**Accelerometer Signals Classification for Activities**

## Introduction

Purpose of the research is to classify signals comes from mobile devices to determine next classes of activities:

* + still;
  + cycling;
  + walking;
  + running;
  + going up/down stairs.

The research can be used in applied areas such as: healthcare, tracking systems, gaming etc. Signals from accelerometer can be evaluated in different measurement aspects:

* + time under an activity;
  + activity intensity;
  + count of steps/cycles.

## 2. Goal

Understand the “AR.mat” dataset in the context of final problem and based on data mining technique and machine learning algorithms prepare model to classify new accelerometer signals.

## 3. Dataset

This is a dataset of accelerometer sample acquired for Activity Recognition (AR) algorithm. It has collected raw measurements (one for each Cartesian axis: x, y, z). Since our algorithm required the framing of the signals, the frame duration has been set equal to 4 s. For the AR case a set of about 14 hours has been employed. Acquisitions were performed by 8 users who kept the smartphones in four different positions and orientations:

* facing towards the user;
* towards the opposite side;
* pointing up;
* pointing down.

The whole database, already exported in Matlab environment, is downloadable and available for possible further experiments and comparisons.

## 4. Research stages

The research includes the following stages:

1. Data extraction (load dataset from file, determine the type of variables);
2. Data exploration (exploring data as a part of future preparation process, find general trends, outliers, correlation, describe and visualize data);
3. Data preparation (clean and transform data in terms of feature selection, scaling, data manipulation).
4. Modeling (build model, validate according to selected technique).
5. Conclusions

## 5. Outcomes

Summarized results of each suggested research steps are introduced in this section.

### Data extraction

The dataset was extracted from AR.mat file. Raw data presented as a dictionary with variety [2; 287] length chunks of variety size [n, 3] matrixes, where n lies in range [39; 88]. Each matrix row represents a mobile device position in space by 3 components: x, y, z.

Here is a fragment of the dictionary names and count of matrix under each of them:

fc\_cycling1: 82

fc\_cycling10: 105

fc\_cycling11: 85

fc\_cycling12: 93

fc\_cycling13: 173

fc\_cycling14: 163

…

fc\_walking237: 55

fc\_walking238: 191

fc\_walking239: 143

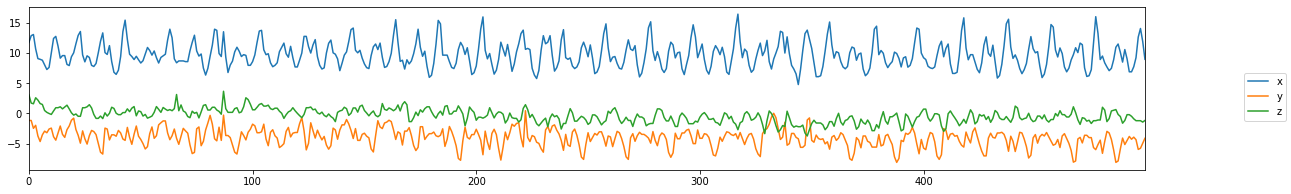
fc\_walking240: 71

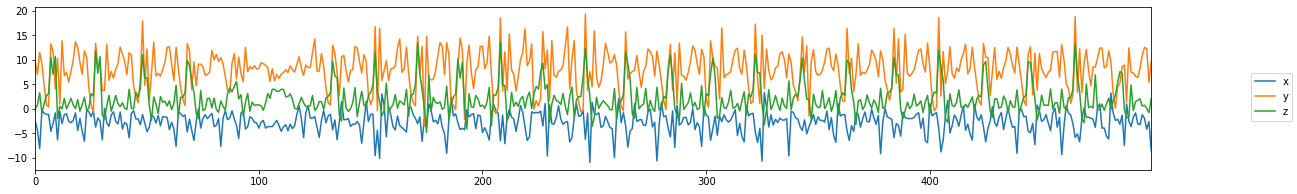
fc\_walking241: 101

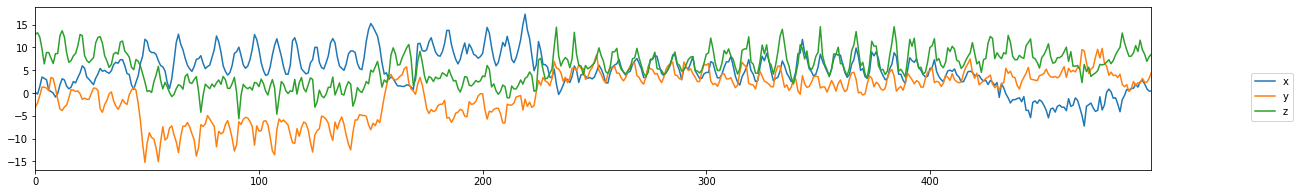
fc\_walking242: 81

As you can see, classes of activities have a different count of matrixes so it means they have different measurements in an aspect of tester, phone position and time. So, we extracted class of activity and experiment number separately. We used the experiment number to split the data in a way where some experiment will be hidden for model training.

