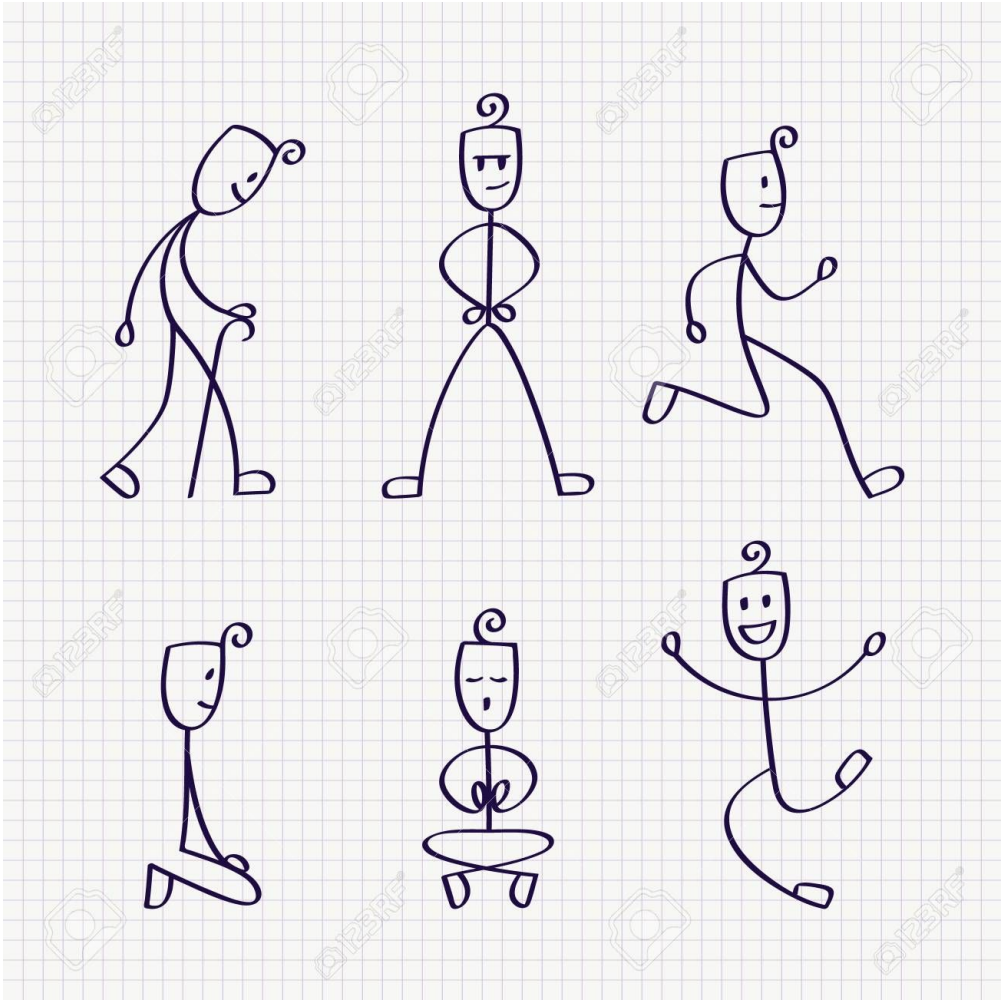
CS205: Health Analytics

# Project Report

# BIOMETRICS BASED ACTION PREDICTION



Aditi Mithal (105027598)
Claudia Seidel (004945986)
Debleena Sengupta (004945991)
Jayanth (405026111)

# Introduction

The emergence of smart watches has taken the health and fitness world by a storm. With access to a plethora of real time health data, a variety of apps and other wearable technology has come to the market to collect and analyze this data. In this project, we collected data via a variety of sensors (accelerometer, gyroscope, etc.) on a smartwatch, and used this data to train a machine learning model to predict if a person is performing one of **4 actions: walking, sitting, standing, or lying down.**

# Overview

Through the course of this project, we developed a machine learning model to predict activity based on sensor data from a smart watch **over a series of iterations**. In the subsequent sections, we will describe the following four steps:

1. Motivation behind each model
2. How the model performed
3. What problems we saw with the model
4. How we addressed those problems to get to the next iteration of the model

Using data set from one user, we created a train set, validation set, and a **personal test set**.
Using data set from another user, we created an **impersonal test set**.

For each of the models we created, we trained and validated based on data collected by one user and **tested the model using data from a different user.**

We began our search for a good model with 5 classification models that took data from all sensors as features. Out of the 5 models, the Random Forest model perform the best, producing around 99% accuracy on the personal test set but only roughly 35% on the personal test set. In order to improve the score of the test set, we decided to build an ensemble Random Forest model (described in more detail below). With this model, we obtained around 98% personal test accuracy and were able to achieve a better impersonal test accuracy of around 48%. Furthermore, to improve the impersonal test accuracy, we decided to explore feature engineering. As per the literature review, we figured out that data samples which are feature engineered using mean and standard deviation for a given time window have shown good results in the past. We delved into extracting new features by applying this technique and saw an impersonal test accuracy of ~58% and a personal test accuracy of around ~99%. We then decided to explore LSTM (long short term memory) models by leveraging the sequential-characteristic of the data. LSTMs inherently ensure feature selection and feature engineering. This resulted in an increase in performance/accuracy by ~15% bringing the **impersonal test accuracy up to 63.55%** with **personal test accuracy of 95.89%.**

# Literature Review

A vast variety of paths fall under the umbrella of “human activity recognition”, but the focus of many academics, including ourselves, is human health. This field itself is also broad, with examples of it including anything from posture tracking and counting calories, and there have been many different studies conducted on the subject. These range from tracking and analysis of actual day-to-day activities within the home (e.g. washing hands, doing laundry, preparing different meals) [1] to trying to model human energy expenditure during physical activity among different demographics [2, 3]. One often-seen

take on these kinds of research questions is collecting data with accelerometers clipped to the body, with the most common locations being on the waist/hip, wrist, or ankle [4, 5]. The method itself is simple, and the data collection is done with an easily accessible sensor, making it one of the most convenient approaches. This data also provides a strong foundation for research, establishing solid findings that can either stand on their own or serve as a basis for further academic inquiry. Steps forward in wearable sensor technology have shifted the dynamics of this, though. Smart wristbands and watches now come equipped with a host of useful sensors- including, but definitely not limited to, accelerometers- within one easily worn device, and are becoming an increasingly popular choice for collecting and working with human activity data. Their ubiquity and the constant improvements made in the wearable technology industry put them at the forefront of the activity tracking area, with some academics even considering that smart watches could replace previously established methods in the field [6].

