Machine Learning Project
Analyzing runners profiles
First Report

Jessa Bekker (s0215494)
Koen De Cock (s0214395)

November 2013

1 Introduction

This research addresses a part of the research about detecting changes in a runner's movement pattern using data from accelerometers attached to the ankle and hip. It focuses on inferring whether the runner is trained or not and on which surface he/she is running.

The research focuses on which features and which classification techniques are good for this task. The experiments consist of five phases: 1) Preprocess the data to extract the part where the the runner is running. 2) Extract features from the preprocessed data. 3) Decide which features will be used for learning. 4) Use the selected features for learning. 5) evaluate the results.

The methods to learn about good features and classification techniques are discussed in more detail in section 2. The results of the executed experiments are presented in section 3 and discussed in section 4. In section 5 avenues for future work are offered.

2 Methods

To answer the research question, a testing environment was implemented in Python 2.7. In this environment data can be preprocessed to extract the part when the test subject is running. Different features can be calculated from the preprocessed data. The features can then be used for different classification algorithms. The accuracy of the features-classifier combination, is estimated with cross-validation. The combined classifier of trained and surface, is seen as a traditional classifier with 6 possible classes.

2.1 Preprocessing

The module `accproc` is used for loading the data and doing the first step of preprocessing: subtracting the mean and calculating the total acceleration. The running part is filtered from the data using the variance in the magnitude of the total acceleration, using the assumption that the variance will be large when the subject is running. Sliding windows are used to calculate the variation at each time unit and for smoothing these variations for tolerating small irregularities. Candidates for the running part are selected by taking pieces where all the smoothed variances are above a certain threshold. The threshold is higher for ankle measurements because a higher acceleration variance and noise is expected. The longest candidate is the chosen one. A part is removed at the start and the end to remove irregularities when the runner starts or stops running. This part of the data is returned. If no running part is found or the running part is too short to be taken into account, the data will not be used for learning because of the assumption something was wrong with the accelerometer or the runner did not run.

2.2 Features

The possible features are divided in four classes: time-domain features, frequency-domain features, peak features and velocity features.

2.2.1 Time-Domain Features

For each axis and the total acceleration, the following features can be calculated:

- Average
- Covariance between axises
- Minimum
- Maximum
- Median

The derivation of these features is straight forward.

2.2.2 Frequency-Domain Features

Each axis and the total acceleration can be converted to the frequency domain. When doing this, the second half of the frequencies are removed because they are considered noise. The zero frequency is also removed. Once this conversion is done the following features can be calculated in a straightforward manner:

- Magnitude of first N frequencies. (In our implementation, N is 10)
- The N main frequencies. (In our implementation, N is 10)
- Covariance between axes

2.2.3 Peak Features

Peaks in the total acceleration can be extract using the `detectPeaksGCDC` method of `accproc`. The possibilities are limited to the combinations of the different detection types (simple, cwt) and using smoothing or not. If smoothing is used, the type (hilbert, sg, butter) can be chosen together with using the correct data or the smoothed for the exact peaks. If peak extraction fails, the features for the failed peaks are set to None. If peak extraction succeeds, features are calculated as statistics on 1) the magnitudes of the peaks and 2) the distance between consecutive peaks. The statistics are the same as for the time-domain features, except for covariance since the different peaks are not about different axes but about different types of peak extraction. Variance however, is calculated.

2.2.4 Velocity Features

From the acceleration of each axis and the total acceleration, an approximation of the velocity can be calculated. This is done using dead reckoning: before the runner starts running velocity is zero and then the acceleration is cumulatively added to obtain the velocity. The velocity features are optimized for the ankle data. every time there is a peak, the assumption is made that the foot hits the ground and thus the velocity is zero.This results however in wrong calculation for the hip velocity From the velocities, the same features as for acceleration in the time domain are calculated.

2.2.5 Detection of Good Features

To determine which features are good, three methods are used. Selection of the K best features, selection of the K best features which are not too correlated and recursive feature elimination with cross-validation.

For selecting the K best features, `feature_selection.selectKBest` from `scikit-learn` is used. The measure used for deciding the score of each feature is the ANOVA F-test[5].

