Methods for predicting the pace of a runner

Martina Betti - 1799160
Leonardo Masci - 1822292
Stefano D'Arrigo - 1960500
Juan Mata Naranjo - 1939671

July 19, 2021

Abstract

The aim of this project is to predict the pace of a runner in terms of bpm in order to provide music by the same bpm. To achieve this, many different models are explored: ARIMA, Naive-Bayes Gaussian Regressor, Ridge Regression, Support Vector Machine, Random Forest, Gradient Boosted Decision Trees and Poisson Regressor. In the end, ARIMA is considered the most appropriate model for this analysis, as it falls under the time dependence method. Under the assumption of independence, the Gradient Boosted Decision Tree yields the best results in terms of RMSE among all runners and models, although this model with the data described throughout the report has lower interpretability.

1 Introduction

The aim of this project is to predict the pace of a runner in terms of bpm in order to provide music by the same bpm. As is stated in literature, "the use of music and specifically tempo-matched music has been shown to affect running performance" [5], so starting from this we want to find music that matches the usual running speed of a runner. Naturally, before achieving this final goal a prediction on the running pace has to be implemented. For this reason, we mainly focus our efforts on this first task.

This report is divided in 7 sections, excluding this one. Firstly, we take a look at the procedure of data collection (paragraph 2) and the subsequent preprocessing (paragraph 3). We explore two different approaches to predict steps/min, namely an assumption of dependence (the target variable depends on the previous observations) or independence (target variable is independent from previous observations). This distinction is relevant in producing the corresponding data sets. We proceed with some explanatory data analysis (paragraph 4) and by illustrating the model and methodology followed in this work (paragraph 5). Here, a focus on the above-mentioned assumptions can be found, as well as a more in-depth exploration of the models employed in the later stages. Lastly, the final results are presented (paragraph 6), some appropriate comments on the topic of interpretable machine learning are provided (paragraph 7) and the conclusions are drawn (paragraph 8).

2 Data Collection

Data collection has been performed independently by each member of the team (4 components) by going running regularly over an interval of 3 months. We have set several conditions beforehand in order to make the data collection as homogeneous as possible and therefore facilitate the later data pre-processing. We have however also allowed some degrees of freedom to each of the runners in order to obtain a more robust final model, since this model should be independent of the runner and its conditions. Between the strong conditions we have set ourselves beforehand we can highlight: (i) Running sessions should have a similar duration for each runner, (ii) the running session should follow a similar path throughout most of the sessions, (iii) data collection device should be kept in the same position throughout all sessions, however all runners do not have to keep the devices in the same location, the models will have to account for this factor.

The data collection itself has been done through two different apps, which collect data simultaneously: **Arduino Science Journal App** and **Google Fit App**. We have decided to use these two apps since we consider them to provide us with sufficiently granular data (data is collected in very short intervals of time) and the features they offer us complement each other very well. From the Arduino Science Journal App we have decided to collect all the features available (AccX, AccY, AccZ, DecibleSource, Accelerometer, etc.) while from the Google Fit App we will only use two variables: (i) Altitude and (ii) Distance. In the next sections we will further explain how we process this data in order to obtain features which can be used for Machine Learning purposes.

The number of sessions for which we were able to collect data can be depicted below:

Team Member	**Number Sessions**	**Length Session (average min)**
Runner 1	22	30
Runner 2	20	6
Runner 3	13	20
Runner 4	5	25

Table 1: Number of sessions and average session length for each team member.

As we can observe, the data collection process for this project has also been challenging, not all members could complete as many sessions as expected and not all the sessions lasted as long as expected. For this reason we have decided to discard the sessions collected from the Runner 4.

3 Data Processing

Due to the nature of the data, the data processing part of this project has been crucial and extensive. This section not only includes cleaning up the data of missing but also the generation of additional features which can improve the performance of the final models. Given the importance of this part, we will go step by step in explaining our procedure and the reasoning behind each of the decisions:

- **Missing Data Imputation:** The Arduino Science Journal App does not measure all data features at the same time, e.g. AccX is measured at time t while AccY is measured at time t', leaving the features AccX and AccY missing at times t' and t respectively. The imputation strategy used was linear interpolation with respect to the time.
- **Session Start and End:** Given that there is an initial setup time for our device before the actual session starts, we have decided to remove the initial and final *noise* which would otherwise strongly bias our model. This has been done automatically by inferring a drastic change in pattern of the feature AccY. This drastic change will denote the beginning and end of the session (given the low number of sessions we made sure that this automatic process is in line with expectations).

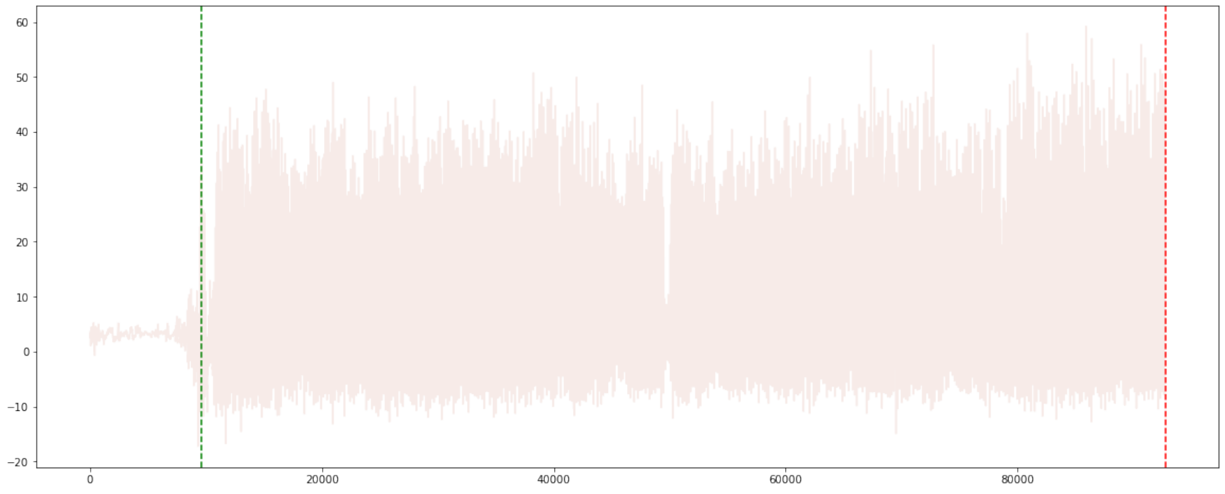


Figure 1: Automatically inferring when session starts and ends.

- **Merging Google Fit and Arduino Data:** Of course we need to merge both sources of data. However, the time at which these two dataframes have been collected are not equivalent. For this reason we have decided to maintain the time of the Arduino Data (which is much more granular than that of Google Fit) and replace missing data of distance and altitude on their new time stamps using once again a linear interpolation approach. It is also relevant to highlight that Google Fit data duplicates most of its measure over time, which strongly reduced the number of *correct* measures we could use from Google Fit.
- **Additional Feature Variables:** We generate two additional features (i) speed: computed as the coefficient between the distance traveled and the elapsed time and (ii) delta altitude: the altitude change between two intervals of time.
- **Generate Target Variable:** As already mentioned previously, the main goal of the project is to predict the tempo/pace at which the runner is running in order to provide music with a similar beat. To achieve this we need to infer when a step is taken by each of the runners. We decided to infer this information by detecting peaks in the one of the AccX, AccY or AccZ features (depending on the runner) and based on the position of the phone understand whether a peak should be translated into 1 or 2 steps. In the following figure we can observe how this detection has been made (in this specific example each peak denotes one step):

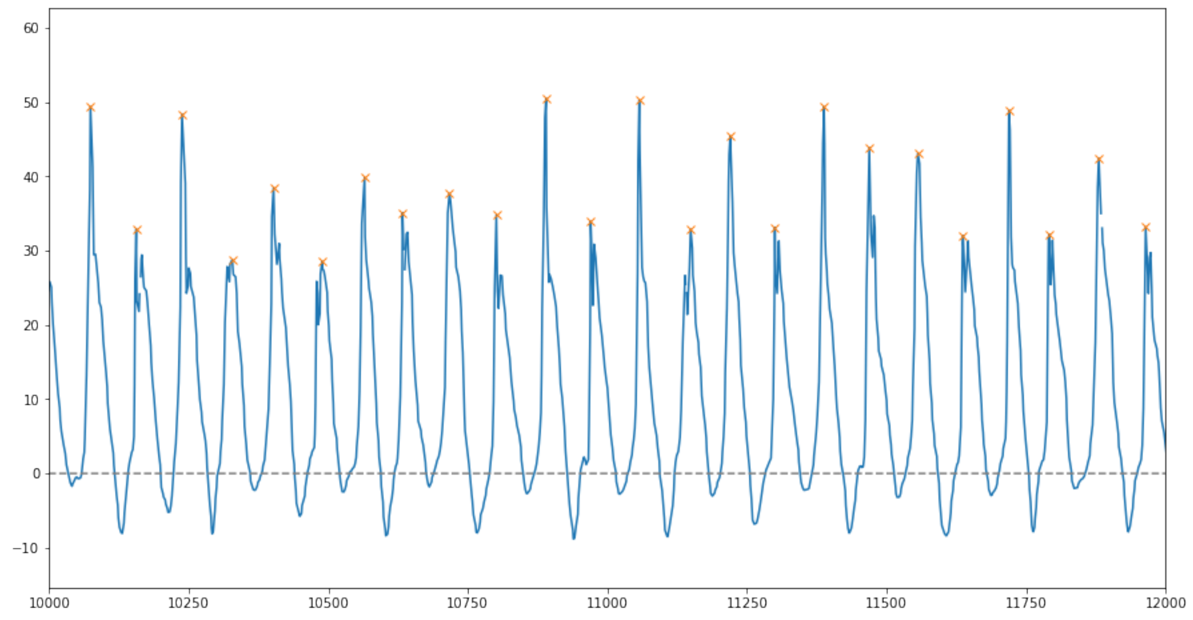


Figure 2: Peak Detection, each peak is translated into 1 step in this example

Once the data for each session have been properly processed, we need to aggregate the data of each of the sessions together. We explored different intervals of time to aggregate our data (1, 2 and 5 seconds). Due to preliminary results and the number of final observations, we decided to aggregate data in intervals of 1 second. The target variable (step sum) has also been aggregated in time intervals of 1 second, however using a moving window of 1 minute, resulting ultimately in a target variable with units of steps/min.

It is also relevant to highlight that we will be using two different types of data sets since we will explore two different approaches to predict steps/min (more detail on this distinction in later sections). For the moment lets focus on the difference between these two data sets:

1. **Approach 1 (Assuming Dependence)**: In this case we assume that the target variable depends on all the previous observations, also on the observations from previous sessions. For this reason all the processed sessions will be concatenated together forming a single time-series of N sessions with p features.
2. **Approach 2 (Assuming Independence)**: We will also consider the scenario in which the target variable is independent from the past (or at least from the late past), allowing us to tackle this machine learning project using more familiar methods. In this case each row (observation) will contain the target variable for that specific time interval, and the features will be regarding the last 3 intervals of time.

The data will be aggregated over intervals of time of 1 second by taking the average, median, max, min and standard deviation of the time series data over that specific time interval. An example of all the variables we are able to generate from the AccX variable can be found below:

timestamp	**AccX min**	**AccX max**	**AccX mean**	**AccX median**	**AccX std**	**step sum**
17:30:32	-7.53	35.04	3.10	0.35	8.88	180
17:30:33	-5.97	36.03	4.17	2.64	8.52	179
17:30:34	-8.21	48.51	5.51	3.51	10.93	178
17:30:35	-8.39	29.29	2.83	1.82	9.15	178
17:30:36	-5.13	39.03	4.43	2.35	8.16	178
17:30:37	-11.57	37.40	6.53	4.99	9.38	177
17:30:38	-6.29	33.83	2.46	1.51	7.46	176
17:30:39	-8.57	32.83	3.64	2.89	9.09	176
17:30:40	-8.88	54.70	7.19	5.08	12.10	176

Table 2: Example of features generates from AccX only. This is the dataset used for Approach 1 (Assuming Dependence)

timestamp	**AccX mean previous 1**	**AccX mean previous 2**	**AccX mean previous 3**	**AccX std previous 1**	**AccX std previous 2**	**AccX std previous 3**	**step sum**
17:30:32	5.93	4.19	3.26	11.18	8.53	11.18	180
17:30:33	3.10	5.93	4.19	8.88	11.18	8.53	179
17:30:34	4.17	3.10	5.93	8.52	8.88	11.18	178
17:30:35	5.51	4.17	3.10	10.93	8.52	8.88	178
17:30:36	2.83	5.51	4.17	9.15	10.93	8.52	178
17:30:37	4.43	2.83	5.51	8.16	9.15	10.93	177
17:30:38	6.53	4.43	2.83	9.38	8.16	9.15	176
17:30:39	2.46	6.53	4.43	7.46	9.38	8.16	176
17:30:40	3.64	2.46	6.53	9.09	7.46	9.38	176

Table 3: Example of features generates from AccX only. This is the dataset used for Approach 2 (Assuming Independence)

As we can observe from Tables 2, 3, in approach 1 we are considering the complete time series, instead in the second approach we are using the summarized data of the 3 previous time intervals in order to predict step sum. This will allow us to shuffle the data and estimate step sum using more familiar machine learning models.

4 Explanatory Data Analysis

One of the first things that have to be reviewed is how the data is distributed, paying particular attention to the target variable, number of steps per minute. The distribution of the target variable (steps per minute) and the features' summaries for all the three runners can be found in Figures 3, 4, 5.

