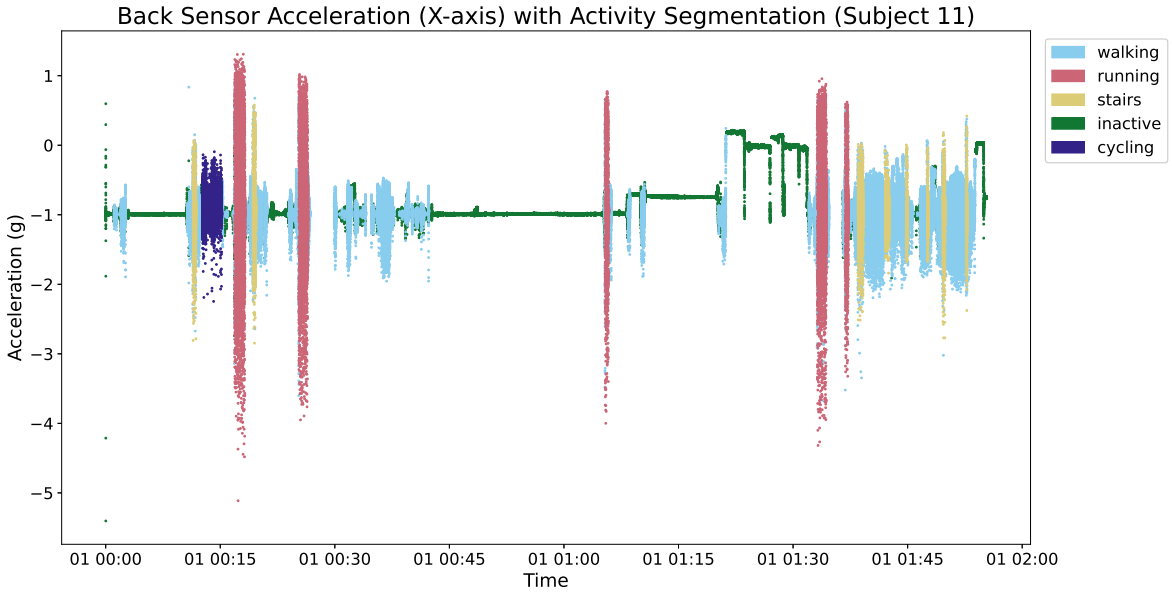

Figure 3: Variation of `back_x` over time for a single subject across all the activities that the subject performs.

The effectiveness of this engineered feature is demonstrated in Figure 4, which illustrates the distribution of `back_x_std` across all activities. The figure highlights the capability of this feature to discriminate between activities.

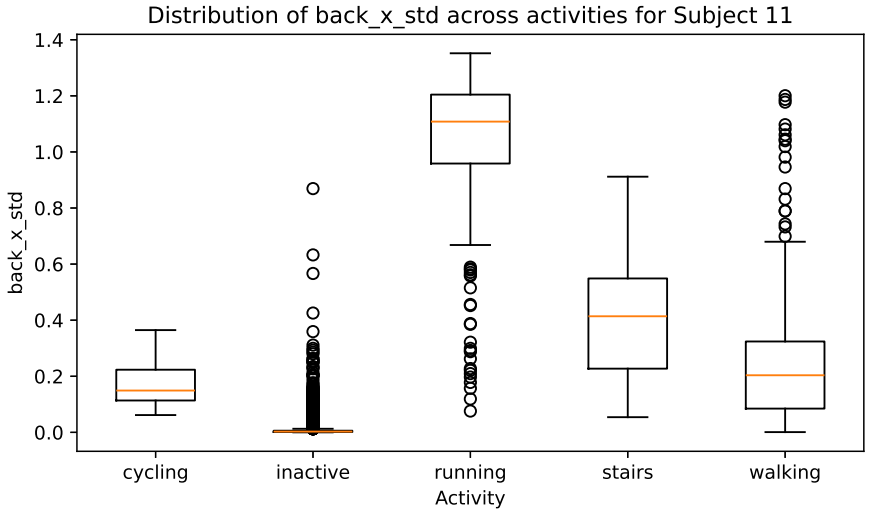

Figure 4: Distribution of `back_x_std` for a single subject across all the activities.

3 Methods

3.1 Data Splitting

To handle the group structure present in the dataset, 'GroupShuffleSplit' is used to create a test set comprising 20% of the data while ensuring no overlap of group memberships between the train_val and test sets. Within the remaining 80%, 'StratifiedGroupKFold' is used for 5-fold cross-validation, maintaining both class distribution and group boundaries in each fold.