For selecting the K best features which are not too correlated, the 10K best features are set as possible features. The correlation of those features is calculated. One by one the features with the highest score are selected and the features which absolute correlation with that feature is above a certain threshold (0.2) are removed as possibilities. This method is expected to work better than the previous one because selecting features independently from one another gives rise to features which do not result in better learning when combined because they are too similar.

For recursive feature elimination with cross-validation [4], `feature_selection.RFECV` form `scikit-learn` is used. The algorithm recursively runs the classifier and removes the least helpful features. It is expected to be the best feature selector because it takes into account the learner and the combination of features. However this function requires the learner to have a fit method that updates a coef_ attribute that holds the fitted parameters. It can not be used for all learners. In our testing environment, it can only be used for Support Vector Machines and Logistic Regression.

2.3 Classification algorithms

The paper of Albert et al.[1] describes the performance of five classification algorithms on their problem. They classified falls, which used similar acceleration data. It is expected that the classification of running yields similar results. Their best algorithms were Support Vector Machines, Sparse Multivariate Logistic Regression, Decision Trees and K Nearest Neighbors. The same learners are tested to classify trained/untrained runners and running surfaces. In our test environment implementations of the `scikit-learn` package are used for the learners. All features are scaled from 0 to 1 before feeding them to the algorithms.

2.3.1 Support Vector Machine

A large number of features is extracted from each data-set. A SVM-classifier, which is effective in high dimensional spaces, even if the number of dimensions is bigger than the number of samples (which is the case here), is expected to excel in classifying this data. It was shown by the paper of Albert et al.[1] to give good results in classifying falls. The svm implementation of `scikit-learn` is used with a linear kernel because it gives better results than the other provided kernels (polynomial, rbf and sigmoid).

2.3.2 Decision Tree

Decision tree-classifiers are worse in classifying highly dimensional data, but this can be improved by selecting the best features. The advantage is that the decision tree can be shown graphically, which provides insight into which decisions are made, involving which features. The implementation of `scikit-learn` is used with the Gini heuristic as the optimal split parameter.

It tends to overfit to the data, which can be muffled by choosing parameters such as the maximum depth of the tree and the minimum samples needed in a node to split, which are provided by the used implementation. Non-supported methods reduce overfitting in a better way. These methods involve better stopping criteria like significance tests or MDL-based methods, or even better: post-pruning the tree

2.3.3 K Nearest Neighbors

It is hard to predict exactly what relation the different features will have with the classes. The advantage of KNN in this case is that it does not hold any assumptions on the characteristics of the classes. A disadvantage is that it is computationally hard. KNN also tends to overfit the data and tends to be fooled by irrelevant features and it suffers severely from the curse of dimensionality. The implementation of `scikit-learn` is used with distance as weights for the neighbors, because similar instances tend to have a similar class.

The higher the dimensionality, the higher the probability that the classifier is confused by irrelevant attributes. This could be improved by weighting the attributes according to their importance. The importance could be estimated by the "wrapper" approach or by using statistical formulas like the one from Langley where attributes that have a low variance inside classes are preferred. This is however not available in the used implementation.

2.3.4 Logistic Regression

Also known as regularized logistic regression. The paper of Albert et al.[1] mentions this algorithm also achieves good results. Like SVM, it copes well with a large number of features. The implementation of `scikit-learn` is used with the default parameters.

2.4 Handling missing values

All except one types of peak extraction we use, occasionally fail on the training set, which results in missing features. One way to handle this would be to discard the sample on which it fails. This would turn the training set of 120 samples into one of 90 samples. Another possibility is to only use peak extraction that never fails. But there is no guarantee that the default peak extraction (which is the one that never fails on the training set) would not fail on a sample that is not in the training set. For not losing to much data or peak related features, we opted for a third option: estimating the missing features. The estimation currently is use, is taking the mean of the value for this feature in the other samples.