Advances in technology for data collection also lead to advances in data processing. Machine learning techniques are favored for analyzing human activity data, particularly more flexible methods involving deep learning and neural networks. Across the different kinds of approaches and technology, there is one consistent key part of analysis: the importance of setting up impersonal as opposed to just personal models. It is common for personal models to significantly outperform impersonal models in accuracy [7], which can create an illusion of extremely high accuracy. In reality, models with a personal basis will often underperform once tested outside of the lab/research environment that they were constructed and trained in [8]. We made sure to take this into account in our work, setting up measures for impersonal and personal models during the testing of our multiple Long Short-Term Memory (LSTM) layers, informed by previous work with similar aims that made use of convolutional neural networks [9].

# Dataset

## Data Collection

The dataset was collected in turns by each project member, collecting around 700 MB worth of data. The data was collected for four different activities - sitting, standing, walking and laying down. To generalize the dataset, the watch bearer collected data at different time intervals and also, at different locations to introduce variation to the data set. For example, for the laying down activity, data was collected on different surfaces, such as on a couch or bed. Different sensors have different sampling rate and also, the sensors don't have fixed sampling rate, for example accelerometer samples either every 3ms or 4ms, leading to non-periodic data. The final dataset size for each sensor was as following:

| | |
|---|---|
| accelerometer | 129580 obs. of 6 variables |
| any_motion | 891 obs. of 18 variables |
| gravity | 129425 obs. of 6 variables |
| gyroscope | 120791 obs. of 6 variables |
| light | 112 obs. of 6 variables |
| linear_acceleration | 129425 obs. of 6 variables |
| magnetic_field | 61671 obs. of 6 variables |
| merged_data | 144832 obs. of 38 variables |
| orientation | 120834 obs. of 6 variables |
| rotation_vector | 120834 obs. of 8 variables |
| step_counter | 708 obs. of 4 variables |

# Visualization

We created two types of visualization plots. One was used to plot the correlation between different values per sensor (for example, check correlation between the accelerometer X, Y, Z positions). The other plot was used to visualize the time-series pattern across the labels and the sensor parameters.

## Correlation Visualization

The visualization plots show correlation between different parameters of the sensors. This information is important to understand which parameters should be selected for our learning model. If the features are correlated, then this means there is redundant data and we can pick one feature to represent both of the correlated features.

In Figures 1-7, X2, X3 and X4 specify the XYZ positions at any point in time and X5 specifies an integer value for the accuracy of the sensor data. A reading of 3 in X5 implies that the sensor data collected is accurate while a reading of 0 means that the collected data is not that accurate.

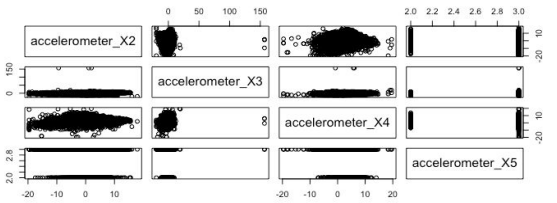



Fig 1: Accelerometer parameters

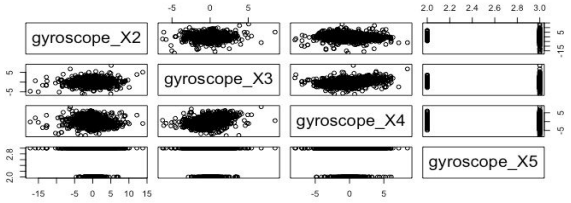



Fig 2: Gyroscope parameters

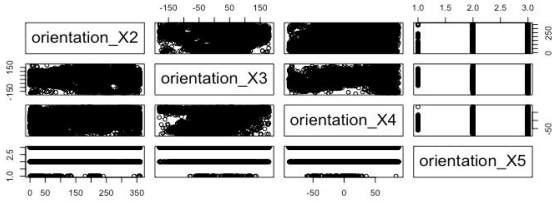



Fig 3: Orientation parameters

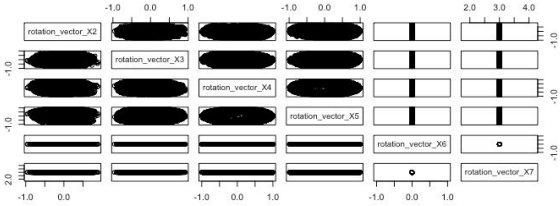



Fig 4: Rotation vector parameters

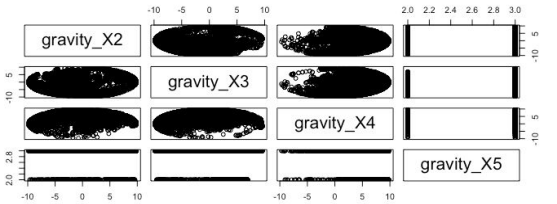



Fig 5: Gravity parameters

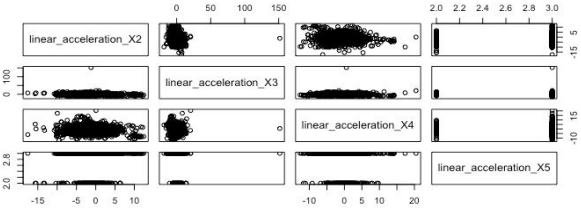



Fig 6: Linear acceleration parameters

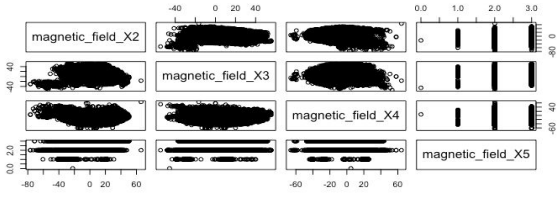



Fig 7: Magnetic Field parameters

## Time Series Visualization

Figures 8-14, below, show the sensor readings across time points to understand how the sensors are affecting the activity label. We can observe a correlation between the true label and sensor reading. In Fig 8 the epoch time where the label reads 4 indicated “lying_down” period matches with the sensor readings in Figures 9-14 . The activity recorded by the sensor is consistent with the true label across the time span for which the “lying_down” activity has been recorded, i.e. almost constant readings with very low variations.

