Machine Learning Engineer Nanodegree

Capstone Project

Predicting heart rate dynamics and internal training stress from cycling data

Ruud Goorden
January 20th, 2021

I. Definition

Project Overview

Introduction

In the sport of cycling powermeters are important instruments that are used to guide the training process and track performance progression. By using a powermeter the actual mechanical work can be monitored by the coach and rider by analyzing individual rides and derived training load metrics.

Before powermeters were commercialized training with power was only accessible for professional cyclists. Non-professionals had other ways to setup training programs and training guidance. For this case heart rate prescription was often used in training prescriptions.

Heart rate of an individual varies with the intensity of the exercise in a nonlinear manner, and understanding the relationship between heart rate response and exercise intensity can be of interest to coaches and riders. In addition to being an indicator of response to exercise intensity, heart rate is also related to other physiological responses such as oxygen consumption, energy expenditure, and cardiac output. Therefore, it is useful to also take into account heart rate responses for training prescription.

Coaches and riders can benefit from predicting heart rate dynamics before a ride (training) takes place. This insight could be used for planning purposes further downstream when multiple rides are simulated and can help to estimate training stress in advance. Moreover predicted heart rate dynamics could ex-post be compared to realized heart rate dynamics which might give further insight in the rider's physical condition.

Problem Statement

Modelling individual heart rate response to a workout is a challenge since many variables play a role. Environmental variables have an influence like i.e. temperature, humidity and altitude (hypoxic conditions), inclination and surface (e.g. cobbles vs tarmac). Moreover nutritional variables in the refueling strategy play a role. Examples are hydration, food / carb ingestion, caffeine ingestion etc. In addition during a ride mechanical factors like cadence (better said angular velocity) for generating power plays a role. Riding at a lower cadence tends to put more stress on the muscular system instead of the cardio-pulmonary system which tends to lower heart rate for the same power output. Riding up a mountain will recruit muscles from in the upper body which need oxygen and therefor heart rate can increase. Also changing breathing rate for the same workload can alter heart rate.

As said next to factors inside the ride also factors outside of the ride play an important role. Some of these are also environmentally related and nutritionally related like during a ride. Dynamics can differ for example for a well fueled rider versus a rider which is not well-fueled. Also the riders diet (lots of carbs / fat) can have an influence. The degree to which the rider is adapted to these diets (factors in general) also play a role. Moreover the training status of the rider plays an important role. Heart rate response after a period of intense training which is possibly accompanied by fatigue which gives different dynamics than compared to when the rider is ''fresh''.

Mental aspects can also play a role. Arousal of the rider before a certain interval starts can have an influence, but also when the rider experiences stress (from other life factors) this might alter the heart rate dynamics during the ride. In addition certain type of medication (e.g. beta blockers) can influence heart rate dynamics.

From an operational perspective modelling on the type of data (second-by-second) means modelling on a lot of individual data points and the size of the dataset can rapidly increase. It is not a coincidence that modelling directly on this data level is not (widely) implemented in commercial cycling software since there is quite a computational burden. Usually models available in those applications are based on aggregate data inferred from individual ride data. This can bring some additional potential problems however, since often users cannot judge the effect anomalies have on the aggregate metrics which are used in those models.

The goal of this project is to *predict heart rate dynamics* and *training stress before* a ride takes place

So, before a ride takes place a coach / rider can get an insight in predicted heart rate dynamics and evaluate internal training stress. This can aid in training and workout planning strategies. Also these predictions could be used ex-post to compare it to actual realized heart rate dynamics and training stress. If necessary future workouts might be adapted given the predictions.

Since cycling performance and physiology is related to an *individual* we construct an *individual* level model for *heart rate dynamics* and *training stress* calculation by using cycling power data and other relevant data which can be pre-planned or can be made available before rides take place.

Metrics

We will evaluate the heart rate dynamics model and training stress models based on the total RMSE and RMSE distributions of the rides in our validation data coming from a period after the model building period (out-of-time sample).

We chose RMSE as metric since we want to penalize both low and high deviations and penalize according to the deviations instead of averaging. We specifically also assess the RMSE distribution of the rides since a bad predicted ride can be compensated by a good predicted ride and on average perform well on total level, which we won't know if only looking at the total RMSE. Calculation of RMSE is shown in Figure 1. For each of the rides in our out-of-time sample we can generate these calculations and generate a distribution, next to an overall out of time sample RMSE.

Figure 1 RMSE Calculation

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

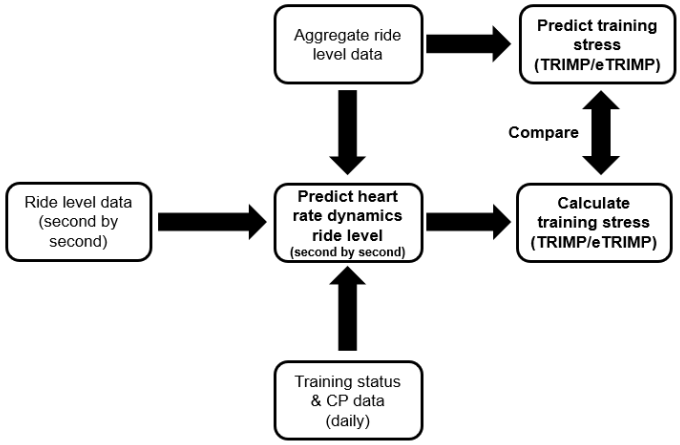
$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values
$y_1, y_2, \ldots, y_n$ are observed values
n is the number of observations

Also we will show predictions visually to get a feeling on the dynamics of the predictions for both well predicted rides and less good predicted rides to find directions for further improvements.

Training stress for a session can be expressed a.o. by several heart rate related metrics. TRIMP (TRaining IMPulse) [1] and eTRIMP [2] are examples used for this purpose. Both metrics are related to internal load, or stress. TRIMP is based on the intensity of the exercise as calculated by the product of the average percentage of heart rate reserve and the duration of exercise. eTRIMP is calculated as the product of the cumulated training duration (in minutes) for 5 heart rate zones multiplied by a coefficient relative to each zone. In addition to the ride level heart rate dynamics models we will evaluate the total aggregate heart rate load metrics (TRIMP and eTRIMP) based on RMSE.

Figure 2 shows a schematic overview of the modelling process for heart rate dynamics and predicting training stress.

Figure 2 Modelling framework



First we will develop models for predicting heart rate dynamics. From these predictions we can calculate training stress related metrics. We will also estimate a simple training stress models from aggregate data and compare the performance of these to the (aggregate) predictions of the final heart rate dynamics model. More information on the data will be presented in the next section.

II. Analysis

Data Exploration

For our project we have three data sources. These sources are data exports in .csv file format from an open source cycling software tool called Golden Cheetah (www.goldencheetah.org). These three data sources relate to: 1) individual ride files, 2) aggregate ride metrics, 3) training load and stress metrics and estimates of the rider's "threshold" (Critical Power).

The individual ride files contain several variables measures second by second which all come from a cycling head unit. The cycling head unit registered data from several sensors on the rider's bike, more specifically data coming from a power meter (watts and cadence), heart rate (heart rate strap), speed (sensor on wheel / GPS) and temperature (build in sensor in headunit). Moreover the individual ride files contained data fields which are extra fields if a rider had extra sensors like for muscle oxygenation. The fields considered from the individual ride files in this project are:

- seconds (time riding)
- heart rate (beats per minute)
- power (watts)
- cadence (rate per minute)
- temperature (Celsius)
- altitude (meter)
- slope (percentage)

Speed was not taken into account. The reason for this was that it was often not registered but -more important- as a variable speed is less useful for predicting heart rate, because it is an indirect way of measuring effort. This effort is expressed by power (watts). Imagine riding 40 km/h with headwind or 40 km/h with tailwind. No other variables were measured.