2.5 Evaluation Method

In a number of papers [1] [2], cross-validation is used to evaluate the performance of the classifiers. Therefor 3-fold stratified cross-validation is used. As described in the lecture-notes, cross-validation yields an accurate estimate for the accuracy of the classifiers, though is somewhat pessimistic because the training sets used to evaluate the method are smaller than the actual one. If non stratified cross-validation were used, another problem would be that the training and the testing set would be complementary.

Because cross-validation is still an approximation, confidence intervals will be used to validate the results.

3 Results

Table 1 shows the results of the different classifiers. Table 2 shows the result when the 10 best features are filtered out. Table 3 shows the result when the 10 best, not too correlated features are filtered out. Table 4 shows the results when recursive feature elimination is applied.

Tables 5, 6 and 7 show the selected features for (un)trained, surface and combined classification respectively when using KBest and KBestUncorrelated. The features used when recursive feature elimination is applied are too many to show, but they always contain the features used by KBest and KBestUncorrelated.

	(Un)trained	Surface	Combined
SVM	0.9000	0.6167	0.5417
DT	0.8250	0.5667	0.3333
KNN	0.9000	0.5083	0.4500
LR	0.9000	0.5833	0.5167

Table 1: The mean accuracy after cross-validation for the different classifiers on the different problems *with all features included*

	(Un)trained	Surface	Combined
SVM	0.9083	0.4667	0.4833
DT	0.8500	0.4833	0.4500
KNN	0.8833	0.4750	0.5133
LR	0.9000	0.4750	0.4417

Table 2: The mean accuracy after cross-validation for the different classifiers on the different problems *with only the 10 best features included*

	(Un)trained	Surface	Combined
SVM	0.9167	0.5833	0.5176
DT	0.8417	0.5500	0.4083
KNN	0.9333	0.5167	0.5083
LR	0.9167	0.5667	0.4583

Table 3: The mean accuracy after cross-validation for the different classifiers on the different problems *with only the 10 best, not too correlated features included*

	(Un)trained	Surface	Combined
SVM	0.9917	0.7167	0.7917
LR	1.0000	0.8083	0.6333

Table 4: The mean accuracy after cross-validation for the different classifiers on the different problems with only the best features,*using recursive feature elimination and cross-validation*, included

KBest	KBestUncorrelated
ankle.Atotal.min	hip.simple_sg_cor.minPeak
ankle.MF8.Ax	ankle.simple_sg_ncor.medianPeak
ankle.MF9.Ax	hip.MF0.Atotal
ankle.simple_butter_ncor.medianPeak	ankle.MF7.Ax
ankle.simple_notSmooth.medianPeak	ankle.AxAy.covar
ankle.simple_sg_ncor.medianPeak	ankle.MF3.Ax
hip.AyAz.covar	ankle.AtotalAy.covar
hip.MF0.Atotal	hip.Az.F6
hip.simple_sg_cor.minPeak	
hip.simple_sg_ncor.maxDist	

Table 5: The selected features for (un)trained classification

KBest	KBestUncorrelated
ankle.AtotalAtotal.covar	hip.simple_butter_cor.medianDist
ankle.MF5.Atotal	ankle.cwt_sg_ncor.medianPeak
ankle.cwt_hilbert_cor.medianPeak	hip.MF1.Az
ankle.cwt_sg_ncor.medianPeak	hip.MF2.Ay
hip.MF1.Ay	hip.MF3.Ay
hip.simple_butter_cor.medianDist	ankle.simple_hilbert_ncor.medianPeak
hip.simple_butter_ncor.medianDist	ankle.VyVy.covar
hip.simple_hilbert_cor.medianDist	ankle.Vy.median
hip.simple_hilbert_ncor.medianDist	
hip.simple_notSmooth.medianDist	

Table 6: The selected features for surface classification

KBest	KBestUncorrelated
ankle.simple_butter_ncor.medianPeak	hip.simple_butter_cor.medianDist
ankle.simple_sg_ncor.medianPeak	ankle.simple_sg_ncor.medianPeak
hip.MF0.Atotal	hip.AyAz.covar
hip.MF0.Ay	ankle.MF7.Ax
hip.MF1.Ay	ankle.simple_sg_cor.varPeak
hip.simple_butter_cor.medianDist	hip.MF2.Atotal
hip.simple_butter_ncor.medianDist	ankle.AxAy.covar
hip.simple_hilbert_cor.medianDist	
hip.simple_hilbert_ncor.medianDist	
hip.simple_notSmooth.medianDist	

Table 7: The selected features for combined classification

4 Discussion

Classifying the surface is clearly a more difficult problem than trained/untrained runners. The results of classifying in six classes (combination trained-surface) seem to be better than the separate classifications.

