Report Pervasive Computing Assignment 2

1) Creation of the dataset

1. Data recording

Measurement tool: I downloaded the app “Sensor Logger” because it is able to track the acceleration in a 3-dimensional space and the sampling rate can be accurately chosen. Furthermore, recorded tracks can be simply exported as csv files.

Sampling rate for the recording: According to [2], 100 % of human movements shall lay below 20 Hz, 99 % below 15 Hz, and 98 % below 10 Hz. More than 120 Hz exceeds human response. Even further, machine learning methods, like the J48, achieved for a sampling rate of 20 Hz the best results. For 30 Hz the methods performed better but not in a significant way [2].
Beyond that, there is lot of human activity recognition research, which also tracks human body movements, which applies sampling rates in a range of 1 to 20 Hz [3].
Therefore, the sampling rate was set to **20 Hz**.

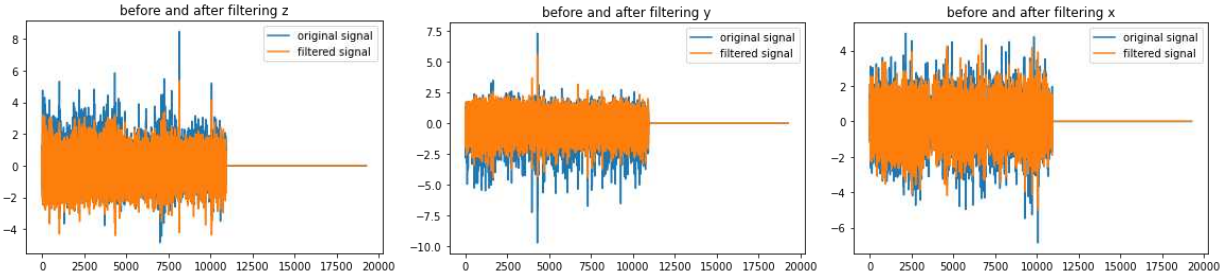
Data recording: I went along a way for ca. 50 minutes per location/class. For each of them I walked along the same way such that the condition for all of them are similar. For the hand classes I simply held the smartphone during walking in the belonging hand. For the pocket classes I chose the front pockets of my trousers.
As it was recommended in the exercise session, I changed the orientation of my smartphone for each location. The left hand held the upside down smartphone such that the camera pointed to the opposite direction of my body like it would catch the left outer side. Also, for the right hand I held the smartphone upside down, this time the camera pointed to the environment which was right to my body. For the left pocket the smartphone was nearly upright with the camera pointing to my leg. For the right pocket the smartphone was upside down with the camera pointing to my leg.

2. Pre-processing, segmentation, feature extraction

Pre-processing and segmentation

Cutting: For each signal I cut the first and last 5 seconds away to be sure that only the process of walking is considered.

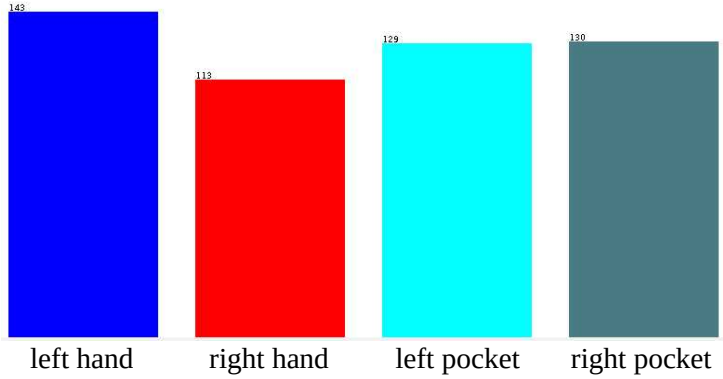
Filtering: The low-pass filter enables removal of noise and gravity acceleration [1]. I had a closer look at the low-pass **Butterworth filter**. As mentioned in the sampling rate choice, 99 % of human movenments shall be covered by 20 Hz [2]. Therefore, the filter threshold was set to that value. Besides that, [4] outlines that strong phase distortions might happen when a unsuitable filter order and frequency are selected. In their case that was caused by setting the order to 3 and the frequency to 4 Hz when measuring leg acceleration [4].
[3] also mentioned that the filtering for each of the 3 dimensions shall happen separately.
I considered the findings of [3] and to be sure that there are no strong signal delays for the individual dimensions I also plotted them. Here is an example:



It becomes clear that for an **order of 9** and a **frequency of 20**, there are no strong shifts in the signal.
(Side remark: The end of the signal is 0-padded as all signals are stored in one array together and therefore need the same length. Snippets which only contain 0s are excluded from the classifcation process.)

