Machine Learning in Practice

Parkinson's Freezing of Gait Prediction

Steffen de Jong (s1065975)
Johannes Otte (s1105539)
Guillem Soler Sanz (s1102526)
Marta Español López (s1103330)

June 2023

Radboud Universiteit

1 Introduction

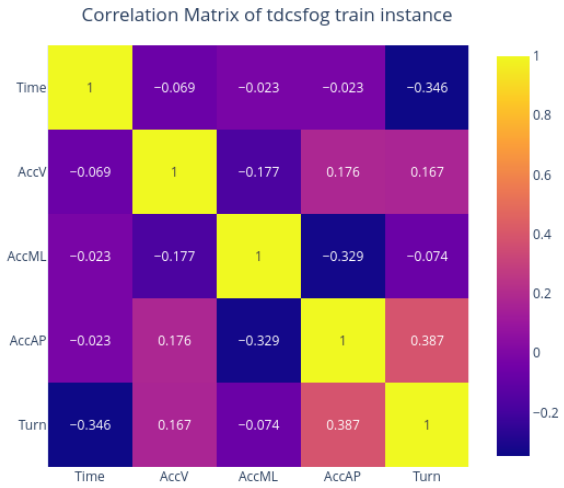
Our participation in the Kaggle competition "Parkinson's Freezing of Gait Prediction," which aimed to predict three different types of Freezing of Gait (FOG) events, provided valuable insights into understanding the causes and potential treatments for this debilitating motor symptom. FOG, prevalent among individuals with movement disorders, especially in advanced stages of Parkinson's disease, remains one of the most challenging symptoms to manage. By analyzing a time series dataset encompassing accelerometer data, contextual information, and other metadata, we aimed to accurately predict three distinct types of FOG events: Start Hesitation, Turn, and Walking. The evaluation metric for our predictions was Mean Average Precision (MAP).

2 Method

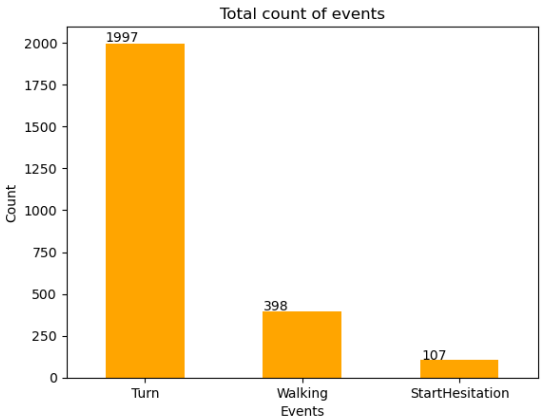
In this Section, we will describe the different strategies and techniques we explored during this project. First, some data exploration was performed to analyze the data and gain a better understanding. For our first approach, we started off with a decision tree to perform feature selection and then used a Support Vector Machine (SVM) to carry on the classification task. We chose a notebook using gradient boosted trees as a comparison baseline which we tried to advance as well.

2.1 Exploratory Data Analysis

To get a better understanding of the data, some exploration was performed. First, a simple plot of of a random sample for both datasets was made, where we could observe how the three kinds of accelerometer data (AccAP: anteposterior, AccV: vertical, AccML: mediolateral) actually look like, Figures 2a and 2b can be found in Appendix A. To check whether the classes were balanced, we plot how often each different event happens in the data: Start Hesitation, Turn or Walking. It can be seen in Figure 1b that the data is not equally distributed and that the class Turn is over-represented, the Turn event seems to happen more often than the other, this may lead the model to over-predict the label Turn for a given instance. On Figure 1a it can be appreciated that some variables are slightly correlated. As it could be expected, it seems that when there is a turn there is a negative effect on time, and a positive effect on the AccAP. And AccAP has a negative relationship with AccML, which we would understand as a decrease on the ante-posterior movement when there is medio-lateral movement.



(a) Correlation matrix for tdcsfog instance.



(b) The distribution for the events: StartHesitation, Turn and Walking. For all the time series.