Since the dataset only consist of 120 samples, it is hard to decide anything with a certain confidence, although some conclusions can be made.

When all features are used, DT does not score very well, it is with 90% certainty the worst learner. It is thus no surprise it benefits from feature selection. With the plain best K features selected, with 90% confidence, it will perform better on detecting the surface and with 95% it scores better on the combined classification. The other learners seems to suffer from the KBest feature selection, although the differences are only statistically relevant for surface detection by SVM and LR. SVM and LR are known for handling large dimensions rather well.

The picture becomes better when KBestUncorrelated is applied. Almost all estimated accuracies rise compared to normal KBest. Only for the combined classification, this trend is not so unanimous. Compared to including all features applying KBestUncorrelated only seems to help KNN.

The result with recursive feature elimination are significantly higher than without feature selection. Especially for LR where all results are higher with 99.5% confidence. However, bear in mind that the feature selection is optimized using cross-validation, which would naturally lead to a higher cross-validation score. Although these results are not statistically significant.

4.1 Classification Algorithms

If the estimated accuracies are considered true accuracies, one could conclude SVM is in general the best learner, followed closely by LR. SVM scores especially good to the combined classification, compared to the other learners. KNN scores alright, especially when only the best, not too correlated features are taken into account. DT scores, in general, the worst. Most of these conclusions are, however, not statistically significant, due to the small dataset.

4.1.1 Support Vector Machines

The SVM-classifier shows some of the best results. Figures 1a and 1b show the decision surfaces when only the two best features are used.[1] For the surface-classification, even with the two best (f-test), uncorrelated features selected, the sample-points are still mainly interleaved. This shows the difficulty that results in the poorer cross validation scores than with trained/not trained.

As can be seen in the figure, the SVM-classifier tries to find hyperplanes to seperate the classes. There is no overfitting and with enough features the classifier becomes very effective.

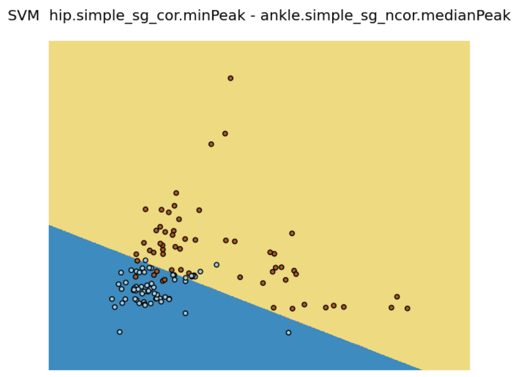
However, because the proportion of features to samples (548 to 120) is quite high, the results are not optimal. This can be helped by selecting the most important features or increasing the number of samples.

4.1.2 Decision Trees

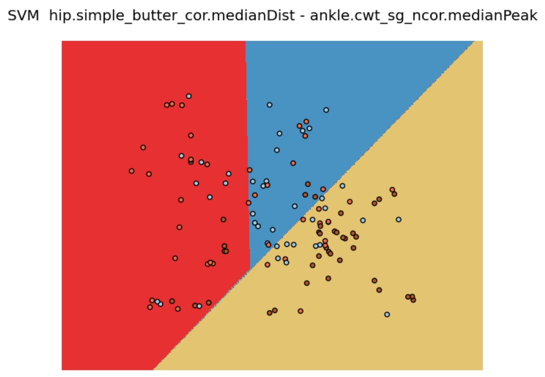
The decision tree-classifier performs generally worst. Especially classifying into the combined categories is too difficult. Selecting only the best features somewhat improves this.

