# SpringBoard Capstone Project 2: Human Activity Recognition
# Milestone Report 2

**Written by:** Sofy Weisenberg
**Date:** 04/19/20

This Capstone Project milestone report will cover the following project sub-topics:


- Statistical Prep: Aggregating Data
- Selecting Features of Interest
- Visualizing Distributions of Aggregated Feature Statistics
- Bootstrap Hypothesis Testing
    - Motivation
    - Performing Hypothesis Tests
- Machine Learning Preprocessing
    - Splitting the Data: Training and Test Sets
    - Feature Scaling
    - Categorical Class Encoding
- Fitting an SVM Model
    - Hyperparameter Tuning
    - Model Evaluation
    - Visualizing Misclassified Activities


## Statistical Prep: Aggregating Data

For statistical analysis purposes and in order to visualize some of the distributions of the different classes and subjects, each activity for a given subject will be summarized (time, mean, median, std, max, min) across the full activity time window for all features. Each of these summary statistics will be calculated in a new dataframe "separated_df_X".

Additionally, it was observed in the Exporatory Data Analysis, that the active and passive classes were more distinguishable as "super-clusters" containing the individual activity class clusters. This means that additional statistical analysis can be done by assigning a new super-class label to each observation. This label will be in a new column "superClass" and will either be 0 for passive or 1 for active.

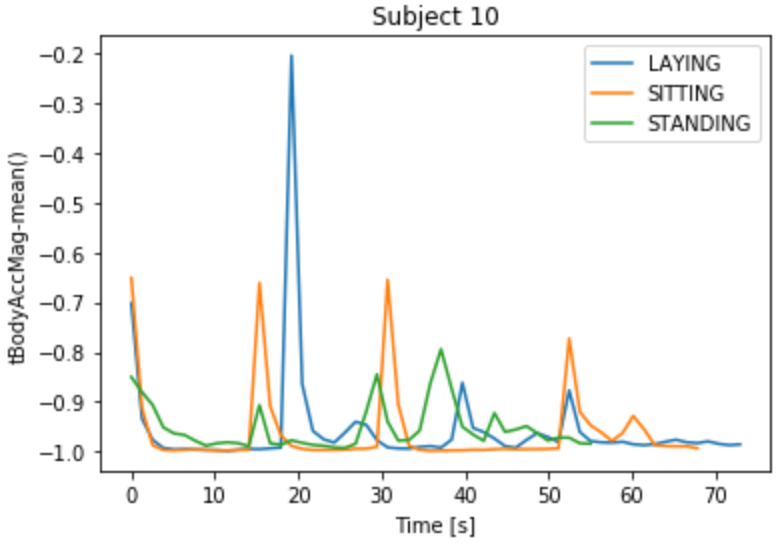
Again, each of the summary statistics will be calculated in a new dataframe "superClass_df_X" -- this time separated by superClass.

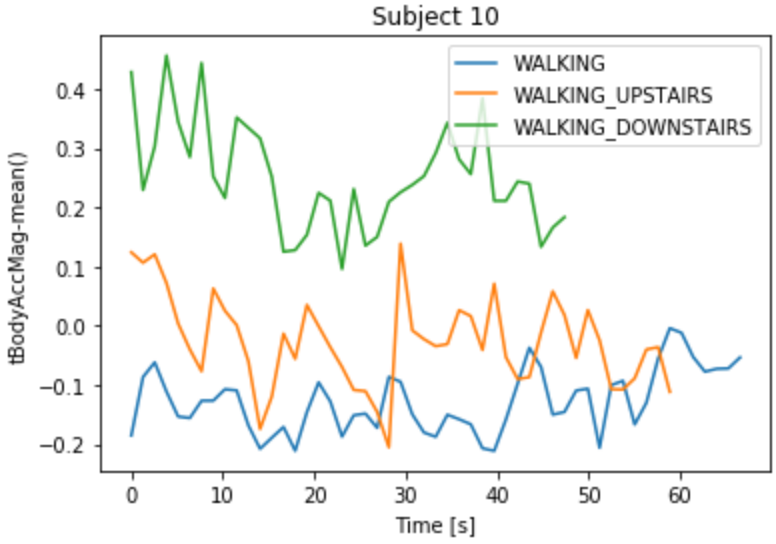
## Selecting Features of Interest

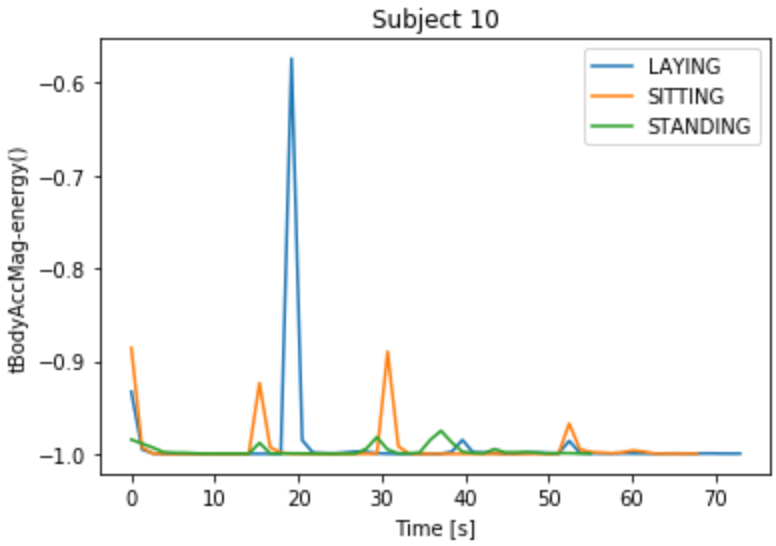
To get an idea of the usefulness of this newly aggregated data, it is helpful first to select and plot single features of interest across Activities and group by superClass. A plotting function is created and called for a random individual subject. This function call may be executed multiple times to get a visual concept of variability amongst subjects.