We can also observe that for time duration where the labels are walking/standing the sensor readings are very close and are harder to distinguish between for these activities along with lot of spikes/variance during the period compared to idle activity like laying down.

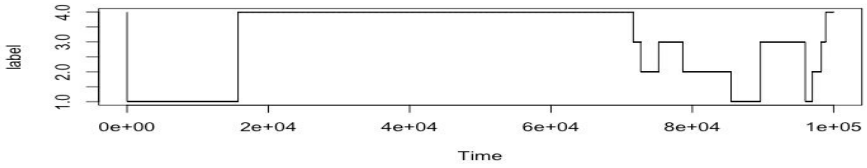



Fig 8: True labels

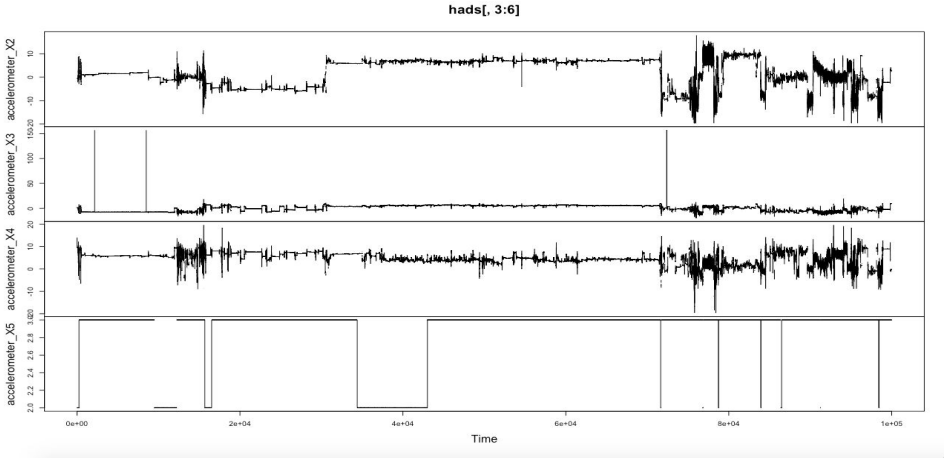



Fig 9: Accelerometer readings

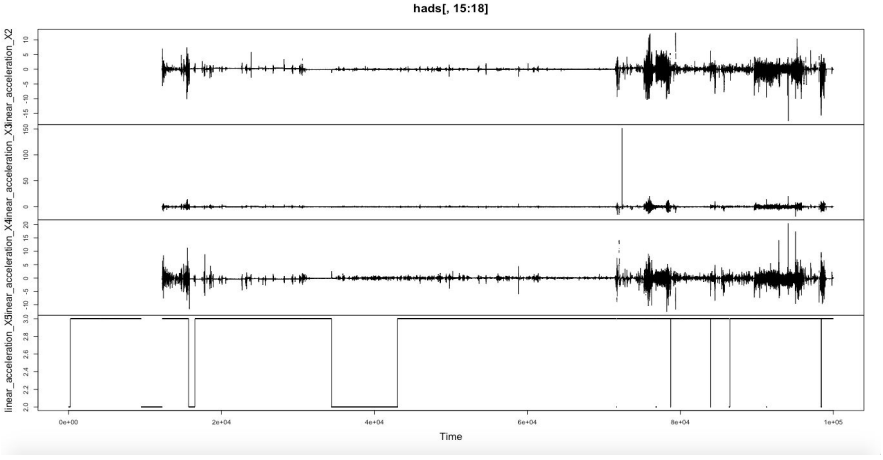



Fig 10: Linear acceleration readings

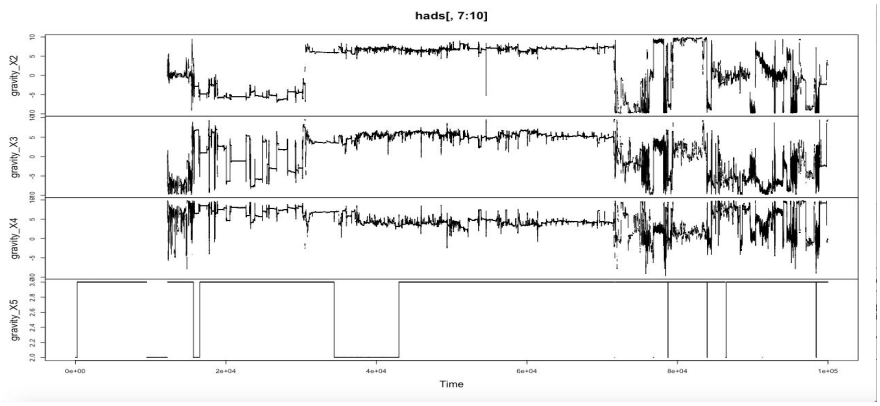



Fig 11: Gravity readings

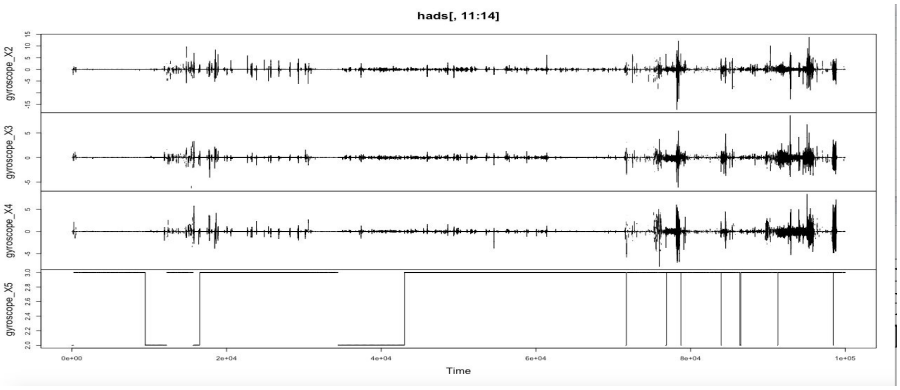



Fig 12: Gyroscope readings

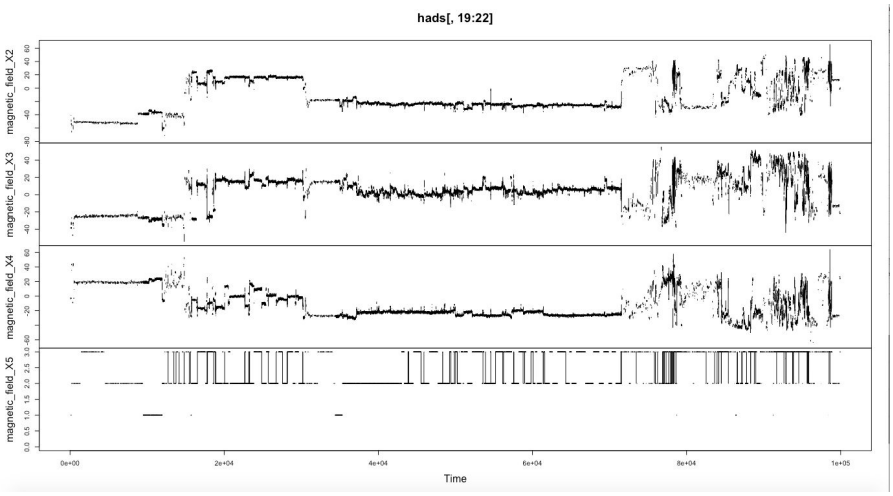



Fig 13: Magnetic Field readings

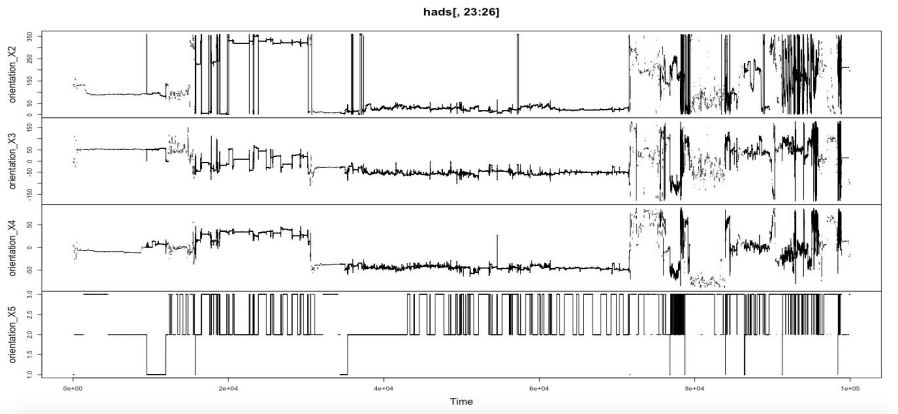



Fig 14: Orientation readings

## Data Processing

After analyzing the results from above, we decided to use data from the following sensors:

1. Accelerometer
2. Gravity
3. Gyroscope
4. Linear Acceleration
5. Magnetic Field
6. Orientation
7. Rotation Vector

After this preprocessing, we created a random sample of 100K data points from our whole data and split it into 90-10% train-test. Figure 15 shows a snapshot of the training data distribution we sampled for modelling.


Figure 15: Training Data Distribution

# Initial Machine Learning Models

## Motivation

After looking at our **raw data**, we observed the following:

1. All sensors have different sampling frequencies
2. Even though they collect data at the same time intervals for the same activity, the timestamps don't match. For example, for the accelerometer, the data was recorded sometimes after 2ms, sometimes 3/4ms. This motivated us to perform the following steps:

We combined all the sensor data to create a single data set. To do this, we first merged the raw dataset into a single file by doing outer joins on the timestamp. We did the outer joins for merging because all the sensors have different sampling frequencies and we did not want to lose any data or sensor for training our model. This resulted in lot of NaNs in our data as some the sensors did not record values, when other sensors were fired (due to different sampling frequencies). We will address the NaNs issue in later parts.