Figures 2a and 2b show the decision surfaces, again with only the two best, uncorrelated features. They clearly show the vertical and horizontal decision edges.

[1]The number of features here is limited by the 2-dimensionality of the graph.

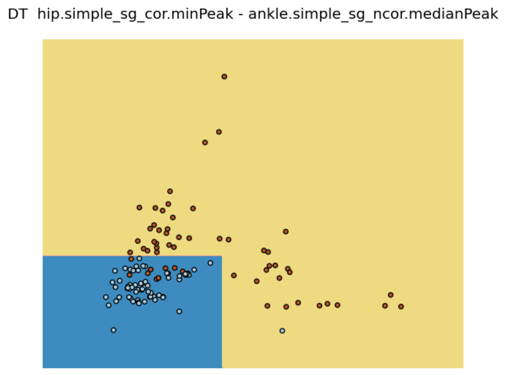


(a) trained/not trained

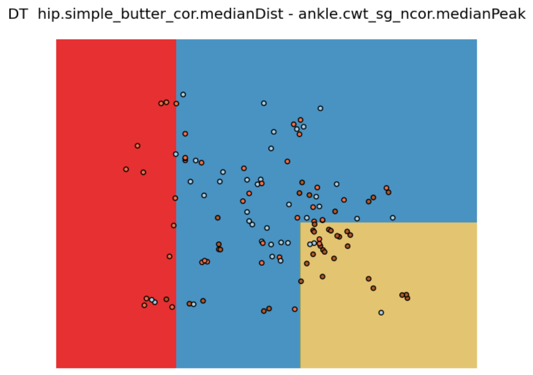


(b) asphalt/track/woodchip

Figure 1: The decision surface of the SVM classifier with the two best features

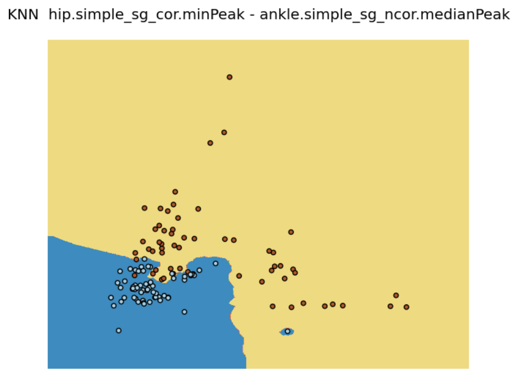


(a) trained/not trained

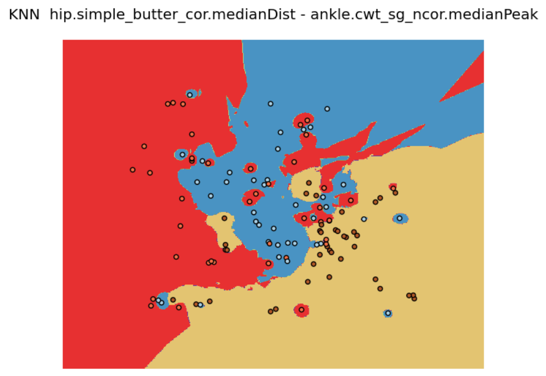


(b) asphalt/track/woodchip

Figure 2: The decision surface of the DT classifier with the two best features



(a) trained/not trained



(b) asphalt/track/woodchip

Figure 3: The decision surface of the KNN classifier with the two best features

4.1.3 K-Nearest Neighbors

KNN has slightly better results than DT, but slightly worse than SVM and SMLR. Figures 3a and 3b show the decision surface for two features. It seems to do quite well in approaching the most likely surface. However, small islands appear around outliers. This means the classifier is overfitting to the data. Figure 4 shows the neighbors of each data-set in the KNN-algorithm. Most points are located around the diagonal, which is probably due to data-sets of the same person lying next to each other on the axes.

4.1.4 Logistic Regression

The results of the logistic regression-classifier are similar the ones of the SVM-classifier, though it seems less accurate in predicting the combined classification. This is probably due to error propagation, which occurs when a multinomial logit classifier has a large number of sub-models[6].