Next, two features have been chosen as those that are both human interpretable and likely to show larger differences among the classes, based on a rough understanding of the processed input signals:

- tBodyAccMag-mean() is the total body acceleration magnitude (Euclidean norm of the three-dimensional signal), averaged over each 2.56 s time window
- tBodyAccMag-energy() the total body acceleration magnitude energy measure (sum of the squares divided by the number of values)







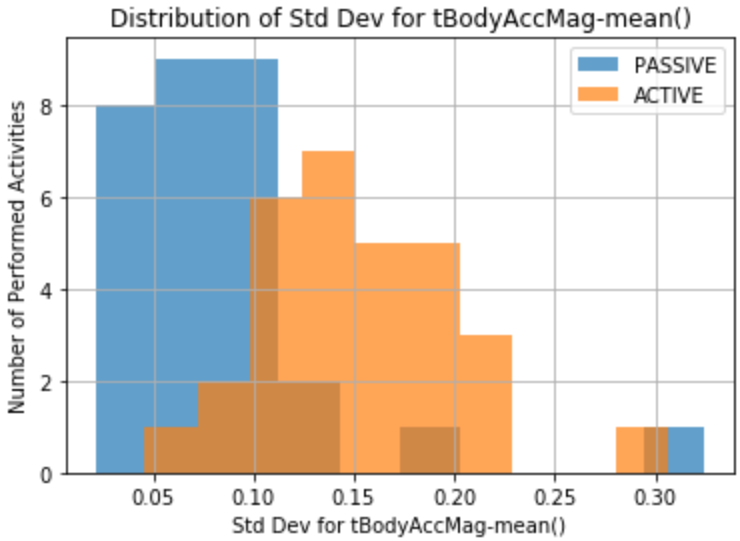
From the plots above, it can be seen that the various activities appear to be somewhat separable in the range of their values.

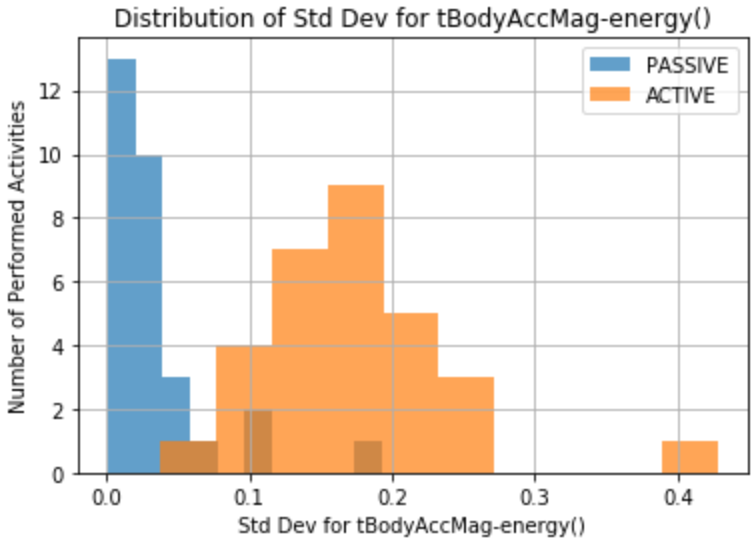
As an additional exercise, summarizing the data contained in the bandsEnergy() features (energy of a frequency interval within the 64 bins of the FFT of each window) could likely give some Activity class-specific differentiators. This would require a different aggregation strategy as the Fourier transformed variables contain 42 bandsEnergy() features, each in its own column. The assumption is that one or more of these frequency bands would consistently stand out as a frequency "signature" for a given Activity class. This has not been attempted here.

# Visualizing Distributions of Aggregated Feature Statistics

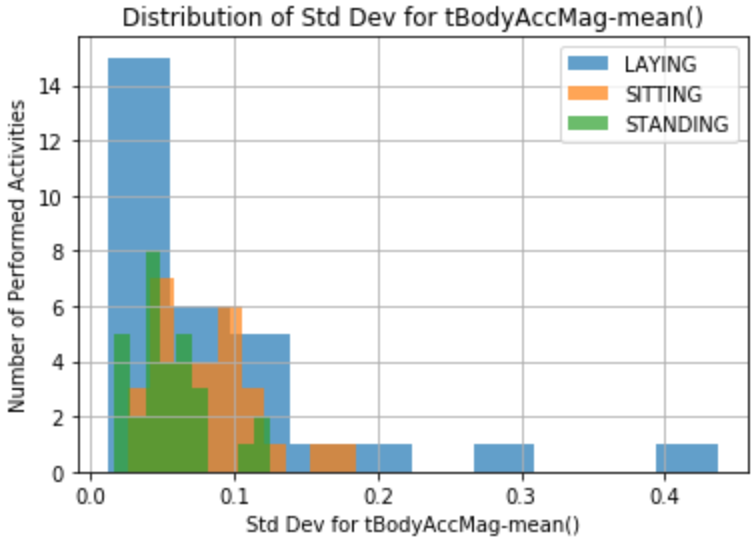
Next, in statistically trying to distinguish the classes, it may be useful to visually compare the distributions of the above plotted features across each of the various activities. For example, it is likely that the std dev of tBodyAccMag signals from the active classes will be generally higher than those of the passive classes.