Figure 1: Depicts the EDA performed.

2.2 Pre-Processing

The main problem that we encountered when running notebooks was memory overload. Therefore, we looked at solutions for memory optimization. We found a function for reducing the size of the data frames that converts the data types of the columns in the notebook to an equivalent data type that requires less memory. Later we implemented a file-saving strategy using parquet files in combination with the use of garbage collection, that achieves that only the training data that is currently needed is in memory (notebook no. 10, see Table 1).
Regarding actual pre-processing of the data for the task goal, we only managed to implement a variant that accounts for the different sample rates of the datasets tdcsfog and defog (in notebook no. 7, Table 1).
Most of the features have been extracted from "Prediction of freezing of gait in parkinson's disease using wearables and machine learning", Borzì L., et al. [1]. The features created are the following:

- **Duration:** This variable, which reflects the event duration, is derived by deducting the start time from the end time. It offers details regarding the duration of a specific task or activity.
- **Stride:** Produced by making the sum of 'AccV', 'AccML' and 'AccAP' variables. The variable represents the overall acceleration along all axes.
- **Step :** The 'Step' variable represents a modified version of the 'Stride' variable, taking into account the magnitude of the combined acceleration.
- **Moving average:** Three variables, 'AccV-ma', 'AccML-ma', and 'AccAP-ma', are calculated by taking the rolling mean with a window size of 2 on the 'AccV', 'AccML', and 'AccAP' variables, respectively. This helps to smooth out the data and capture trends over time.
- **Delta with time:** Three variables, 'AccV-delta', 'AccML-delta', and 'AccAP-delta', are computed by calculating the difference between consecutive values of 'AccV', 'AccML', and 'AccAP', respectively. It represents the rate of change or the incremental difference in acceleration values over time.

2.3 Feature Selection

To reduce the size of the data set feature selection was performed and the most important features were selected to train our model.

2.3.1 Tsflex

Tsflex was often used in other Kaggle notebooks and was achieving moderate scores. Tsflex is a Python library that provides efficient and flexible time series feature extraction. It offers a wide range of feature extraction methods and allows for easy customization and combination of features. By using tsflex, we were able to extract meaningful features from our time series data, contributing to the overall performance of our models. Our implementation was mainly based upon notebook no. 2 from Table 1 as a baseline.

We established in the code a default window size and stride of five seconds as this is a good chunking size according to the literature [2]. Furthermore, the usage of a window size of 5s allowed for a reduction in computational cost and memory usage. This was an important consideration for us as we encountered challenges related to computational resources and memory limitations. Two types of variables were created with tsflex:

- **Basic feats:** It represents a collection of basic features extracted from acceleration signals (AccV, AccML, and AccAP).
- **EMG feats:** It represents a collection of EMG features extracted from acceleration signals (AccV, AccML, and AccAP).

As said above, both features are computed using a window size of 5000ms and a stride of 5000ms. Then we combined the features into a single collection of feature descriptors for further analysis.

2.3.2 Important feature selection using Decision Tree

To gain insight from the most significant features within the dataset, we employed a decision tree algorithm to fit the data and extract the important features. In our analysis, we obtained the top 20 most important features, which are presented in Table 2. These features provide valuable information about the characteristics that strongly influence the outcome of our model. The implementation details can be found in notebook no. 8 from Table 1. This notebook contains the code and step-by-step procedures used for fitting the decision tree model and extracting the important features.
By employing decision trees for feature selection, we aimed to enhance our understanding of the dataset and identify the most informative variables as decision trees are easy to interpret. These important features should serve as the basis for our subsequent analysis and modeling steps, enabling us to develop more accurate and robust predictive models.

2.4 Classification

In the existing literature, there are a few different approaches that are commonly used for event detection in motion-based data. Most of them are in one of the following three categories: Deep Learning, tree-based, and classification. For this project, we decided to explore the option of decision trees for feature extraction [3] in combination with a Support Vector Machine (SVM) according to the paper by Borzi et al. [1]. This specific approach was chosen because the paper thoroughly analyzes different approaches with the result that this combination performs very well.