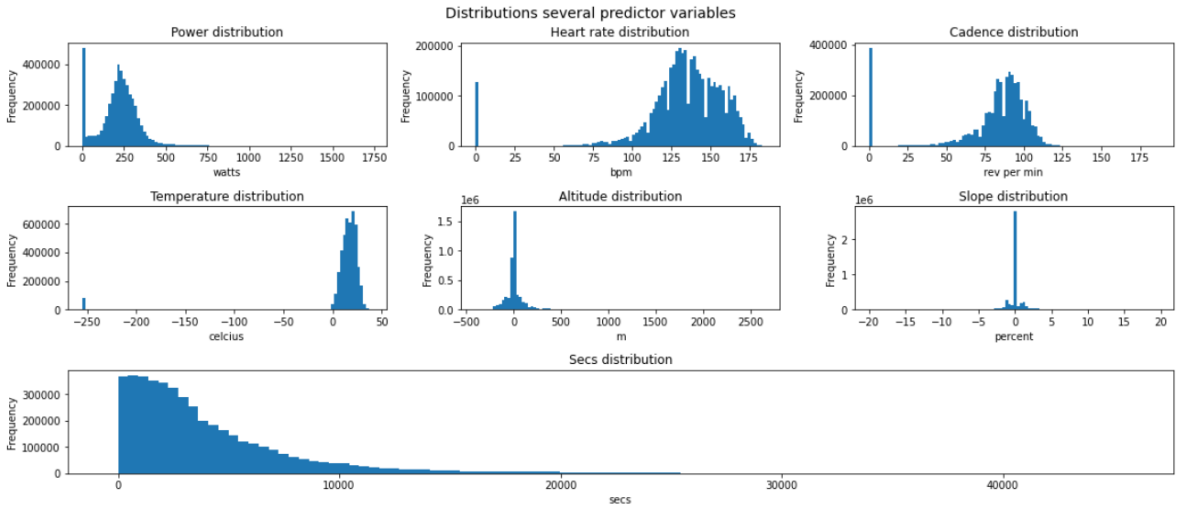
The aggregate ride data had several metrics available with simple averages from the ride files and cycling specific domain variables like for example XPower and BikeScore [3]. The training load and stress metrics (data on daily level) reflected the training status of the rider both from an mechanical point of view (external load, wattage derived) as well as from a physiological stress point of view (internal stress, heart rate derived). In addition we had estimates of the rider's power "threshold" (Critical Power) on a daily basis. In total we had 839 ride files covering a time span of 3 years (January 2017- December 2020).

Exploratory Visualization

We explored univariate statistics and bivariate relations for the variables in this study. In **Figure 3** the distributions from the individual ride data are displayed.

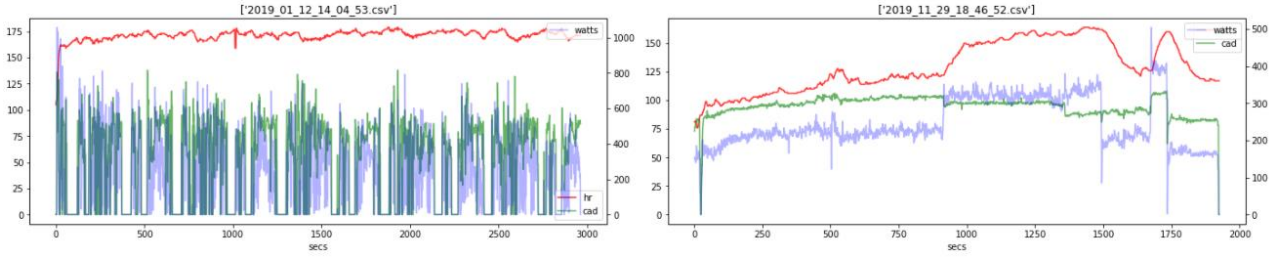
Already some possible outlier points / anomalies can be seen at for example temperature (-255) and heart rate (0). Most of the rides come from riding on the flat and only a few rides contained rides in a mountain environment.

Figure 3 Variable distributions



The data comes from ride files. **In Figure 4** two examples of those ride files with sensor data from heart rate (hr), power (watts) and cadence (cad) are depicted.

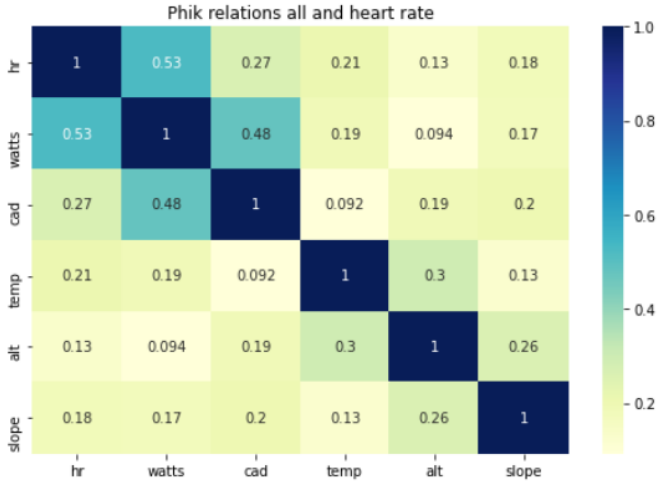
Figure 4 example ride plots



As can be seen these ride examples show different characteristics. The ride on the left shows much more variable power output and cadence while the right ride in the figure has much more stable patterns. Rides show clear differences between structured rides and unstructured rides. Examples of unstructured rides are for example races in which the rider accelerates, slows down, etc. due to either his/her own intention or for example by environmental conditions (hills) and peloton dynamics.

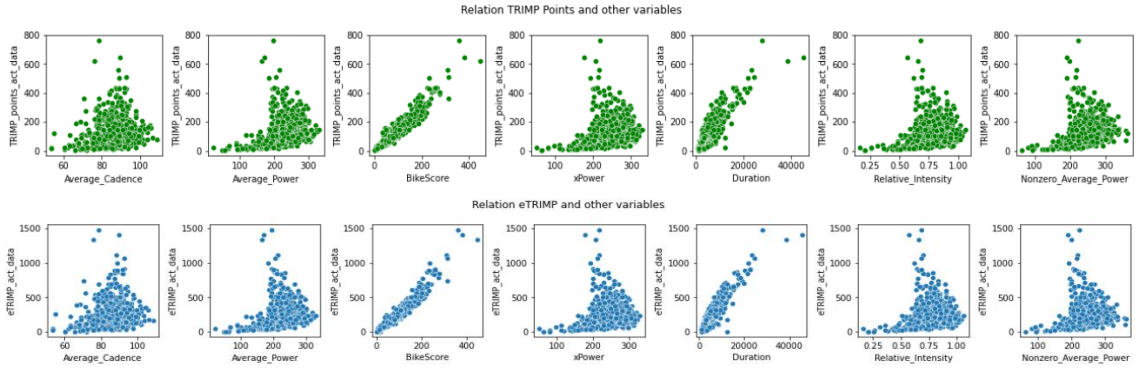
Figure 5 shows bivariate Phi_K relations [4]. Phi_K can take into account non-linearity while correlations look at linear relations. We see that power output (watts) seem mostly related to heart rate. Power and cadence also see related (for example when not pedaling, power output is zero).

Figure 5 Phi_K relations



For aggregate data **Figure 6** shows bivariate relations between variables considered for predicting training stress (TRIMP).

