Human Activity Recognition from Accelerometer Data

Sagar Raichandani
Data Science Initiative, Brown University

Introduction

Lower back pain is an increasingly common, expensive, and debilitating condition that affects all age groups. One study estimated lower back pain to be the leading cause of disability globally, with more than 83 million years of cumulative global burden[5]. The inspiration for the project is drawn from the work of Sani, Wiratunga, et al. on the SELFBACK system[1] that identifies physical activities from accelerometer data to help manage lower back pain. To overcome the practical limitations of obtaining personalized user activity data, Sani, et al. utilize nearest neighbor similarity to generate a personalized model for the user based on subject independent training sets. Their novel approach allowed them to achieve F1 scores of more than 0.9 for certain activities, and an overall improvement of more than 5% compared to other algorithms that are designed for personalized classification based on subject independent data.

The dataset[2] was compiled by the team of the original paper[1], and sourced from the UCI Machine Learning Repository. My objective through this project is to develop a subject independent (unseen user) classifier that identifies physical activity based on an accelerometer's readings. While the researchers predicted the L2 classes (as in Figure 1), the target variable for this project is the L1 classification only.

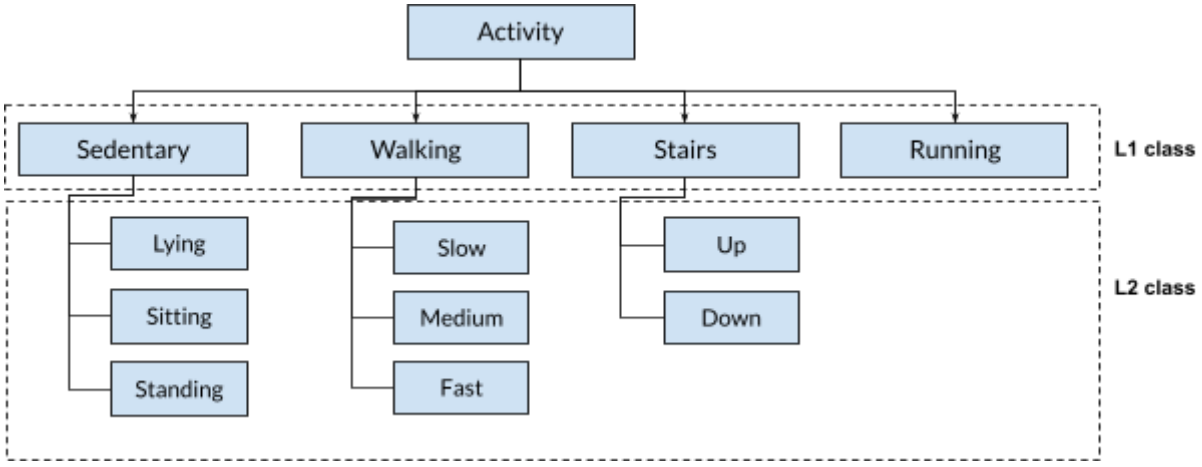

Fig 1. Flowchart of the activity classification present in the dataset– L1 activity classes is the project objective.

The dataset contains 9 activity classes (L1+L2 classes); 6 ambulatory and 3 sedentary, performed by 33 users. In total for the wrist-bound accelerometers there are 4.8 million rows, and 4 features in addition to the target variable activity class. The features are presented in table 1 below.

Feature	Description
Time	Time of signal measurement from accelerometer
x	Acceleration in g's along x-axis in 3D plane [-8g to + 8g]
y	Acceleration in g's along y-axis in 3D plane [-8g to + 8g]
z	Acceleration in g's along z-axis in 3D plane [-8g to + 8g]
Class	Activity type

Table 1. Dataset features and their descriptions.

Essentially, the time-series data is the output of tri-axial accelerometer readings with a sampling rate of 100 Hz. (i.e. 100 samples/second). Figure 2 presents a snapshot of the unprocessed readings of the accelerometer.