From them we can observe that the distributions for all runners are very different. It is also obvious that the distribution of steps for the first and third runner can almost be described using a Gaussian distribution, while that of the second runner could maybe be considered as a mixture of Gaussians (2 or 3).

There also seem to be several outliers (mainly in the case of the second runner) or inaccurate data which would be best to remove. These outliers have been deleted based on an best guess threshold.

Regarding the summaries of the features, we can appreciate a certain degree of similarity among the runners except for the DecibelSource, which could easily depend on the fact that data was collected along different paths with different noise levels. In general, it seems fair to assume that the set of rules established for data collection have worked well enough to guarantee a certain level of homogeneity.

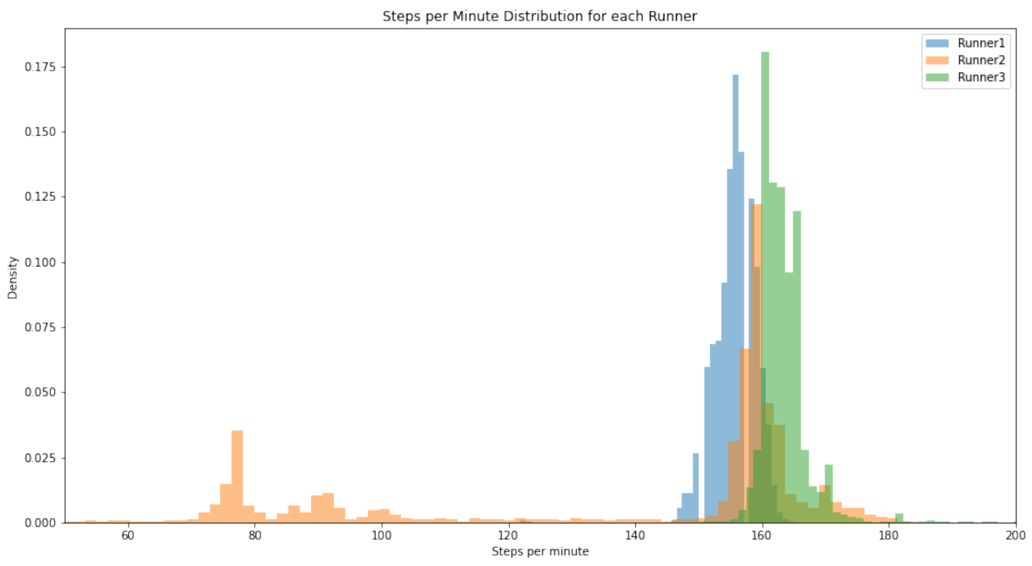


Figure 3: Steps per minute density distribution for all runners.

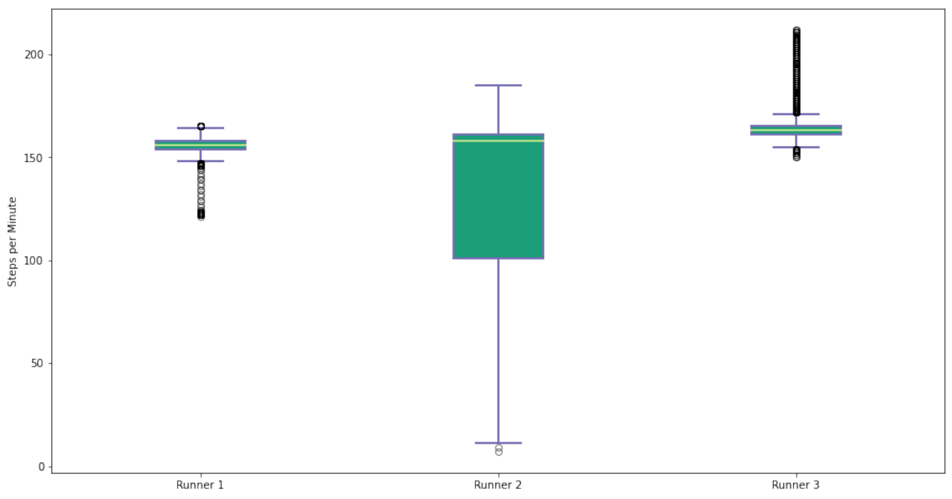


Figure 4: Steps per minute boxplot for all runners.

5 Model and Methodology

As highlighted in the data processing section, we follow two approaches to estimate our target variable. In this section we review these two methods in detail and outline all the machine learning models used for prediction.

5.1 Assuming dependence

In the first approach we assume that the steps per minute can be estimated using all the past information, not only including data of the same session but also from previous sessions. Under this assumption the best models are time series models, since it is reasonable to assume that some aspects of the past patterns will continue into the future. Although this is an unconventional use of a time series, we still felt that this could be a promising approach and decided to focus on the ARIMA model.

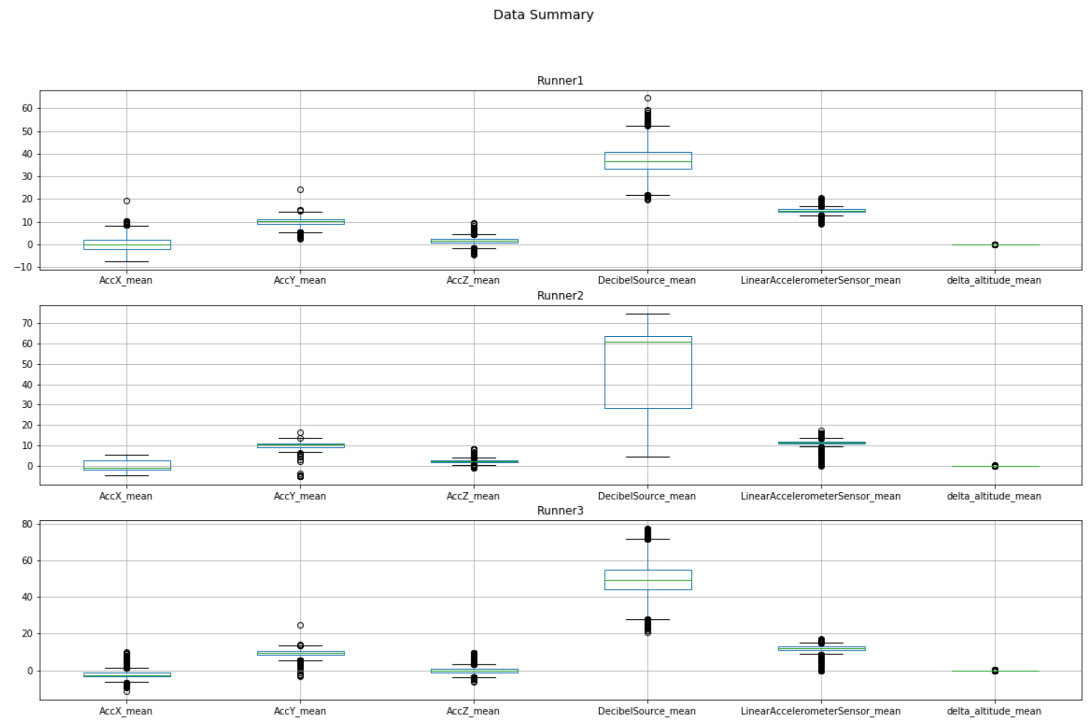


Figure 5: Features Summary.

It is worth mentioning that given the high reliance of time series models on the definition of "lag", the datasets have been processed to obtain summaries by the minute so as to obtain more meaningful and interpretable results. More specifically, we have calculated the mean number of steps per minute.

5.1.1 Properties of time series

- **Stationarity**: A time series is stationary if it is not a function of time, meaning that the mean of the observed values doesn't have a strong increasing or decreasing trend. We used the Augmented Dickey-Fuller test to check this property and got a low p-value for all runners. By rejecting the null hypothesis we can assume all three of the time series to be stationary, therefore we set the ARIMA differencing parameter (i) to zero.
- **Autocorrelation**: In a time series, autocorrelation measures the linear relationship between lagged values. In our case, the autocorrelation is cyclic but not seasonal, since it does not repeat itself at fixed frequency (each run has a ever-so-slightly different duration). More details on autocorrelation are provided in the interpretability section (Paragraph 7).

	Runner1	**Runner2**	**Runner3**
p-value	0.0186	3.6328e-20	1.4392e-13

Table 4: Augmented Dickey-Fuller test results.

5.1.2 ARIMA Model

As previously mentioned, we focus our efforts on the autoregressive integrated moving average (ARIMA) model. In such a model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself.

Autoregressive models

Thus, an autoregressive model of order p can be written as:

$$y_t = c + \phi_1 y_t - 1 + \phi_2 y_t - 2 + ... + \phi_p y_t - p + \epsilon_t \tag{1}$$

Moving Average models

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model. It can be written as:

$$y_t = c + \epsilon_t + \theta_1 \epsilon_t - 1 + \theta_2 \epsilon_t - 2 + ... + \theta_q \epsilon_t - q \tag{2}$$

where ϵ_t is white noise.

Model Selection

Lastly, the maximum likelihood estimation (MLE) is used for evaluating ARIMA models. ARIMA also returns Akaike's Information Criterion and the Bayesian Information Criterion. These information criteria tend not to be good guides for selecting the appropriate order of differencing (d) of a model, but only for selecting the values of p and q.

Taking all of this into consideration, we use the AutoArima function to set an interval for the values of p and q and automatically return the best model based on its AIC. In the end, the training of the model on each of the runners returns different values for the parameters, which allows for a more accurate prediction. The resulting parameters can be found in Table 5.

	Runner1	**Runner2**	**Runner3**
AR	1	1	0
I	0	0	0
MA	2	0	2

Table 5: Configuration Parameters for ARIMA model. AR: Autoregression parameter, I: differencing level, MA: Moving avarage parameter.

In conclusion, our three models are best described by the following equations:

- **Model 1**
$$\hat{Y} = c + \phi_1 y_{t-1} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} \tag{3}$$
- **Model 2**
$$\hat{Y} = c + \phi_1 y_{t-1} \tag{4}$$
- **Model 3**
$$\hat{Y} = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} \tag{5}$$

5.1.3 Adding exogenous variables

Even though the model can be trained by just looking at the target variable, one can try to improve the model's performance by adding some exogenous variables to the ARIMA (endogenous) model. For each runner we selected the most correlated feature with respect to the target variable, which ended up being AccZ for Runner 1 and Linear Acceleration for the other two. By comparing the results it emerged that slightly better results are obtained only for one runner (Runner 3), while the results got worse for the other two.

We tried to understand which is the factor in the exogenous feature choice that determines the improvement of the results. We hypothesize this could depend on a stronger correlation for one runner compared to the other two,

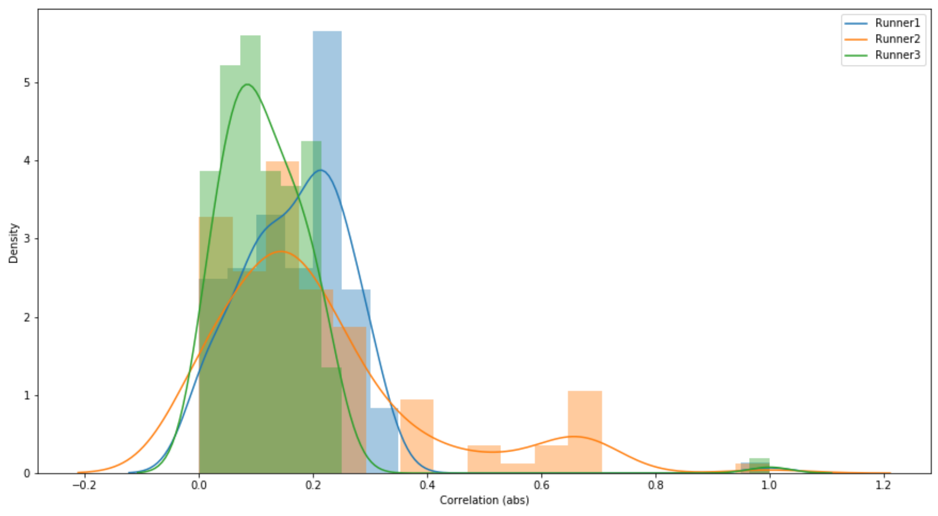


Figure 6: Distribution of the correlation between predictors and steps per minute.

but, as can be seen in Figure 6, Runner 2 has a stronger correlation between the selected exogenous variable and the number of steps per minute (0.400) compared to Runner 3 (-0.268), which does not explain the latter's improved results. Moreover, the exogenous feature selected for Runner 2 and Runner 3 is the same (Linear Acceleration), so a causation mechanism due to the selected sensor can not be hypothesized.

Our interpretation of these results is that the addition of an exogenous variable to an ARIMA could potentially improve the model performances in the case of a strong predictor, while given a set of weak predictors the integration of a single feature to the model may simply add some noise. Finally, we decided not to include any exogenous variable to the models, because it takes away some level of interpretability (we are not able to predict the effect) while not yielding good enough results. In addition, we heavily focus on the role of predictors in the next section.

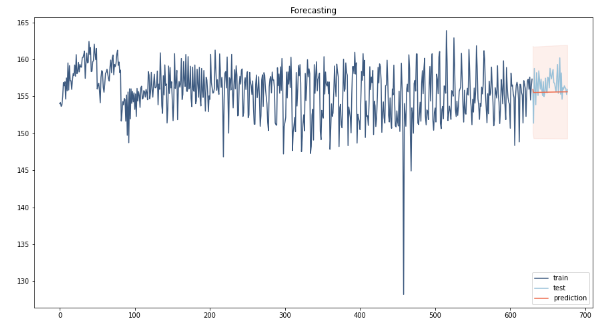
5.2 Assuming independence

On the other hand, in this approach we assume that the steps per intervals of time can be estimated from the summaries of the previous three time intervals. In other words, the current number of steps is conditionally independent from the far past. This is a very strong and very likely inaccurate assumption, however we hope that this approach can still give good enough results while also making the interpretability of our results clearer. It is also relevant to highlight that for this approach we will follow two strategies, classification and regression. Following the logic of the last homework, we considered that despite the nature of this problem being regression, classification might also give valuable results.