2.5 Notebooks

In Table 1 you can find submitted notebooks that contain the main implementations:

index	Notebook name	Description of added features	Private score	Public score
1	Gait Prediction - Ver. 1	Baseline 1	0.237	0.289
2	time-series tsflex	Baseline 2	0.181	0.237
3	Fast LGBM-GroupKfold tsflex+Memory optimization	Baseline 3	0.230	0.263
4	EDA_FoG_MLiP_Group18	Simple EDA	-	-
5	Gait Prediction Different Seed - Ver. 3	EDA on numpy seed, set fixed random_state	0.227	0.255
6	Gait Prediction KFold Train Interactive 2 - Ver. 1	GroupKFold Train-Valid split, data persistence	0.231	0.252
7	Gait Prediction KFold Samplerate Train Interactive - Ver. 12	Above + variable samplerate	0.239	0.261
8	GroupKfold Cross-Validation tsflex	Implement features from [1]	-	-
9	LGBM-GroupKFold tsflex+Memory optimization - Ver. 12	Other LGBM-Model, added own features, memory reduction, local test	0.229	0.253
10	Reducing DataFrame memory size by 65%	Make the dataset smaller without losing information	-	-
11	Feature selection + svm	Tried to train a SVM classifier	-	-

Table 1: **Reference to the main notebooks used.** Red scores indicate notebooks that were not submitted by us.

We picked the baseline model because it was the best performing public notebook. It implements a lightGBM multi output regressor that was tuned on a few hyper-parameters and achieves a score of 0.283. It includes the basic seglearn features (described above) using the tsflex library and creates features by combining the accelerometer data with the other data tables. The first improvement to the baseline was achieved by chance when rerunning the notebook, resulting in a public score of 0.289. This led to an exploratory analysis into the effects of random factors on the public score. For this we created a new notebook (No. 5, Table 1) where we first looked into the effect that changing the seed has on the build of k-means-cluster and eventually the score. The observed effects on the public score were not big but could be of greater impact in a more complex model. We therefore decided to also fixate the random_state in the LGBMModel to be able to fully eliminate random effects and be able to better compare different models while developing, even though that is not recommended to do, as boosting does not work correctly anymore when there is no sampling [source].

Through the discussion forum we found out that tackling the class imbalance should be the hardest challenge of this task. Therefore, a smart cross-validation (CV) strategy should be chosen, to be able to generalize the model well. We also realized that the provided train-test split does not really work to evaluate the models properly because the test files did not include all labels which lets the MAP metric fail. Therefore, the focus was then on implementing said strategy.

In notebook no. 6 we therefore implemented a Group5Fold Train-Validation-Split on the subjects. By storing the folds as a parquet file on the kaggle file system and loading it before training we achieved to implement a baseline which we could use to compare different models on. Notebook no. 7 is the first variation regarding pre-processing that accounted for the different sample rates of the tdcsfog and defog datasets by dividing by the magnitude of gravity on the tdcsfog dataset. We found that better local CV scores were not a really good indicator for a better public score. At the same time we implemented our own time-based features derived from Borzi et al. [1]. In notebook no. 8 you can find how the features stated in the "Pre-Processing" chapter have been created and combined. The same notebook implements the decision tree for feature selection in order to obtain the most valuables variables.

Based on these newly implemented features we wanted to compare the architecture proposed in [1] with the baseline lightGBM model. The implementation for an SVM based on the foundation of notebook no. 8 can be found in notebook no. 11 from Table 1.