Windowing: [1] states that for window lengths of at least **3 seconds** the signal-to-noise ratio (SNR) for walking activities is higher than for smaller windows. Also, [1] uses 3 second long windows. Moreover, I also roughly counted how many seconds I need to make one leg/arm swing. It was a bit more than 1 second. Through 3 second windows it becomes probable that at least one walking activity is captured even when there is a short pause between the movements. The windows don‘t overlap. Each window counts as one sample from which features are extracted for classification.

This gives the following class distribution:



The classes are more or less equally distributed. As no class clearly dominates or is weakly represented, the class distribution can be used for the classification process as it is.

Feature extraction

For this assignment I concentrated on **time-domain features**. All features are extracted for each dimension of the filtered signal windows with exception of **magnitude mean** and **magnitude variance**. Those were introduced in the exercise session.

[3] presents different feature options of which I implemented the **maximum, minimum, sum, absolute sum, standard deviation, euclidian norm, zero crossing rate, mean, absolute mean**. (Side remark: Skewness and kurtosis were also implemented first. However, often they delivered only 0s. Therefore, they were excluded from the classification process.)

Furthermore, [3] gave me the idea of dimensionality reduction via **Kernel Discriminant Analysis (KDA)** for time-domain data. KDA aims to handle high within-class variance which is an issue of

human activity recognition [1]. It produces a low-dimensional outcome whose number of dimensions is the number of classes minus 1, in this case 3.

2) Classification

class legend:

- 'left_hand': 1
- 'right_hand': 2
- 'left_pocket': 3
- 'right_pocket': 4

kNN

default: 0.726

number of neighbors:
3 → 0.701
5 → 0.719
8 → 0.716
9 → 0.729 (keep for further hyper-parameter experiments with kNN)
10 → 0.721
11 → 0.719
20 → 0.690
30 → 0.623

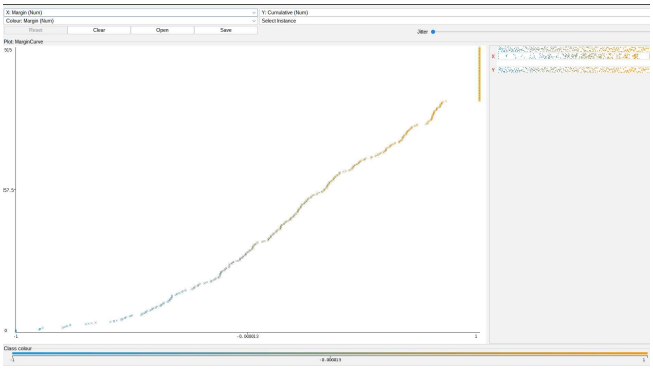
weighting:
1/distance → 0.733 (keep)
1-distance → 0.727

nearest neighbor search algorithm: no impact on F-measure

```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         371               72.0388 %
Incorrectly Classified Instances       144               27.9612 %
Kappa statistic                          0.6261
Mean absolute error                      0.1935
Root mean squared error                  0.3112
Relative absolute error                 51.7079 %
Root relative squared error             71.9427 %
Total Number of Instances              515

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,769    0,008    0,973      0,769    0,859      0,824    0,973     0,952     1
                 0,699    0,007    0,963      0,699    0,810      0,782    0,940     0,913     2
                 0,721    0,168    0,589      0,721    0,648      0,519    0,886     0,682     3
                 0,685    0,190    0,549      0,685    0,610      0,463    0,830     0,610     4
Weighted Avg.    0,720    0,094    0,768      0,720    0,733      0,647    0,908     0,789

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 110   1  13  19 |   a = 1
   1  79  15  18 |   b = 2
   0   0  93  36 |   c = 3
   2   2  37  89 |   d = 4
```

Looking at the kNN it becomes clear that it reaches a relatively good score (F-measure of 0.733). The confusion matrix tells us that false positives for classes a/1/left_hand and b/2/right_hand are

very low (look at the first two columns). Looking at the entries of c/3/left_pocket together with d/4/right_pocket it seems to be the case that the samples of those two classes are relatively close in kNN space such that an optimal separation of them is not possible.