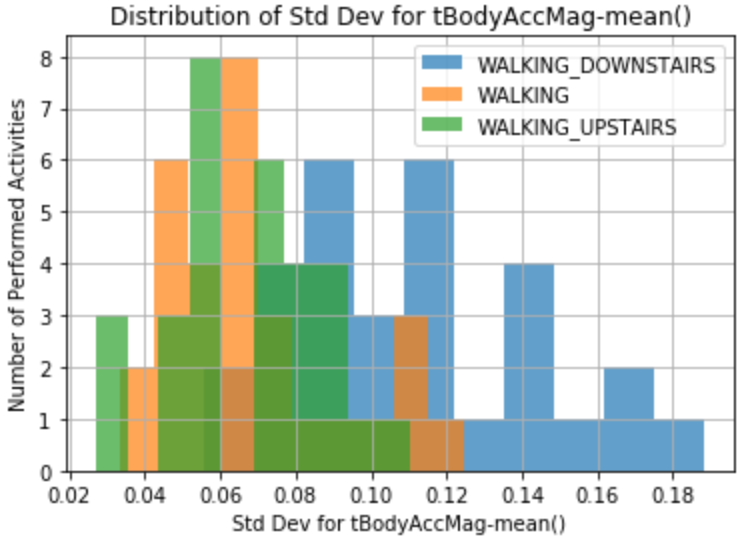
Histograms of the standard deviation for the 2 variables of interest may be plotted to compare by the superClass distinction.

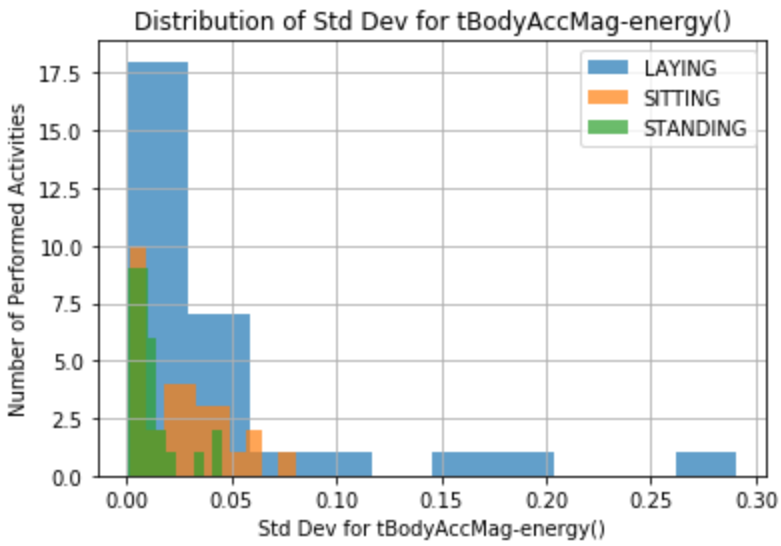


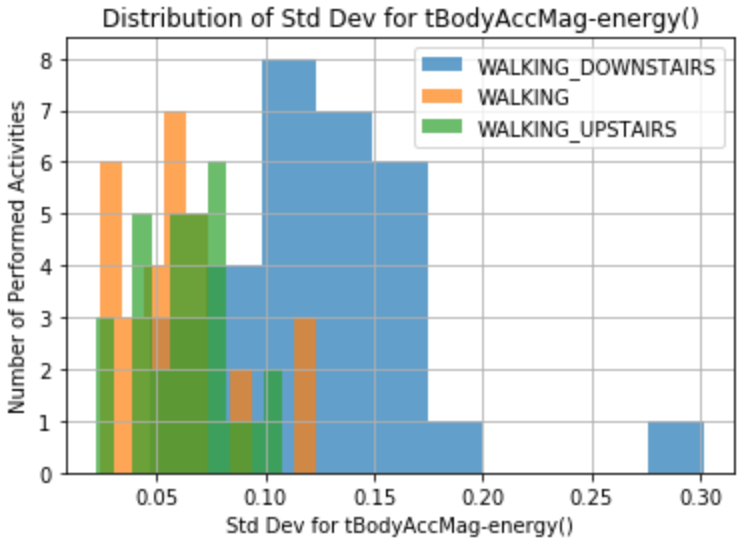




Plotting similar histograms for these feature summary statistics (focusing on std deviation here for conciseness), this time with each Activity separated out individually. The differences of these distributions are more nuanced than those of the aggregated superClasses, as is to be expected.


Distribution of Std Dev for tBodyAccMag-mean()
LAYING
SITTING
STANDING
Number of Performed Activities
Std Dev for tBodyAccMag-mean()


Distribution of Std Dev for tBodyAccMag-mean()
WALKING_DOWNSTAIRS
WALKING
WALKING_UPSTAIRS
Number of Performed Activities
Std Dev for tBodyAccMag-mean()


Distribution of Std Dev for tBodyAccMag-energy()
LAYING
SITTING
STANDING
Number of Performed Activities
Std Dev for tBodyAccMag-energy()
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.00
0.05
0.10
0.15
0.20
0.25
0.30


Distribution of Std Dev for tBodyAccMag-energy()
WALKING_DOWNSTAIRS
WALKING
WALKING_UPSTAIRS
Number of Performed Activities
Std Dev for tBodyAccMag-energy()
0
1
2
3
4
5
6
7
8
0.05
0.10
0.15
0.20
0.25
0.30

# Bootstrap Hypothesis Testing

## Motivation

One hypothesis worth testing is that Activity or superClass distinctions can be made based on the distribution of one or more of their single feature's summary statistics, across a subject sample of 30 participants.

For example, one hypothesis: passive and active classes have statistically significantly different mean std. dev for the tBodyAccMag-mean() and tBodyAccMag-energy() features. The histogram points to the possibility that this hypothesis is true, but a hypothesis test would be required to prove it with statistical significance. It is also possible that statistically significant differences can be proven between superClasses, but might fail to proven between certain similar activities within a superClass (WALKING and WALKING_UPSTAIRS, for example).

Since the distributions of std. dev. for either superClass do not meet the normality assumption of traditional frequentist inference and are relatively small (N=90, 30 subjects performing each of 3 activities), this a good candidate for bootstrapping. A bootstrap resampling of these distributions would allow for statistical inference testing of the difference of means.