Notebook no. 9 represents the most advanced of the notebooks that we submitted. Here we added our own features to the Baseline 3 model which is a lightGBM implementation similar to Baseline 1. We added a local test strategy to this notebook which made sure we had a fixed 5-folded training + validation set but also a big enough test set

that ensures to have enough variability to be a good generalization predictor. We figured that subjects can not only be clustered by their features stemming from the data in the subjects data table. Indeed, FOG is a rarely occurring event and patients experience in a different amount of severity. This means there a patients that never have any FOG event, there are some that only experience it while turning and some experience all three types of FOG events (Start Hesitation, Turn, Walking). Therefore, we grouped the event data by subject and time series id. We then counted the occurrences that each patient has annotated for each of the FOG types. We did aggregate the time globally but also counted the occurrences in different time series. We did not manage to implement this as a feature but manually created a test set from this data picking out subjects so that:

- The test set holds at least five percent of the time series
- All the labels occur in the test set
- The most distinguishable subject types with (0,0,0) and (n,m,k) [1] labels are well represented.
- The time series type (0,0,0) is most prominent as it makes out most of the data.

3 Results

The results of the feature selection can be seen in Table 2. The table shows the top 20 most influential features sorted by feature importance according to the decision tree model.

Feature	**Importance**
AccAP mean w=5000	0.132030
AccV slope sign changes w=5000	0.082534
AccAP ma	0.070561
AccAP maximum w=5000	0.051215
AccML delta	0.049009
AccAP	0.048363
AccML ma	0.041627
AccV delta	0.039900
AccV kurt w=5000	0.038747
AccAP delta	0.038654
AccV ma	0.036420
AccML	0.035404
Stride	0.030452
AccV	0.030284
AccV minimum w=5000	0.027409
AccV root mean square w=5000	0.026370
AccAP waveform length w=5000	0.018873
Step	0.017878

Table 2: **Selected features.** Ordered by importance according to a DecisionTree model. Where "w=5000" refers to the window in milliseconds.

We furthermore managed to implement a good cross-validation strategy, additional time-based features and made an effort to implement different models based on decision trees and gradient boosting. Unfortunately, we were not able to implement a model that achieved a higher score than the Baseline 1 model which had a public score of 0.237 and a private score of 0.289.

4 Discussion

Our first approach of implementing a SVM didn't work because of execution time restraints. That is why it was decided to stick with the LGBM classification model. As can be seen from Table 1 all the notebooks reduced their performance on the private score which shows that the baselines already were not good at generalizing. The better performance of the baseline model over the same model with another seed (private score of 0.289 vs. 0.267) can be explained by the fact that the baseline hyperparameters were tuned on the exact features that were produced with the underlying seed. Therefore, when you change the seed but do not tune the model again the performance can be significantly worse. We realized fixating the random state of the boosted tree probably was not a good idea

[1] n,m,k representing the count of time series of the subject that report this event at least once, where m is biggest, as the turn label is the most prominent in the data

instead multiple experiments (an ablation study) should be executed.
We still tried to implement a RandomizedSearchCV for hyperparameter tuning in this notebook which turned out to be very challenging because of the combination of MAP evaluation and multi class prediction.

5 Conclusion

After investigating many different approaches we can conclude that the detection of freezing of gait (FOG) events is a challenging task as can be seen from the overall scores on the leader board, the best one of them being 0.514, which barely achieves to predict 50% of the events correctly. This may also have been because of the data, which contained a considerable amount of ambiguous (valid=false) or unannotated (task=false) samples.

As future work, we could try more deep learning based strategies such as LSTM, RNN or CNN. Some more research on improving our initial approach, specifically on feature selection could be done, to find more or new features. Moreover, a better pre-processing and implementing a proper data loader could be interesting to avoid memory issues.

6 Author Contributions

- **Marta:** Explored Kaggle notebooks and interesting literature for the selection of our approach, data exploration, and helped with decision tree implementation.
- **Steffen:** Explored Kaggle notebooks, and did some data exploration, feature extraction, and SVM implementation.
- **Johannes:** Literature search, searched discussions for helpful topics, data exploration, implementation of cv-strategy, data persistence, lightGBM variant notebooks and hyperparameter tuning
- **Guillem:** I explored Kaggle notebooks and did a few data exploration. After that I've created the variables with tsflex library and the ones that are exposed in the preprocessing . Finally, implemented the decision tree algorithm for feature selection.