X: Margin | Y: Cumulative

The margin gives the difference between the probability of true class and the highest probability of other classes (https://weka.sourceforge.io/doc.dev/weka/classifiers/evaluation/MarginCurve.html). The gaphic tells that the kNN is often very sure about its prediction and is actually correct about it as a score of 1 is reached. However, for lots of samples the kNN is uncertain and therefore the margin score is lower. This could come from the fact that for classification 9 neighbors are considered. When at least one neighbor is from another class then the certainty is lowered. Notice, that only very few samples fall into -1, meaning that the network almost never predicts the wrong class with absolute certainty.

Naive Bayes

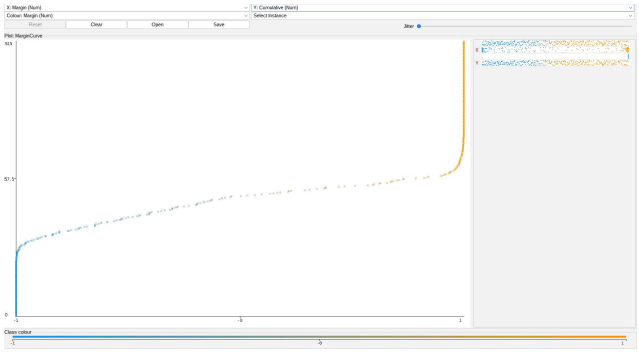
default: 0.511

useKernelEstimator = True → 0.557

```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         291               56.5049 %
Incorrectly Classified Instances       224               43.4951 %
Kappa statistic                          0.423
Mean absolute error                      0.2089
Root mean squared error                  0.4324
Relative absolute error                 55.8377 %
Root relative squared error             99.9761 %
Total Number of Instances              515

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,378    0,062    0,701      0,378    0,491      0,397    0,914     0,721     1
               0,673    0,177    0,517      0,673    0,585      0,454    0,861     0,764     2
               0,729    0,202    0,547      0,729    0,625      0,484    0,854     0,622     3
               0,515    0,135    0,563      0,515    0,538      0,392    0,822     0,552     4
Weighted Avg.  0,565    0,141    0,587      0,565    0,557      0,430    0,864     0,663

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 54 69  6 14 |  a = 1
 21 76 13  3 |  b = 2
  0  0 94 35 |  c = 3
  2  2 59 67 |  d = 4
```

Naive Bayes works with conditional probabilities. It seems to be the case that it is not helpful to know about another feature to determine the current feature's probability because the F-measure scores with 0.557 rather low.
Besides that, the confusion matrix shows that the pocket classes, c/3/left_pocket and d/4/right_pocket, are almost never predicted as hand classes, a/1/left_hand and b/2/right_hand.

Also, the vice versa case delivers only a few misclassifications. This shows us that the Naive Bayes can distinguish relatively well between pocket and hand classes.



X: Margin | Y: Cumulative

This margin curve plot shows that the Naive Bayes is most of the time very convinced as the most samples are at -1, it is 100 % sure about its wrong prediction, or at +1, it is 100 % sure about the ground truth prediction.

Decision tree (J48)

default: 0.776

confidenceFactor:
0.05 → 0.780
0.1 → 0.780 (keep for further hyper-parameter experiments with J48)
0.5 → 0.770

binarySplits, useLaplace, numFolds: no impact on F-measure

unpruned = True → 0.766

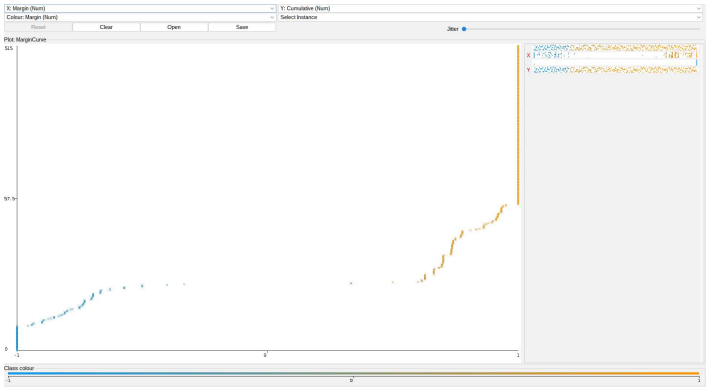
subtreeRaising = False → 0.775

useMDLCorrection = False → 0.783 (keep)