If this hypothesis test fails, it would strengthen the business case for creating a machine learning classification model. If, however statistically siginificant differences CAN be found amongst single features, a more lightweight approach to classification based on statistical inference may make more sense, particularly if the productionized classifier needs to run in real time with limited processing power. This is often the case for wearable devices.

## Performing Hypothesis Tests

Using the above defined function, a bootstrap hypothesis test may be performed. To start, the following hypothesis will be tested: passive and active classes have statistically significantly different mean std. dev for the tBodyAccMag-mean() and tBodyAccMag-energy() features. This statement actually contains 2 hypothesis tests, one for each feature. The hypothesis tests are defined as follows, with null and alternative hypotheses:

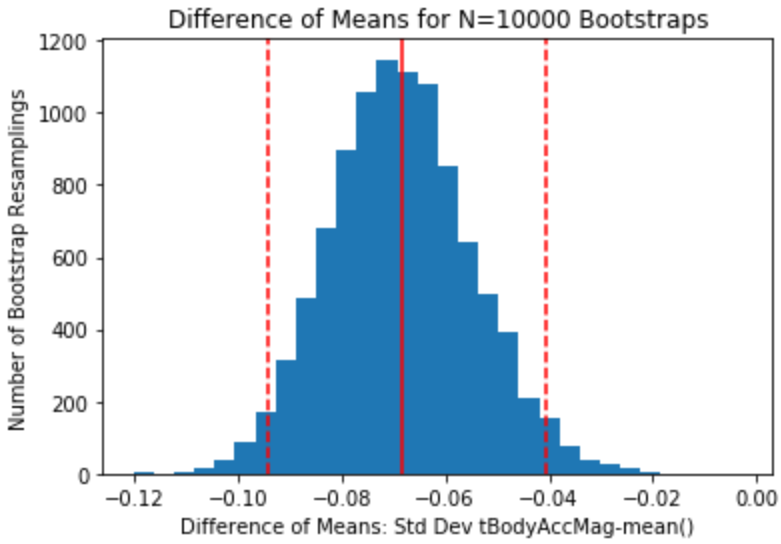
**Hypothesis 1**
Ho: The difference of mean standard deviation for the tBodyAccMag-mean() between passive and active classes is equal to zero.
Ha: The difference of mean standard deviation for the tBodyAccMag-mean() between passive and active classes is not equal to zero.

**Hypothesis 2**
Ho: The difference of mean standard deviation for the tBodyAccMag-energy() between passive and active classes is equal to zero.
Ha: The difference of mean standard deviation for the tBodyAccMag-energy() between passive and active classes is not equal to zero.

Significance level for both tests is 95%. The number of bootstrap samples is N=10000.

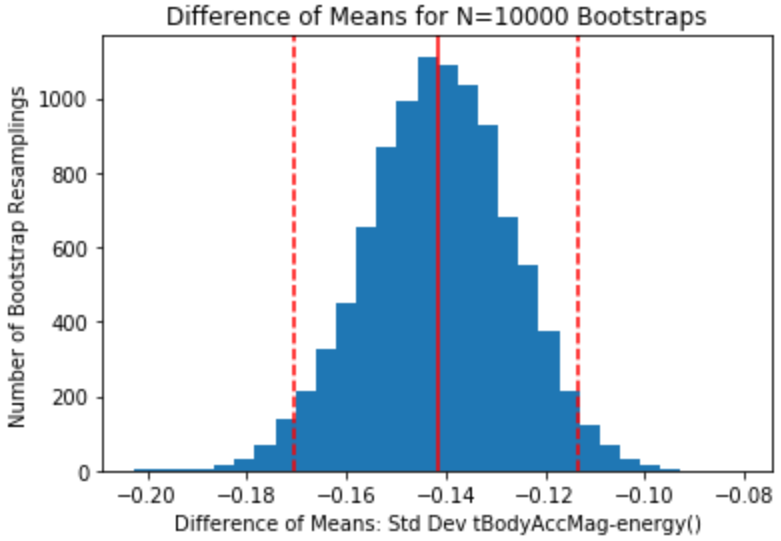
**Hypothesis 1**

The bootstrap 95% CI is [-0.09417189, -0.04060474].




**Hypothesis 2**

The bootstrap 95% CI is [-0.17066601, -0.11332069]




For both of the above hypothesis tests, the N=10000 bootstrap 95% confidence interval does not contain zero. This means that the null hypothesis can be rejected. Therefore, the hypothesis that passive and active classes have statistically significantly different mean std. dev for the tBodyAccMag-mean() and tBodyAccMag-energy() features can be accepted with a statistical significance level of 95%.

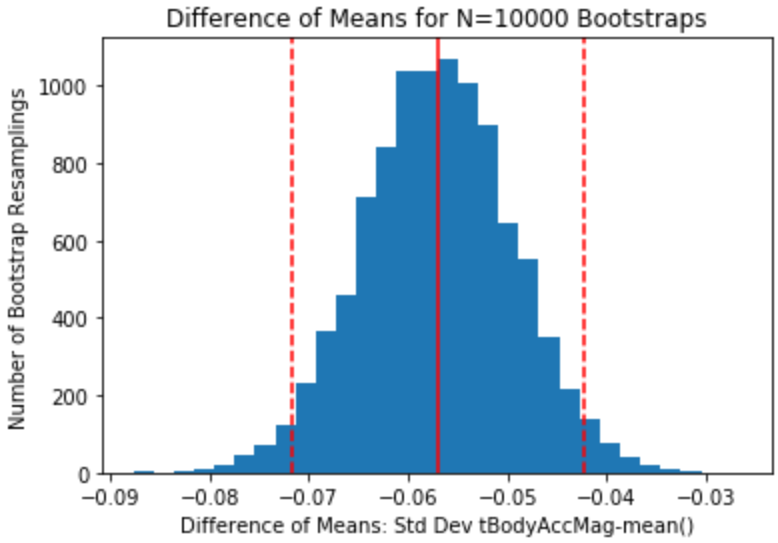
In other words, the distributions of these feature summary statistics are different enough to correctly classify a random Activity as passive or active based only on the std. deviation of one of these features with 95% confidence.