4.2 Features

According to the f-test, features based on peaks are very important. Five different types of peak features are ranked in the top ten for (un)trained classification and seven for the other two classifications. They are however very correlated and filtered out using KBestUncorrelated, which only keeps one feature for hip peaks and one or two for ankle peaks. For (un)trained classification only the peaks magnitude seems important, whereas in in the other classifications, the distance between peaks gets a big role as well. The distance between peaks is important for the hip data and the magnitude for the ankle data.

A second important feature type are the main frequencies of the acceleration. For the (un)trained classification, the x coordinate of the ankle is pretty dominant, whereas for surface classification the main frequencies of the hip are more important.

A third important feature type is the covariance between acceleration axes or total acceleration.

The least important feature category seems to be velocity. Although for the surface classification, two of the velocity features are selected as important features. The selected features are the velocity for the ankle, which is no surprise since the calculation for the hip velocity is incorrect. The velocity is a very rough estimate, so it could be that velocity would provide better information if this estimate would improve.

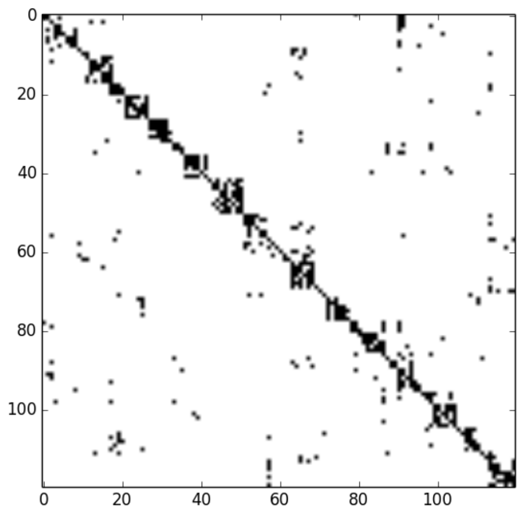
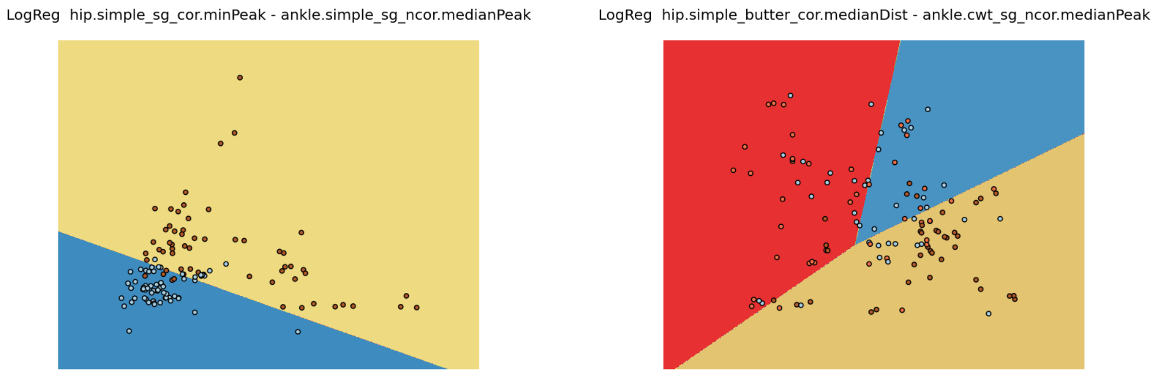


Figure 4: This shows the k neighbors for each dataset with k=5.



(a) trained/not trained

(b) asphalt/track/woodchip

Figure 5: The decision surface of the SMLR classifier with the two best features

5 Future work

Only few statistically significant conclusions could be drawn from our experiments. It would thus be very useful to have more data to experiment with and to obtain statistically significant results.

Because the learners are compared to each other with the same dataset, the results are actually not independent and confidence intervals can not be trusted. A better way of evaluating the results would be to use McNemar's test. To really evaluate the learners and not the hypotheses, the paired t-tests would be better. However the last option requires a lot of independent data to train and to test on.

Using peak detection was done with a blackbox approach. The provided function was used without questions. However since it occasionally does not work, it could be interesting to look into it with more detail. Different, more robust algorithms could be tried. Or using different values for the parameters could possibly change the results.