Figure 6 Distributions aggregate ride data



From these plots it can be seen that BikeScore (metric of ride training load) could be a strong linear predictor variable for predicting training stress. More data exploration and visualizations can be found in:

Notebook : 0 Data exploration.ipynb, 5 Training stress calculations.ipynb

Algorithms and Techniques

To model the heart rate dynamics our main source is the ride level second-by-second data for multiple rides. Each ride has its own characteristics and length of the rides differs. Our goal is to derive a model to predict heart rate at a certain point in the ride giving the second-by-second features of that ride as an input *before* the ride takes place. We formulate our problem as a regression problem in which we want use features to predict the heart rate (continuous variable) at a certain point in the ride. Although there is a time element in the rides, it is not really a classic time series problem since we don't want to predict from a certain point x periods into the future. As such we did not investigate methods in this field further, but concentrated on methods from a regression point of view.

Features we generate will ''fit'' the setup for the regression problem which results in a tabular data format with rows representing the second by second rides and columns the (derived) features. In addition on ride level we merge the aggregate ride features. On a daily level we merged the information regarding the training status from an internal stress (heart rate) and external load (power) perspective along with the rider's modelled "threshold" (Critical Power [5]).

For our (semi-)final model iterations we use linear (regularized) methods and tree methods (normal, boosted (XGBoost, LightGBM), bagging (RandomForest)). We also considered using feedforward neural networks (multilayer perceptron architecture), but the previously mentioned methods also have proven to be able to generate good performance on tabular data, especially also when using hyperparameter tuning. In addition we considered the models dealing with linearity and non-linearity (and interactions) to be sufficient.

We model rides which all have a specific "footprint". That is, points in the ride have dependency. To account for this we used a GroupKFold cross-validation method to take this characteristic into account.

Benchmark

At the date of writing we did not find a benchmark in literature which studied our problem in the way we do in combination with the amount of data, the field setting and not on the methods used. One study which resembled our study most is the study of Hilmkil et al [6] in which they used sensor data from 15 professional road cyclists from field conditions. They also i.e. used power and cadence, but included heart rate 30 second prior to predict heart rate at time t. From the LSTM architecture they got a mean RMSE of 5.62. So they modelled heart rate dynamics *during* a ride and *not before* the ride takes place. Other studies which relate to our study but differ in setup, (amount) of data used and conditions (lab environments) can be found in our literature overview [7,8,9,10,11].

III. Methodology

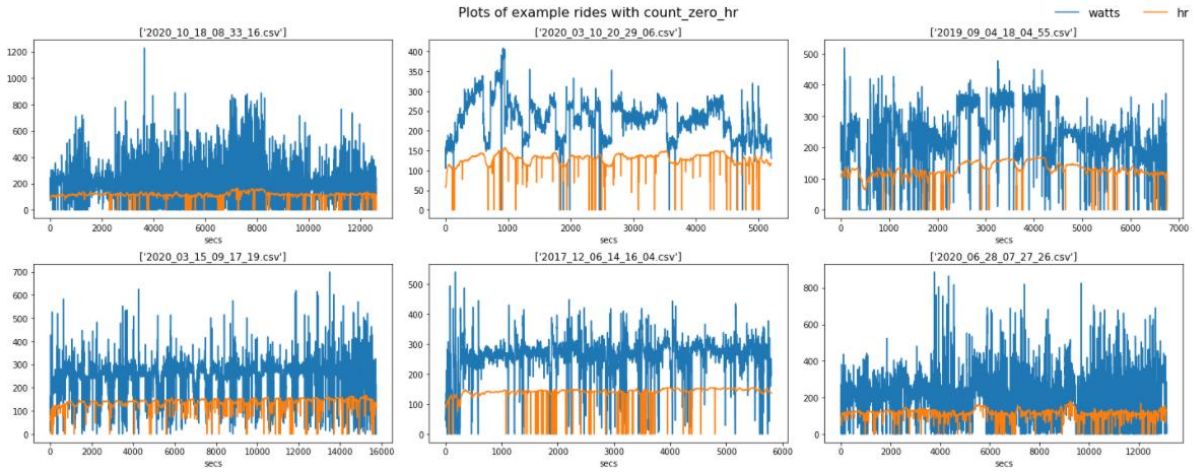
Data Preprocessing

Data cleaning

Before our modelling could take place we had to preprocess the data. From the data exploration we already got some initial insights into potential data problems and anomalies we could expect and that had to be addressed before modelling.

The ride data contained all kinds of different challenges. Sometimes rides did not contain our target variable heart rate or one of the predictor variables. Also issues rose related to sensor dropouts, like temporary dropouts, full dropouts, but also partial dropouts. In addition we had to address other unwanted data parts of the rides, like (long) breaks in which there was no training (ride) activity. Moreover periods occurred in which the sensors reported a complete stable pattern. Identifying all the anomalies and dealing with them resulted in an iterative process with continuously visualizing results of the data cleaning steps and correcting accordingly. Some knowledge on the rider's physiology also helped identifying some of the anomalies. **Figure 7** gives an example of 6 rides with a sudden full or partial drop in the heart rate (hr) sensor.

Figure 7 Example of anomalies



We also made decisions in this process to remove rides which still showed bad data or became very short in time due to the cleaning steps that were taken. In this whole process we used a combination of rules and algorithms. This process meant that we ended with 770 ride files out of 839.

Notebook: 1 Data procesing.ipynb

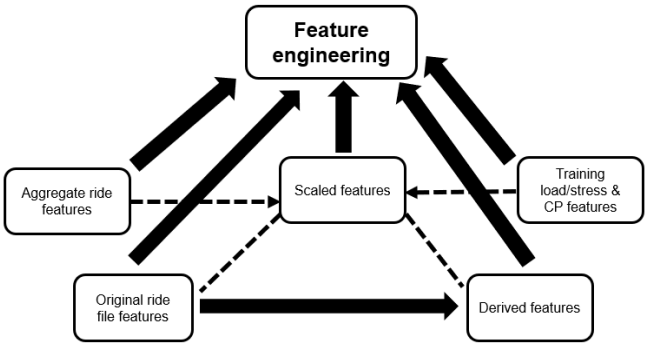
Related .py files : ride_stats_calculation.py, graph_ride_anomalies.py, dash_chart_rides_interactive.py

Feature engineering

From the ride files we had second by second data from different sensors which from which we used the following features: heart rate (hr), power (watts), cadence (cad), altitude (alt) and slope (slope). (temperature was not selected see section **Feature selection**) . From these features we derived new features which were mainly based on, lags, simple and exponential moving averages in several time windows up to one minute. Heart rate is a much more stable signal than power (watts) which is much

more stochastic in nature. Moreover heart rate response lags (mechanical) load increase or decrease. Our feature engineering process is depicted is **Figure 8**

Figure 8 Feature engineering framework



In addition we created some extra feature using log/sqrt transformations and features tied to acceleration. Next to these metrics we implemented XPower by replicating the implementation of Clarke and Skiba [3]. In addition we calculated (using the same methodology) the same metric but then using a 30 second time window instead of 25. This 30 second window represents the underlying window which is used in Normalized Power [12].

We included aggregate ride file metrics. These features were related to metrics like duration of the ride, time of the day of the ride, averages of some of the metrics mentioned above and ride derived load and stress metrics like BikeScore and Relative Intensity [3].

Next to this we had data on the overall training load of the riders. These were related to ATL (Acute Training Load), CTL (Chronic Training Load), and TSB (Training Stress Balance) based on power (watts) and heart rate. These features are derived from rides overtime. For more information on how these features are derived see our literature section [3,12].