Out of 10 features, we considered only 7 features in our initial run, eliminating - light, any_motion and step_counter. This was done because any_motion and step_counter provide a direct label for our model which renders the machine learning algorithm useless as these variables have been designed to make use of the raw data from the sensors we have been tracking. To ensure that we do not leverage undue advantage of the pre-trained data variables, we left them out because (step_counter only fires during walking and hence, will directly give away the label).

Picking right and sufficient features is important so that we get good accuracies but also, don't add noise (for example ambient light sensor is completely irrelevant for the activity prediction here).

## Models

Using the 90K samples in our training set, we fit the following models using all 7 sensor variables.

1. Linear Discriminant Analysis - LDA
2. Quadratic Discriminant Analysis - QDA
3. Logistic Regression/ Softmax Classifier using Neural net
4. Decision Trees
5. Random Forest

We also tried using subset of features on the individual models but the accuracies were not as high even after using all the features from all sensors. We also ensured that over-fitting was not happening by doing cross-validation on our dataset.

## Model Performance

After the model was fit, we tested it on the remaining 10K samples which we call the personal test set. All these models ended up giving accuracies of around 80-95% with NA values (60-70%) in the prediction. Random Forest was also not able to give labels to all inputs (only 30% labels were predicted without NA). However, it gave good **personal test accuracy for more than 99%** of the predicted labels because of it

being a bagging ensemble. LDA also had lower accuracy on the test set compared to QDA, implying the data is not linearly separable and hence, we did not choose to train and test models like Naive Bayes, Plain SVM etc. However, the **impersonal test accuracy was about 35%** so we decided to explore better models.

| Model | Personal Test Accuracy | # NaN Predictions |
|---|---|---|
| LDA | 90.26% | 6920 |
| QDA | 99.51% | 6920 |
| Softmax/Logistic | 92.24% | 6920 |
| Decision Tree | 65.06% | 6920 |
| Random Forest | **99.58**% | 6920 |

Table 1: Results of Model 1

From the table, we can observe that there are so many NaNs in the data set based on our construction, which leads to larger predictions as NaNs. We also, made a plot of variable importance in Decision Classifier to check whether all sensors were important. Indeed it seemed so that all sensors were contributing (in top 10 features).