This result would most likely be the case also for examining two activities that are the most "dissimilar". For example, STANDING and WALKING_DOWNSTAIRS. However, similar activities within the same superClass (WALKING and WALKING_UPSTAIRS, for example) may not have statistically different enough distributions.

**Hypothesis 3**
Ho: The difference of mean standard deviation for the tBodyAccMag-mean() between STANDING and WALKING_DOWNSTAIRS is equal to zero.
Ha: The difference of mean standard deviation for the tBodyAccMag-mean() between STANDING and WALKING_DOWNSTAIRS is not equal to zero.

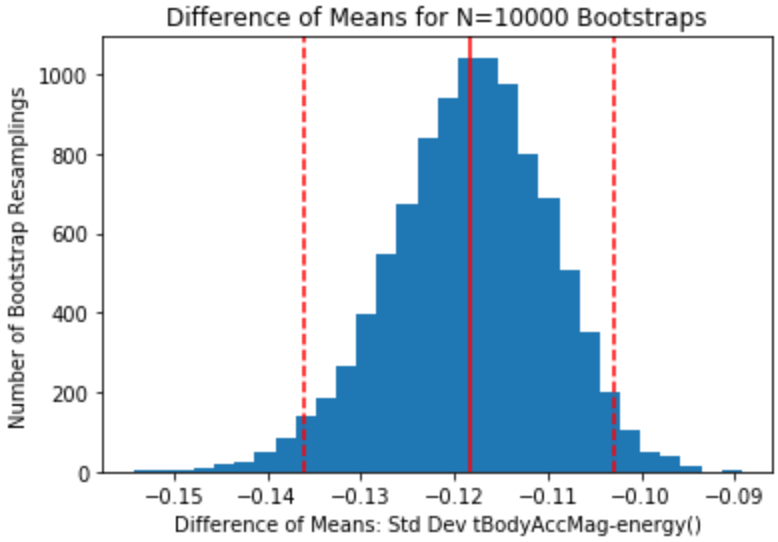
**Hypothesis 4**
Ho: The difference of mean standard deviation for the tBodyAccMag-energy() between STANDING and WALKING_DOWNSTAIRS is equal to zero.
Ha: The difference of mean standard deviation for the tBodyAccMag-energy() between STANDING and WALKING_DOWNSTAIRS is not equal to zero.
Significance level for both tests is 95%. The number of bootstrap samples is N=10000.

**Hypothesis 3**
The bootstrap 95% CI is [-0.071769,  -0.04226481]



**Hypothesis 4**

The bootstrap 95% CI is [-0.13603891, -0.10288204]




**Hypothesis 5**

Ho: The difference of mean standard deviation for the tBodyAccMag-mean() between STANDING and SITTING is equal to zero.

Ha: The difference of mean standard deviation for the tBodyAccMag-mean() between STANDING and SITTING is not equal to zero.

**Hypothesis 6**

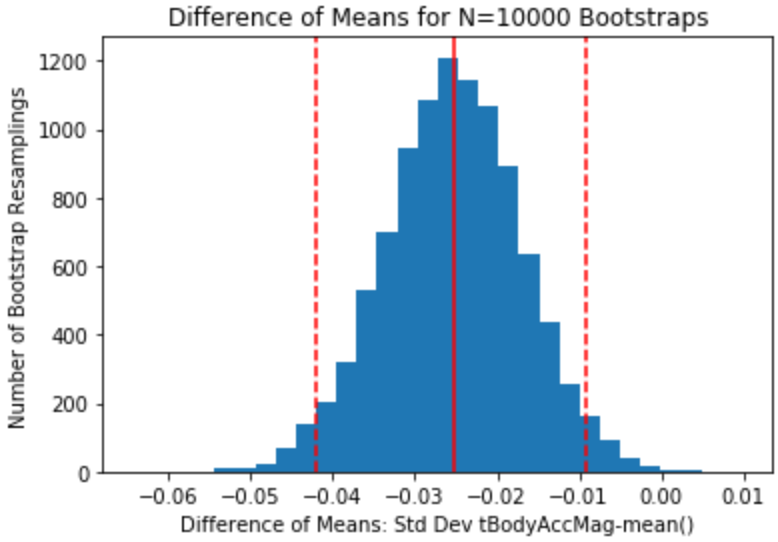
Ho: The difference of mean standard deviation for the tBodyAccMag-energy() between STANDING and SITTING is equal to zero.

Ha: The difference of mean standard deviation for the tBodyAccMag-energy() between STANDING and SITTING is not equal to zero.

Significance level for both tests is 95%. The number of bootstrap samples is N=10000.

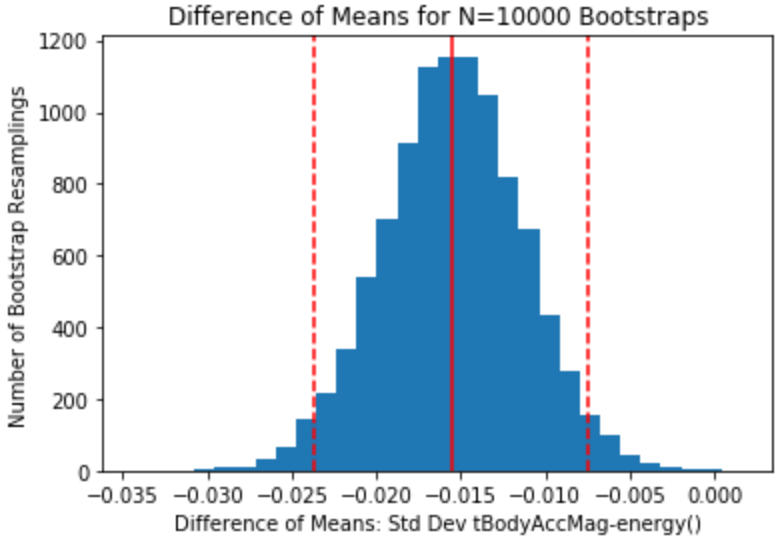
**Hypothesis 5**

The bootstrap 95% CI is [-0.04202838, -0.00906252]




**Hypothesis 6**

The bootstrap 95% CI is [-0.02366883 -0.00743425]




Both null hypotheses can be rejected at a 95% statistical significance level.