Because heart rate dynamics respond a.o. to overall capacity to perform of the rider we also created scaled features related to the actual ''threshold'' (based on watts) of the rider at the time of doing the rides. The idea behind this is, is to correct for the fact that overtime the rider is able to produce more watts (or less watts) at a different heart rate, but that this is tied to the percentage of the actual threshold of the rider at that time. As an example at time t a rider can produce 300 watts having a threshold of 350 watt at heart rate 140. So the heart rate is 140 at 85% of his ''threshold'. At time t+1 , as the rider gets fitter, he is able to produce 320 watts having a threshold 373 watts at heart rate 140. By comparing the watts at time t with time t+1 we would think heart rate would be higher than 140. But we see that in the new situation he is also riding at the same percentage of his threshold (85%) at a heart rate of 140. As the threshold moves also the accompanied heart rate moves (either downward for the same watts as he gets well-trained or upward when less well-trained).

The features were created after the data cleaning process had taken place. In total we had 394 features for our feature selection step.

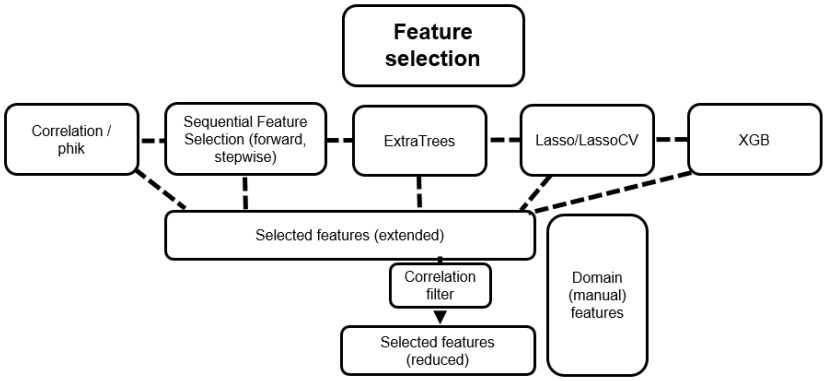
For our aggregate data model we did not apply further feature engineering since from the data explorations we already got a good idea on applying a simple modelling technique to obtain a baseline result to compare with results from the heart rate dynamics model.

Notebook: 2 Feature engineering.ipynb, Additional Replicating xPower Python code.ipynb

Feature selection

In this step we applied several feature selection steps to get insight into important predictors. This process is depicted in **Figure 9.**

Figure 9 Feature selection framework



Before the feature selection we split the dataset into train and test sets based on *ride level* not datapoint level. This is done since within a ride datapoints have a dependency which we want to keep and -in a later modelbuilding process- not leak into our evaluation. We chose to apply some of the feature selection methods only on half of the dataset, which was then split 75% in train and 25% in test. The main argument for this was performance (speed).

We balanced the argument of using only half of the data for this procedure against the fact that we apply a variety of methods and if we slightly stretch the chosen amount of features in each of the methods the probability of not selecting important features is estimated as low.

Bivariate metrics and model driven methods were used. Correlations give insight into linear relations between target and features. Phi_K gave insights into bivariate non-linear relations since the relation between a feature and the target maybe non-linear which is not shown using correlations.

Sequential feature selection methods were used to evaluate which features -if sequentially added- would be picked and to assess to what extent we would see a performance increase (i.e. decrease in RMSE). Both forward and stepwise (floating) showed that using a linear regression model, model performance stabilized after approximately 30 features and then very slowly improved further: meaning you have to add a lot of features to get a slight performance increase. Note this iteration graph was only for a training set.

We also added a linear model -which is sometimes used for feature selection as well- based on regularization. By implementing Lasso(cv) we obtained further potential candidates for further model development. Next to linear regression this method only use main effects and not interaction effects if not explicitly stated.

Two methods which take into account non-linearities and interaction effects, namely ExtraTrees and XGBoost. There are a variety of ways to select the features from tree based algorithms, like based on amount of splits the feature occurs, information gain, Shapely values, permutation importance, etc. , but we chose to used information gain as mainly based on feasibility.

The insight from the sequential feature selection method, meaning at around 30 features performance seemed to stabilize, was also carried forward to selecting features from the other models. Each method

came up with overlapping and non-overlapping features. In all we determined a cutoff of 50 features as maximum for each of the methods.

For each of the feature selection method we used on the train set, we predicted on the test set using the features from the models itself just to get a rough feeling of performance potential. For the Lasso(CV) results we additionally ran two linear regression models. **Figure 10** shows the RMSE distribution for each of the rides in the test set. We also included a graph to get an idea on R2.

ExtraTrees seems to perform best on average, but -apart from the fact that this variant still had all features in the model- please note this is only on a small part of the dataset and a one-off train-test split which has an effect on model performance.

Figure 10 RMSE distributions feature selection methods

Combining the results of the method lets to 141 unique features. We call this feature set the ''extended'' feature list.

Since some of the features that were in the extended list were highly (cor)related we decided to take another step to bring the number of features further down. For this we applied a correlation filter. For each pair of the features which had a correlation of more than 0.9 we selected the feature which had the highest correlation with the target. This brought the number of features down to 54. We call this set the ''reduced'' feature list.

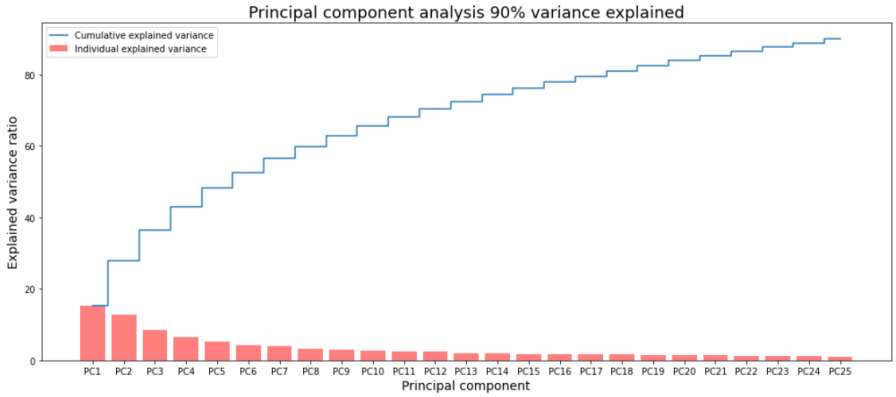
As a final feature set we manually selected features based more on domain specific knowledge and balancing knowledge about the data driven results. This method could be seen as a bit more subjective but could bring some more ''face validity''.

In summary we went forward with 3 feature sets: The extended list coming from the combination of the feature selection methods, the reduced list based on the extra correlation filter on the extended list and a domain, manual selected feature list.

We decided not to include temperature during the rides. We have 31 files which had no temperature recorded. Also from inspecting some of the rides temperature sometimes showed very unreliable patterns. This does not mean we do not see this as a possible predictor of heart rate dynamics, but merely that for reasons mentioned we don't use it here

To conclude we also did a PCA to get an idea on the dimensionality of our data. Results can be seen in **Figure 11**. Around 25 factors could explain 90% of the variability in the features. We did however not use results further in modelling, but focused on the created feature lists.