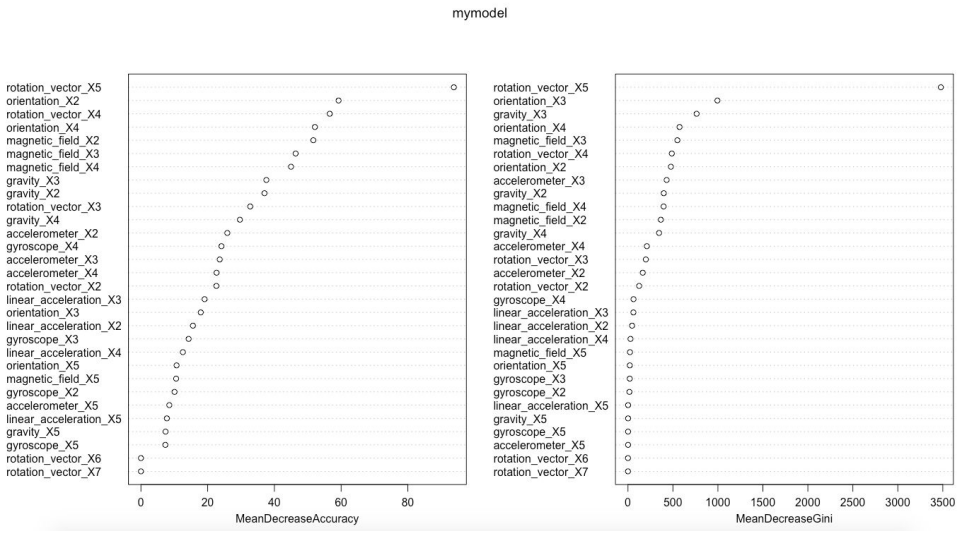



Figure 16: Overall Variable Importance

This motivated us to introduce a different model which would handle the NaN dataset and leverage all sensor's features.

# Ensemble Model

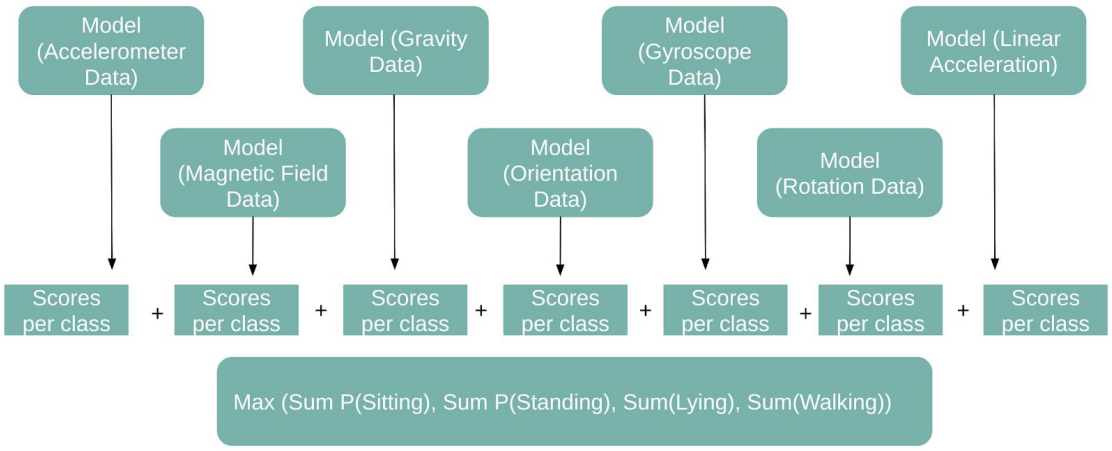



Fig 17: Ensemble Model

## Motivation

Our initial attempt to predict one of the four actions resulted in a good personal test accuracy but low impersonal test accuracy. Our model needed improvement because it did not deal with excess NaNs in the data set based on our construction. Due to our method of preprocessing the data, we had NaN values for sensors that were not active during a certain data collection period. To tackle the issue of NaN in the prediction, we decided to go with an ensemble of classifiers. **Each classifier will handle each sensor variables separately** so that the issue of NaN due to separate sampling frequency across sensors is resolved and we use all the sensor readings for the model assembly.
This model worked in the following way. Each “model” is trained on individual sensor data. During testing, a sample is run through each model, which outputs 4 class probabilities (aka probability of performing one of the 4 activities). Next, the scores for each corresponding outputted label are added together, and then max of these sums determines the final label of the sample.
This method was done to handle the issues of NANs/NA labels in our initial dataset. The steps of this model are summarized in Fig 17.

In Figure 17, “model” refers to one of the five models listed below:

1. Multinomial Logistic Regression/ Softmax Classifier using Neural net
2. Linear Discriminant Analysis - LDA
3. Quadratic Discriminant Analysis - QDA
4. Decision Trees
5. Random Forest

## Model Performance

| Model | Personal Test Accuracy |
|---|---|
| Multinomial Logistic Regression/ Softmax Classifier | 81.48% |
| Linear Discriminant Analysis - LDA | 81% |
| Quadratic Discriminant Analysis - QDA | 88% |
| Decision Trees | 78.48% |
| Random Forest | **98.85**% |

Table 2: Performance of Model 2

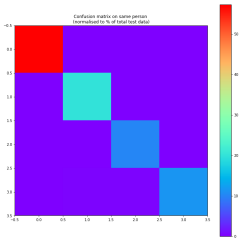
As it can been observed from the table, the Random Forest model gave the best results so we used it as a part of our Ensemble model. This makes sense because the data is nonlinear and Random forest performs very well on non linear data. The **impersonal test accuracy** improved from 35%, in the previous model, to about **48%** with this method.