We tried to concatenate all the matrices within an experiment to define is the signal continuous in it. Here how signal looks like.







*Figure 1: Signal within an experiment (first 500 samples)*

An experiment signal view means that all the data presented in the file was sliced somehow. So, for our purposes we can reslice it not in way [experiment frames count] / [expected frame size], but [experiment frames count] – [expected frame size] – 1. It will allow us to propagate data in case if it’s not enough at all or for some classes.

Each matrix has variety row count. Here are all counts of sizes:

Count of rows: count of matrixes

39: 1,

44: 2,

46: 1,

51: 2,

52: 2,

53: 1,

54: 1,

55: 2,

57: 1,

58: 3,

59: 1,

60: 4,

61: 7,

62: 25,

63: 95,

64: 375,

65: 1423,

66: 5367,

67: 6724,

68: 7,

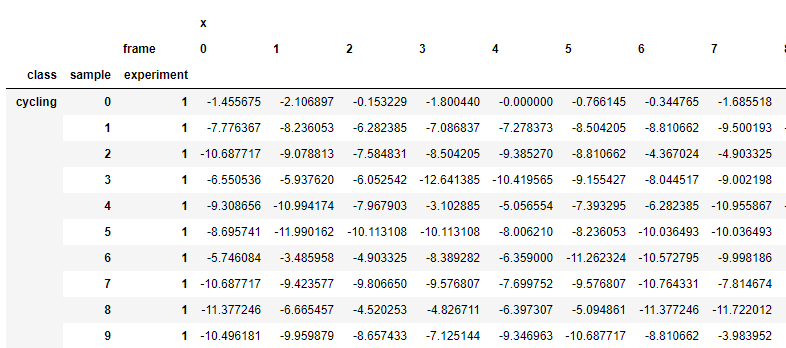
69: 1,

73: 1,

88: 1.

To save most of the data it will be enough to save just matrixes with sizes equal or greater than 60 rows and slice sizes for the rest of the matrixes to 60 rows. Thus, we will have all matrixes with the same size.

To manipulate with all the data in the easiest way we’ve collected all the matrixes in Pandas DataFrame object [1].



*Figure 2: DataFrame object view with all the data (key indexes: class, sample (number of sample in class) and experiment; key columns: axis and frame (60 frames))*

After all data was prepared, we received next samples per class counts:

Cycling 1562

Downstairs 1457

Idle 545

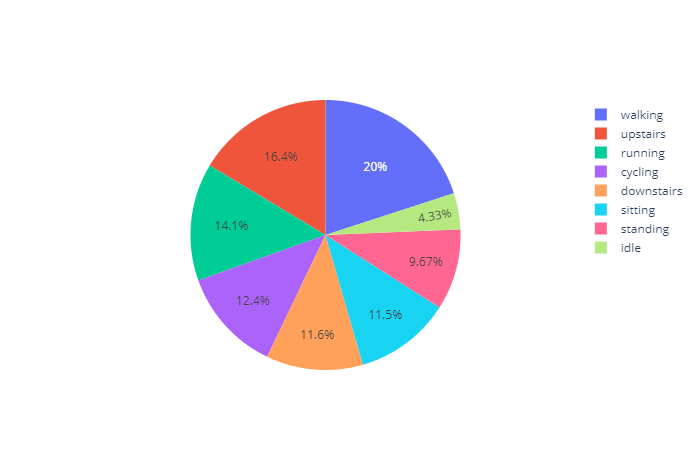
Running 1769

Sitting 1449

Standing 1216

Upstairs 2061

Walking 2514



*Figure 3: Samples per class distribution*

This shows we have enough data for splitting it into train and test sets. But if we take a look how many experiments do we have per class we’ll see an opposite result:

Cycling 17

Downstairs 77

Idle 3

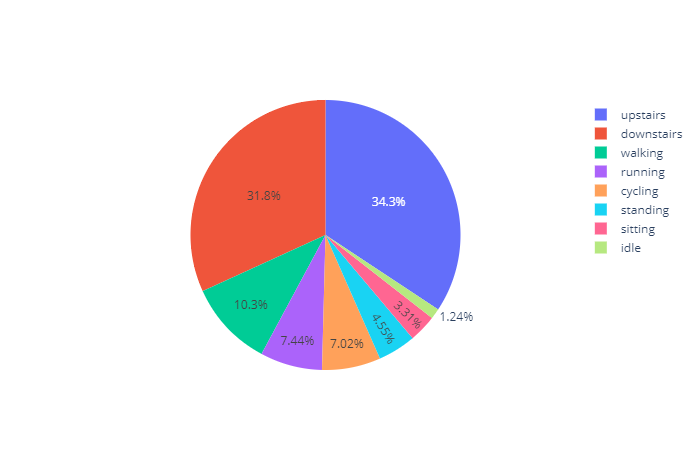
Running 18

Sitting 8

Standing 11

Upstairs 83

Walking 25



*Figure 4: Experiments per class*

Based on “standing”, “sitting” and “idle” are very similar classes of activity there is some sense to merge them together in testing purposes.

The last point that was fixed in the data is duplicates in "downstairs" class. There were two classes “downstairs” and “downsta” which represented the same data. So, we’ve deleted one.

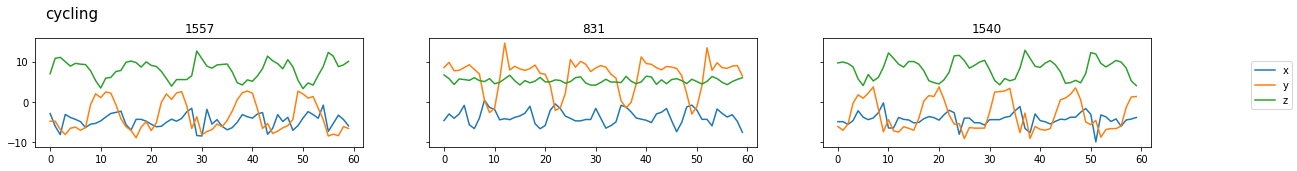
### 5.2. Data exploration

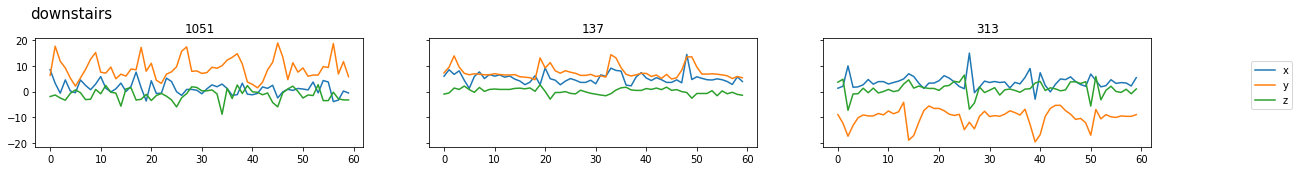
All the matrixes in different experiments based on figure 1 have few differences:

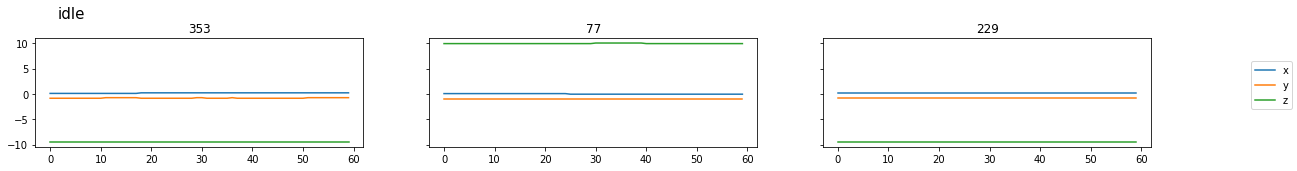
* mixed placement relative to each other;
* different peak orientation (in some chunks signal looks similar, but they are inverted);
* frequencies.