minNumObj:
1 → 0.779
3 → 0.779
5 → 0.777
8 → 0.755
22 → 0.652

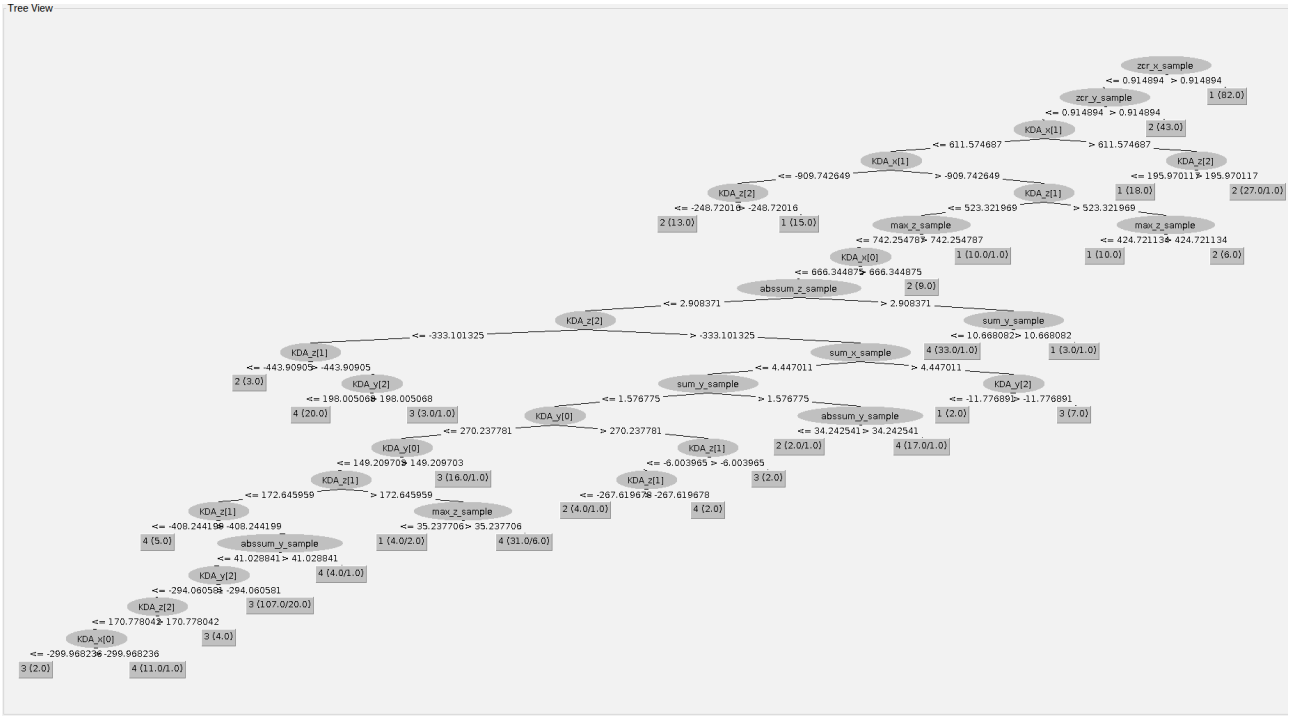
```
Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         403               78.2524 %
Incorrectly Classified Instances       112               21.7476 %
Kappa statistic                          0.7092
Mean absolute error                      0.1197
Root mean squared error                  0.3093
Relative absolute error                 31.9809 %
Root relative squared error             71.5074 %
Total Number of Instances              515

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,923    0,035    0,910      0,923    0,917      0,884    0,952     0,884     1
               0,823    0,032    0,877      0,823    0,849      0,809    0,918     0,798     2
               0,721    0,114    0,679      0,721    0,699      0,595    0,862     0,641     3
               0,654    0,109    0,669      0,654    0,661      0,549    0,830     0,654     4
Weighted Avg.  0,783    0,073    0,784      0,783    0,783      0,711    0,891     0,746

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 132   6   2   3 |   a = 1
   5  93   8   7 |   b = 2
   3   1  93  32 |   c = 3
   5   6  34  85 |   d = 4
```