## Accuracy Analysis

Below is the Confusion Matrix based on the Personal Test set:

| Predicted /Actual | Laying Down | Sitting | Standing | Walking |
|---|---|---|---|---|
| Laying Down | 5725 | 4 | 5 | 1 |
| Sitting | 4 | 1999 | 3 | 4 |
| Standing | 0 | 8 | 1019 | 2 |
| Walking | 9 | 33 | 42 | 1142 |

Table 3: Confusion matrix for Personal Test results of Model 2



Looking at the confusion matrix, we can observe that most of the predicted labels correspond to the true label. Though we see that the confusion matrix is showing good results, we see that there is still room for improvement in the model since the model seems to have some trouble distinguishing between walking and sitting and walking and standing. Lying down is dark red coloured because of

large amount of data for laying down as data is not equally distributed amongst classes.

Using our *Random Forest ensemble model* which takes the score per sensor we get the following results:
**Cross Validation Accuracy - 98.67%** (trained & tested on 90k with 5:1 split)
**Personal Test Accuracy - 98.85%** (tested on unseen personal 10k dataset)
I**mpersonal Test Accuracy - 47.89%** (tested on unseen impersonal 100k dataset from other user)
We did well on the train and personal test set but the accuracy we obtained was still low on impersonal test data. To address this issue, we tried feature engineering which will be discussed in the next section.

# Sensor Contribution Analysis

Before moving onto models with feature engineering, we analyzed the contribution of the combination of the available sensors to achieve the best results. The sensor can be divided into two categories: those which give acceleration information and those that give orientation information.
The sensors which belonged to each of these two categories have been listed below:

1. acceleration information = [accelerometer, gravity, linear_acceleration]
2. orientation information= [rotation vector, gyroscope, orientation, magnetic field]

If we can capture a good accuracy using only one sensor from each of these two categories, our model is complete. To try this, we tested various combinations and concluded that Gravity/ Accelerometer sensor is the best predictor and thus, with them we obtain the highest accuracy and without them, the lowest accuracy. They both are complementary as they belong to same category and capture redundant information.

We also found that **Rotation vector with accelerometer gives 97.86% accuracy** and **Rotation vector with gravity gives *98.6%* accuracy** which is as good as the overall model (implying we don't need other sensors at all) - 98.85%

The following table has the accuracies of single sensor based model in leftmost column 1, the accuracies obtained when we use all the sensors except one to capture the loss of neglecting one given sensor in middle column and the accuracies obtained by using another sensor with accelerometer in column 3. And, this leads to an interesting observation that rotation vector seems to capture a very significant portion of information which leads to the highest score among all of them in column 3, i.e. both acceleration and orientation information are required for activity tracking.

| Sensor | Only this Sensor | Without this Sensor | Accelerometer and this Sensor |
|---|---|---|---|
| Accelerometer | 88.05% | 98.81% | 88.05% |
| Gravity | **89.37**% | **98.08**% | 89.80% |
| Gyroscope | 68.21% | 98.75% | 89.47% |
| Linear Acceleration | 76.81% | 98.88% | 87.75% |
| Magnetic Field | 54.12% | 98.79% | 94.15% |
| Orientation | 87.83% | 98.67% | 97.73% |

| | | | |
|---|---|---|---|
| Rotation Vector | 87.92% | 98.52% | **97.86**% |

Table 4: Performance of Model 2 with selected sensor data

# Feature Engineering

To improve the accuracy and generalize the model across multiple persons on impersonal test dataset, we explored feature engineering for the dataset. We only considered accelerometer sensor for feature engineering to avoid NaNs in other sensor outputs and we found in last analysis, that accelerometer features were sufficient to approach accuracy upto 88.05%. For feature engineering, we created exponential moving average and exponential moving variance for all the axes of accelerometer data. The exponential moving average (EMA) and Exponential Moving Variance (EMVar) are given by:

$$\begin{aligned}\delta &= x_i - \mathrm{EMA}_{i-1} \\ \mathrm{EMA}_i &= \mathrm{EMA}_{i-1} + \alpha \cdot \delta \\ \mathrm{EMVar}_i &= (1-\alpha) \cdot \left(\mathrm{EMVar}_{i-1} + \alpha \cdot \delta^2\right)\end{aligned}$$

Then, we trained the best performing Random Forest Model on all the 10 features (X,Y,Z,EMA(X),EMA(Y),EMA(Z), EMVar(X),EMVar(Y),EMVar(Z), sensor_accuracy). For calculating these new features, we took the value of alpha as 0.5, which will give equal weight to current value and the last calculated metric.

## Model Performance

Using our feature engineered *Random Forest model* which adds exponential moving average metric per feature, we get the following results:
**Personal Test Accuracy - 92.9%** (tested on unseen personal 10k dataset)
I**mpersonal Test Accuracy - 59.46%** (tested on unseen impersonal 100k dataset from other user)

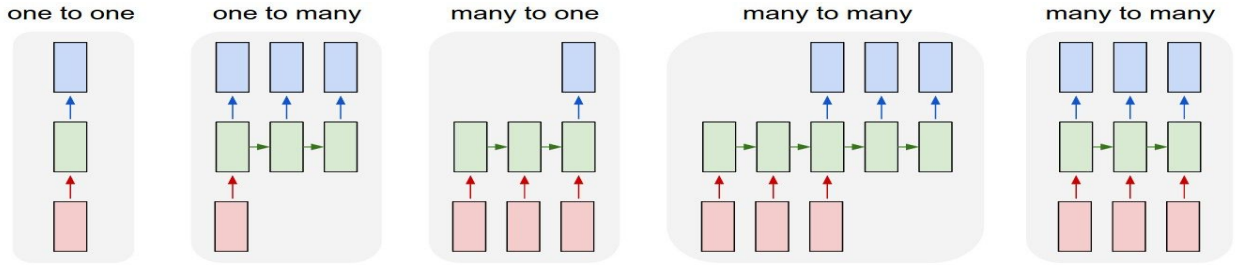
We did well on the train and personal test set but the accuracy we obtained was still below 60% for impersonal dataset, unable to achieve the current literature benchmarks. To address this issue, we explored deep learning models in the following section, which automatically do feature engineering.