Figure 11 Principal component analysis



Notebook: 3 Feature exploration and selection.ipynb

Implementation

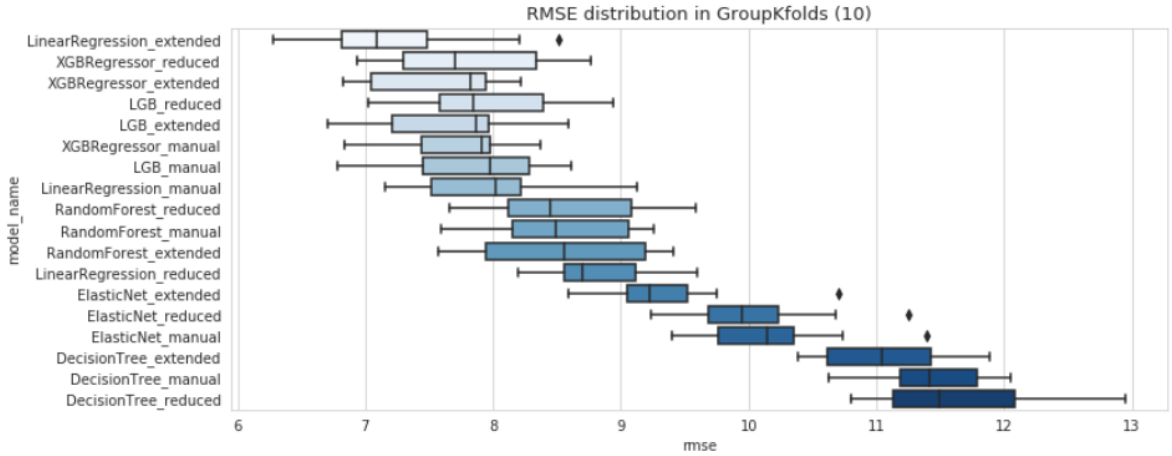
Our process of initial heart rate dynamics model building is characterized by applying GroupKFold cross-validation using different algorithms on a part of the data which was split for training. We split our total dataset in two based on a time period. The last three months will be used as our out- of-time validation set, while the period before (almost 3 years) was used as training set in which we applied GroupKfold. The reason for choosing GroupKFold is that when splitting the data into folds we want to keep the datapoints which belong to a certain ride within the same fold. With ''normal'' cross-validation we would split based on datapoints which will in this case ''leak'' ride characteristics of one ride in the train fold to the test fold. We believe we should take the relation of datapoints within a ride into account to not overestimate our performance metric. We used 10 folds in our procedure.

The algorithms selected are Linear regression, Elastic Net, Decision Trees, XGBoost, LightGBM, and Random Forrest. For the algorithms default settings were applied. The reasoning is that we first wanted to get an idea on generalization error (read RMSE) of the models and set a benchmark instead of already tuning hyperparameters for models with -possibly- more potential. As such this can be characterized as a "spot checking" algorithms procedure.

The choice of the algorithms came from the idea of evaluating a linear (and regularized) model with main effects versus models which deal with non-linearity and interactions. For the tree methods our idea was to use a simple method (Decision Tree) and ensemble based methods (XGBoost, LightGBM and Random Forest) where the latter be less prone to overfitting. Also we wanted to include both bagging (Random Forest) and boosting (XGBoost and LightGBM). Moreover these algorithms are very well suited for tabular data and have won (many) machine learning (e.g. Kaggle) competitions.

Before our initial runs we standardized the input space. Reason for this was mainly that Elastic Net's performance is impacted by standardization of the input. The other algorithms can deal with non-standardized input. Results of this procedure is depicted in **Figure 12**.

Figure 12 RMSE distribution "spot checking" algorithms



We see some interesting patterns in that our linear regression on the extended variable list actually performs quite well. After that we see the boosting algorithms on the extended and reduced variable lists appearing. The two types (XGB, LightGBM) don't differ that much in performance. RandomForest, ElasticNet and Decision Trees seem to stay behind.

For linear regression this was basically our result: there is nothing more to tune. Final evaluation for the linear regression (extended) will be done on the out-of-time set (last 3 months of data) when using the results of hyperparameter tuning on one of the other algorithms. We decided to first test the procedure of hyperparameter tuning for LightGBM. The reason for choosing this algorithm is based on the fact that it scored quite well initially compared to the other algorithms, its memory usage and its speed.

The cross-validation procedure was done on a AWS notebook instance (ml.c5.18xlarge), but note that it also takes around 5 hours to complete.

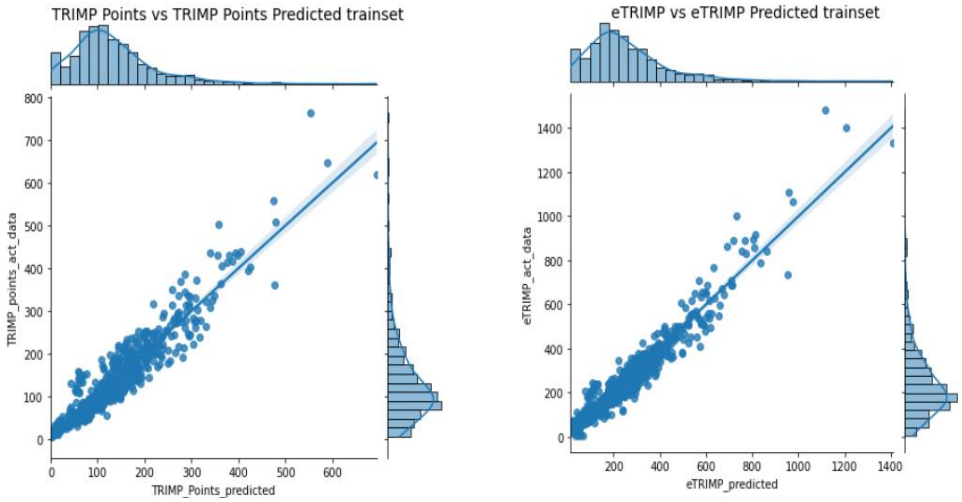
For our training stress model (TRIMP variables as target variables) we created a simple linear regression model which could serve as a baseline for comparison. Predictors were selected based on bivariate analyses (correlation and Phi_K). There were three variables in the models which related most strong to the aggregate TRIMP scores, namely BikeScore, Average_Power and Average_Cadence. **Table 1** shows the results. Note these are results on the *train* set. Performance on the holdout period follow on our Result section.

Table 1 Aggregate model results

Model	Total RMSE	R2
TRIMP	30.7	0.89
e-TRIMP	47.6	0.93

In addition we visualized the predicted scores and actual scores in **Figure 13**

Figure 13 Relation aggregate models actual TRIMP/eTRIMP versus predicted



Notebook: 4. Modelling.ipynb, 5. Training stress calculations.ipynb

Refinement

From our initial heart rate dynamics model implementations there was potential to improve. As stated we have chosen LightGBM to further develop by using hyperparameter tuning. To test the hyperparameter procedure we used Nested GroupKFold cross-validation. This is a special case of cross-validation in which we have an outer cross-validation and an inner cross-validation (which is nested in the outer cross-validation) for hyperparameter optimization model selection. We will use random search for this. One of the benefits of this whole procedure is that overfitting in the hyperparameter search can be mitigated since only a part of the dataset is used by the outer cross-validation procedure. This should provide a less biased estimate of the tuned model's performance on the dataset. A downside is time to process since more models are estimated. We will use 10 folds for the outer part of the cross-validation and 5 folds for the inner part. In addition we use 25 iterations for our random search. Per outer fold we then fit 125 models. In addition we used early stopping rounds (set to a relatively low value given the amount of models) of 5.