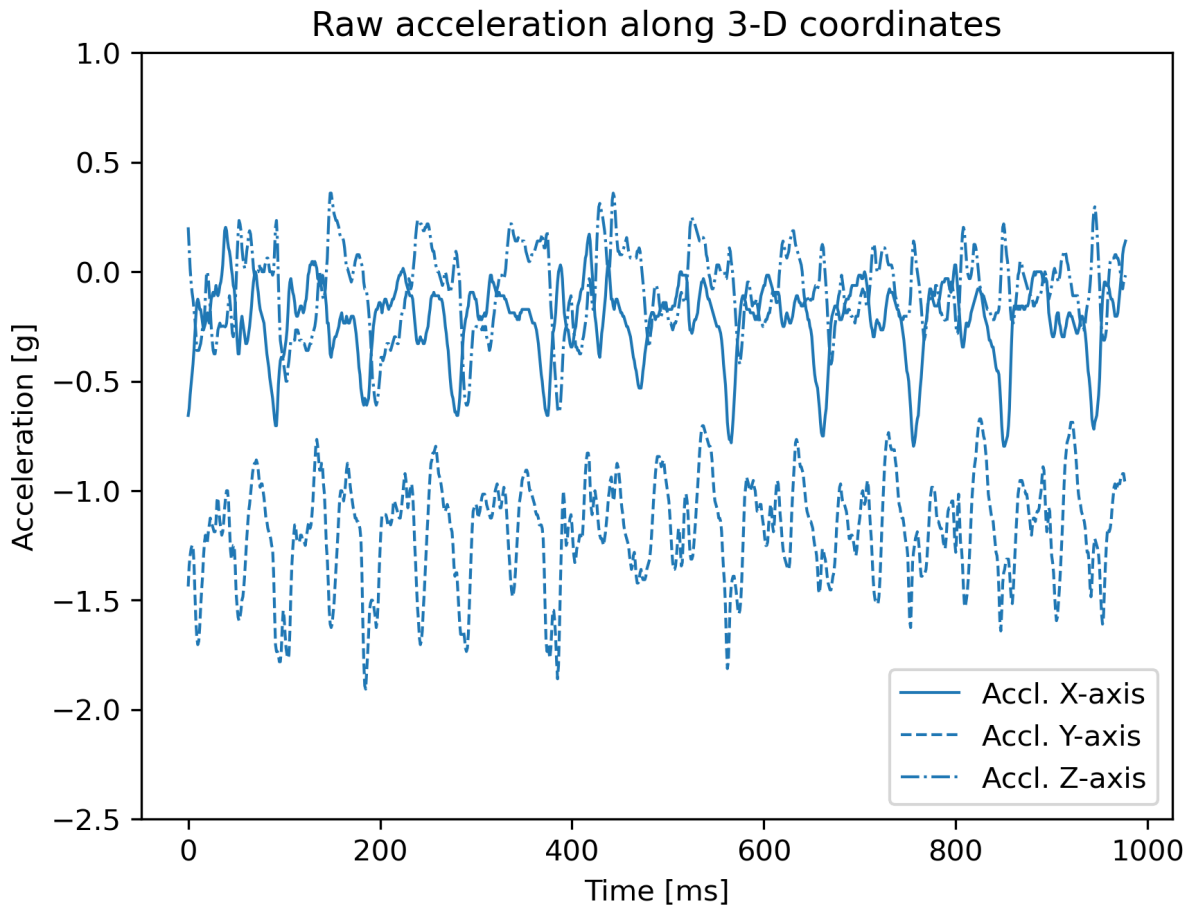

Fig 2. A time-series plot of the acceleration along 3-D coordinates. Time is measured in milliseconds along the horizontal axis, and acceleration [1g = 9.8 m/s^2] is along the vertical axis.

As is evident from figure 2 the readings are not independent nor identically distributed, and hence require some processing before model development. Likewise, the limited number of features necessitates the need for feature engineering to extract information that may be useful for the classifier. These constraints are addressed in the Windowing and Feature Engineering sections that follow.

Windowing

Windowing [3] is the process of dividing (or partitioning) time-series data into uniform "chunks" of a certain time *t*, measured in seconds here, with an overlap factor (0.5 in this case). Each window is constructed in a manner that is specific to a user and an activity class. Each partition w_i is then identical and independent to partitions w_{i-1} and w_{i+1}, and is the basis for feature engineering and activity classification. Figure 3 illustrates this process for the 3 features *x, y,* and *z* from the dataset.