7 Evaluation of the Process

For this second challenge, we still got some organizational issues. We started by doing an extensive overview of existing literature and researching which techniques are usually applied to this kind of analysis. Once, we had a clearer understanding of how to process the data, and which kinds of machine learning algorithms would work best, we divided the tasks into: EDA, pre-processing, decision tree implementation, and SVM implementation. So, each member mainly carried one of them. Although we divided the work earlier than in the past challenge, we still struggled to communicate each other's advances, and since the progress of some influenced the progress of others we lost some time, which could have been useful for improving and/or fixing the current implementation. Even though we did not find the same troubles as in the past challenge with setting up the cluster, since submitting to Kaggle is pretty straightforward, we did encounter memory overload problems. A big struggle was working without code completion while trying to implement models with yet unknown libraries. Achieving data persistence with the kaggle file system was also a challenge.

8 Evaluation of the Supervision

The supervision that we received was helpful, when asked for it, even though we did not ask for much. It would have been helpful to have some insight into whether our approach was appropriate. Or some suggestions about how to improve it or fix, for example, our SVM implementation.

References

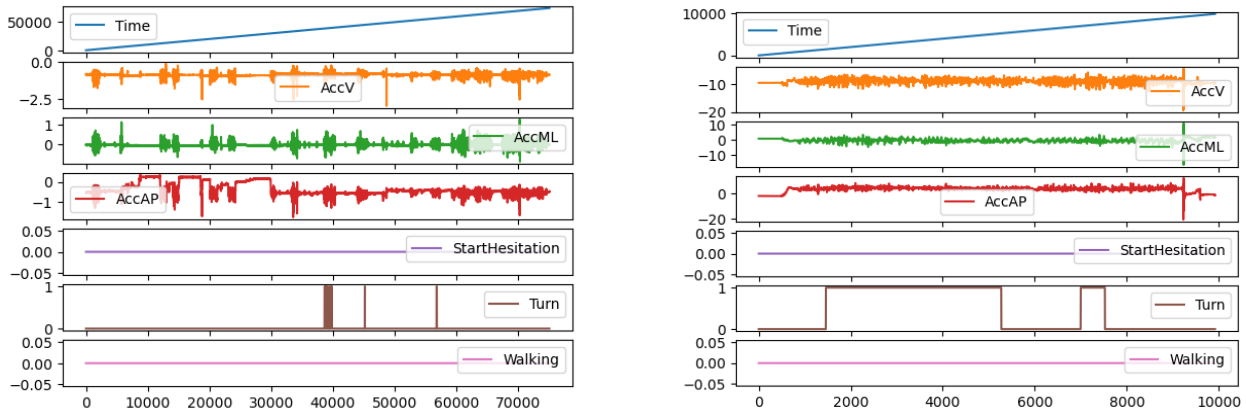
[1] Luigi Borzì, Ivan Mazzetta, Alessandro Zampogna, Antonio Suppa, Gabriella Olmo, and Fernanda Irrera. Prediction of freezing of gait in parkinson's disease using wearables and machine learning. *Sensors*, 21(2):614, 2021.

[2] Pavón I Costa S Arezes P López JM De Arcas G. Sigcha L, Costa N. Deep learning approaches for detecting freezing of gait in parkinson's disease patients through on-body acceleration sensors. *Sensors(Basel)*, 20(7):1895, 2020.

[3] Senden R Heyligers I Grimm B. Lipperts M, van Laarhoven S. Clinical validation of a body-fixed 3d accelerometer and algorithm for activity monitoring in orthopaedic patients. *J Orthop Translat.*, 11:19–29, 2017.

Appendix

A EDA



(a) Instance from the defog time series.

(b) Instance from the tdcsfog time series.

Figure 2: **Depicts two random instances from the two different datasets** (excluding "daily_metadata.csv"). Displaying the waveform of the three different kinds of accelerometer data (AccVvertical, AccMLmediolateral, and AccAPanteroposterior), as well as the three different labels (Start Hesitation, Turn, and Walking) and their distribution along time.