3.2 Data Preprocessing

As all features are continuous, the dataset is normalized using `StandardScaler`, transforming each feature to have zero mean and unit variance. This preprocessing step improves model convergence and ensures consistent scaling across features.

3.3 ML Pipeline

The pipeline is built to compare the performance of four machine learning algorithms: Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, and XGBoost. Macro F1-score is chosen as the evaluation metric due to its ability to provide balanced performance evaluation in the presence of imbalanced datasets. It computes the harmonic mean of precision and recall for each class and averages these values across all classes.

Model	**Hyperparameter**	**Values**
Logistic Regression	C	$10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, \mathbf{10^{3}}, 10^{4}, 10^{5}$
K-Nearest Neighbors	n_neighbors	1,**30**, 100, 300, 1000
	weights	uniform, **distance**
	metric	minkowski, euclidean, **manhattan**
Random Forest	max_depth	$1, 3, 10, \mathbf{30}, 100$
	max_features	$0.25, \mathbf{0.5}, 0.75$
XGBoost	n_estimators	$1, 30, 100, 300, \mathbf{1000}$
	max_depth	$1, 3, \mathbf{10}, 30, 100$

Table 2: Hyperparameters tuned for different models (optimum values in bold)

Hyperparameters for each model are optimized using `GridSearchCV`, with the explored ranges detailed in Table 2. To account for uncertainties due to split randomness and stochastic behavior in certain models (e.g., Random Forest), the evaluation process is repeated across five random seeds. The final scores for each model are averaged to ensure reliable performance estimates.

4 Results

4.1 Model Performance Overview

Figure 5 presents a comparison of model performance in terms of both Macro-F1 score and Accuracy. XGBoost outperforms all other models in both metrics, achieving a Macro-F1 score of 85% and an accuracy of 94%. K-Nearest Neighbors follows closely with a Macro-F1 score of 84% and the same accuracy of 94%. Logistic Regression, however, performs the worst among the four models, with a Macro-F1 score of 81% and accuracy of 93%.

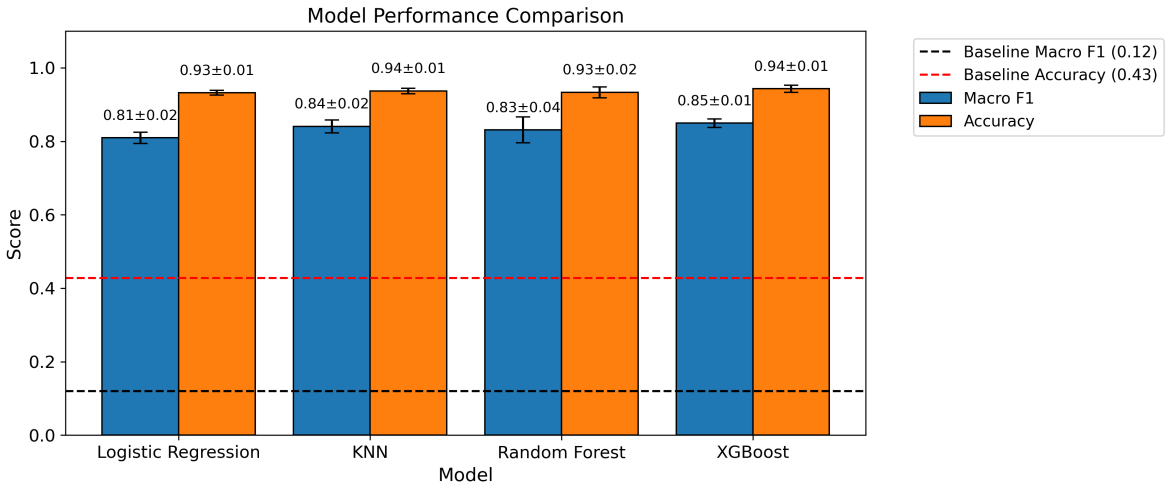

Figure 5: Model performance comparison in terms of Macro-F1 and Accuracy.

In comparison to the baseline, all models significantly outperform it, with the baseline Macro-F1 being 12% and baseline accuracy at 43%. XGBoost's Macro-F1 score of 85% is approximately 36 standard deviations above the baseline Macro-F1, signifying a substantial improvement in performance. As shown in Figure 6, the confusion matrix for XGBoost reveals that most of the diagonal elements are close to one, reflecting accurate predictions. However, the model struggles with distinguishing the true label stairs, often confusing it with walking. This could be due to the similarity in acceleration patterns between these two activities, as the acceleration during both movements is expected to be quite similar.