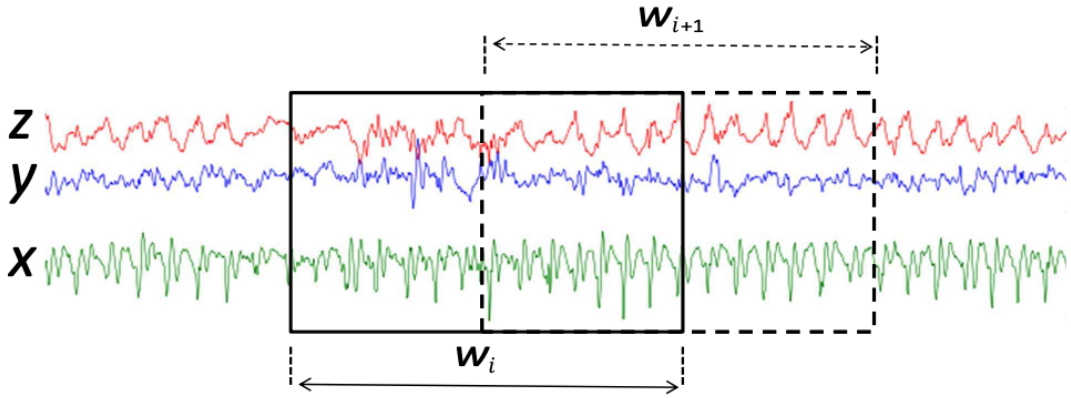

Fig 3. Illustration of accelerometer data windowing.

An important facet here is the length of window duration *t*; ideally, this should be chosen considering a balance between latency and accuracy. Typically, shorter windows have lower latency and accuracy; longer windows have better accuracy but higher latency. In literature [3, 4] the window duration is typically set to 10 secs, and I have considered the same. Since the sampling rate here is 100 Hz, a typical window in the partitioned dataset contains around 1000 points. Consequently, after windowing the dataset reduces from 4.8M rows to about 9.4K.

Feature Engineering

Feature Engineering is the process of extracting useful information from underlying data in order to improve the model's performance. While the scope for feature extraction with this dataset is vast, I have primarily focused on the categories presented in Figure 4 below.

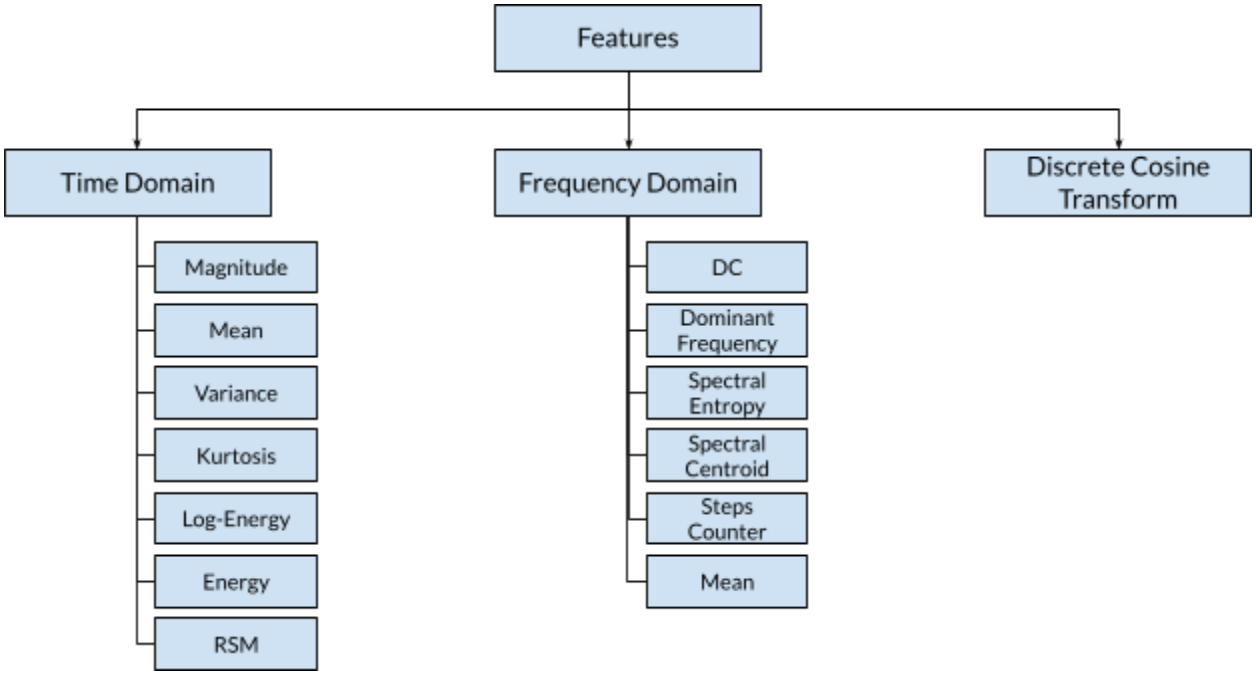

Fig 4. Flow chart of features extracted from the dataset. Each domain represents a class of features built of functions defined in that domain.