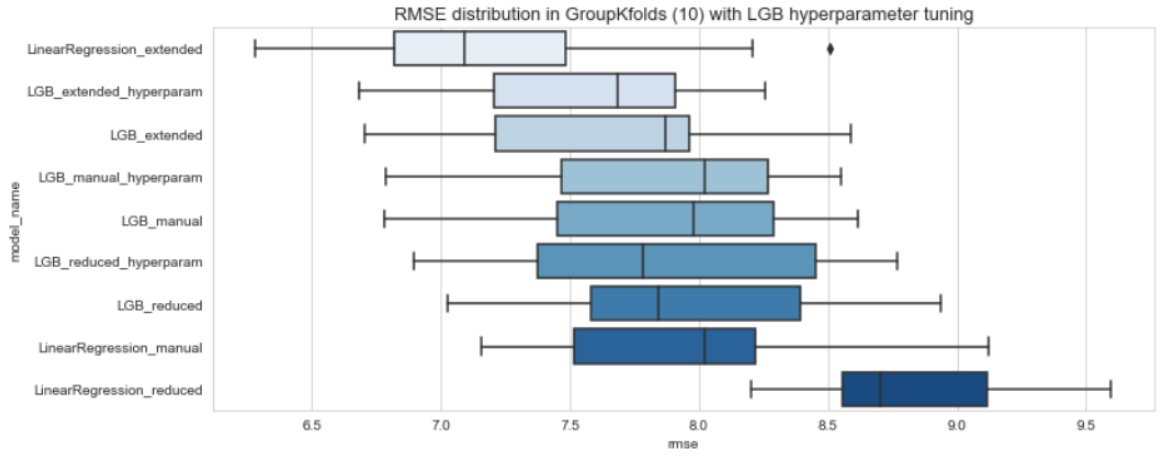
For our random search cross-validation with hyperparameter tuning we constructed a grid shown in **Figure 14.**

Figure 14

```
lgb_param_test ={'num_leaves': sp_randint(6, 50),
         'min_child_samples': sp_randint(100, 500),
         'min_child_weight': [1e-5, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4],
         'subsample': sp_uniform(loc=0.2, scale=0.8),
         'colsample_bytree': sp_uniform(loc=0.4, scale=0.6),
         'reg_alpha': [0, 1e-1, 1, 2, 5, 7, 10, 50, 100],
         'reg_lambda': [0, 1e-1, 1, 5, 10, 20, 50, 100]}
```

The results of the hyperparameter tuning procedure and the previous results (non-tuned together with the linear regression results) are depicted in **Figure 15**

Figure 15 Nested GroupKFold cross-validation with LightGBM hyperparameter tuning



From this we conclude tuning is *likely* to improve results, but only marginally. We chose to further evaluate two models namely the Linear regression with 140 variables (extended feature list) and the more parsimonious LightGBM with 39 variables (manual selected feature list).

We will use the hyperparameters which came up as best most often from the inner folds for each outer fold. Note that this is not a "normal" procedure. After nested cross-validation one "should" do the hyperparameter tuning again, but now on the complete training set. Another practice could be to create ensembles from the best nested cross-validation inner models. But since the inner model configuration seems quite stable we chose to use that hyperparameter result. This final configuration was:

- n_estimators : 80
- colsample_bytree : 0.90
- min_child_samples : 308,
- min_child_weight : 1e-05
- num_leaves : 46
- reg_alpha : 50
- reg_lambda : 0
- subsample: 0.93

We did not tune our aggregate TRIMP models since we had a relatively low amount of possible predictors and using the baseline model is good enough for comparison purpose.

Notebook: 4. Modelling.ipynb, 5. Training stress calculations.ipynb

Related .py files : lightgbm_hyperparam_nested_cv_proc.py

IV. Results

Model Evaluation and Validation

The final results for our two heart rate dynamics models are presented in **Table 2.**

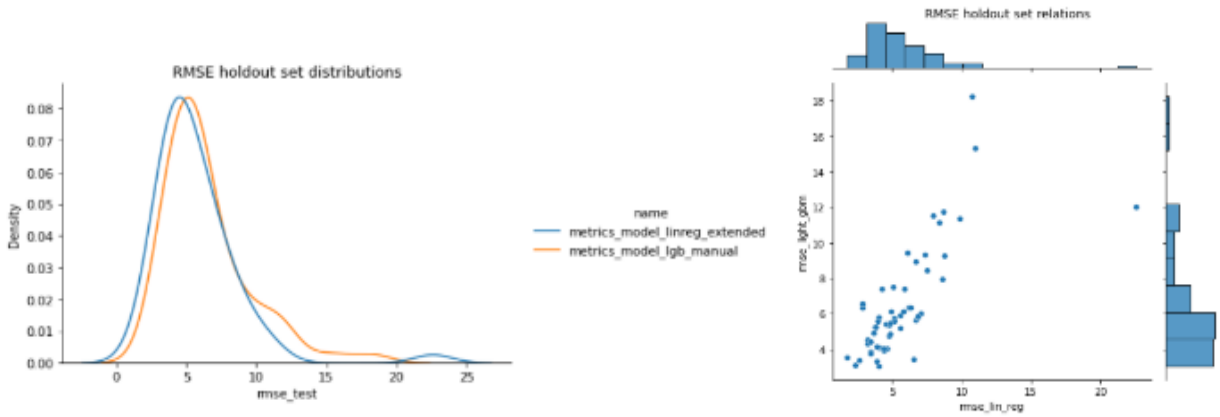
Table 2 Heart rate dynamics final model performance metrics

Model	Total RMSE	Mean RMSE*	R2
Linear regression extended (140 variables)	6.26	5.65	0.84
LightGBM manual (39 variables)	7.66	6.46	0.78

*Mean RMSE per ride

We see that from this table the Linear regression seems to perform best on our holdout period. In **Figure 16** we can see the distribution of the RMSE over the rides and what the relation is of the RMSE between the models.

Figure 16 RMSE distributions final models



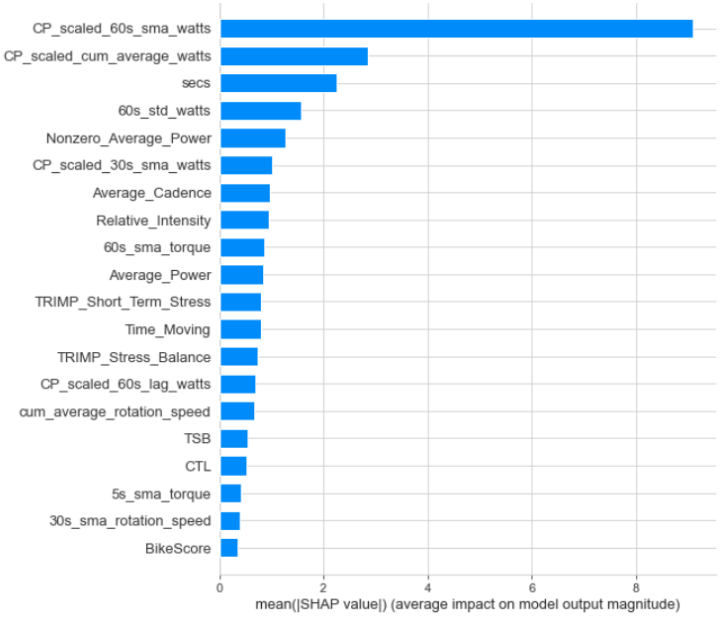
We tested whether the RMSE distributions were statistically different. Since we test two algorithms outcomes on the same data we used a Paired Sample T-test. With a P-value of 0.01 (T-statistic -2.6) we rejected the hypothesis that the mean difference between the samples is zero. The Linear regression extended shows statistically better performance.

The top 20 variables which were important for the LightGBM model predictions are shown in **Figure 17** using SHAP values [13]. The three most influential variables on the model predictions are the CP watt scaled 60 second simple moving average, the CP watt scaled cumulative average and time in the ride (secs).

For the linear regression it is a more "difficult" process to determine how the variables impact the predictions. Since the variables correlate (highly) simply interpreting (standardized) beta coefficients is not an option. Several methods exist to get an idea on feature importance like SHAP(ely) values, dominance analysis and relative importance analysis. All of these methods are very computational intensive so to get an indication on which variables are important we used the results of a stepwise procedure. If we look results from a stepwise selection procedure (*3. Feature exploration and*

selection.ipynb) we see the following 10 variables appearing (in order) CP_scaled_60s_sma_watts, Nonzero_Average_Power, sqrt_secs, Average_Power, CP_scaled_cum_average_watts, exp_cum_total_cad, Relative_Intensity, CP_scaled_NP_EMA, Average_Cadence and CP_scaled_60s_sma_watts. There is quite a **large overlap** between these variables and the variables in LightGBM which come up as important.