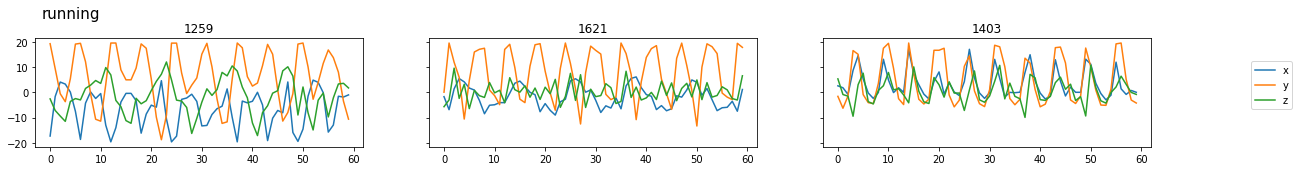
There are two reasons for that: different orientation of device and different testers. For example: if we make 2 measurements with face up and face down device one axis signal will be upside down. And if we measure running of 2 testers one of them might run faster, so it will be more frequent signal by one axis.

Most signals samples are pretty similar within a class and pretty different between different classes. On figure 5 below you can see random samples of signals.

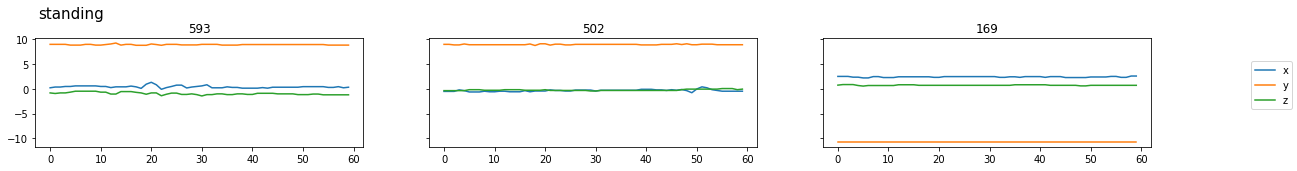


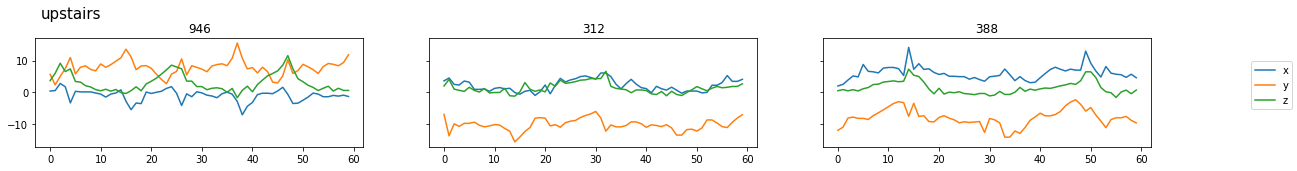


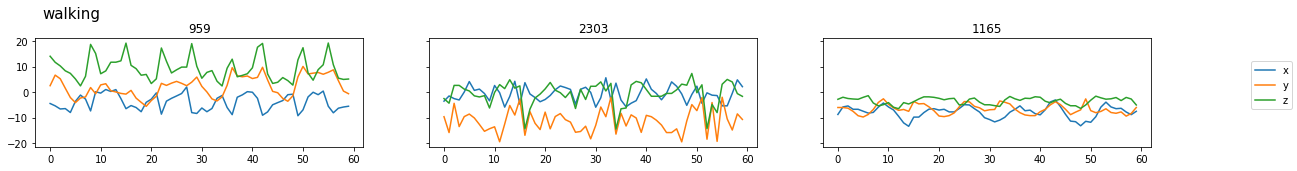










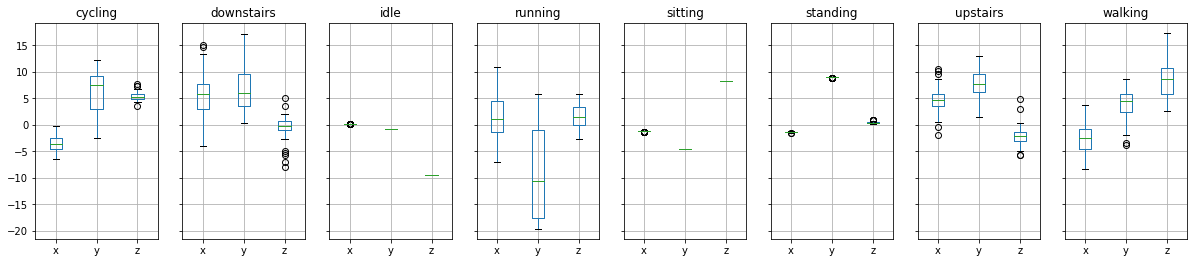


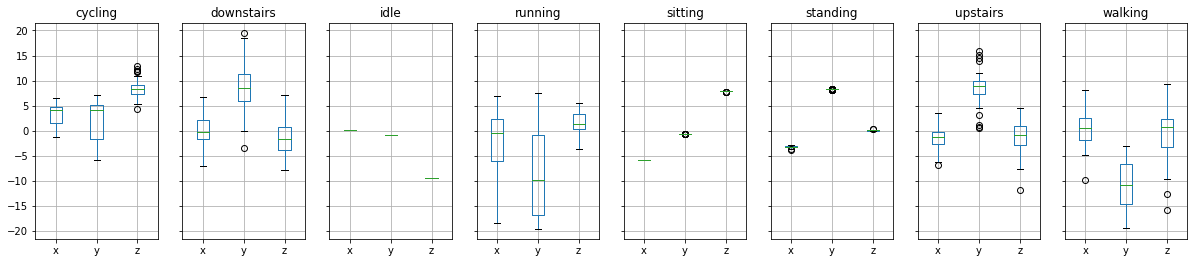
*Figure 5: Signals for different samples within and between classes (number in title show number of sample)*

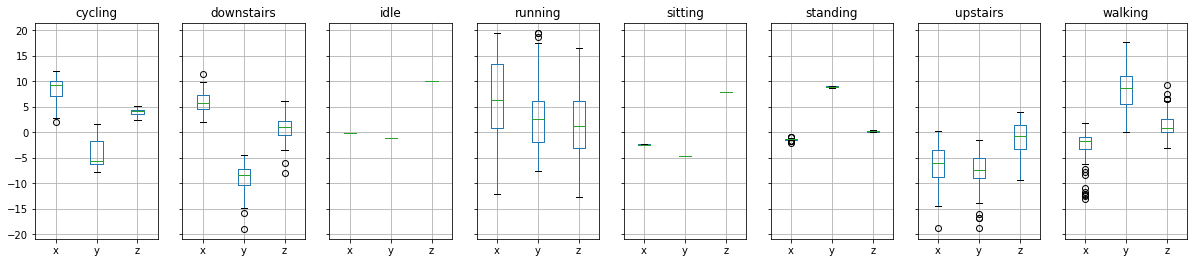
### 5.3. Data preparation

In aspect we have enough data for different classes there is no reason to propagate samples. Each sample has enough length of signal that does visible even few periods. Period length and view very differs from class to class so there is no way to use a fixed period for analysis.

On figure 6 you can see different classes have different distribution of values so we can analyze statistic measures. Each sample is 3 separate signals so we need to build features for each of them. Boxplot graph is the best way to display that.







*Figure 6: Boxplots for different samples within and between classes*

**Statistic features**

On figure 6 you can see that different classes’ signals have different interquartile range, median, outliers. Each feature vector item displays just one evaluation. Since this new vector has smaller size it makes model evaluation easier.

Pros:

* easy to compute;
* long time of computation (to get quantiles signal should be sorted)

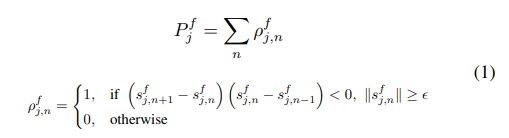
Cons:

* loose signal nature.

**Signal features**

There is one more easy method to calculate signals’ features to save some signal nature is to extract some signal evaluations: mean, standard deviation and number of peaks [2].

Here how we can calculate number of peaks:



Pros:

* easy to compute;

Cons:

* not full nature of signal is saved;

**FFT features**

One more good way to extract features from the signal is to transform it to frequency domain. The most popular way is fast Fourier transform. This transformation returns new representation of a signal as a sequence of complex numbers with amplitude in real part and phase in complex one.