This is a surprising result: even for highly overlapping std. dev. distributions like for SITTING and STANDING, there is enough difference to say with statistical significance that a random std. dev value for the tested variables belongs to one distribution over another. This confirms the high class separability that was seen in the PCA and t-SNE visualizations. It makes this dataset a strong candidate for both statistical inference approaches as well as machine learning classifiers such as SVM. These will be explored in the next section.

# Machine Learning Preprocessing

## Splitting the Data: Training and Test Sets

In order to allow for evaluation of the machine learning (ML) model, it is critical to split the full data set into a training set (used to train the ML model) and a test set (used to evaluate the model's performance).

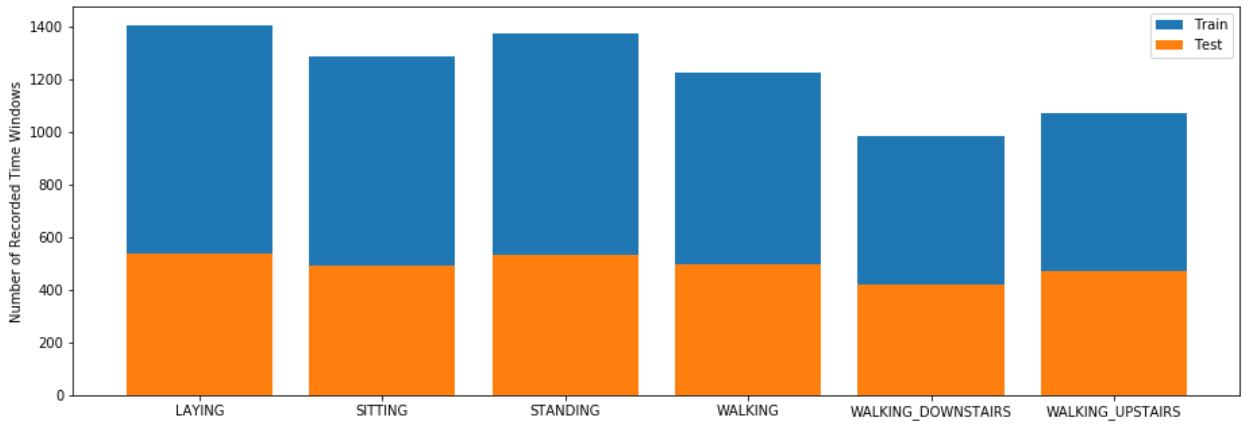
In this case, a stratified split is important (i.e. making sure that the classes are represented in both the training and test sets relatively equally). Also, it must be taken into account that entire activities and subject sets should be kept together. Movement patterns of the same subject should be segregated so as not to bias the test set.

Therefore of the 30 participating subjects' data, 30% (9 subjects) will be segregated into the test set and saved for the evaluation phase. These subjects have been selected randomly (Kaggle has already provided the data in train/test split for this dataset.)

The resulting feature (X) and label (y)  arrays are sized as follows:

| Array | Num. Rows | Num. Columns |
|---|---|---|
| X_train | 7352 | 561 |
| y_train | 7352 | 1 |
| X_test | 2947 | 561 |
| y_test | 2947 | 1 |

And the class distribution among the resulting test/train split is relatively even:



## Feature Scaling

To improve performance of the machine learning model(s) to be applied to this classification problem, first the feature data should be appropriately scaled.

Scaling must be performed both on the training and test data sets in the same manner. Best practice is to use only the training set to identify the correct scaling, and then blindly apply the same transformation to the test set. In this case, the feature matrix will be standardized (i.e. the mean of each feature set to zero and the std dev set to 1) using the sklearn method StandardScaler(). This results in feature sets X_train_scaled, X_test_scaled .

- For each column in the X_train_scaled feature matrix, the mean is set to zero and the std dev is set to 1
- For each column in the X_test_scaled feature matrix, the mean is ***NOT*** set to zero and the std dev is ***NOT*** set to 1, since a transformation matrix (generated from normalizing the training feature matrix range) was applied to the test feature matrix.

## Categorical Class Encoding

Since the class labels (Activity column) for this data are in a categorical string format, they must first be encoded to a numerical format useable for supervised machine learning. Simple integer encoding (i.e. translating each class label into an integer label 1-6) would not be effective on its own if the categorical variable was in the feature set, since such an encoding implies an ordinal relationship between the categories, where in reality there is none. This may result in poor model performance or unexpected results.

Integer encoding followed by one-hot encoding is the preferred method to transform categorical data within the feature matrix. Each integer class label is assigned to a new binary (0/1) column of "dummy variables". Each observation in the dataset is then be labeled with a 1 in only one of these "dummy variable" columns and the rest are labeled with a 0, resulting in sparse matrices of categorical variables for y_train and y_test.

However, in this case where the categorical variable is the dependent varaible (classifier label y), the second step of one-hot encoding is not necessary. In fact the SVC() method in sklearn will not accept a one-hot encoded multi-class array as the label vector.

# Fitting an SVM Model

In previous published papers (most notably: https://upcommons.upc.edu/bitstream/handle/2117/101769/IWAAL2012.pdf), a support vector machine (SVM) was used successfully to harness the engineered features to implement a lightweight multi-class machine learning classifier.