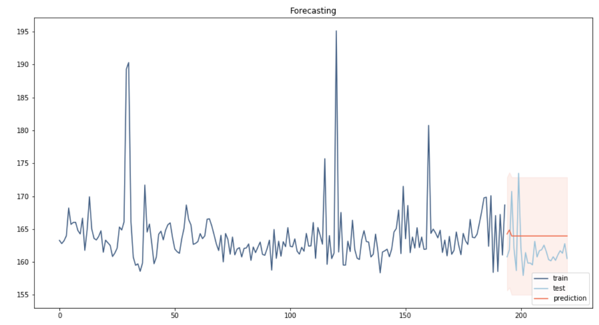
5.2.1 Feature Engineering

Part of the feature engineering has already been executed in the data processing section. However, after generating all of the above mentioned features, we considered that the number of features might be too high and that reducing this number would improve the accuracy of the model and increase its interpretability. We explored several options for this:

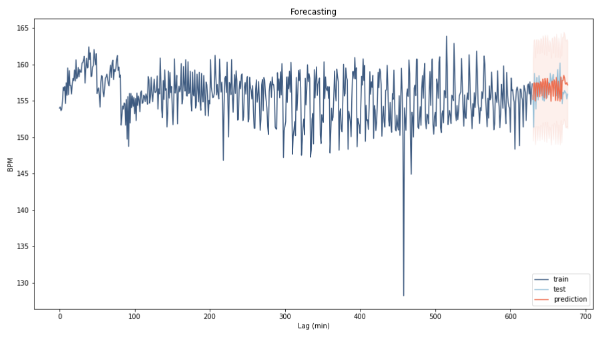
- Reducing dimensionality using SVD and keeping the 30-40 most explicative components. The issue with this method is that we would not be able to interpret the components generated by the SVD, which would significantly reduce the model's interpretability.
- Estimating the most important variables by running multiple Random Forests on subsets of the complete data



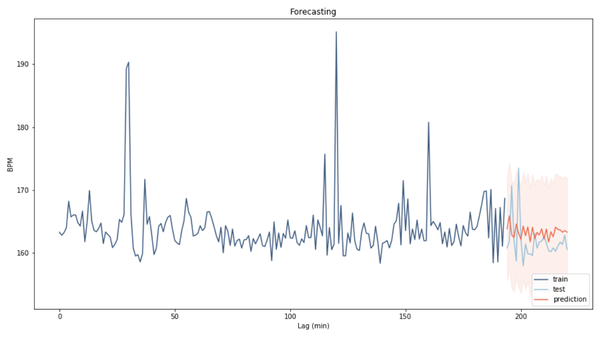
(a)



(b)



(c)



(d)

Figure 7: Predictions for Runner 1 (a, c) and Runner 3 (b, d) for ARIMA model without exogenous variables (a, b) and with exogenous variables (c, d).

set, and choosing the variables that are selected as the most important variables with highest frequency. This selection is done by taking the mean position of each feature variable and ranking it accordingly.

After trying out these two approaches we decided not to leave out any feature since the results with a lower number of features were providing worse results (on training and test data), as depicted in Table 6.

Features Type	**Estimator (with best parameters)**	**Accuracy on Train**	**Accuracy on Test**	**F1 on Test**
Reduced features	RidgeClassifier	0.548	0.538	0.581
Reduced features	XGBClassifier	0.784	0.793	0.787
Reduced features	LogisticRegression	0.711	0.714	0.704
Complete features	RidgeClassifier	0.644	0.636	0.675
Complete features	XGBClassifier	0.800	0.802	0.795
Complete features	LogisticRegression	0.766	0.754	0.747

Table 6: Comparing results with complete features and with only the most important variables. Results using classification and for only one of the runners.

Instead, we decided to keep all of the features and do our best to explain the model accordingly, since real-world models will also have this high number of features and interpretability should still be achieved.

5.2.2 Proposed Models

The models proposed for our modeling task are the following:

- **Naive-Bayes Gaussian Regressor:** This is our baseline model, which assumes conditional independence over each class for the observed features. This specific Naive-Bayes regressor assumes these conditional probabilities to follow a Gaussian distribution. We do not expect this model to perform very well over this data, but rather it will serve as a benchmark for the remaining models.
- **Ridge Regression:** The ridge linear regression model is a modification of the linear regression model. The ridge regression contains an additional factor, the ridge factor. It tries to mitigate the collinearity between feature variables, which would otherwise decrease the performance of the linear model. Mathematically, the closed form solution of the Rigde model is the normal equation with the additional ridge factor:

  $$\hat{\beta}^{ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \tag{6}$$

  Due to the high collinearity between our features we expect this model to perform pretty well.
- **Support Vector Machine:** In contrast to OLS, the objective function of Support Vector Regressors is to minimize the coefficients — more specifically, the L2-norm of the coefficient vector — not the squared error. The error term is instead handled in the constraints, where we set the absolute error less than or equal to a specified margin, called the maximum error, ϵ. The optimization problem is:

  $$\min \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}|\xi_i| \tag{7}$$

  with the constraint similar to that of the classification problem:

  $$|y_i - \omega_i x_i| \leq \epsilon_i + |\xi_i| \tag{8}$$

  We can also explore alternative feature spaces using the kernel trick, as in the classification problem.
- **Random Forest:** A random forest is an ensemble of individual decision trees which are then combined together in somewhat of a voting system in order to provide the final predicted class. The key ingredient in random forests is that the individual decision trees need to be uncorrelated, in order for the votes not to be biased. This property is obtained through a series of bootstrapping procedures.
- **Gradient Boosted Decision Trees:** As the name suggests, we are generating a model which includes the creation of multiple decision trees, each of which are modelled using the output prediction of the previous. This technique is named as boosting. Gradient suggests the method used to find the optimal prediction value which minimizes the loss function.
- **Poisson Regressor:** The only difference between this model and a normal linear regression model is how the expectation of the target variable is modelled given the observed covariates. As we know, in the linear regression model we have $\mathbb{E}(Y|x) = \boldsymbol{\beta x^T}$, while in the Poisson regressor the expectation is modelled as $\mathbb{E}(Y|x) = e^{\boldsymbol{\beta x^T}}$.

5.2.3 Hyper-parameter Tuning

The champion model and hyper-parameters have been calculated using a combination between 5-Fold Cross Validation and a Grid Search with a 70-30 splitting scheme. Table 7 contains an exhaustive overview of the parameters explored.

The best models have been evaluated using the root mean squared error.

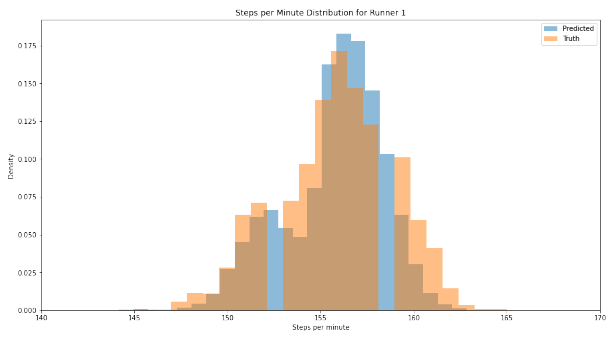
6 Results

The idea behind the project is to predict the bpm of a runner during his/her next running session. We have scored the performances of the models by evaluating the RMSE on the predictions (bpm) for the last running session when assuming dependence, and have similarly evaluated the RMSE on a random sample of our data for the independence approach.

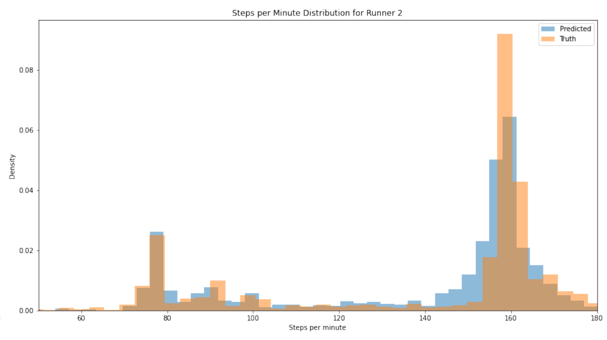
After inspecting all of the previously mentioned models, parameters, classification/regression, assuming dependence versus assuming independence, etc., the results obtained are outlined in Table 8.

Regressor	Configuration				
	Parameter	**Values**	**Config. 1**	**Config. 2**	**Config. 3**
Poisson Regressor	alpha	0, 0.001, 0.01, 0.1, 0.5, 1, 5, 10, 50	5	0.1	50
	max_iter	10000	10000	10000	10000
Ridge Regression	alpha	0.0001, 0.01, 0.1, 0.25, 0.5 1, 5, 10, 20, 50, 100	100	1	100
Gaussian Mixture	num components	1, 2, 3, 4, 5	5	5	5
	max iter	10000	10000	10000	10000
	covariance type	tied	tied	tied	tied
Gradient Boost Regressor	max depth	3, 6, 9, 20	9	6	0
	min child weight	1	1	1	1
	learning rate	0.01, 0.1, 0.2, 0.5	0.2	0.2	0
Support Vector Machine	kernel	linear, poly, rbf, sigmoid	rbf	rbf	rbf
	degree	1, 2, 3, 4, 5	1	1	1
	C	0.001, 0.5, 1, 50	50	50	50
Random Forest Regressor	num estimators	10, 50, 100, 200	200	100	200
	max depth	3, 6, 9, 12	12	12	12
	min samples split	2, 4, 6	6	4	2
	max features	auto, sqrt, log2	sqrt	sqrt	sqrt

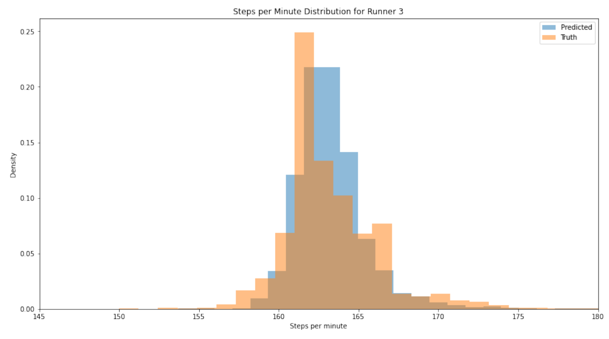
Table 7: Configuration parameters for the considered regressors. The Gaussian Naive-Bayes regressor has no parameter to tune and hence is hidden from the table. The *Config. {1, 2, 3}* columns contain the best parameters for the models trained on the Runner 1, 2 and 3 data, respectively.



(a)



(b)



(c)

Figure 8: Comparison of distribution of predicted steps per minute versus the actual steps per minute.

The distribution of the predicted steps per minute using the Gradient Boost Decision Tree (one of the models with best performance) for each runner compared to its actual distribution can be seen in Figure 8.

Before proceeding, we spend some words on the classification task. Although it was previously explored, we realized that the regression approach is a more natural approach, since the target variable is continuous. Furthermore, the classification approach comes with a huge limitation: after training the model, we are able to assign new observations only to the classes we have previously observed. There is no way to predict a new class - hence a new number of

	Estimator	Train RMSE	Test RMSE
Runner 1	***Assuming Dependence***		
	ARIMA	3.286	1.986
	Assuming Independence		
	Regression		
	PoissonRegressor	2.455	2.428
	Ridge	2.452	2.435
	GaussianMixture	154.009	153.885
	XGBRegressor	1.776	1.697
	GaussianNB	5.147	4.938
	SVR	2.349	2.301
	RandomForestRegressor	1.927	1.873
Runner 2	***Assuming Dependence***		
	ARIMA	25.161	8.268
	Assuming Independence		
	Regression		
	PoissonRegressor	24.312	21.883
	Ridge	22.146	21.229
	GaussianMixture	140.500	141.629
	XGBRegressor	13.943	12.764
	GaussianNB	29.656	32.122
	SVR	27.381	26.895
	RandomForestRegressor	12.870	12.263
Runner 3	***Assuming Dependence***		
	ARIMA	4.525	3.845
	Assuming Independence		
	Regression		
	PoissonRegressor	4.532	4.893
	Ridge	4.729	6.263
	GaussianMixture	162.007	161.480
	XGBRegressor	2.733	2.561
	GaussianNB	6.518	6.388
	SVR	4.321	4.096
	RandomForestRegressor	2.933	2.996

Table 8: Results on best parameters over all inspected models.

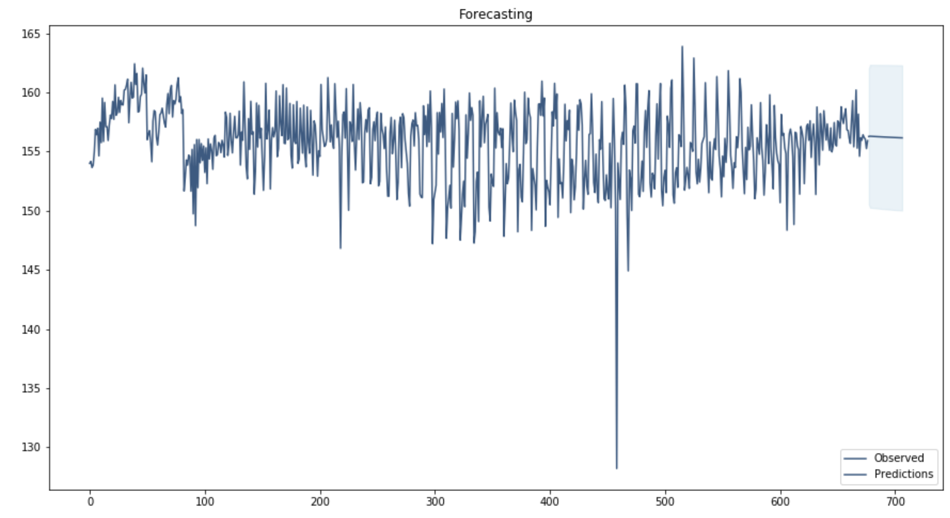
steps - if we observe a new combination of features' values. Let us clarify this point with an example: during the data collection the runner's maximum number of steps is, say, 180; once the model is trained, the runner has a more intense session and the maximum number of steps recorded is 208; in this case, the model is always going to fail in the prediction. Obviously, thinking about the initial theoretical goal of the project - develop a model that estimates the number of steps per minute as the user runs - this behaviour is not desirable.