Here's a short description of each of the feature types:

1. **Time domain-** Analysis of functions that depend upon and vary with time. These can be statistical features of the data such as mean, variance, skewness, and more advanced quantities like energy of a discrete-time signal, root mean square, power, etc.
2. **Frequency domain-** Analysis of functions that are defined by frequency– or how much of a signal lies within a frequency for a range of frequencies. Most of these are the result of the application of the Fast Fourier Transform on the signal data.
3. **Discrete Cosine Transform-** DCT expresses a finite discrete signal as a sum of cosine functions at different frequencies. DCT compresses all the energy of the original signal into as few coefficients as possible and returns an ordered set of coefficients such that the most significant information is in the lowest indices of the ordered set. For the purpose of feature representation, only the absolute magnitude of the coefficient is of interest.

While feature engineering is an on-going portion of this project, at present the number of features have expanded from 3 to 45. Therefore, with windowing and feature engineering completed we now have a dataset with around 9,400 rows and 47 columns (45 numerical features, 1 target variable, and 1 group variable- i.e. user).

Exploratory Data Analysis

Now that we have reduced our dataset and expanded the usable features, we begin EDA to get a better understanding of the underlying class imbalance, distribution of the extracted quantities, and relation among features and with the target variable. We start with class ratios to get a sense of any skew or need for stratification– Figure 5 illustrates the class balance as a bar chart.

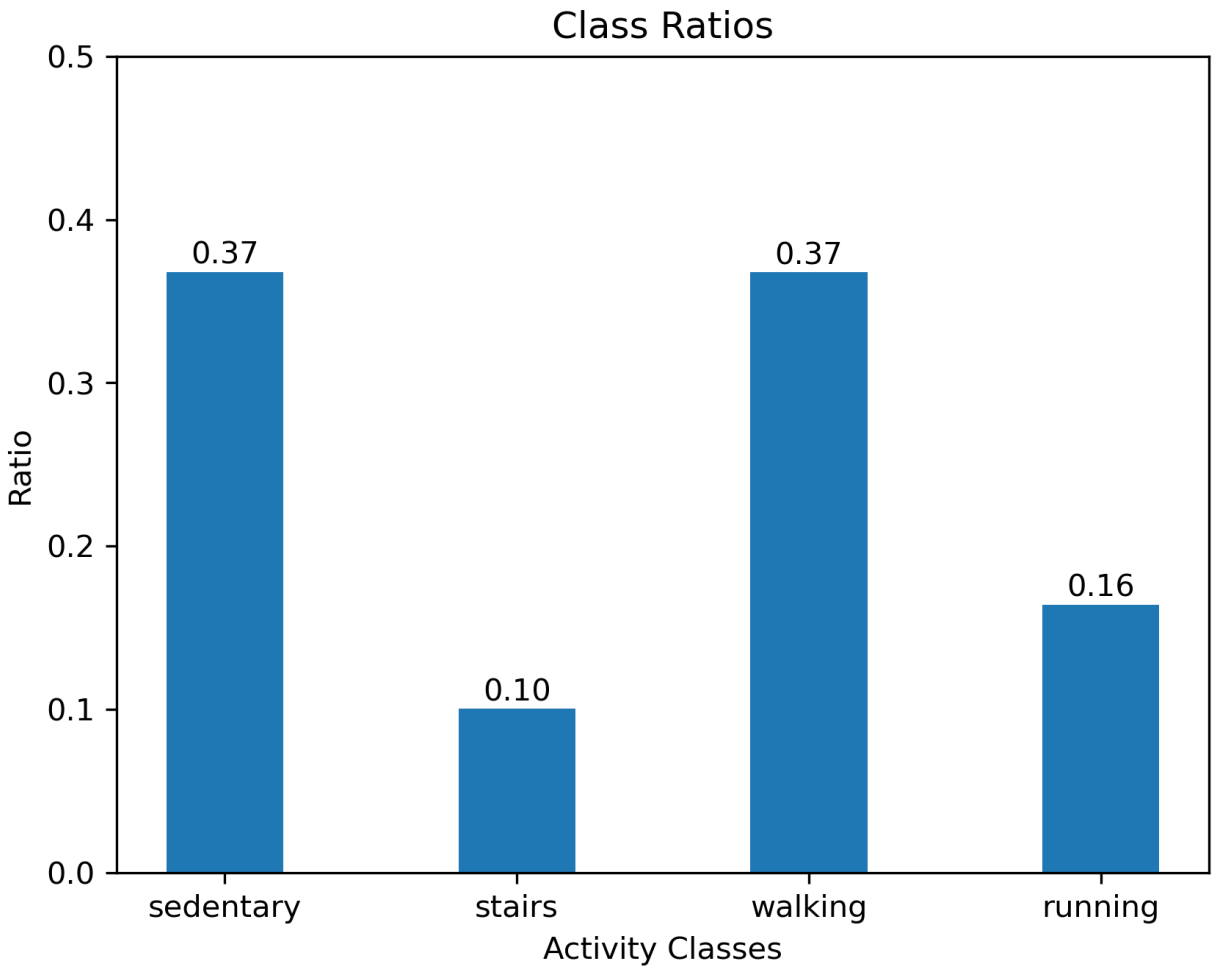

Fig 5. A bar chart of the class ratios based on L1 classification. Here the ratio on the vertical axis represents the portion of class data points off the total. The horizontal axis represents class labels.