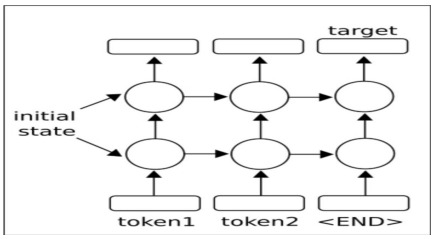
# Deep Learning Model

Compared to a classical approach, using a Recurrent Neural Networks (RNN) with Long Short-Term Memory cells (LSTMs) require no or almost no feature engineering. Data can be fed directly into the neural network who acts like a black box, modeling the problem correctly. Other research on the activity recognition dataset used mostly use a big amount of feature engineering, which is rather a signal processing approach combined with classical data science techniques. Prior research[9] uses Deep CNN on UCI ML Smartphone dataset to achieve an accuracy of upto 97% on personal test set. The approach here is rather very simple in terms of how much did the data was preprocessed.

## Motivation

As explained in this article[10], an RNN takes many input vectors to process them and output other vectors. It can be roughly pictured like in the image below, imagining each rectangle has a vectorial depth and other special hidden quirks in the image below. In our case, the "many to one" architecture is used: we accept time series of feature vectors (one vector per time step) to convert them to a probability vector at the output for classification. Note that a "one to one" architecture would be a standard feedforward neural network.

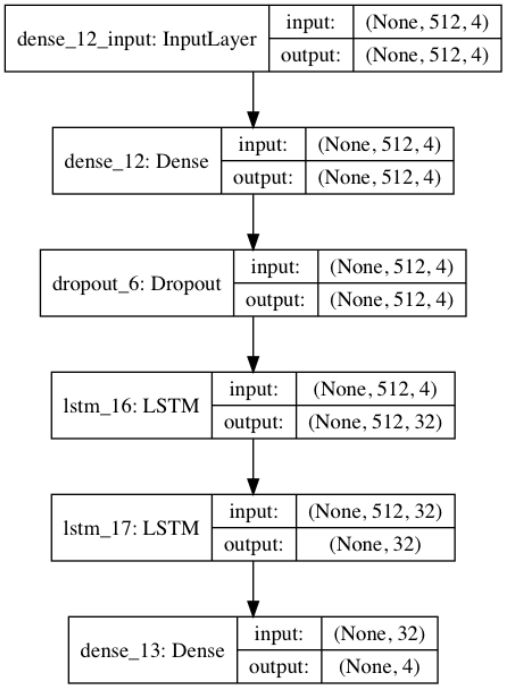



An LSTM is an improved RNN. It is more complex, but easier to train, avoiding what is called the vanishing gradient problem by using gradient highways. A many to one 2 layer stacked lstm architecture is like as on the right.

Vertically stacking layers in an LSTM (where the horizontal axis is time) could potentially help as the input to LSTM cells in later layers won't just be the raw input itself, it'll be some transformed version of the input that could potentially represent different/higher-level features.

## LSTM Architecture

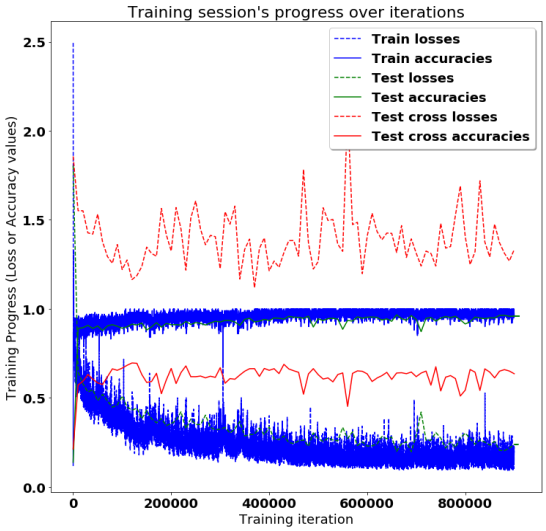



For the model, we only used accelerometer sensor features for fixed 512 time steps (around 2 seconds) with the same activity. The dataset is then, converted to the format of (Number of dataset points x timesteps (512) x Sensor Channels (4)). The output is then, one hot encoding for the four activities. The neural net architecture mainly, consists of double stacked LSTM layers over hidden channel of 32 dimensions. The architecture was programmed in Pytorch.

The model has 4 layer - affine layer with dropout, followed with 2 layers of LSTM layers and a last affine layer with sigmoid activation. The loss function was softmax cross entropy loss function with L2 regularization, proportioned to 0.0015. The model was trained with ADAM optimizer with 0.0025 learning rate with batch size of 100 over 100

epochs, looping 100 times over the whole training dataset. We used dropout value of 0.2 i.e. randomly drop 20% neurons to regularize the model and reduce overfitting.