On the other hand, a regression model is able to correctly predict, or at least guess, the new target value, clearly provided that the learnt regression function didn't overfit the training data.

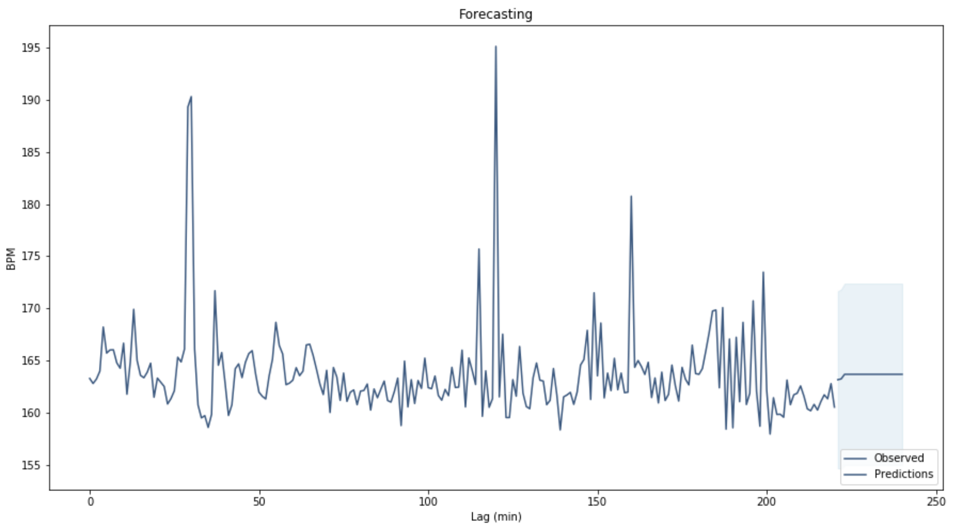
Thus, in the following section only the regression's results will be shown and commented. Nevertheless, since it is a step in the problem analysis and being an additional, though different, insight on the data, the classification results are given in the Appendix A.

6.1 BPM prediction

A simple application of our predictions is to provide each runner with the predicted BPM throughout their next running session and suggest a specific playlist with a suitable BPM range. The predictions for the three runners are shown in Figure 9.



(a)



(b)

Figure 9: Next running session forecasting for Runner 1 (a) and Runner 3 (b).

The predicted BPM throughout the runner's sessions is pretty stable throughout the session so we could potentially suggest a 155 BPM playlist to Runner 1 and a 165 BPM playlist to Runner 3.

7 Interpretable Machine Learning

Several models have proven to return excellent results on our test data. Given the large set of models inspected, we will not review the interpretability of all of them, but rather we further inspect only the models that have shown the best overall performance (considering all runners) and that we consider could be the most interesting from an interpretation point of view:

- **Gradient Boost Decision Tree**: Similarly to any model based on decision tree, it is simple to follow the paths followed by a prediction in order to understand the reason behind model decisions.

- **Poisson Regressor**: Given the linear nature of this model, global and local interpretability should not be hard to obtain.
- **ARIMA**: Predictions obtained from time series models are exclusively dependent on the behaviour of the target variable during the past, which itself guarantees a high level of interpretability.

For the Gradient Boost Decision Tree regressor and the Poisson Regressor identifying the most relevant variables is straight-forward. In the Poisson Regressor case this is due to its linear nature, which allows us to interpret the coefficients with the highest absolute value as the most important variables for the prediction (of course this only applies when the data is standardized). Instead, the variable importance in the Gradient Boost Decision Tree case is estimated based on the decrease in accuracy or mean squared error - for classification and regression respectively - when removing one of the variables from the model.

7.1 Methods

LOCO This method tries to estimate the prediction power of the model when removing one variable compared to not removing the variable, i.e. keeping the original set of features. This is quantified by taking the median over the following quantity:

$$|Y - \hat{f}^{-j}(X)| - |Y - \hat{f}(X)| \tag{9}$$

where Y is the true target variable of a test data set, and $\hat{f}(X)$ and $\hat{f}^{-j}(X)$ are the predicted values considering all feature variables and removing one feature, respectively. A bootstrap approach has been used to estimate the corresponding confidence interval (correcting for multiplicity by using Bonferroni).

LIME The Local Interpretable Model-agnostic Explanations (LIME) [14] is a method to locally explain the individual predictions of a classifier or a regressor by training local surrogate models. The idea behind LIME is to treat whichever model as a black-box, interpreting it through its behaviour in the neighbourhood of the considered prediction. More specifically, given a prediction to be explained, LIME slightly changes the values of its features, in order to obtain samples close but not equal; then, it feeds the target model with the perturbed samples, getting back the predictions. On the tuples new samples-predictions, the algorithm fits an interpretable model, like Lasso, weighted by the proximity of the sampled instances to the instance of interest: this way, LIME learns a local approximation of the black-box's predictions, being then able to distinguish the impact that each feature has on the considered prediction.

For instance, if the black-box model is a binary classifier, LIME provides a local decision boundary and tells which feature is in favour to the predicted class and which is against. The case of explaining a regressor is less intuitive, since the prediction is no longer discrete but continuous: LIME outputs the positive or negative effect a feature has on the final prediction, i.e. if each feature increases or lowers the predicted value.

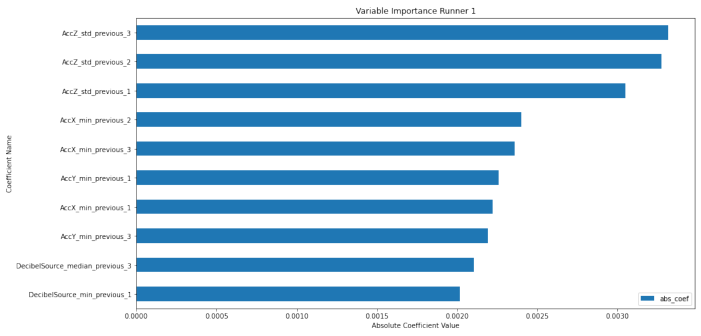
The mathematical formulation of LIME is the following: let us consider an observation $x \in \mathbb{R}^d$ and a black-box model $f : \mathbb{R}^d \to \mathbb{R}$. Let us also consider a class G of interpretable models and $g \in G$; actually, we denote each model through its explanation g. As not every explanation (model) in G may be simple enough to be interpretable, we let $\Omega(g)$ be a measure of complexity of the explanation g. As previously mentioned, we perturb x in order to obtain a neighbourhood of close samples Z_x. For each $z \in Z_x$, we define a measure of proximity $\pi_x(z)$, which weights z on the basis of its distance from x, i.e. the farther z, the less weighted. Ultimately, let $\mathcal{L}(f, g, \pi_x)$ be a measure of how unfaithful g is in approximating f in the locality defined by π_x, i.e. how well g approximates the predictions of f in the neighbourhood Z_x. A LIME explanation of x is then:

$$\xi(x) = \arg_{g \in G} \min \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{10}$$

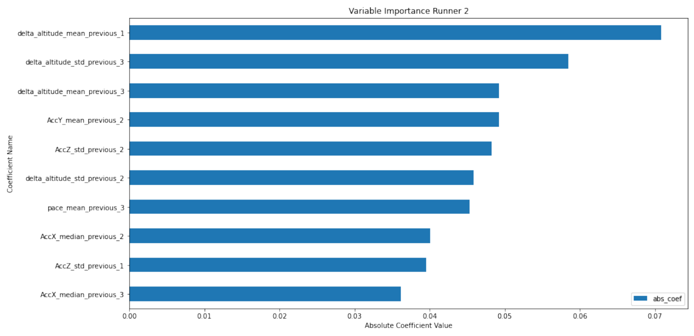
7.2 Models interpretation

7.2.1 Poisson Regressor:

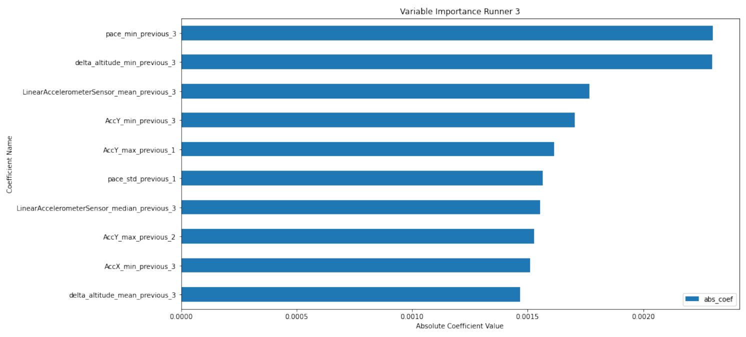
The first thing we inspect is the variable's importance based on the absolute value of the resulting modelling coefficients. In order to compare coefficients accordingly, it is important to highlight that the data needs to be standardized before training the optimal parameters, otherwise the coefficients would depend on the scale of the feature itself and the coefficients would not be comparable. The most relevant features for each runner when applying the Poisson regression model can be found in Figure 10.



(a)



(b)



(c)

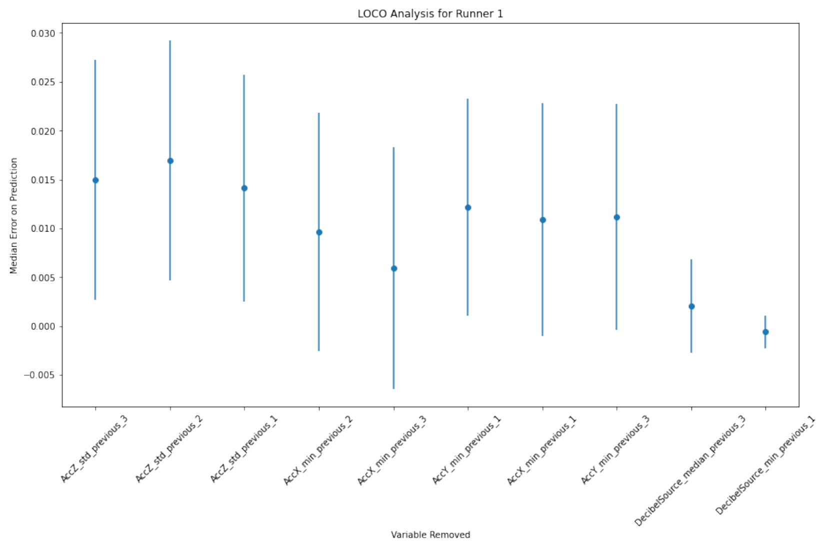
Figure 10: Variable Importance on the Poisson Regressor for each Runner.

From these plots we can distinguish between the most relevant variables for each of the runners, despite no significant difference being observed. It is also interesting to observe how the Decibels are a relevant variable for Runner 1, a feature which we initially considered would not be relevant. However, it must be noted that the position in which Runner 1 held the data recording device was closer to the ground than that of the Runner 2 and 3, which allowed for a better capturing of the noise generated while running.

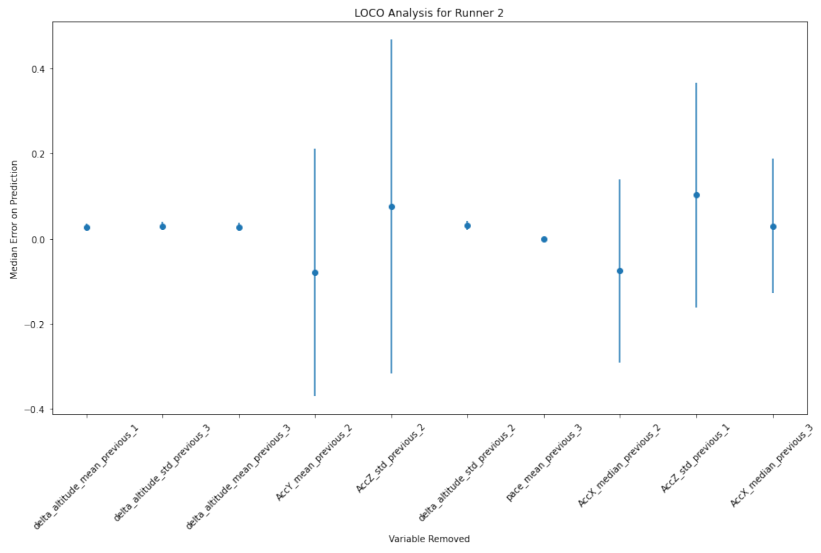
Next, we try to quantify in a more appropriate manner the actual impact that each single variable has on the final model predictions by using the LOCO method (Leave-one-covariate-out).

The results obtained from LOCO (Figure 11) are not very revealing, as none of the variables investigated seem to have a substantial overall impact on the model performance. This indicates that the model is performing well due to the large number of features and because each feature has a minor contribution on the prediction, but none of them are essential for the final result.