Next, we take a look at the change in magnitude of acceleration (time domain feature) by activity class intensity. Figure 6 illustrates that as activity intensity increases, so does the magnitude of acceleration– bearing with reasonable intuition. The difference between classes *stairs* and *walking* is quite limited, and we may need the addition of another feature to create a stronger discrimination between the two classes.

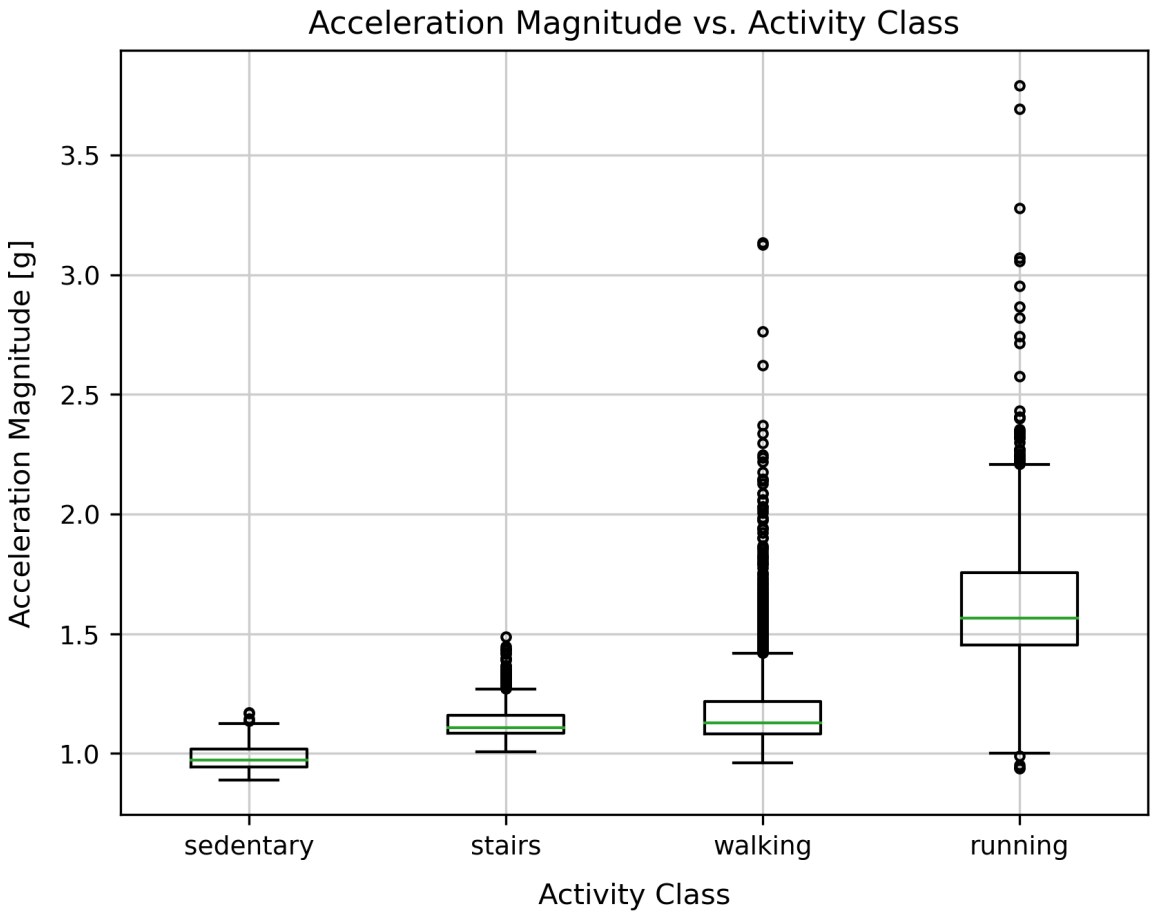

Fig 6. Box-plot comparing the acceleration magnitude of each activity- with class on the horizontal axis and acceleration [g] on the vertical axis. As the activity intensity increases so does the reading on the accelerometer.