The decision tree reaches a F-measure of 0.783. The classification of the first two classes (the hand classes) work very well as there are only a few off-diagnoal values related to them. Even further, they can be well distinguished from the pocket classes, c/3 and d/4.
The pocket classes can't be distinguished from each other very well. For kNN exactly the same problem happened. However, at least the pocket classes are very well separable from the hand classes for J48. Therefore, it outperforms the kNN.



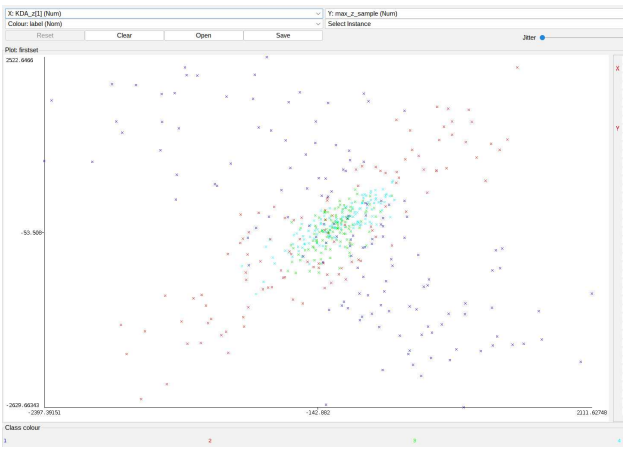
X: Margin | Y: Cumulative

As J48 is a decision tree, it is most of the time quite certain about its predictions and because those predictions often fit with the ground truth by scoring +1. However, the tree is not sure about its predictions all the time that can happen when one leaf node already learned in the training that it might not contain only one class.

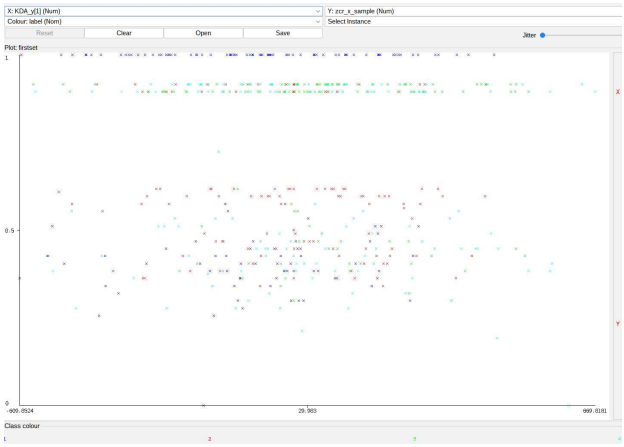


In the tree visualization it becomes clear that only the features zero-crossing rate, maximum, sum, absolute sum, and KDA are needed for classification. KDA is the feature with the most node representations. Through that, the KDA's authorization assumption is supported for human motion measurement tasks.

X: KDA_x[1] | Y: KDA_z[2]



X: KDA_z[1] | Y: max_z



X: KDA_y[1] | Y: zcr_x

Looking for interesting feature relation, KDA delivers the most interesting ones. In the first graphic it becomes visible that the second dimension of the x-signal KDA and the third dimension of the z-signal KDA already lead in combination to some kind of cluster building. Class 1/left_hand is visualized by some kind of negative correlation between the two features and class 2/right_hand by a positive one.
The second plot underlines this observation with the difference that on the x-axis is the second dimension of the z-signal KDA and on the y-axis the maximum for the z-signal.
The last illustration shows that the zero-crossing rate for the x-signal separates the class 1/left_hand and pocket classes, 3/left_pocket mixed with class 4/right_pocket, part-wise from the other classes.