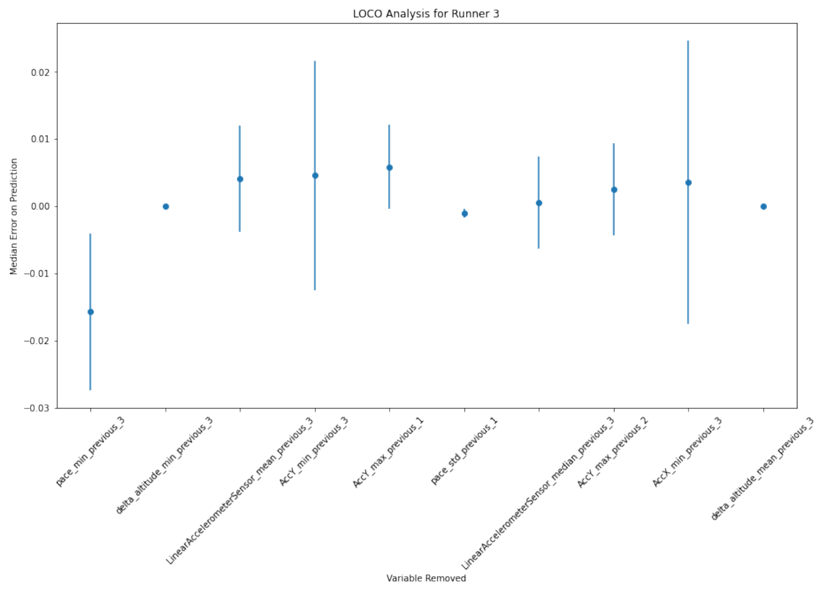
In the Figures 12, 13, 14, we show and comment an example of right prediction explained through the LIME algorithm. Again, LIME provides a local interpretation for each individual prediction. Recall that LIME looks at the neighbourhood of the prediction through the perturbation of the original features' values: what the results tell is that features can have a positive impact on the prediction (green bars in the plot), i.e. tend to increase the predicted value, or a negative impact (red bars), i.e. tend to decrease the predicted value as they vary in the given interval; as final general remark about the plots, the x axis represents the strength of the effect, while the y axis gives the interval of



(a)



(b)



(c)

Figure 11: LOCO Analysis on the Gradient Boost Decision Trees.

the considered feature. The overall results give insight on the applicability of the method and on the interpretation of the model, even if the complexity of the features' interactions doesn't allow to always easily interpret the explanation We refer to the graphs' descriptions for further details.

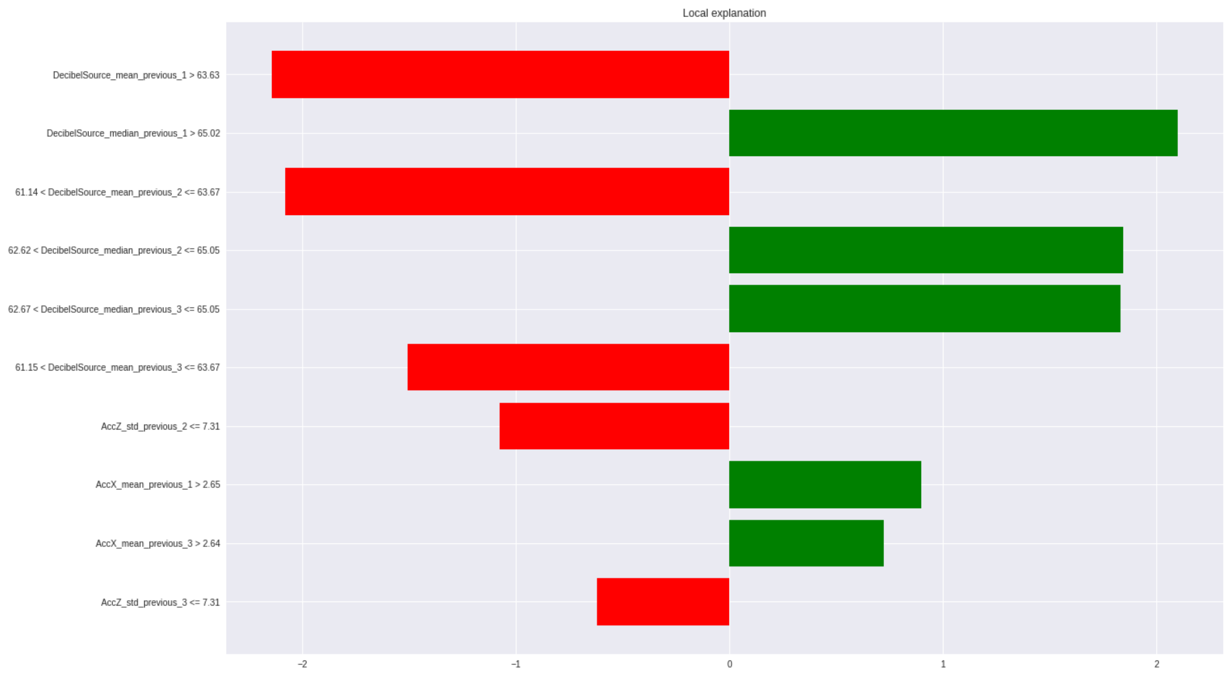


Figure 12

	0.25	0.50	0.75	1.00
DecibelSource_mean_previous_1	28.606295	61.133198	63.624916	74.869361
DecibelSource_median_previous_1	26.946124	62.621506	65.023759	75.131380
DecibelSource_mean_previous_2	28.606295	61.132084	63.625027	74.869361
DecibelSource_median_previous_2	26.946124	62.622486	65.023759	75.131380
DecibelSource_median_previous_3	26.947816	62.621392	65.022824	75.131380
DecibelSource_mean_previous_3	28.608413	61.132059	63.624588	74.869361
AccZ_std_previous_2	7.314392	8.199448	9.196135	16.482395
AccX_mean_previous_1	-2.005544	-0.924596	2.659280	5.387416
AccX_mean_previous_3	-2.003484	-0.919444	2.659128	5.387416
AccZ_std_previous_3	7.314427	8.199900	9.196640	16.482395

Table 9: Example of explanation of a right prediction of the Poisson regressor on the Runner 1 data. The table shows the values of the quartiles for the selected features. Comparing them with the plot, we can explain the prediction as follows: values of the DecibelSource_mean_previous_1 above the third quartile have a negative effect on the predicted value, while values of DecibelSource_median_previous_1 above the third quartile have a negative effect; since the DecibelSource is a very noisy feature, its impact on the prediction is not consistent, being at the same time positive and negative; the LIME explanation helps to point this out. Variations of the AccZ_std lower than the first quartile lead to a negative impact: low variability along the z axis, lower number of steps. Finally, the latter two features are self-explained, since high values of them can be easily related to an increase of the number of steps.

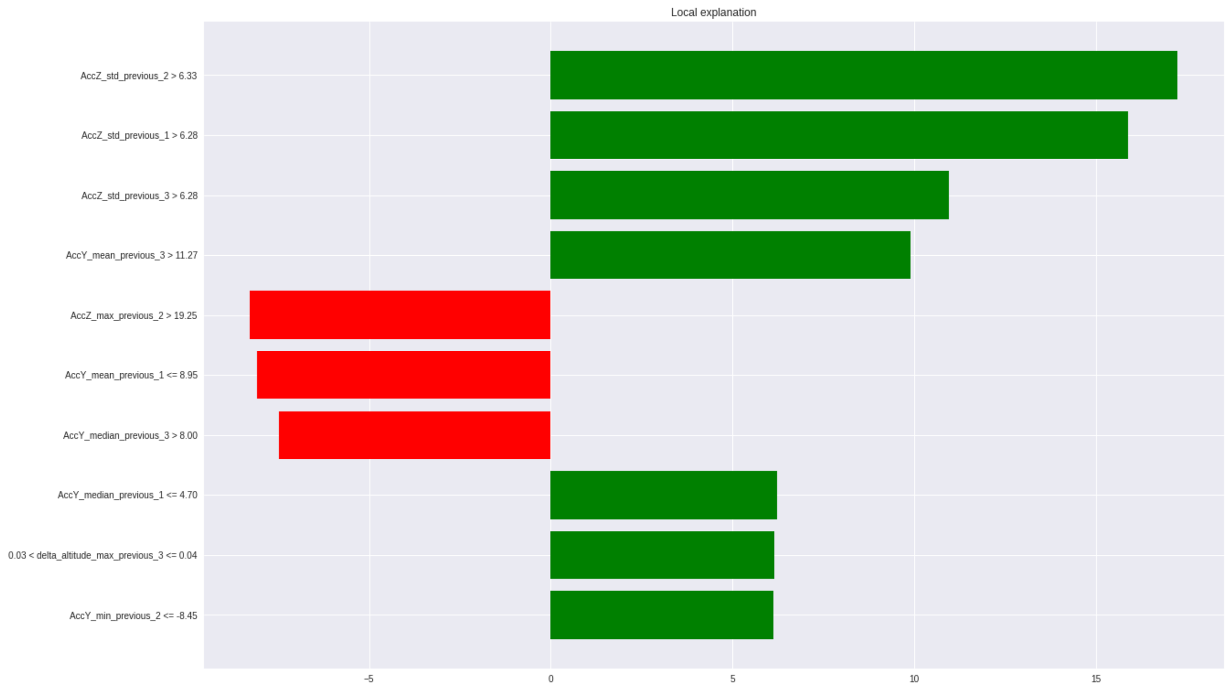


Figure 13

	0.25	0.50	0.75	1.00
AccZ_std_previous_2	4.559571	5.335836	6.220600	16.501408
AccZ_std_previous_1	4.559797	5.335836	6.219915	16.501408
AccZ_std_previous_3	4.557489	5.333084	6.220600	16.501408
AccY_mean_previous_3	8.983644	10.131218	11.320573	15.282459
AccZ_max_previous_2	12.973492	16.002213	19.169712	52.525055
AccY_mean_previous_1	8.989186	10.132677	11.327558	15.282459
AccY_median_previous_3	4.730194	6.260605	8.026151	14.738380
AccY_median_previous_1	4.731231	6.266237	8.033684	14.738380
delta_altitude_max_previous_3	0.032934	0.036948	0.051485	0.384184
AccY_min_previous_2	-8.413975	-7.371307	-6.365128	-2.696930

Table 10: Example of explanation of a right prediction of the Poisson regressor on the Runner 2 data. As in the case of the Runner 1, high variance along the z axis pushes the predicted value up; the same effect is observed for the AccY_mean_previous_3 above the third quartile. On the contrary, the next three features have a negative impact, even if their role is not completely clear, being in contrast with the other contributions. Finally, it is interesting to remark that values of the delta_max_altitude in the first half of the distribution, i.e. low difference of altitude, have a positive contribution: the slope of the ground indeed affects the trend of the run. In general, we observe a high relative strength of the top ten features with the prediction, result that is in agreement with what was observed through other methods.

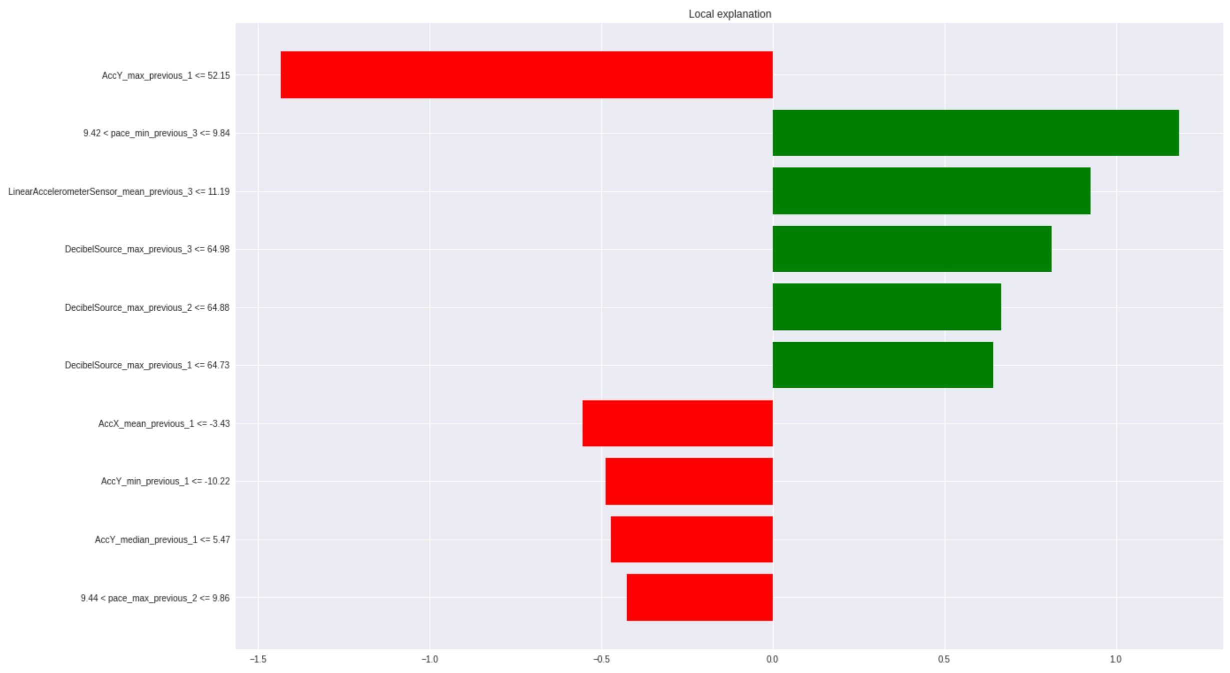


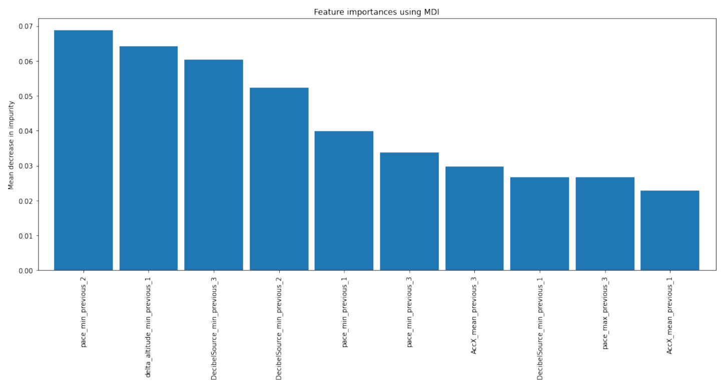
Figure 14

	0.25	0.50	0.75	1.00
AccY_max_previous_1	52.453912	60.850613	68.791840	78.121643
pace_min_previous_3	9.435942	9.852833	10.629557	188.595705
LinearAccelerometerSensor_mean_previous_3	11.189480	11.965895	12.831279	16.936035
DecibelSource_max_previous_3	64.782902	71.583838	76.430676	82.604984
DecibelSource_max_previous_2	64.782902	71.577824	76.429333	82.604984
DecibelSource_max_previous_1	64.782902	71.575564	76.429180	82.604984
AccX_mean_previous_1	-3.429515	-2.502657	-1.417658	9.862628
AccY_min_previous_1	-10.268481	-8.817657	-7.486538	9.521140
AccY_median_previous_1	5.437556	6.814412	8.230022	12.894026
pace_max_previous_2	9.444775	9.874932	10.663896	188.643988