Similarly, let's consider the feature dominant frequency from the frequency domain features. To put it succinctly, dominant frequency is the one that carries the most energy with respect to all other frequencies in the spectrum of the signal. In figure 7 below, we see that there is sufficient discrimination between dominant frequencies of all classes except *stairs* and *walking*. Further, it's evident that as activity increases so does the frequency– again, bearing with our intuition this makes sense as higher activity intensity is usually associated with more rapid hand movements thereby shifting the dominant frequency to higher ranges.

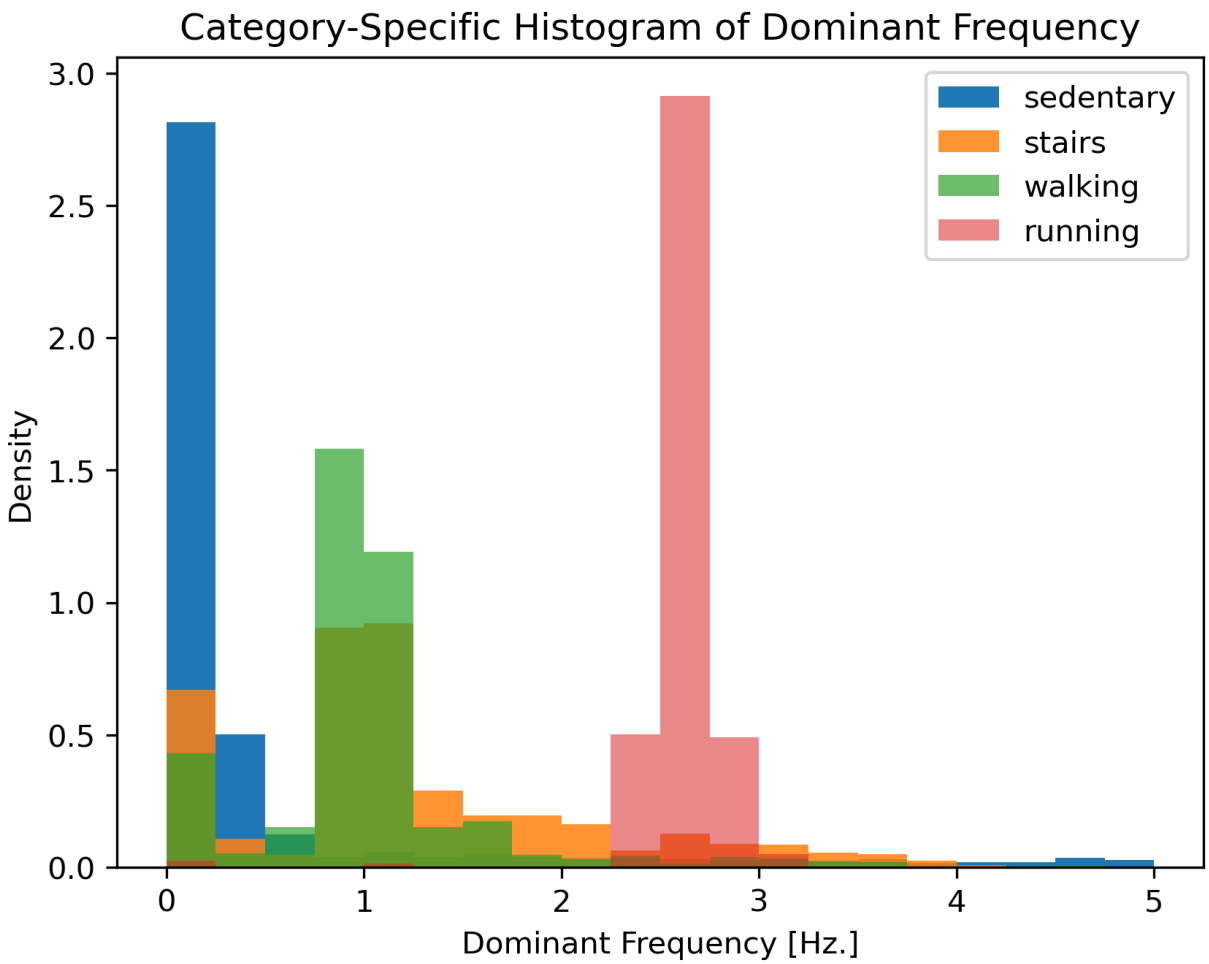

Fig 7. A category-specific histogram of the dominant frequency for each activity class. With dominant frequency on the horizontal axis and density on the vertical axis, each activity exhibits a different dominant frequency with an increase in dominant frequency as activity intensity increases.