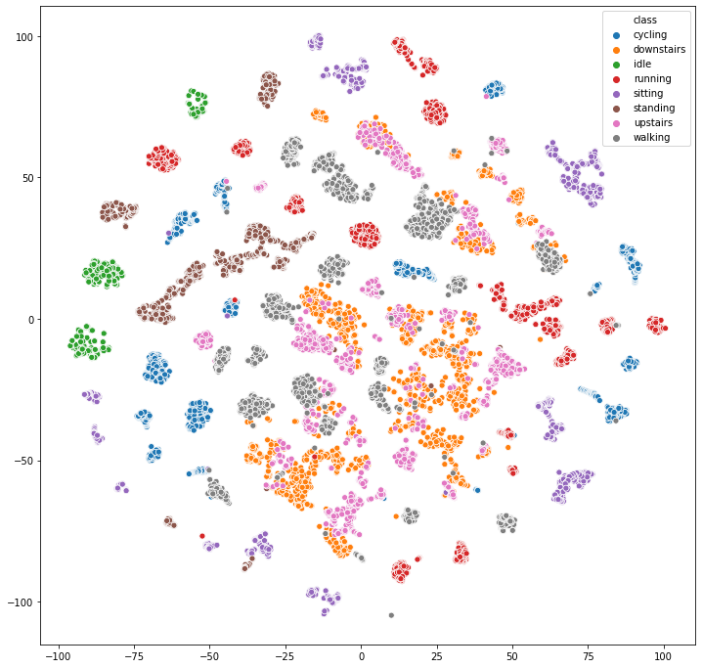
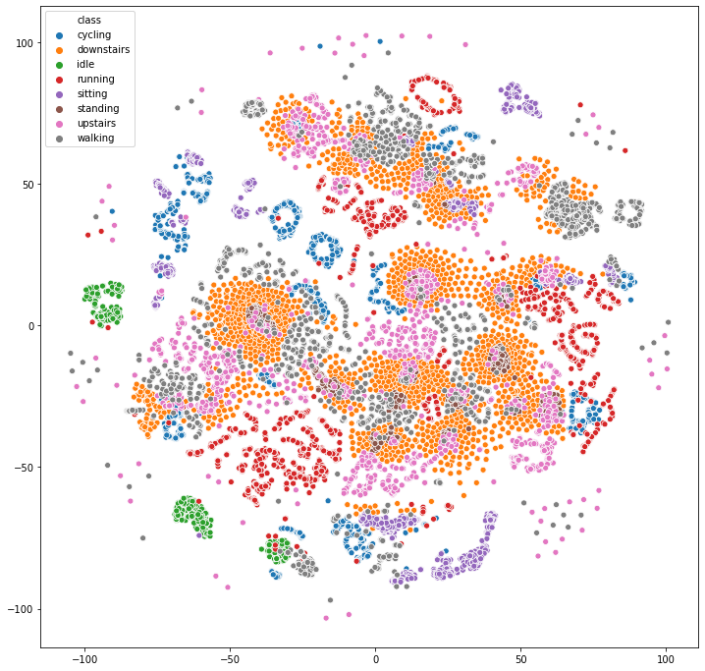
Pros:

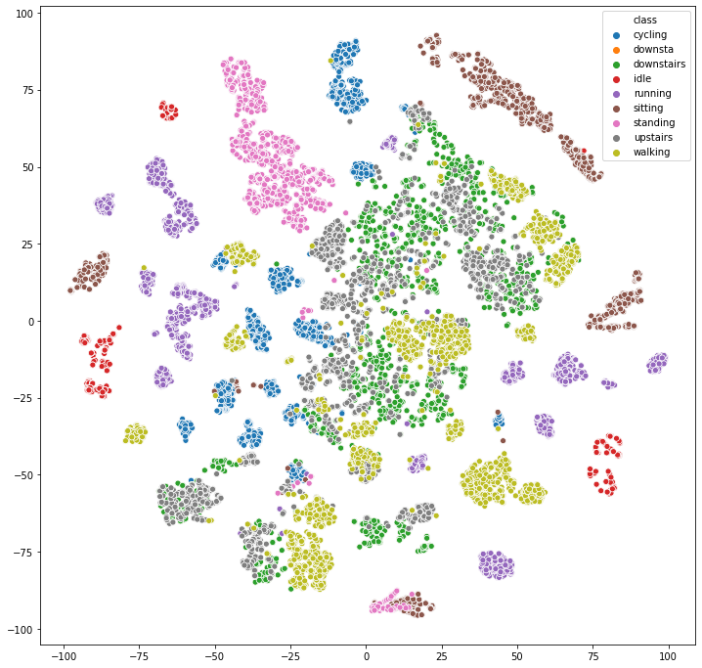
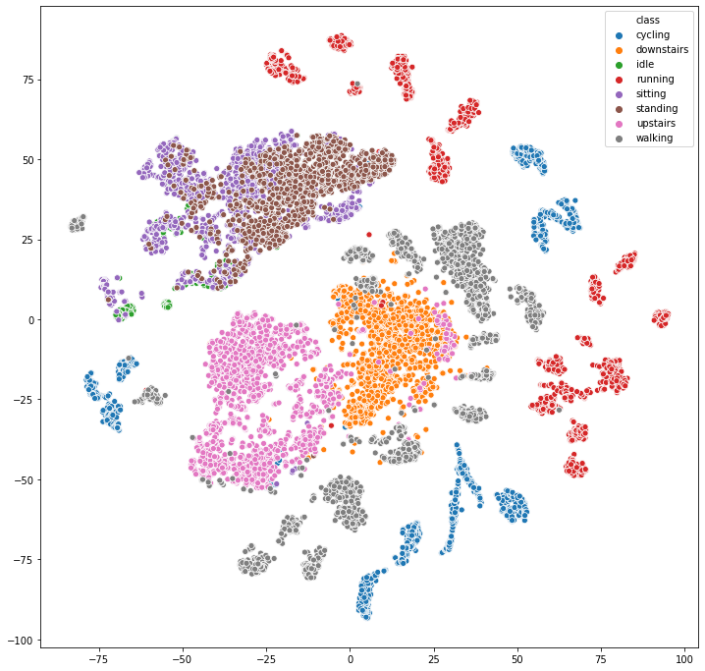
* full signal nature is saved;
* easy to compute.