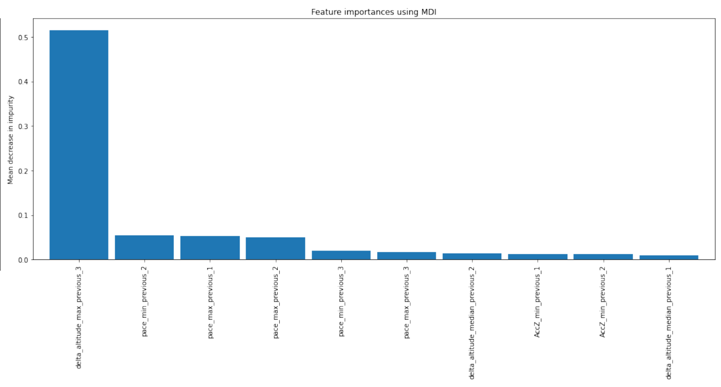
Table 11: Example of explanation of a right prediction of the Poisson regressor on the Runner 3 data. The analysis of the result proceeds the same way shown with the other two runners. It is valuable to point out that, as for Runner 1, the impact strength of the features on the prediction is much lower than the one of Runner 2, confirming the results of the variable importance in Figure 10.

7.2.2 Gradient Boosted Decision Trees:

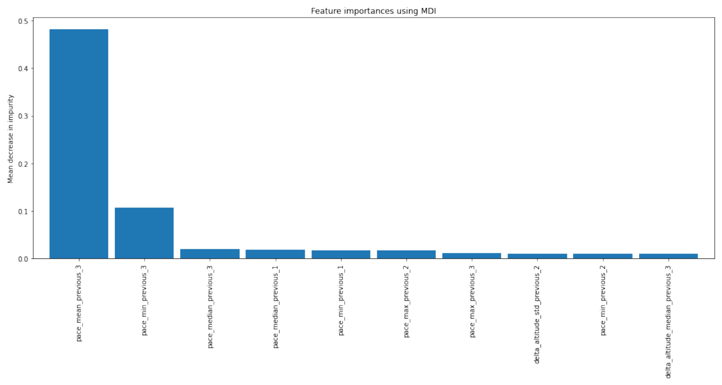
Several methods will be used to try to properly understand and interpret this model. The first step is to get a grasp of the most relevant variables, as depicted in Figure 15.



(a)



(b)



(c)

Figure 15: Variable Importance on the Gradient Boosted Decision Tree for each Runner.

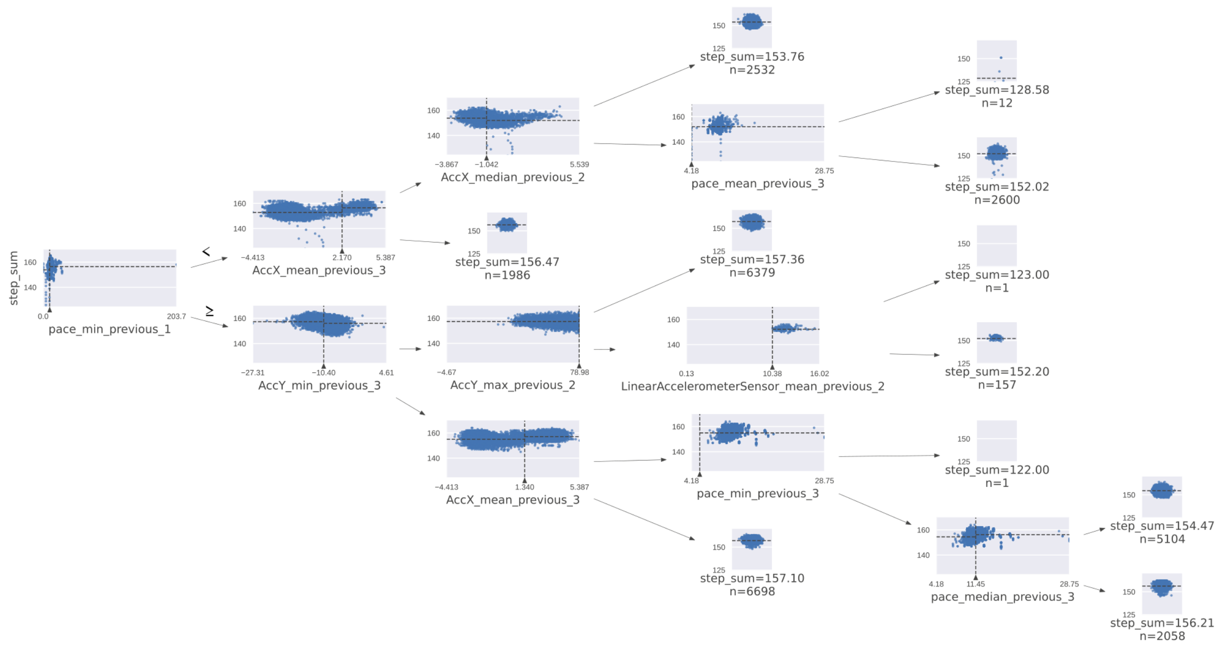
We can observe that the most relevant variables are different among the different runners. We can however observe a larger distinction between the most relevant and the least relevant features, specially for Runner 2 and 3. For Runner 1 there is no clear feature which we can denote as the most relevant one.

Decision tree based models can also be easily interpreted by looking at the paths that have lead to a certain decision. The plots in Figure 16 represent one decision tree for each of the runners.

It is worth to spend some time commenting the decision trees' examples in Figures 16a, 16b, 16c: each node of the tree has a scatter plot representation of the target variable versus each considered feature, providing insight on how separable the observations are with respect to the features: the dashed lines in each plot make the splitting value clearer. Ultimately, the leaves show the predicted values, which seem in line with the overall results previously commented, even with such shallow trees.

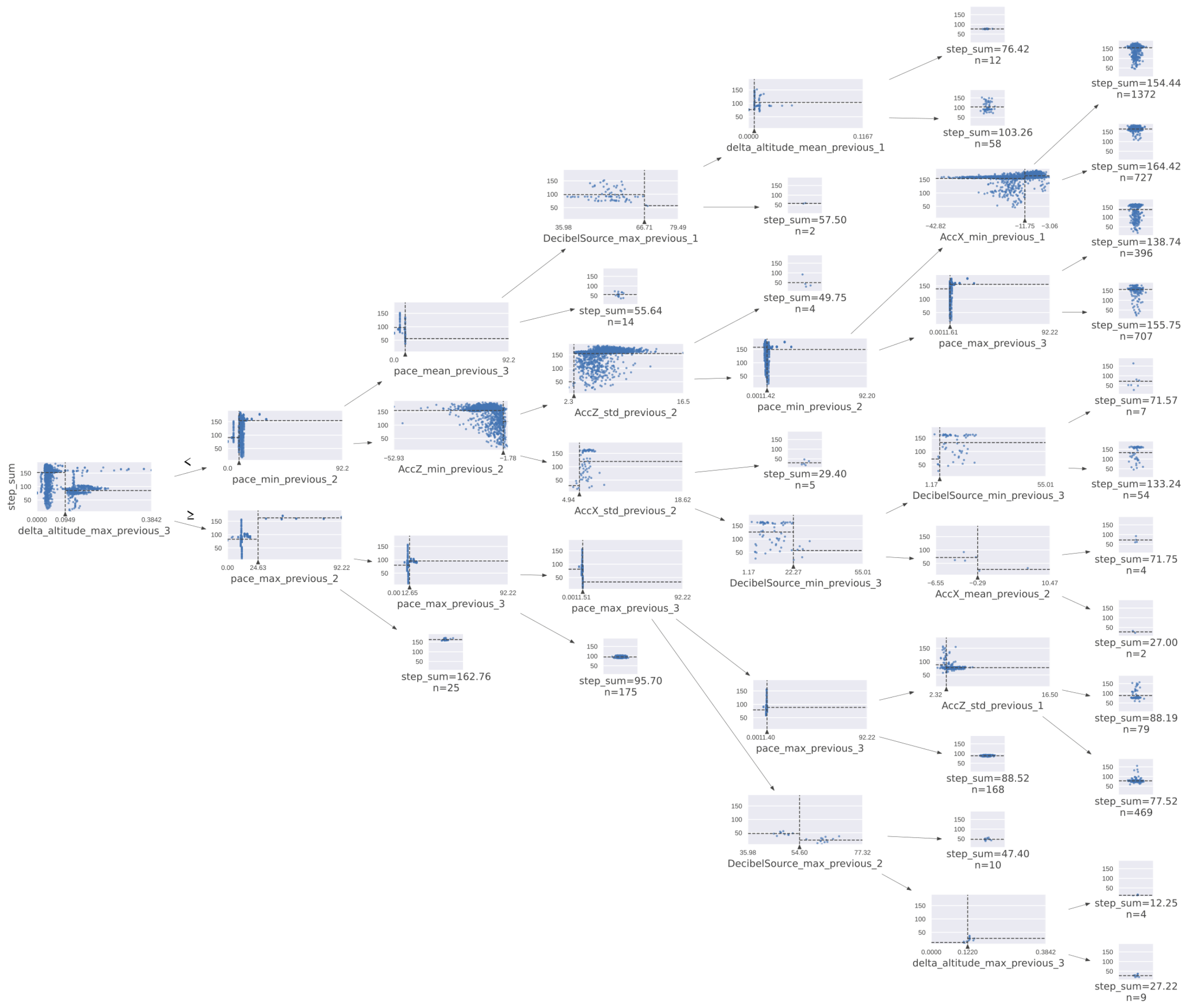
Moving forward, similarly to the Poisson case, LOCO does not conclude which variable has a higher impact on the prediction for any of the models. Model interpretability at a global scale is not obvious for any of the runners nor for any model.

Even looking locally at an example of explained prediction through LIME (Figures 18, 19, 20), the features' interactions are not clear, making us conclude that the predictions are the product of underlying dependencies among the features that vary instant by instant, describing the complex motion of each runner.


step_sum
pace_min_previous_1
<
≥
AccX_mean_previous_3
AccY_min_previous_3
AccX_median_previous_2
step_sum=156.47
n=1986
AccY_max_previous_2
AccX_mean_previous_3
step_sum=153.76
n=2532
pace_mean_previous_3
step_sum=157.36
n=6379
LinearAccelerometerSensor_mean_previous_2
pace_min_previous_3
step_sum=157.10
n=6698
step_sum=128.58
n=12
step_sum=152.02
n=2600
step_sum=123.00
n=1
step_sum=152.20
n=157
step_sum=122.00
n=1
pace_median_previous_3
step_sum=154.47
n=5104
step_sum=156.21
n=2058

(a)