As a baseline for comparison of deep learning techniques that will be implemented later in this project, the SVM model is fit to the training data and evaluated. Deep learning models can effectively work on the raw time series data, but require heavier computational resources, while traditional machine learning models perform better when feature engineering is performed beforehand with sufficient domain knowledge applied. This will be done using the support vector classifier SVC() method in sklearn.

## Hyperparameter Tuning

In order to optimize performance of the SVC() model, a grid search across several model hyperparameters is performed. The hyperparameters chosen for this search have the following value ranges, based on commonly selected values in the literature:

- Regularization parameter **C**: [1, 10, 1000, 10000]
- Kernel coefficient **gamma**: [1e-6, 1e-4, 1e-2]
- A linear kernel vs. (non-linear) RBF kernel, since it is not known a-priori if the classes are linearly separable

The GridSearchCV() method is used to scan across hyperparameter combinations using a 5-fold cross validation to evaluate a mean model performance for each combination. The best performing set of hyperparameters is chosen for the final model. Since the classes are more or less evenly distributed in the train/test sets, the accuracy metric is suitable for performance evaluation. The results of the grid search are as follows:

Best Score (accuracy): 0.9383
Best Params:  {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}

## Model Evaluation

The SVC() model is fit using the optimized parameter set and the .predict() method is used on the fitted model. The confusion matrix and classification report will be used to compare model to future models.

First, evaluating the method on the training features results in the following:

| | LAY | SIT | STA | WALK | WALK_D | WALK_U |
|---|---|---|---|---|---|---|
| **LAY** | 1407 | 0 | 0 | 0 | 0 | 0 |
| **SIT** | 0 | 1276 | 10 | 0 | 0 | 0 |
| **STA** | 0 | 13 | 1361 | 0 | 0 | 0 |
| **WALK** | 0 | 0 | 0 | 1226 | 0 | 0 |
| **WALK_D** | 0 | 0 | 0 | 0 | 986 | 0 |
| **WALK_U** | 0 | 0 | 0 | 0 | 0 | 1073 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **LAY** | 1 | 1 | 1 | 1407 |
| **SIT** | 0.99 | 0.99 | 0.99 | 1286 |
| **STA** | 0.99 | 0.99 | 0.99 | 1374 |
| **WALK** | 1 | 1 | 1 | 1226 |
| **WALK_D** | 1 | 1 | 1 | 986 |
| **WALK_U** | 1 | 1 | 1 | 1073 |
| | | | | |
| **accuracy** | | | 1 | 7352 |
| **macro avg** | 1 | 1 | 1 | 7352 |
| **weighted avg** | 1 | 1 | 1 | 7352 |

The above evaluation shows a nearly perfect classification, as expected on the data set used to train the model. There is some misclassification between the SITTING and STANDING classes.

Next, the model is evaluated on the test set which it has never encountered.

| | LAY | SIT | STA | WALK | WALK_D | WALK_U |
|---|---|---|---|---|---|---|
| **LAY** | 537 | 0 | 0 | 0 | 0 | 0 |
| **SIT** | 0 | 436 | 54 | 0 | 0 | 1 |
| **STA** | 0 | 15 | 517 | 0 | 0 | 0 |
| **WALK** | 0 | 0 | 0 | 493 | 3 | 0 |
| **WALK_D** | 0 | 0 | 0 | 5 | 398 | 17 |
| **WALK_U** | 0 | 0 | 0 | 16 | 2 | 453 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **LAY** | 1 | 1 | 1 | 537 |
| **SIT** | 0.97 | 0.89 | 0.93 | 491 |
| **STA** | 0.91 | 0.97 | 0.94 | 532 |
| **WALK** | 0.96 | 0.99 | 0.98 | 496 |
| **WALK_D** | 0.99 | 0.95 | 0.97 | 420 |
| **WALK_U** | 0.96 | 0.96 | 0.96 | 471 |
| | | | | |
| **accuracy** | | | 0.96 | 2947 |
| **macro avg** | 0.96 | 0.96 | 0.96 | 2947 |
| **weighted avg** | 0.96 | 0.96 | 0.96 | 2947 |