Cons:

* pretty bigger size of features;
* window function should be applied because of not fully periodic nature of signal.

To show how new features could distributes in space we can decries feature vectors size and draw them on 2d plot. We used T-SNE method to convert high-dimensional vectors to 2 components:



*Figure 7: Signals scatter for original signals, statistic features, signal features and for FFT features*

On figures above you can see all features except the original signal gives good distribution by clusters with a single class. But on most plots few classes pretty mixes, such as “sitting”, “standing” and “idle” or “upstairs” and “downstairs” . As “sitting”, “standing” and “idle” are classes of inactivity we’ll treat them as “still” class, but “upstairs” and “downstairs” we’ll leave separated because they reflect different activities and the difference between them could be significant in healthcare or sports aspect.

### 5.4. Modeling

The simulation was based on 4 algorithms:

1. KNN
2. Linear regression
3. Support-vector machine
4. Random forest

And on next features:

1. Statistic
2. Signal
3. FFT
4. Statistic and signal features combination
5. Statistic, signal, and FFT features combination

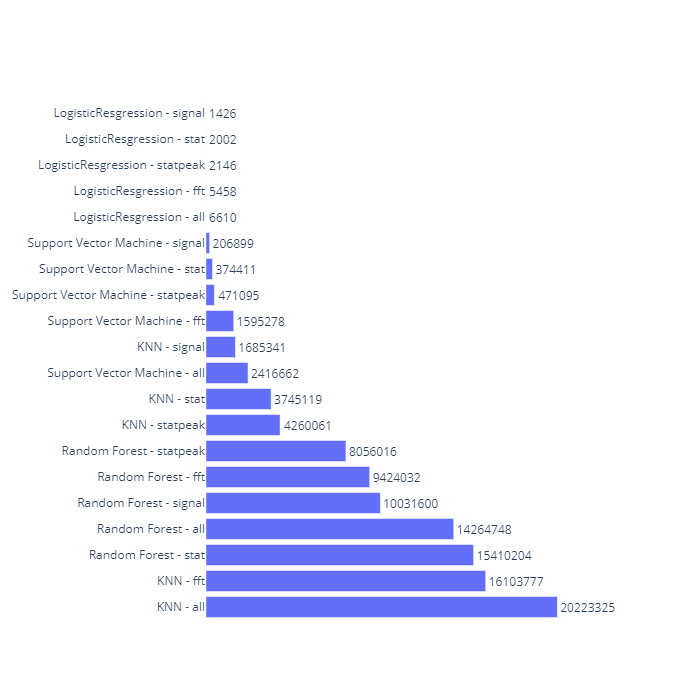
The results of the algorithms using the optimal settings of hyper parameters are given below.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Feature | Mean accuracy | F1 score (weighted) |
| KNN | FFT | 0.9257 | 0.9256 |
| Linear regression | FFT | 0.9257 | 0.9244 |
| Random forest | FFT | 0.9177 | 0.9164 |
| Support-vector machine | FFT | 0.9440 | 0.9439 |