Missing values could be handled in a better way. For example, in a decision tree, when a node splits on a feature that is missing in one of the samples, that sample could follow the majority, or be partially be assigned to both subnodes.

Instead of using six classes for the combined classification we could also experiment with multi-label classification or see whether the results of predicting (un)trained and surface independently from one another are better.

5.1 Optimizing Learners

5.2 SVM

Now we used a linear kernel for SVM. We could experiment more thoroughly with other kernels to obtain better results.

5.2.1 DT

At this moment overfitting in the decision tree-classifier is avoided by tuning parameters like the maximum depth etc. This could be done more optimal by the use better stopping criteria like significance tests or MDL-based methods, or even better: post-pruning. The used scikit-learn package does not support pruning, but there are some projects that seek to help this.[3]

5.3 KNN

We could get rid of the irrelevance problem by weighting the importance of the attributes according to their importance as was descriped in section 2.3.3.

Using prototypes could also possibly improve results because it can get rid of noise.

5.4 Ensemble methods

Ensemble methods seek to get better classifications by taking the results of multiple classifiers (different learners, or variants of the same learner) and combining them by means of voting or other ways. This could potentially improve the results in some cases.

Appendices

A Time Spent

Jessa:

What	**time**
Literature Study	4:00
Writing report 1	3:45
Installing and learning how to use python and scikit-learn	12:30
Feature extraction	30:00
Data management	15:30
Feature selection	5:30
Analyzing results and writing report 2	15:00
Total	86:15

Koen:

What	**time**
Literature Study	5:00
Writing report 1	3:45
Installing and learning how to use python and scikit-learn	12:30
Classifiers	37:00
Velocity features	6:00
Feature selection	8:00
Analyzing results and writing report 2	14:45
Total	86:00

B Decision trees

Figures 6 and 7 show some decision trees by the DT-classifier.

References

[1] Mark V. Albert, Konrad Kording, Megan Herrmann, and Arun Jayaraman. Fall classification by machine learning using mobile phones. *PLoS ONE*, 7(5):e36556, 05 2012.

[2] Stephen J. Preece, John Yannis Goulermas, Laurence P. J. Kenney, and David Howard. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. *IEEE Trans. Biomed. Engineering*, 56(3):871–879, 2009.

[3] sgenoud. Adding a pruning method to the tree. `https://github.com/scikit-learn/scikit-learn/pull/941`, 2013. [Online; accessed 14-December-2013].

[4] J. Weston and I. Guyon. Support vector machine—recursive feature elimination (svm-rfe), January 10 2012. US Patent 8,095,483.

[5] Wikipedia. F-test — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=F-test&oldid=582604780`, 2013. [Online; accessed 14-December-2013].

[6] Wikipedia. Multinomial logistic regression — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Multinomial_logistic_regression&oldid=583285110`, 2013. [Online; accessed 14-December-2013].