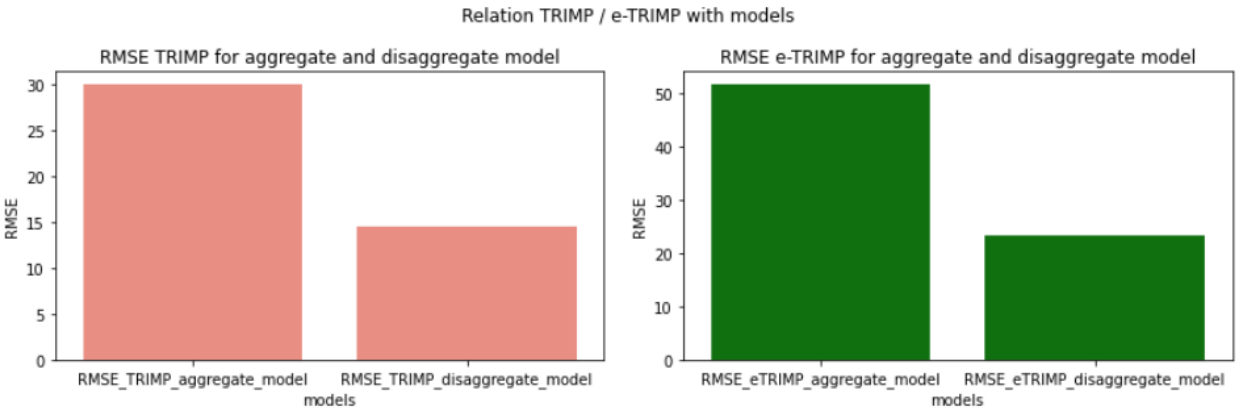
Figure 17 SHAP values LightGBM



To assess the model predictions on the level of training stress we calculated the TRIMP and eTRIMP scores for the rides in our out-of-time validation set and compared them to the actual scores. As a baseline we created a model using aggregate data. **Figure 18** compares the RMSE for the TRIMP and e-TRIMP using the aggregate models and the disaggregate (heart rate dynamic) model (linear regression). The heart rate dynamics model has *around half* of the RMSE as models on aggregate level.

To conclude: inferring TRIMP and e-TRIMP scores (training stress) from our heart rate dynamics model compared to the aggregate model resulted in *more accurate predictions*.

Figure 18 RMSE comparison heart rate dynamics model versus aggregate model TRIMP/eTRIMP



Notebooks: 4. Modelling.ipynb, 5 Training stress calculations.ipynb

Justification

The only benchmark for our heart rate dynamics model we had from a similar, but yet different study compared to the goal- was an RMSE of 5.26 Hilmkil et al [6]. Given the fact that we did not use (lagged) heart rate itself to predict future heart rate in a ride we believe that our results are not that bad given the RMSE's of around 6. Hilmkil et al [6] used deep learning methods (LSTMs). To us this is an indication that deep learning methods on an apparent time series related problem do not always perform better when transformed to a tabular problem and that using more complicated model architectures by default is not a necessary way to go.

For the linear regression model we had quite some variables in the model and also variables which are highly collinear. For explanation purposes using "traditional'' techniques (like interpreting beta coefficients) this causes problems. However, from a perspective of pure prediction under these circumstances (also large sample size, a lot of predictors) this does not have to be a large problem [14].

It can be debated whether the linear regression model should have been picked as 'best' model from a practical perspective to further evaluate using the TRIMP score calculations since the LightGBM (with about 1/3rd of the regression variables) performed also quite good. From our feature selection process using forward selection methods for linear regression and from our boosting method in LightGBM we saw that model performance dropped quite rapidly. After adding around 30 features in forward (and stepwise) linear regression we saw that the RMSE curve (learning curve) basically ''flattened''. We also saw this in relation to the number of boosting rounds in our LightGBM in which we actually could train further although visually the learning curve flattened. However a small experiment – not documented- showed that after around 120 boosting rounds the drop (although again it was very subtle, but seemed like a ''threshold'' point) in RMSE continued. These small number of features and small number of boosting rounds also ''relate'' to the PCA showed that 25 factors explained around 90% of the variance in the data.

More details about the latter paragraph can be found in the notebooks mentioned below.

Notebook: 3.Feature exploration and selection.ipynb ,4. Modelling.ipynb

V. Conclusion

Our heart rate dynamics model(s) showed promising results. Also using the predictions coming from these disaggregate models seem to perform very well compared to predictions from aggregate metrics to estimate training stress (TRIMP).

To conclude we show some results of the top 10 best (lowest RMSE) predicted rides (**Figure 19**) and the top 10 worst predicted rides (**Figure 20**) from our holdout period.

Figure 19 Top 10 best predicted rides

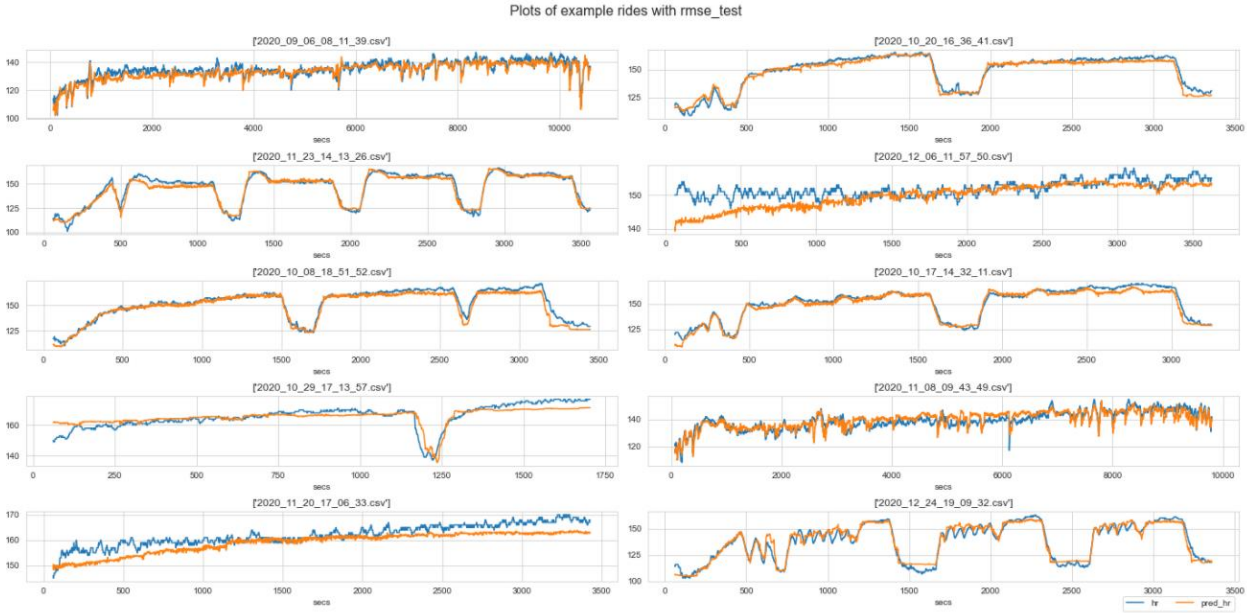


Figure 20 Top 10 worst predicted rides

We can see that from the top 10 worst predicted rides two general problems arise.

1) the base level of the predictions sometimes is too low or to too high while the pattern in itself is not really bad
2) the amount of variability (predictions are ''spikey'') in the predictions sometimes is too high

There are a multitude of factors which might influence these results. These will be discussed in the **Improvement** section along with how possible improvements could be made.