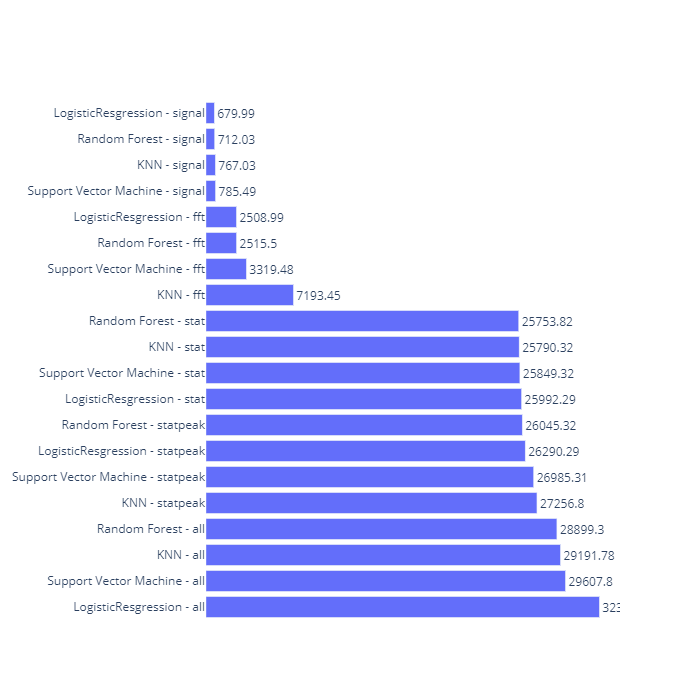
Figure 8: Evaluations of base models

### 5.5. Outcomes

We build pretty accurate models to classify the accelerometer signal dataset. To choose which model should be deployed we tested them by 2 more parameters and here is what did we get:

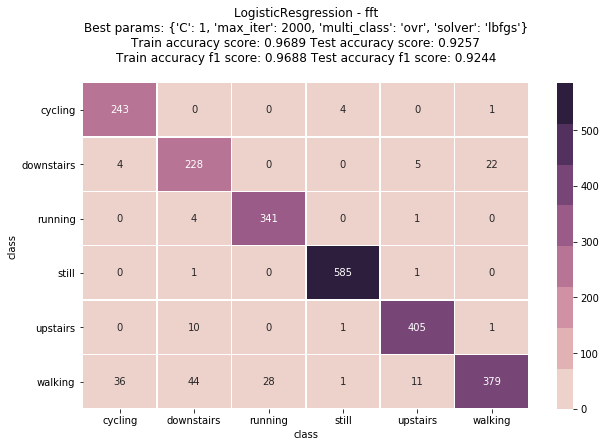


*Figure 9: Size of models*

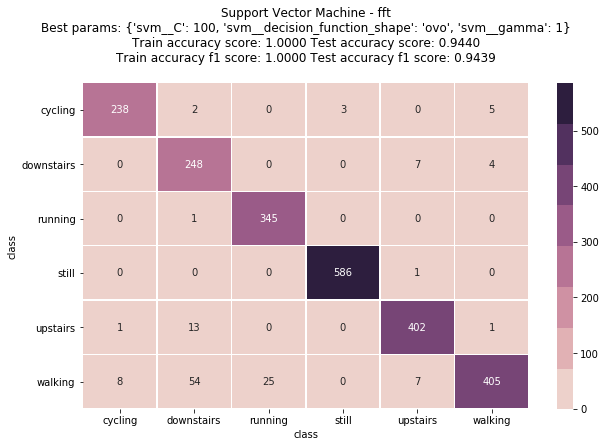


*Figure 10: Time of feature evaluation and prediction*

Finally, we get product of time, size and accuracy scores and as the result we can recommend 2 modes: support-vector machine and logistic regression. On figures below you can see confusion matrixes of both with the best hyperparameters combination.



*Figure 11: Logistic regression*

**

*Figure 12: SVM*

# Bibliography

|  |  |
| --- | --- |
| [1] | «Pandas DataFrame,» [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html. |
| [2] | S. M. I. A. D. F. L. a. A. S. M. I. Igor Bisio, «Enabling IoT for In-Home Rehabilitation: Accelerometer Signals Classification Methods for Activity and Movement Recognition». |