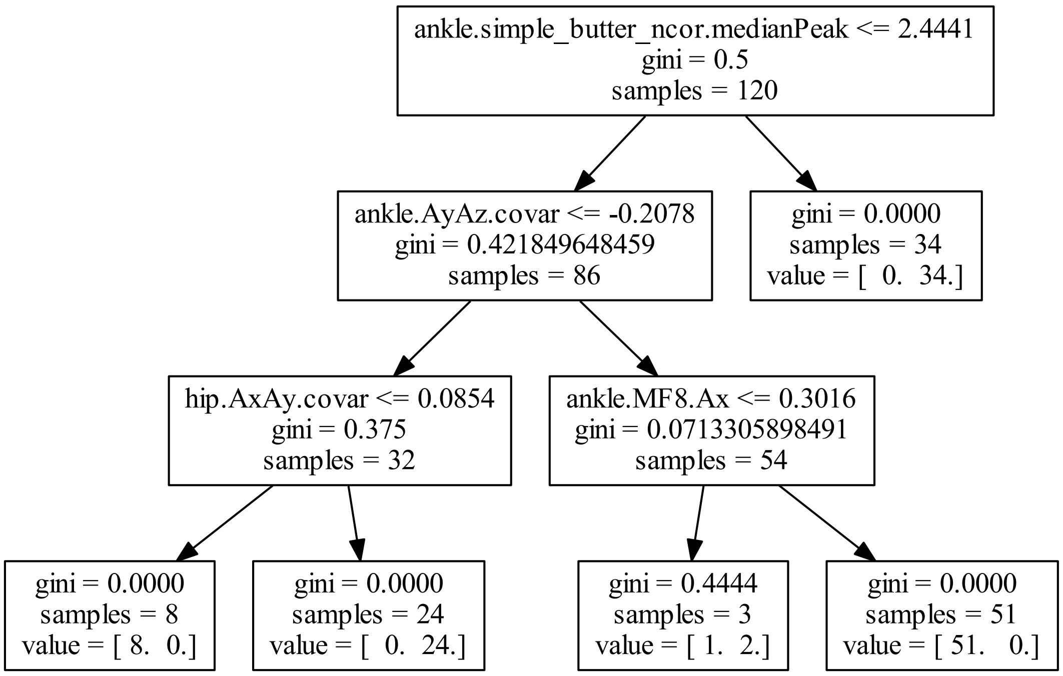


Figure 6: The decision tree trained to classify samples in not Trained/Trained

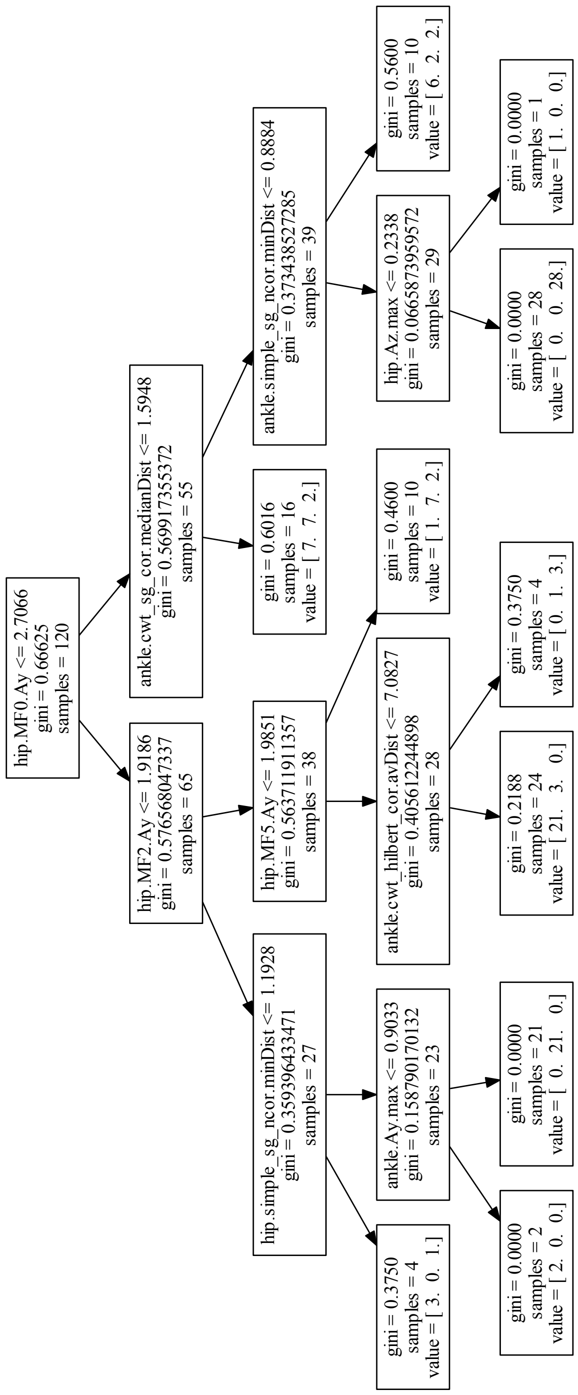


Figure 7: The decision tree trained to classify samples in asphalt, track or woochip. The tree has been limited to increase readability.