Lastly, let's take a look at the relationship between the log-energy (time domain feature) and the number of steps taken (frequency domain feature). In figure 8 below, we see that as activity increases the number steps taken per window increases; however the spread about log-energy remains quite high. While this scatter plot highlights some of discrimination between classes *stairs* and *walking*, it is still not sufficiently pronounced.

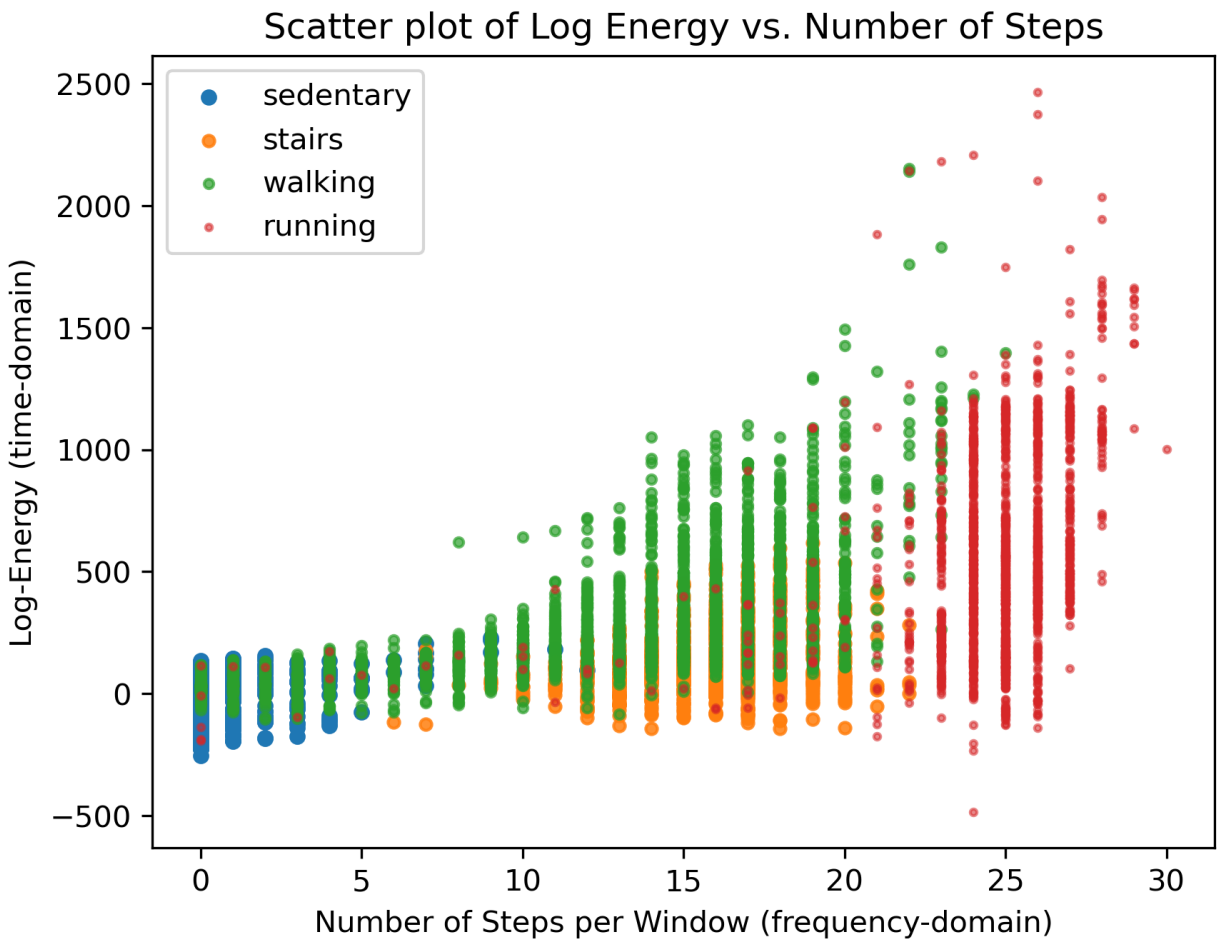

Fig 8. A scatter plot of the relationship between the number of steps taken and log-energy. Each point in the graph represents one instance of the number of steps and log-energy for a window. Broadly, as the activity intensity increases so does the number of steps taken per window. Log-energy, however, has a very high spread.