## Model Performance

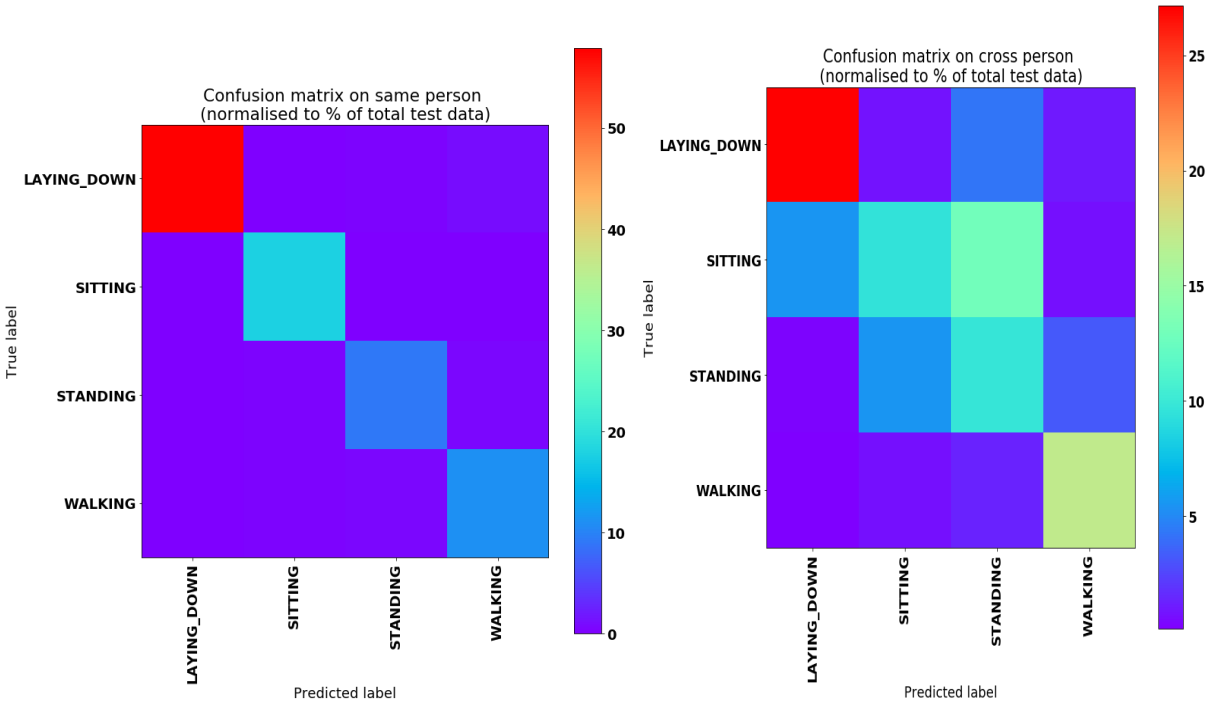



The side image plots the losses and accuracy during training across 100 epochs. The personal test and impersonal (cross) test metrics were calculated separately in each training epoch and plotted together for comparison. We can clearly see that the training loss continues to decrease to almost 0, while the training accuracy and personal test accuracy peak towards 90%, along with cross test accuracy stabilizing around 60%. The final accuracy results using the architecture was:

**Personal Test Accuracy - 95.89%** (tested on unseen personal 10k dataset)
**Impersonal Test Accuracy - 63.55%** (tested on unseen impersonal 100k dataset from other user)

## Accuracy Analysis

Below are the confusion matrices based on the personal test set on the left and impersonal test set on the right:



Looking at the left confusion matrix for personal test data, we can observe that most of the predicted labels correspond to the true label. However, the model makes a lot of mistakes on impersonal dataset and still, there is lot of room for improvement in the architecture for generalized model. But, the result of 63.55% just using accelerometer features without any feature engineering in deep learning model is impressive.

# Conclusion

| Model | Personal Test Accuracy | Impersonal Test Accuracy |
|---|---|---|
| Random Forest Ensemble (on all sensors) | 98.85% | 47.89% |
| Random Forest (on Accelerometer) | 88.05% | 56.42% |
| Random Forest (on Accelerometer with feature eng) | 92.9% | 59.46% |
| LSTM (on Accelerometer) | 95.89% | 63.55% |

Table 5: Performance of best models on impersonal test

In this project, we have created a series of machine learning models to predict the type of activity a person is doing, based on sensor data from a smartwatch. We created models that used all sensor data, partial sensor data, and feature engineered sensor data, which train well to specific users. Ensembles perform really good for personal data without leveraging sequential temporal dependency. We also explored *LSTM models* to leverage the temporal dependence in the data set. We were able to achieve the best results using the LSTM model, which gave around *63.55% accuracy on impersonal test data set* only using accelerometer data by capturing the latent features from the dataset in the neural net architecture. In the end, we agree with Weiss et al. [7] that "personal models vastly outperform impersonal models for activity recognition, but at the cost of requiring each user to provide labeled training data".

# References


1. E.M. Tapia, S.S. Intille, and K. Larson. "Activity Recognition in the Home Using Simple and Ubiquitous Sensors." Massachusetts Institute of Technology, 2004.

2. S. Liu, R. Gao, and P. Freedson. "Computational Methods for Estimating Energy Expenditure in Human Physical Activities." *Medicine and Science in Sports and Exercise* 44.11 (2012): 2138-2146.

3. T. White, K. Westgate, N.J. Wareham, and S. Brage. "Estimation of Physical Activity Energy Expenditure during Free-Living from Wrist Accelerometry in UK Adults." MRC Epidemiology Unit, University of Cambridge, 2016.

4. M.E. Rosenberger, W.L. Haskell, F. Albinali, S. Mota, J. Nawyn, and S.S. Intille. “Estimating Activity and Sedentary Behavior From an Accelerometer on the Hip or Wrist.” *Medicine and Science in Sports and Exercise* 45.5 (2013): 964-975.

5. A. Mannini, S.S. Intille, M. Rosenberger, A.M. Sabatini, and W. Haskell. “Activity recognition using a single accelerometer placed at the wrist or ankle.” *Medicine and Science in Sports and Exercise* 45.11 (2013): 2193-2203.

6. B. Mortazavi, E. Nemati, K. VanderWall, H.G. Flores-Rodriguez, J.Y.J. Cai, J. Lucier, A. Naeim, and M. Sarrafzadeh. “Can Smartwatches Replace Smartphones for Posture Tracking?” *Open Access Sensors* 15 (2015): 26783-26800.

7. G.M. Weiss, J.L. Timko, C.M. Gallagher, K. Yoneda, A.J. Schreiber. “Smartwatch-based Activity Recognition: A Machine Learning Approach.” *Proceedings of the 2016 IEEE International Conference on Biomedical and Health Informatics (BHI 2016)*, Las Vegas, NV, 426-429.

8. D. Anguita, A. Ghio, L. Oneto, X. Parra and J.L. Reyes-Ortiz. “A Public Domain Dataset for Human Activity Recognition Using Smartphones.” 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

9. W. Jiang and Z. Yin. “Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks.” Proceedings of the 23rd ACM international conference on Multimedia 2015. Brisbane, Australia 1307-1310 October 2015.

10. Andrej Karpathy. “The Unreasonable Effectiveness of Recurrent Neural Networks.” http://karpathy.github.io/2015/05/21/rnn-effectiveness/. May 2015.