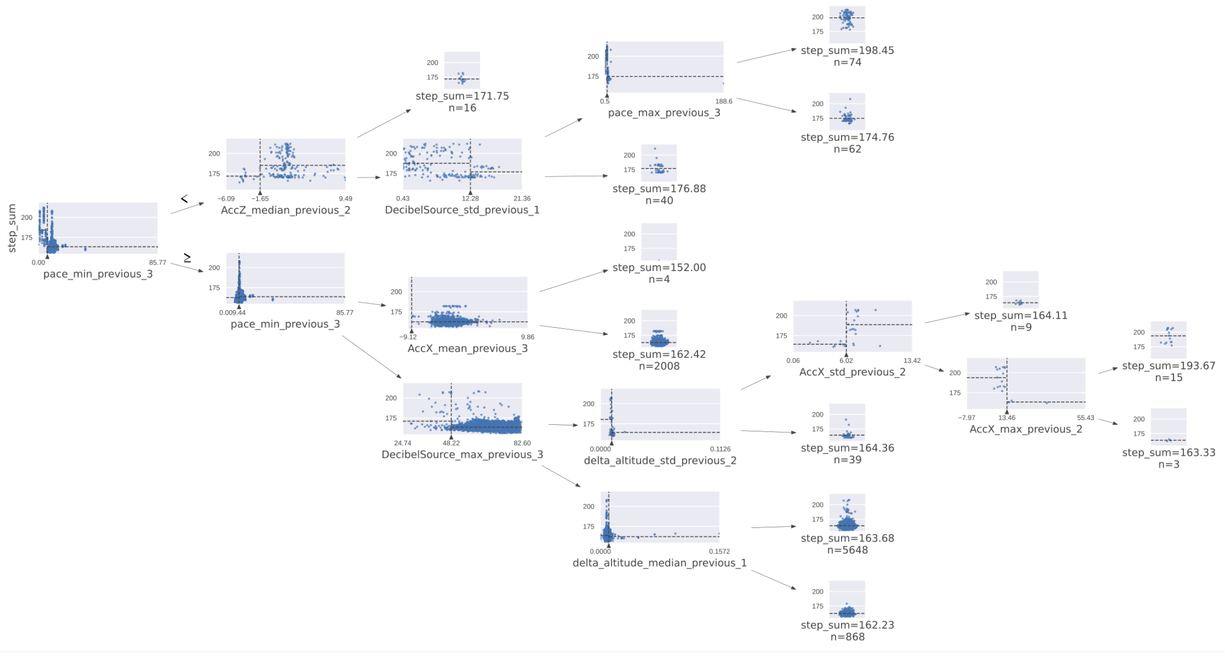
Figure 16



step_sum
delta_altitude_max_previous_3
<
≥
pace_min_previous_2
pace_max_previous_2
pace_mean_previous_3
AccZ_min_previous_2
pace_max_previous_3
step_sum=162.76
n=25
DecibelSource_max_previous_1
step_sum=55.64
n=14
AccZ_std_previous_2
AccX_std_previous_2
pace_max_previous_3
step_sum=95.70
n=175
delta_altitude_mean_previous_1
step_sum=57.50
n=2
step_sum=49.75
n=4
pace_min_previous_2
step_sum=29.40
n=5
DecibelSource_min_previous_3
pace_max_previous_3
DecibelSource_max_previous_2
step_sum=76.42
n=12
step_sum=103.26
n=58
AccX_min_previous_1
pace_max_previous_3
DecibelSource_min_previous_3
AccX_mean_previous_2
AccZ_std_previous_1
step_sum=88.52
n=168
step_sum=47.40
n=10
delta_altitude_max_previous_3
step_sum=154.44
n=1372
step_sum=164.42
n=727
step_sum=138.74
n=396
step_sum=155.75
n=707
step_sum=71.57
n=7
step_sum=133.24
n=54
step_sum=71.75
n=4
step_sum=27.00
n=2
step_sum=88.19
n=79
step_sum=77.52
n=469
step_sum=12.25
n=4
step_sum=27.22
n=9

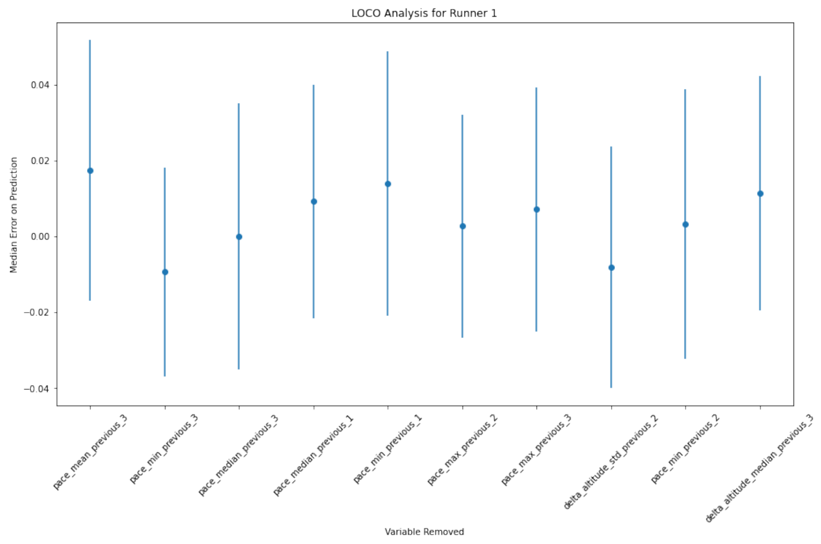
(b)

Figure 16

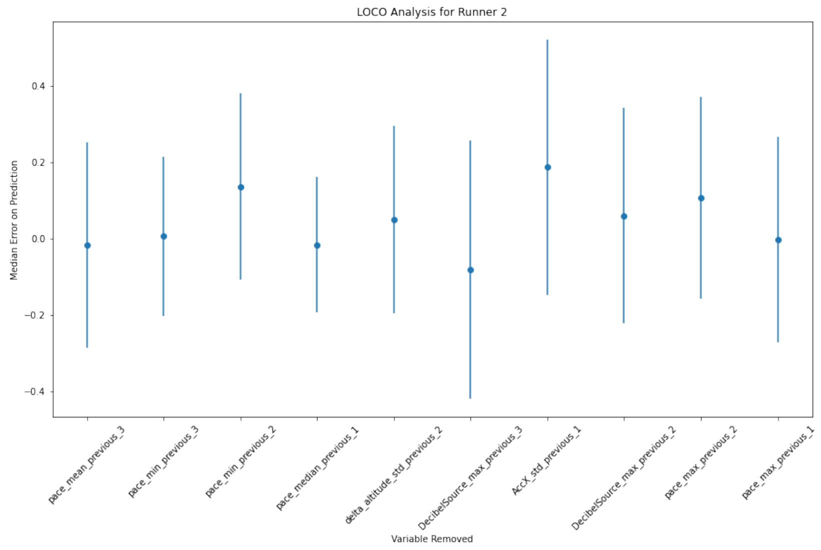


(c)

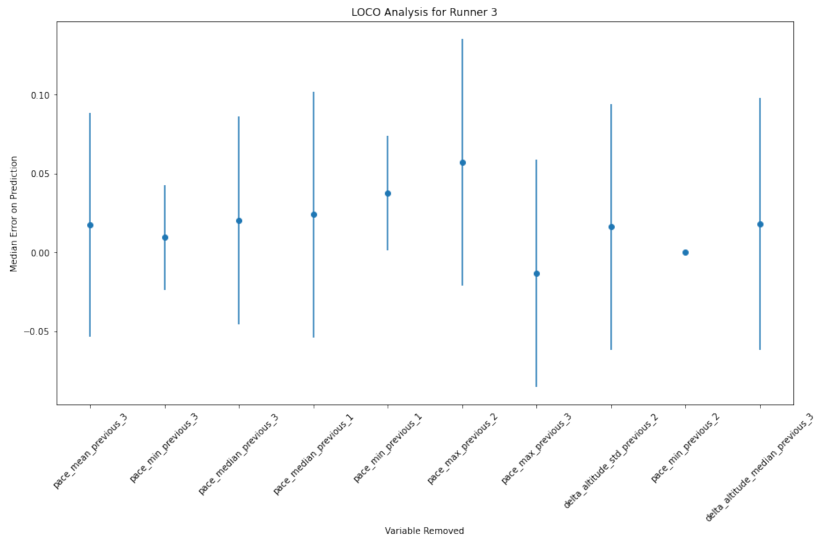
Figure 16: Visualisation of a decision tree of the Gradient Boosted Decision Trees regressor for the Runners 1 (16a), 2 (16b) and 3 (16c).



(a)



(b)



(c)

Figure 17: LOCO Analysis on the Gradient Boost Decision Trees.

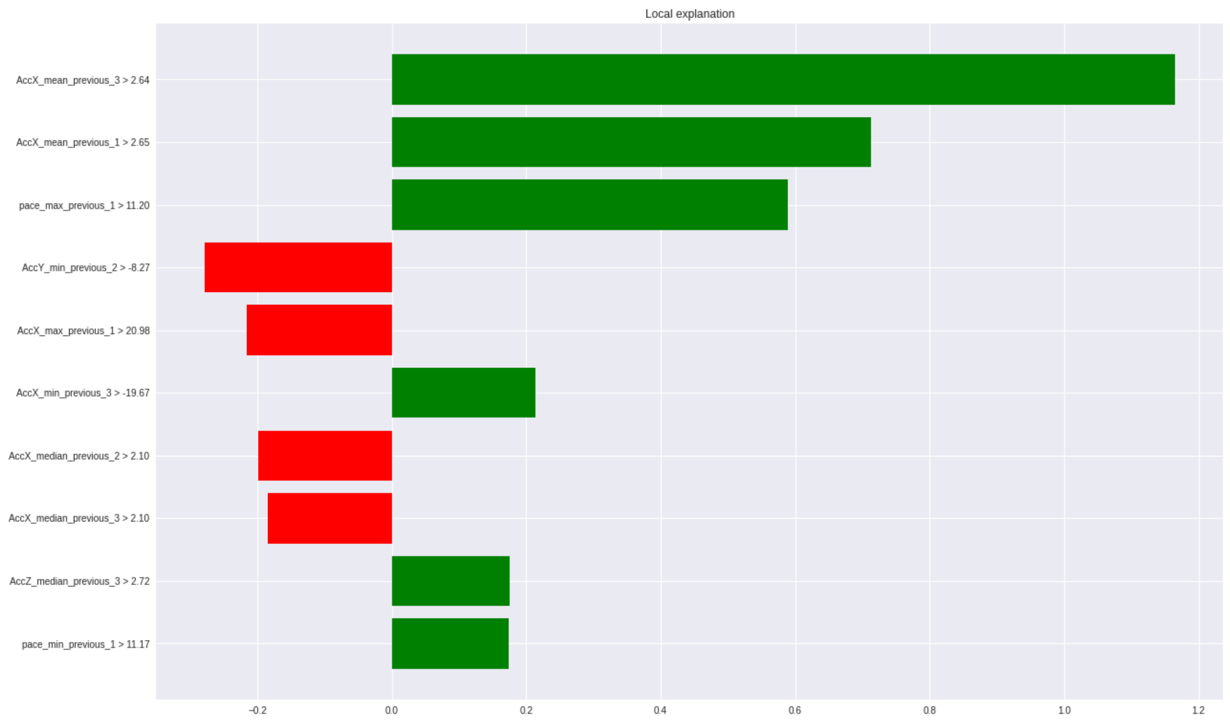


Figure 18

	0.25	0.50	0.75	1.00
AccX_mean_previous_3	-2.003484	-0.919444	2.659128	5.387416
AccX_mean_previous_1	-2.005544	-0.924596	2.659280	5.387416
pace_max_previous_1	9.863627	10.470661	11.208103	203.755353
AccY_min_previous_2	-10.775879	-9.419754	-8.246704	7.309311
AccX_max_previous_1	11.429459	16.223160	21.017448	75.625000
AccX_min_previous_3	-30.147972	-24.556152	-19.710022	0.589813
AccX_median_previous_2	-1.405139	-0.351898	2.104737	5.539478
AccX_median_previous_3	-1.404799	-0.350708	2.104737	5.539478
AccZ_median_previous_3	1.503219	2.108658	2.722488	8.378242
pace_min_previous_1	9.833796	10.446246	11.180259	203.703201

Table 12: Example of explanation of a right prediction of the Gradient Boosted Decision Trees regressor on the Runner 1 data.

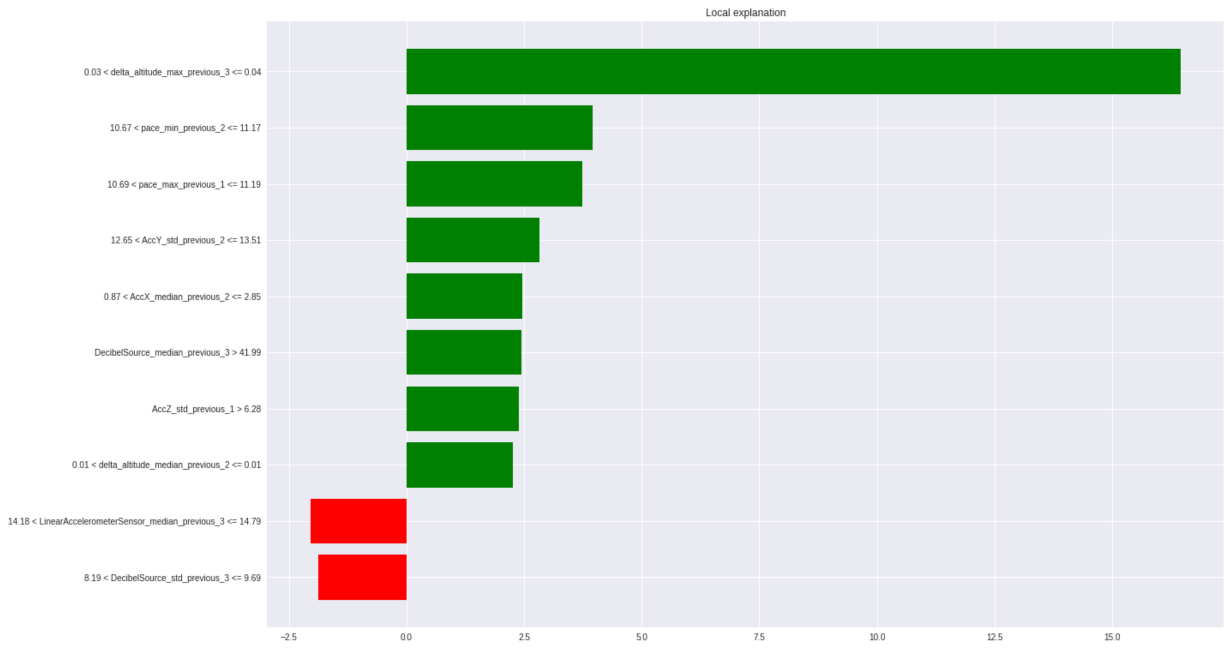


Figure 19

	0.25	0.50	0.75	1.00
delta_altitude_max_previous_3	0.032934	0.036948	0.051485	0.384184
pace_min_previous_2	10.688981	11.172398	11.499987	92.197823
pace_max_previous_1	10.702999	11.216108	11.531142	92.221427
AccY_std_previous_2	12.648713	13.533451	14.354262	17.775650
AccX_median_previous_2	-2.100231	0.760387	2.921104	8.744235
DecibelSource_median_previous_3	34.068169	37.739332	42.061704	64.246932
AccZ_std_previous_1	4.559797	5.335836	6.219915	16.501408
delta_altitude_median_previous_2	0.011816	0.012629	0.014674	0.102449
LinearAccelerometerSensor_median_previous_3	14.183056	14.790767	15.337957	21.531273
DecibelSource_std_previous_3	8.164489	9.703077	11.268200	19.118404

Table 13: Example of explanation of a right prediction of the Gradient Boosted Decision Trees regressor on the Runner 2 data.