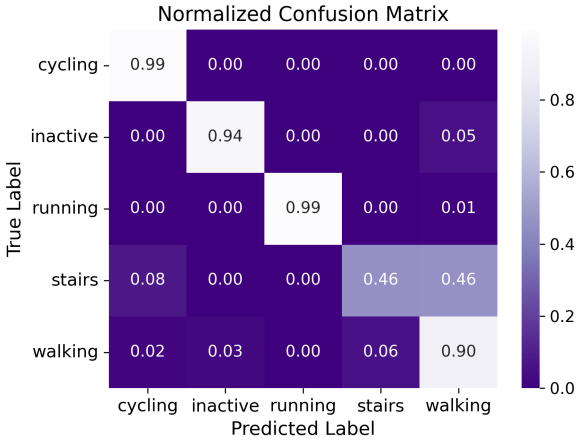

Figure 6: Confusion Matrix for XGBoost

4.2 Global Feature Importance

To evaluate the global feature importance in the XGBoost model, I use three distinct methods:

1. **Perturbation Importance**

   The perturbation importance method ranks features based on their impact on model performance when shuffled. This analysis shows that `back_x_std` and `thigh_z_rms` consistently emerge as highly important features. These features likely capture the core patterns of human motion, which are essential for distinguishing between the different activity classes.

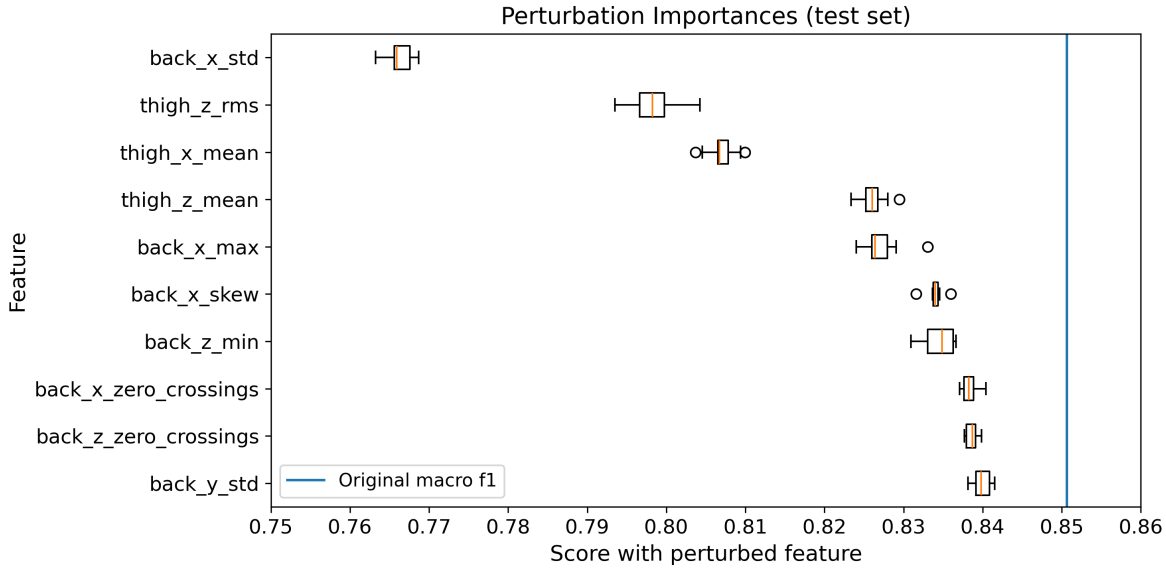

Figure 7: Top-10 important features based on perturbation feature importance.

2. **SHAP (Shapley Additive Explanations)**

   The SHAP analysis further reinforces the importance of `back_x_std` and `thigh_x_mean`, which are top-ranked across the board. These features likely capture differences in the movement and posture of individuals, making them crucial for accurate classification. Notably, `thigh_x_mean` stands out as particularly important, suggesting that the horizontal movement of the thigh plays a significant role in differentiating activities.