Multilayer Preceptron

default: 0.729

learningRate:
0.01 → 0.612
0.05 → 0.732
0.1 → 0.737
0.2 → 0.743 (keep for further hyper-parameter experiments with MLP)

momentum:
0.0 → 0.726
0.01 → 0.732
0.1 → 0.735
0.3 → 0.720

batchSize: no impact on F-measure

trainingTime:
400 → 0.737
600 → 0.754 (keep)
1000 → 0.752

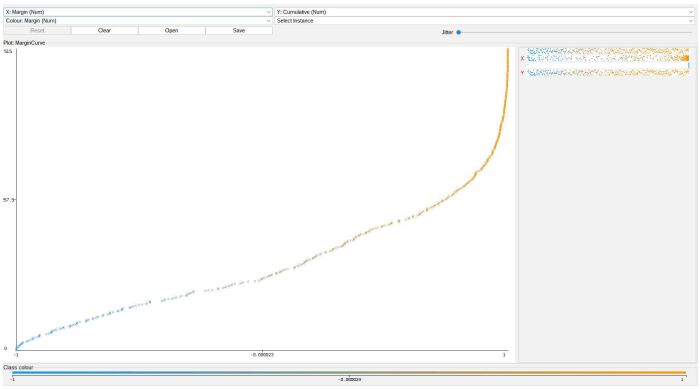
hiddenLayers:
6 → 0.674

10 → 0.714
30 → 0.755
40 → 0.770 (keep)
50 → 0.757
70 → 0.739

decay = True → 0.434

```
Time taken to build model: 17.08 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         394               76.5049 %
Incorrectly Classified Instances       121               23.4951 %
Kappa statistic                          0.6859
Mean absolute error                      0.1478
Root mean squared error                  0.3053
Relative absolute error                 39.4925 %
Root relative squared error             70.5769 %
Total Number of Instances              515

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,853    0,022    0,938      0,853    0,894      0,857    0,968     0,944     1
               0,788    0,022    0,908      0,788    0,844      0,807    0,922     0,886     2
               0,767    0,130    0,664      0,767    0,712      0,610    0,899     0,731     3
               0,646    0,140    0,609      0,646    0,627      0,496    0,838     0,592     4
Weighted Avg.  0,765    0,079    0,780      0,765    0,770      0,693    0,908     0,789

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 122   1   6  14 |   a = 1
   1  89   8  15 |   b = 2
   3   2  99  25 |   c = 3
   4   6  36  84 |   d = 4
```

The MLP reaches the second best F-measure with 0.770. It shows a very similar confusion matrix compared to the best performer J48. It only performs slightly worse.



X: Margin | Y: Cumulative

MLP gives probabilties for each prediction candidate/class. This becomes also in the illustration visible as lots of margin values lay between -1 and +1.
Moreover, the model is often quite sure about its prediction as lots of margin values are over 0.8 .

Conclusion: All methods showed that the pocket classes aren't perfectly separable from each other. I explain this with the fact that I wore lose trousers during the recording. Therefore, the phone might be not much moved through my legs.
The best performer is J48 (F-measure of 0.783) closely followed by MLP (F-measure of 0.770). It is able to classify each of the hand classes very well and separates the pocket classes from them. The

pocket classes are moderately well classified and not perfectly separated from each other. There is a good chance that with tight trousers a more precise classification would be achieved.
(Side remark: The best classifier's output is stored in the appendix file.)

References

[1] Khan, A., Siddiqi, M., & Lee, S.-W. (2013). Exploratory data analysis of acceleration signals to select light-weight and accurate features for real-time activity recognition on smartphones. *Sensors*, *13*(10), 13099–13122. https://doi.org/10.3390/s131013099

[2] Khusainov, R., Azzi, D., Achumba, I., & Bersch, S. (2013). Real-time human ambulation, activity, and physiological monitoring: Taxonomy of issues, techniques, applications, challenges and limitations. *Sensors*, *13*(10), 12852–12902. https://doi.org/10.3390/s131012852

[3] Sousa Lima, W., Souto, E., El-Khatib, K., Jalali, R., & Gama, J. (2019). Human activity recognition using inertial sensors in a smartphone: An overview. *Sensors*, *19*(14), 3213. https://doi.org/10.3390/s19143213

[4] Wang, W., Guo, Y., Huang, B., Zhao, G., Liu, B., & Wang, L. (2011). Analysis of filtering methods for 3D acceleration signals in Body Sensor Network. *International Symposium on Bioelectronics and Bioinformations 2011*. https://doi.org/10.1109/isbb.2011.6107697