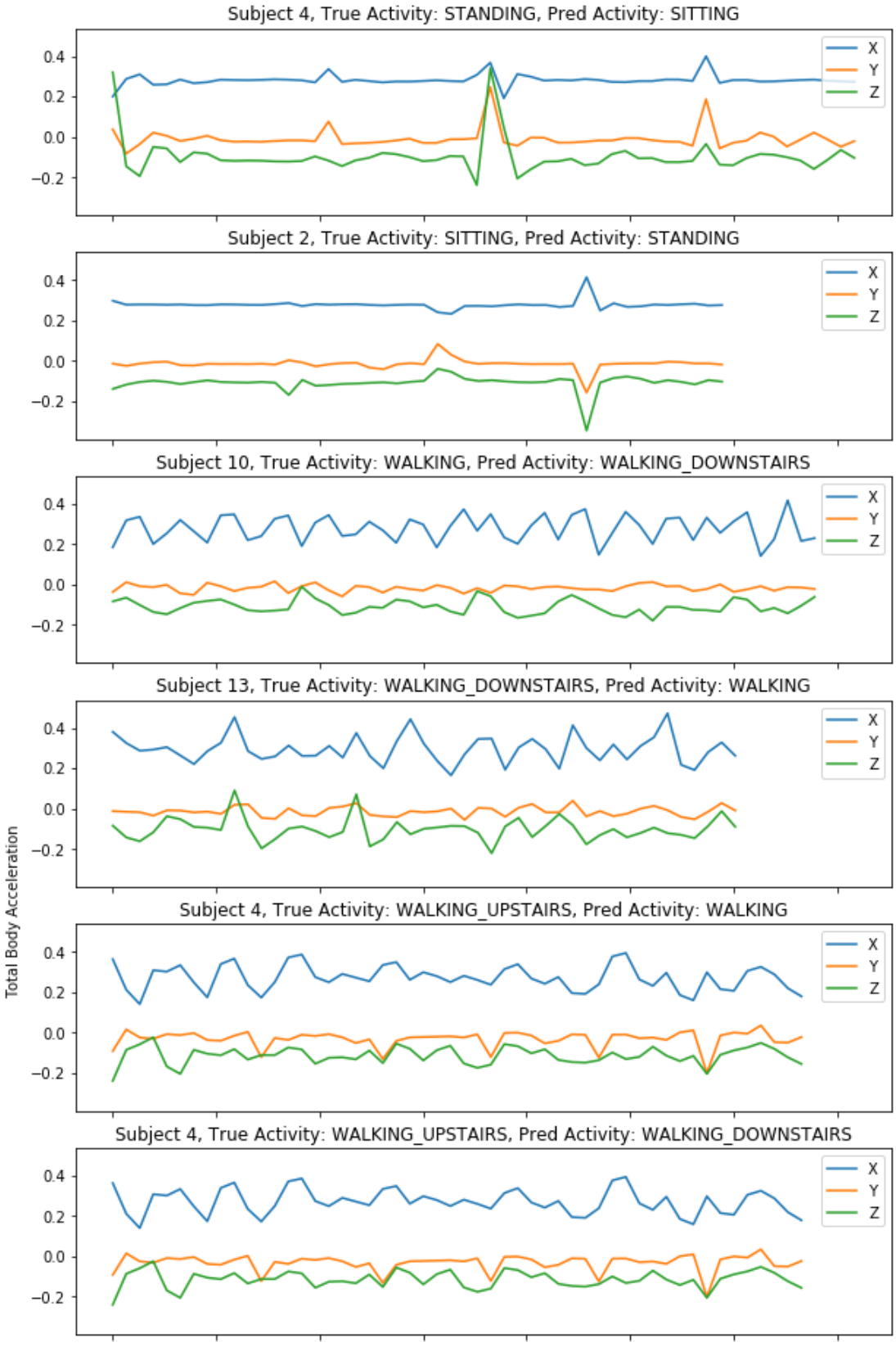
The evaluation on the test set shows some additional misclassifications, but overall appears to be a strong classifier with many classes being perfectly distinguished from one another (no misclassifications). Overall model accuracy is 0.96.

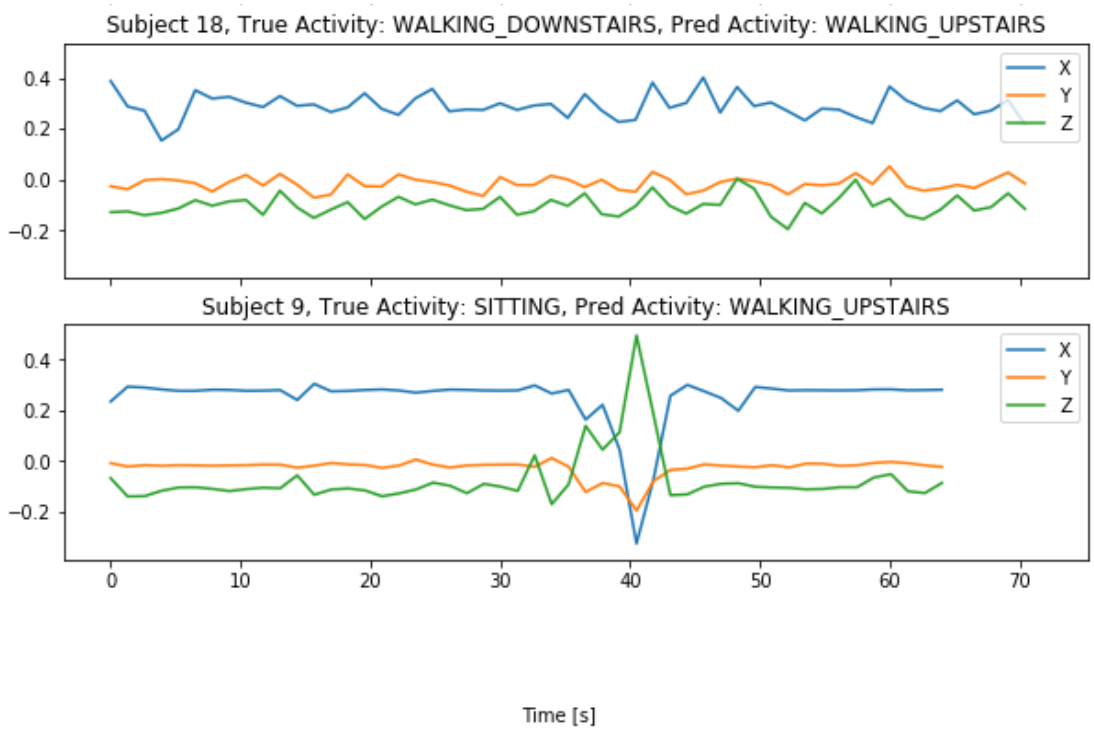
## Visualizing Misclassified Activities

To better understand the limitations of the evaluated SVM model, it is helpful to visualize some of the signal data from activity instances that were misclassified by the model. These may be compared to activities that were correctly predicted. Since each window is predicted separately, however, it may not be straightforward to see these patterns of misclassification.

Here are examples from the training set misclassifications:

And a few examples from the test set misclassifications:




Subject 18, True Activity: WALKING_DOWNSTAIRS, Pred Activity: WALKING_UPSTAIRS
X
Y
Z
0.4
0.2
0.0
−0.2
Subject 9, True Activity: SITTING, Pred Activity: WALKING_UPSTAIRS
X
Y
Z
0.4
0.2
0.0
−0.2
0
10
20
30
40
50
60
70
Time [s]