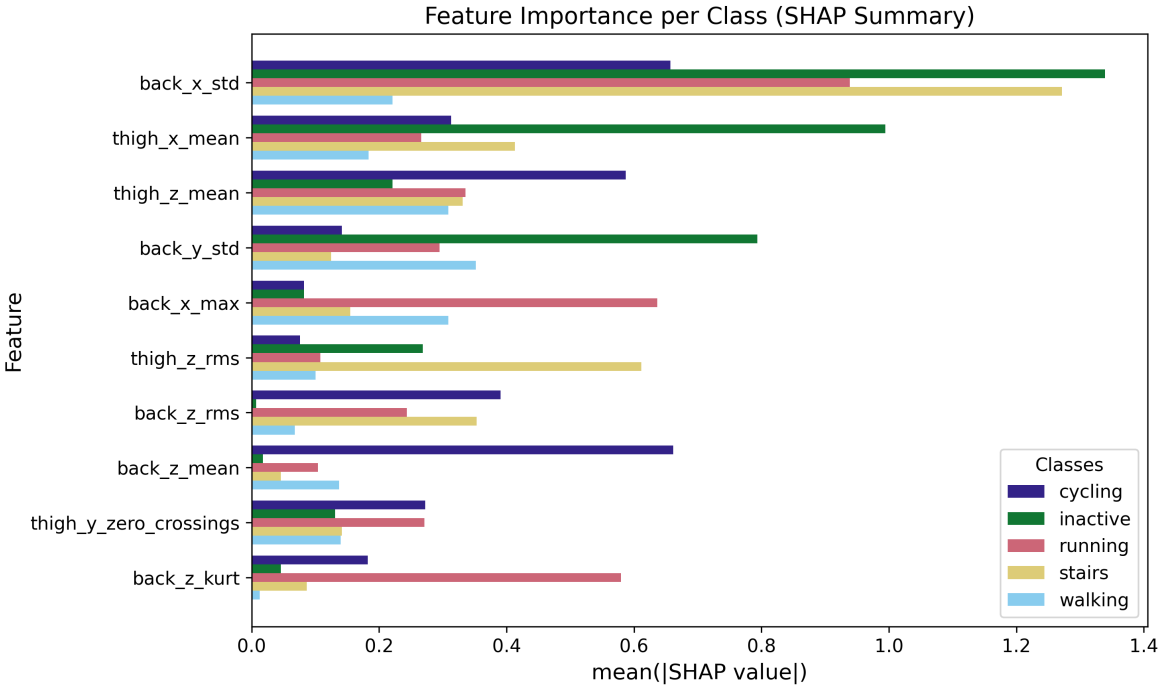

Figure 8: Top-10 important features based on mean SHAP value.

3. **XGBoost-Specific Metrics**

   XGBoost provides several feature importance metrics, including gain, weight, cover, total_gain, and total_cover. While I calculate all of these metrics, I only include Total Gain in this report. This metric measures the total contribution of each feature to improving model accuracy. Similar to the SHAP and perturbation importance results, `back_x_std` and `thigh_x_mean` are consistently identified as key contributors. This indicates that these features hold intrinsic relevance for the given classification task.