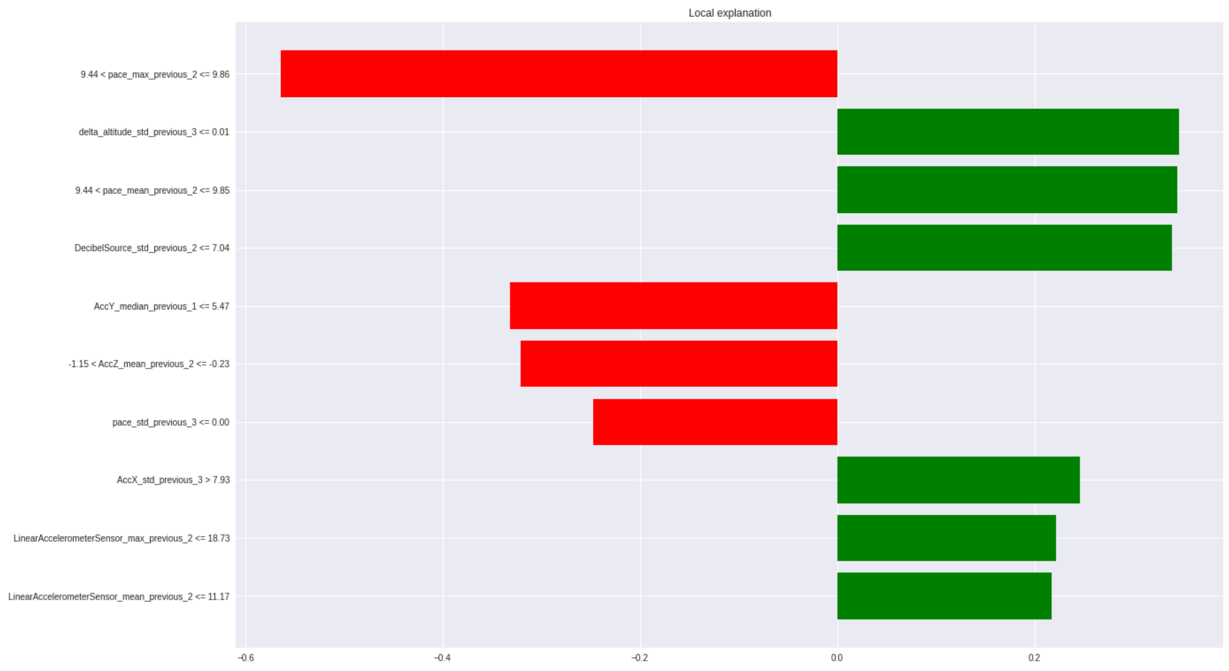


Figure 20

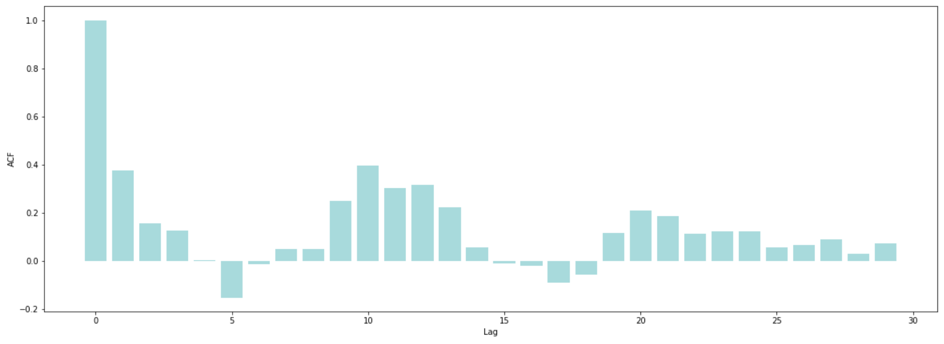
	0.25	0.50	0.75	1.00
pace_max_previous_2	9.444775	9.874932	10.663896	188.643988
delta_altitude_std_previous_3	0.009470	0.010483	0.011628	0.213798
pace_mean_previous_2	9.440547	9.866907	10.642766	188.608152
DecibelSource_std_previous_2	7.099582	9.370426	12.033387	21.364426
AccY_median_previous_1	5.437556	6.814412	8.230022	12.894026
AccZ_mean_previous_2	-1.162231	-0.232943	0.641774	9.492944
pace_std_previous_3	0.000662	0.000711	0.000775	89.619023
AccX_std_previous_3	6.163269	7.026168	7.979630	13.427067
LinearAccelerometerSensor_max_previous_2	18.773299	21.482191	24.431895	40.939379
LinearAccelerometerSensor_mean_previous_2	11.192041	11.967790	12.833475	16.936035

Table 14: Example of explanation of a right prediction of the Gradient Boosted Decision Trees regressor on the Runner 3 data.

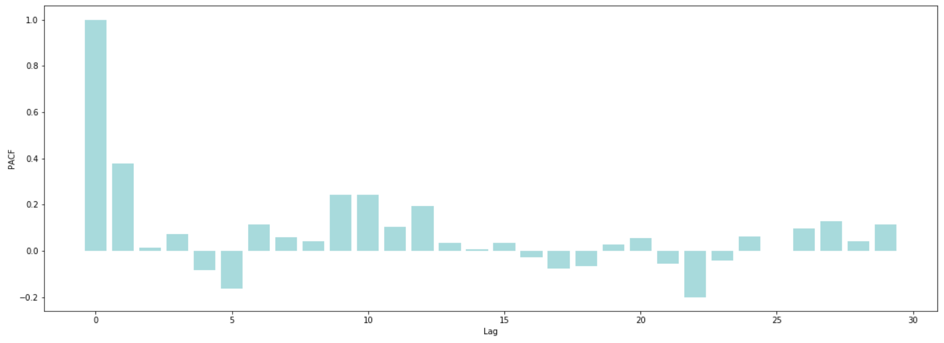
7.2.3 ARIMA

ARIMA models try to find and describe patterns through a variable number of parameters that entirely depends on the target variable complexity, which already excludes any kind of interpretability limit related to the number of features. Moreover, ARIMA's parameters can often be explained graphically and numerically, especially when dealing with regular and well behaved time series. As mentioned in section 5.1, all the parameters describing ARIMA models relate to data itself, to some extent. More specifically, we can retrieve important information on time series through the Augmented Dickey-Fuller, ACF and PACF. Let's dig on how these three analysis can return an accurate information on the number of parameters needed for the AR, I and MA models:

- AR: If we consider a time series that was generated by an autoregression process with a lag of k, we expect the ACF to be strong to a lag of k and an inertial behaviour would continue to subsequent lag values with a weakened effect. Since the PACF only describes the direct relationship between an observation and its lag, there should be no correlation for lag values beyond k.
- I: This parameter can be defined by the level of differentiation that guarantees the stationarity of the time series, which can be tested by the Augmented Dickey-Fuller.
- MA: If we consider a time series that was generated by a moving average process with a lag of k, we would expect the ACF to show a strong correlation with recent values (up to the lag of k), then a sharp decline to low or no correlation. For the PACF, we would expect the plot to show a strong relationship to the lag and a trailing off of correlation from the lag onwards.



(a)



(b)

Figure 21: Runner 1 ACF(a) and (b)PACF with order (1,0,2)

We can observe the expected behaviour indeed: coherently with the order obtained for Runner 1 (1,0,2), we have a sharp decrease in both the ACF and the PACF after lag 1 and small or no correlation after lag 2.

To summarize, even though parameters tuning is probably the most reliable way of getting the optimized parameters values, those can easily be inferred or confirmed from data as well, making ARIMA models highly interpretable.

8 Conclusions

From the results we can conclude that the main objective of this project, which is to estimate the steps per minute, can be attained with a very low error rate. We have been able to achieve this through numerous approaches. The two main approaches followed in this project were assuming independence and dependence (as explained in detail throughout the report). None of the two approaches have proven to provide the best results in all cases, on the contrary they have provided really similar results in most cases, with the predictions for Runner 2 being the less precise ones. Overall, we consider the time dependence method to be the most appropriate for this analysis since it considers all the past data and can capture underlying trends which we would not be able to identify otherwise.

On the other hand, assuming *almost* independence among observations has also delivered very promising results and we can confidently say that this approach could also estimate the results in a very accurate fashion (overall, the Gradient Boost Decision Tree was able to return the best results among all runners and models). These good results in terms of RMSE however have a downside: interpretability. Due to the high number of variables the global model interpretability has significantly declined. This can be observed from the LOCO analysis and from the fact that there is no clear distinction between important and less important features (features importance is rather flat throughout all the model variables). This has lead us to explore other local interpretability methods such as LIME. This method allowed us to get a better intuition on the model behaviour.

In terms of future work there are several things that could be improved upon:

- Homogenize the data collection phase, i.e. set a fixed position for the phone. This would have allowed us to better compare model performance and obtain a better understanding of the feature variables.
- Add additional features such as heart beats and environmental conditions that might influence the runners performance.
- Integrate predictions with actual music and see if the performance of the runner is improved with customized playlists.

References

[1] Athanasopoulos, G., Hyndman R. (2018). *Forecasting: Principles and Practice*

[2] Bäärnhielm, A. (2017). *Multiple time-series forecasting on mobile network data using an RNN-RBM model*

[3] Barbieri, S., Kemp, J., Perez-Concha, O. et al. *Benchmarking Deep Learning Architectures for Predicting Readmission to the ICU and Describing Patients-at-Risk.* Sci Rep 10, 1111 (2020). https://doi.org/10.1038/s41598-020-58053-z

[4] Bly, Kristopher, *The effect of music playlist tempo on self-paced running, mood, and attentional focus tendencies* (2013), Ithaca College Theses. 12. https://digitalcommons.ithaca.edu/ic_theses/12

[5] Buhmann J, Moens B, Van Dyck E, Dotov D, Leman M (2018) *Optimizing beat synchronized running to music.* PLoS ONE 13(12): e0208702. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0208702

[6] Laugel, Thibault & Lesot, Marie & Marsala, Christophe & Renard, Xavier & Detyniecki, Marcin. (2017). *Inverse Classification for Comparison-based Interpretability in Machine Learning.*

[7] L. Pantiskas, K. Verstoep and H. Bal, *"Interpretable Multivariate Time Series Forecasting with Temporal Attention Convolutional Neural Networks,"* 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 2020, pp. 1687-1694, doi: 10.1109/SSCI47803.2020.9308570.

[8] Katardjiev, N. (2018). *High-variance multivariate time series forecasting using machine learning (Dissertation).* Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-353827

[9] Karimi, A., Barthe, G., Balle, B. & Valera, I.. (2020). *Model-Agnostic Counterfactual Explanations for Consequential Decisions. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, in *Proceedings of Machine Learning Research* 108:895-905 Available from http://proceedings.mlr.press/v108/karimi20a.html

[10] Parmezan, Antonio. (2019). *Re: What are the best machine learning algorithms for time series forecasting?*. Retrieved from: https://www.researchgate.net/post/What-are-the-best-machine-learning-algorithms-for-time-series-forecasting/5d7e74010f95f1bedb676868/citation/download.

[11] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. 2020. *Causal Interpretability for Machine Learning - Problems, Methods and Evaluation.* SIGKDD Explor. Newsl. 22, 1 (June 2020), 18–33. DOI:https://doi.org/10.1145/3400051.3400058

[12] Rudin, C. *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.* Nat Mach Intell 1, 206–215 (2019). https://doi.org/10.1038/s42256-019-0048-x

[13] Wagner, N., Michalewicz, Z., Schellenberg, S., Chiriac, C. and Mohais, A. (2011), *"Intelligent techniques for forecasting multiple time series in real-world systems"*, International Journal of Intelligent Computing and Cybernetics, Vol. 4 No. 3, pp. 284-310. https://doi.org/10.1108/17563781111159996

[14] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. *""Why should i trust you?" Explaining the predictions of any classifier."* Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.

A Classification - Prediction Results

Runner	Estimator	Train RSME	Test RMSE	Test Accuracy	Test F1
1	GaussianNB	4.210	4.221	0.144	0.109
1	RandomForestClassifier	1.501	1.360	0.480	0.476
1	RidgeClassifier	6.454	6.523	0.112	0.132
1	SVC	3.181	3.351	0.102	0.092
2	LogisticRegression	22.780	21.852	0.162	0.141
2	RandomForestClassifier	15.438	15.593	0.257	0.235
2	RidgeClassifier	41.582	41.749	0.020	0.021
2	SVC	23.799	23.135	0.155	0.161
2	XGBClassifier	16.843	16.567	0.260	0.239

Table 15: Some results obtained through the classification approach. The values presented in this table were computed on the same dataset used for the regression task, in order to be consistent and let compare the results. A previous analysis, not shown in the report, was conducted on fine-grained values of the number of steps, which was computed in time intervals of 1s, 2s, 5s. Clearly, this way we had much less classes (i.e. values of the number of steps per second) and hence much higher values of accuracy and F1-score; as pointed out throughout the report, this approach turned out not to be robust and consistent.