Notebook: 4. Modelling.ipynb

Related .py files : graph_rides_model_performance.py

Reflection

The project turned out to be quite big. The data processing part took a very long time, because all kind of anomalies occur. This resulted in a lot of code for which I simply did not have the time to refactor so it could perform better and was potentially better readable. I solved this partly by commenting a lot of what operations were done in the notebook, at the beginning or between the lines of code.

Since there was a lot of data and to speed up processing, and prevent memory errors we made use of frequently deleting data we did not need anymore, run code to reduce memory usage and sometimes do analysis (e.g. feature selection) on part of the data. In addition to prevent running things again we saved intermediate results (pickle files). This made the code very verbose and gave it a certain repeating character at times.

The size of the dataset in relation to computing resources in the beginning of the project also made it a lengthy process for which I should have made the decision earlier to move it to a cloud platform. However, also on a cloud platform (AWS) it still took quite a lot of time. This is a lesson for project planning.

I found the topic of trying to predict heart rate response to workouts fascinating to work on. Since introduction of the powermeter in cycling the whole training and racing process and the ''improvements'' in derived metrics are very much directed towards power while the physiological component seems -in my opinion- to be pushed more to the background. This is by itself strange since in order to perform one of the (most important) factors to be able to perform is the physiological component.

In the end I found the results to be quite satisfactory and promising, at least from the experimental perspective. Nevertheless I still see a lot of improvement potential both in the model and in my way of working and so it turned out to be a useful learning experience.

Improvement

We have done quite some experiments and analysis, but still there is plenty of room to further improve the project outcomes. Below a list of possible enhancements and directions for experimentation.

- First of all we would suggest to do a more fundamental error analysis of the model: where and why does the model predict well. We already found some rides with power dropouts during interval sets. These kinds of dropouts can "relatively'' easy be detected to due to expected power output in structured intervals, but it becomes harder with unstructured like rides (e.g. races) in which a pattern ''that is expected'' is more difficult to pinpoint
- Enhance anomaly detection methods. We have used a combination of rules and algorithms to construct a clean model training set. However, after inspection still some data anomalies exist for which extra rules / algorithms could be made. However -more on ride level-, more strict ruling on whether to use a ride or not for model building could be made
- We observed that there was as stabilization in the RMSE drop (learning curve) after around 30 variables in the model (and also around approx.. 30 boosting rounds). After that RMSE dropped (without really starting to overfit yet) very slowly indicating that considering the current features available we would only squeeze more out of the model just by adding a lot of variables. To really make a substantial progress in RMSE for further studies would not be -in our opinion- to look at other algorithms, but to first look at better (other) features

- Building on the previous improvement suggestion possibly some more domain related features could be created, but also one could look at feature generation coming from different fields like signal processing theory. We also see some more possibilities by creating indices between different rolling periods in our current feature set. New features could also be generated based on the time series dimension (aggregate features). A python library for this purpose is for example tsfresh
- Also it is interesting to see how the selected model variables would perform on other rider's data. For coaches this could be to experiment on their own rider data. Also open source cycling data is available which is a huge source for scientists and practitioners to experiment further. See https://github.com/GoldenCheetah/OpenData
- Since heart rate level and dynamics are also determined by the rider's physiological condition - well-trained means a higher wattage for the same heart rate- it would be interesting to use a moving window approach. The window shifts forward on a continuous base and the model parameters are re-estimated given the ''current'' state / condition of the rider. In this way ''correcting'' for critical power or other ''threshold'' metrics might not even be ''necessary'', because the physiological condition is also shifting in the window. A side effect is however less data to estimate the model parameters
- We did not include (outdoor) temperature in our models since we did not deem the quality of this data to be very good. New experiments could be done to see how the model would improve on rides with valid temperature readings
- It would be interesting to also include other ''meta'' variables like HRV (Heart Rate variability) to see how the general level of the heart rate prediction might be affected by these ''internal'' variables. In our study we did not have this data for the rider. In addition metrics on the riders subjective feeling before rides take place could be taken into account. Other examples are sleep metrics
- More and more new wearables come to market. Glucose monitors and core body temperature sensors are an example of this. These can be worn during workouts. Including this data in the future might further enhance model predictions. In this way the effect of nutrition / hydration could be taken into account. If however used for predicting heart rate response before a ride takes place this might be difficult since the variables will have to be estimated before they can be used as input for the model. To still be able to use some of this information to some extent would be to measure before a ride
- To increase model performance an option would be to first cluster similar rides (either by labelling in advance or algorithmically) and to develop models based on these clusters. Models for each of these clusters might enhance performance instead of modelling on all rides. Also the dimension of structured workouts versus unstructured workouts could be an option maybe also from a practical training prescription perspective
- We have included some metrics related to mechanical and internal training load/stress (e.g. ATL, CTL, TSB). However, also other aggregate metrics which have a lagged effect in it with the current day on which the workout is done, can be tested
- We used modelled CP (extended model) from Golden Cheetah and we observed that in general scaling wattage to ''correct for riders form'' did give stronger relations with heart rate. However the modelled CP values also causes deviations due to periods of non-hard efforts in the window of observation. Also such strong deviations in rider's threshold value is not realistic. Possible new experiments could focus on more subtle CP deviations as factor to correct

Literature

1) Thomas W. Calvert, Eric W. Banister, Margaret V. Savage, Tim Bach, A Systems Model of the Effects of Training on Physical Performance, IEEE Transactions on Systems, Man, and Cybernetics, volume SMC-6, issue 2, 1976, pages 94–102

2) Edwards S (1993) High performance training and racing. In: Edwards S, editor. The heart rate monitor book. Sacramento: Feet Fleet Press 113-123

3) Rationale and resources for teaching the mathematical modeling of athletic training and performance, David C Clarke, Philip F Skiba, Advances in Physiology Education, 2013 Jun; 37(2): 134-52. doi: 10.1152/advan.00078.2011

4) A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics, H. Baak, R. Koopman, H. Snoek and S.Klous, Nov 2018, https://arxiv.org/abs/1811.11440

5) Application of Critical Power in Sport, March 2011, International Journal of Sports and Psychology and Performance 6(1):128-26

6) Towards Machine Learning on data from Professional Cyclists (2018), Hilmki Al, Ivarsson O, Johansson M, Kuylenstierna D, van Erp T, Accepted for the 12th World Congress on Performance Analysis of Sports, Opatija, Croatia, 2018, https://arxiv.org/abs/1808.00198

7) Zakynthinaki MS, Stirling JR (2008) Stochastic optimization for the calculation of the time dependency of the physiological demand during exercise and recovery. Comput Phys Commun 179(12):888–894

8) Stirling JR, Zakynthinaki MS, Refoyo I, Sampedro J (2008) A model of heart rate kinetics in response to exercise. J Nonlinear Math Phys 15(3):426–436

9) Mazzoleni MJ, Battaglini CL, Mann BP (2015) Modeling heart rate dynamics in response to changes in exercise intensity. In: Proceedings of the ASME 2015 International Design Engineering Technical Conferences and Computers in Engineering Conference, DETC2015–47587

10) Lefever J, Berckmans D, Aerts J (2014) Time-variant modelling of heart rate responses to exercise intensity during road cycling. Eur J Sport Sci 14(S1):S406–S412

11) Modelling and predicting heart rate dynamics across a broad range of transient exercise intensities during cycling, Sports engineering, 19, 117–127(2016)

12) Training and racing with a powermeter, 3rd edition, March 2019, Hunter Allen (auteur), Andrew R Coggan PhD (auteur), Stephen McGregor PhD (auteur), VeloPress

13) Interpretable Machine Learning, a guide for making black box models explainable, C. Molnar, https://christophm.github.io/interpretable-ml-book/

14) The Effect of Multicollinearity on Prediction in Regression Models, D Mundfrom, M.L de Poy Smith, Jan 2018, Research Gate, DOI 10.31523/glmj.044001.003