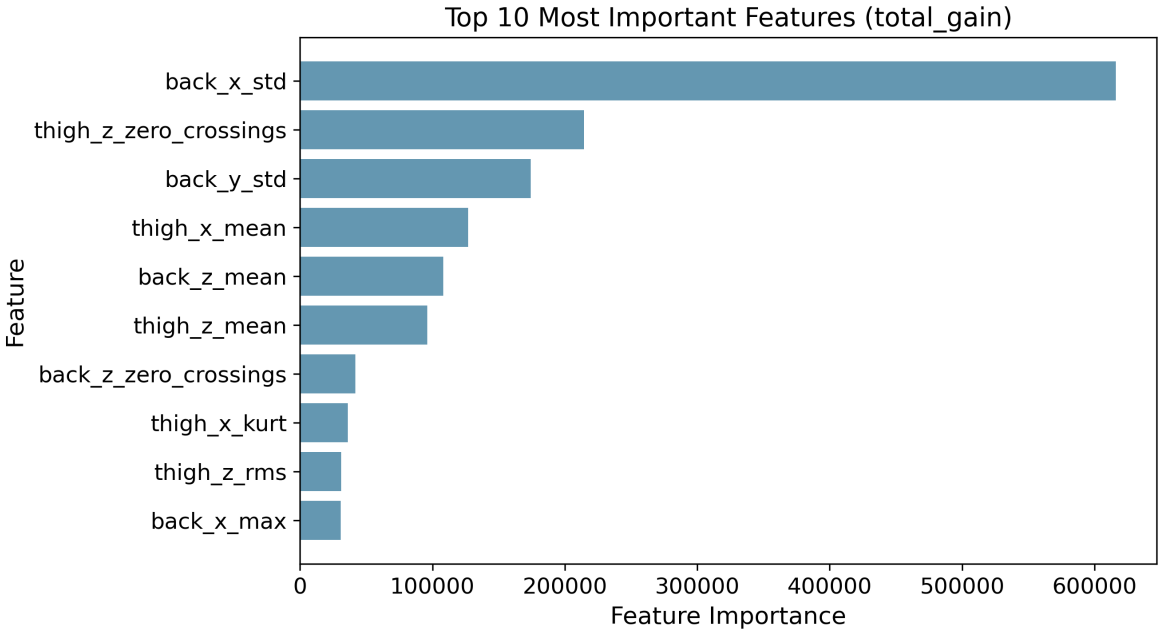

Figure 9: Top-10 important features based on XGBoost's total_gain importance measure.

4.3 Local Feature Importance

I use SHAP force plots corresponding to 'cycling' class to analyze the contribution of individual features for 2 specific test samples. Figure 10 shows how features like `back_x_std` and `thigh_x_skew` positively impact the prediction (red), while features like `back_z_mean` and `thigh_z_mean` pull the prediction lower (blue) for random sample #1. For random sample #2 (Figure 11), features like `thigh_z_mean` and `back_z_mean` positively impact the prediction, while features like `thigh_y_rms` and `back_x_skew` pull the prediction lower.

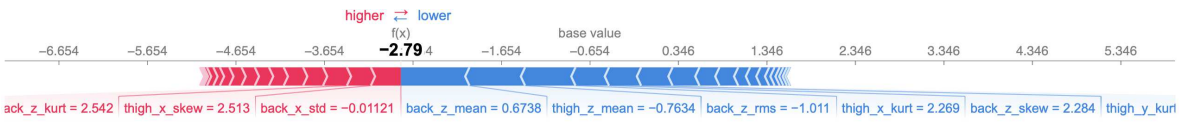

Figure 10: SHAP force plot for random sample #1 corresponding to 'cycling' class

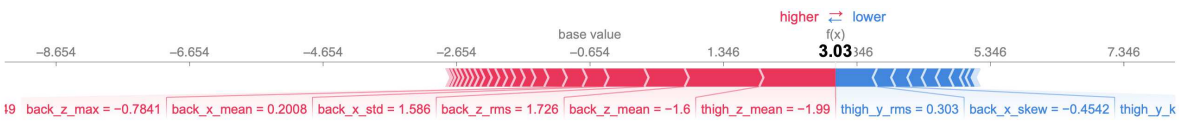

Figure 11: SHAP force plot for random sample #2 corresponding to 'cycling' class

5 GitHub Repository

https://github.com/devraj-raghuvanshi/human-activity-recognition

6 Outlook

To improve the model's performance and interpretability, I could treat the window length as a hyperparameter and explore values in the range of 1 to 10 seconds. Additional feature engineering, such as extracting frequency domain features, could provide a more comprehensive signal representation. To address class imbalance, I would implement techniques like Synthetic Minority Oversampling Technique (SMOTE). Exploring advanced classification algorithms, such as ensemble methods or neural networks, and incorporating temporal models like recurrent neural networks (RNNs) could further enhance the results. Collecting additional data for underrepresented classes to augment feature sets would also likely improve performance. These steps would help mitigate the current limitations and refine the overall modeling approach.

References

[1] Sebastian Scheurer, Salvatore Tedesco, Brendan O'Flynn, and Kenneth N Brown. Comparing person-specific and independent models on subject-dependent and independent human activity recognition performance. *Sensors*, 20(13):3647, 2020.

[2] Aleksej Logacjov, Kerstin Bach, Atle Kongsvold, Hilde Bremseth Bårdstu, and Paul Jarle Mork. Harth: a human activity recognition dataset for machine learning. *Sensors*, 21(23):7853, 2021.

[3] Human activity recognition trondheim (harth) dataset. *UCI Machine Learning Repository*, 2023. https://archive.ics.uci.edu/dataset/779/harth/.

[4] Benish Fida, Ivan Bernabucci, Daniele Bibbo, Silvia Conforto, and Maurizio Schmid. Varying behavior of different window sizes on the classification of static and dynamic physical activities from a single accelerometer. *Medical engineering & physics*, 37(7):705–711, 2015.

[5] Kerstin Bach, Atle Kongsvold, Hilde Bårdstu, Ellen Marie Bardal, Håkon S Kjærnli, Sverre Herland, Aleksej Logacjov, and Paul Jarle Mork. A machine learning classifier for detection of physical activity types and postures during free-living. *Journal for the Measurement of Physical Behaviour*, 5(1):24–31, 2021.