Data Preprocessing

In data preprocessing we have two main steps: splitting the data into train, validation and test sets, and scaling the features based on their type. Let's take a look at each of the steps individually

1. Splitting
   Bearing that our target variable is the activity class and our goal is to predict the activity for a previously unseen user there are three factors affecting our dataset:
   - User defines group structure
   - Data is not shuffled as windows are built sequentially
   - Class imbalance (as shown in figure 5)

While each of these conditions are true, we find that since our goal is to predict activity for a previously unseen user we must optimize for this as opposed to the need for stratification to meet class imbalance. Consequently, we build the test set using sklearn's *GroupShuffleSplit* method with *n_splits* (number of splits) set to 1 and *test_size* (number of groups) set to 6, thereby achieving an approximate 20% split from the base of 33 users. The flowchart below (figure 9) summarizes this process

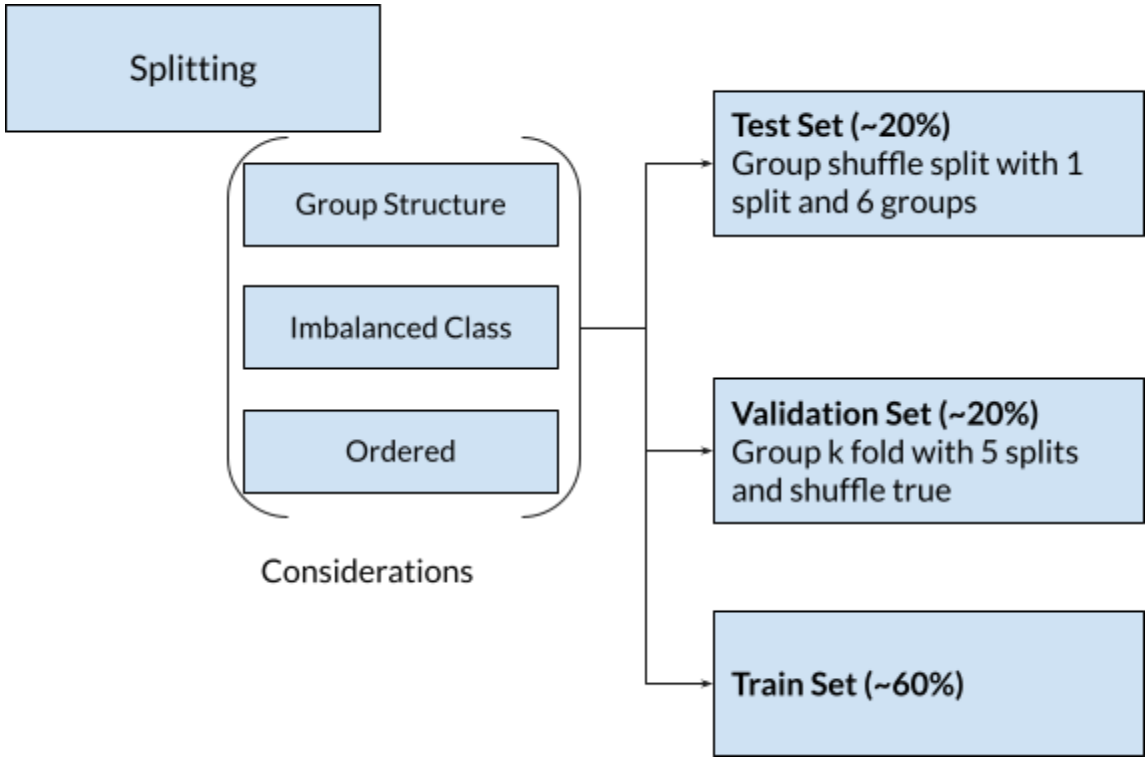

Fig 9. Summary of the considerations, methods, and parameter arguments for splitting.

2. Feature Scaling

   In the feature set for this problem, all features are numerical and continuous- hence, sklearn's *StandardScaler* is the optimal choice for scaling. The *StandardScaler*, scales the feature values by reducing the mean to 0 and the standard deviation to 1. This process has no impact on the shape of the feature set in this case as, unlike *OneHotEncoder* or *OrdinalEncoder* the *StandardScaler* does not add any new splits in the underlying features. Thus, the number of features continues to remain at 45 after scaling.

References

[1] kNN Sampling for Personalized Human Activity Recognition
Sadiq Sani, Nirmalie Wiratunga, Stewart Massie, and Kay Cooper
[2] UCI Machine Learning Repository: selfBACK data set
[3] Physical Activity Recognition from Accelerometer Data Using a Multi-Scale Ensemble Method
Yonglei Zheng, Weng-Keen Wong, Xinze Guan, Stewart Trost
[4] Activity Recognition from acceleration data based on Discrete Cosine Transform and SVM
Zhenyu He, Lianwen Jin
[5] Placing the global burden of lower back pain in context

GitHub

https://github.com/raichandanisagar/human-activity-recognition