Predicting Running Performance using Machine Learning

Hippolyte Gisserot-Boukhlef

June 19, 2022

Abstract

Many specialists consider that the most critical aspect in running performance is to be able to calibrate properly the athlete's training plan. Either at an amateur or a professional level, a fair number of counter-performances are still due to potential under- or overestimation, preventing the runner from practicing at the most relevant pace.

This issue is even more present when the athlete prepares a brand-new distance, on which he/she has no reference. Let's take the example of a runner who has a 90 minutes personal best on half-marathon (21.097 km, 13.1 miles) and who wishes to run his/her first marathon (42.195 km, 26.2 miles). What would be a reasonable goal regarding his/her half-marathon level? 3h10? 3h20? 3h30? The answer depends on many factors such as, of course, the athlete's past performances on other distances (on average, faster runner on half-marathon logically perform better on marathon), but also his/her age (older runners tend to perform relatively better over longer distances), weight (heavier runners tend to struggle more as the distance increases), gender, invisible genetics...

Our objective then will be to try and find the main performance factors and develop a powerful prediction tool for runners who wish to start their career on a new distance. To do this, we decided to use the self-service databases of the French Athletics Federation and rely on a bunch of machine learning algorithms to understand and explain the vast amount of collected information: linear regression, nearest neighbors, decision trees, random forests, (extreme) gradient boosting and neural networks.

Using some relevant evaluation metrics that we detailed below, we identified the random forest algorithm to be the most efficient on the assigned task. And more than that, we also noticed that this algorithm exceeded in terms of precision the official French Athletics Federation scale, which is used in any accredited competitions when it is necessary to establish equivalences between different disciplines. Given these very encouraging results, we decided to create our own running performance prediction tool.

1 Data pre-processing

1.1 Web-scrapping

To conduct our analysis, we decided to use the self-service databases of the French Athletics Federation [1], which give access to all the performances of any athlete (amateur or professional) who participated in races accredited by the federation, and this since their beginnings in running.

After scrapping, a first set of raw data was extracted. It has the following columns: runner's ID, birthdate, height (cm), weight (kg), gender, distance, performance and date. Each row then corresponds to one given performance (see Table 1).

ID	Birthdate	Height	Weight	Gender	Distance	Time	Date
0	1997	-	-	M	800m	1'45"94	44366
0	1997	-	-	M	800m	1'45"03	43663
0	1997	-	-	M	800m	1'45"35	43334
0	1997	-	-	M	800m	1'45"76	42860
0	1997	-	-	M	800m	1'45"05	42525
...	...	...	...	...	...	...	...
106948	1957	-	-	M	100 Km Route	19h14'50"	42637
106948	1957	-	-	M	100 Km Route	18h38'43"	42273
106948	1957	-	-	M	24 Heures	109km824m	44381
106948	1957	-	-	M	24 Heures	103km420m	43261
106949	1966	-	-	M	100 Km Route	19h59'44"	43736

Table 1: Raw data set structure

1.2 Data organization

Next, we had to reorganize this data set in order to make it usable for the analysis we wanted to conduct. The idea was, for each individual, to identify his/her whole time personal bests over all available distances and then look back in the past at the various personal bests achieved over other distances (these will be used as performance predictors).

As a result, we ended up with the following sets of variables:

- Explanatory (X) variable:
  - $PredDist$: distance on which performance is to be predicted (e.g.: if I want to predict a performance over the marathon distance, this variable will take the value 42.195)
  - $AgePred$: projected age at the time of the predicted race (e.g.: if I plan to run this marathon race at age 30, then the variable will take value 30)
  - $PerfTime$: personal best, in seconds, over the distance used to predict future performance (e.g.: if I ran a 10km race in 35 minutes, then this variable will take value $35 \times 60 = 2100$)
  - $PerfDist$: distance over which the personal best used for prediction was ran (e.g.: in that specific case, this variable will be equal to 10).
  - $DeltaAgePerf$: number of years that separate the personal best from the predicted performance (e.g.: if I set my 10km personal best 2 years ago, then this variable will be set equal to 2)
    Note: we decided to limit this variable to 3, in order to maintain a certain homogeneity in the level of performance
  - $Height$ (in centimeters)
  - $Weight$ (in kilograms)
  - $Gender$: dummy variable set equal to 1 when the individual was a man and 0 if she was a woman.
  - $Alpha$, $Beta$ and $IsAlphaBeta$:
    A characteristic element of running performance is the profile of the runner. A runner is said to be a "speedster" if he or she is relatively more efficient over short distances, and an "endurance monster" if on the contrary his or her potential is expressed more when distances increase. Note that, for a given athlete, we can reasonably model performance as a function of distance as follows:

$$Time = \gamma Distance^{\beta} \iff \ln Time = \ln \gamma + \beta \ln Distance = \alpha + \beta \ln Distance$$

$$\text{Where } \alpha > 0 \text{ and } \beta > 1$$

    Therefore, each time an athlete had at least 2 personal bests to explain a future performance, we could compute the α and β coefficients by running a simple linear regression. Then, the $IsAlphaBeta$ variable was set to 1. Otherwise, $Alpha$, $Beta$ and $IsAlphaBeta$ were set equal to 0.

- Response (y) variable: performance, in seconds, over the distance of interest (given by the *PredDist* variable).

Finally, we got a data set with a total of 93,029 rows and 12 columns (see Table 2), that we divided into training and test sub-samples (respectively 75%/25% of the whole data set).

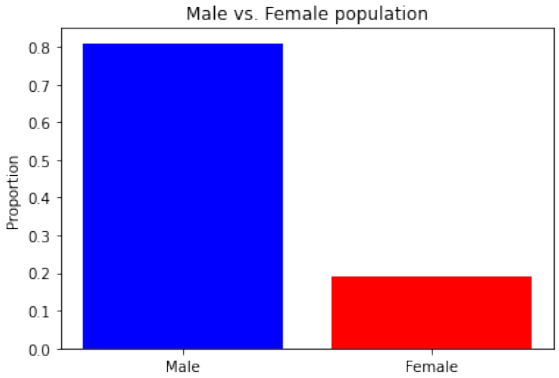
PerformanceID	y	PredDist	AgePred	PerfTime	PerfDist	DeltaAgePerf	Height	Weight	Gender	Alpha	Beta	IsAlphaBeta
29749	2301	10	51	1076.4	5	2	164	57	1	0	0	0
52283	2326	10	36	11146.0	42.195	1	183	75	1	0	0	0
28369	117.91	0.8	20	159.5	1	1	185	65	1	5.067339	1.122975	1
28369	117.91	0.8	20	242.3	1.5	0	185	65	1	5.067339	1.122975	1
28369	117.91	0.8	20	551.7	3	1	185	65	1	5.067339	1.122975	1
...	...	...	...	...	...	...	...	...	...	...	...	...
42507	2265	10	41	5186.7	21.097	2	170	52	1	5.376283	1.042123	1
42507	2265	10	41	10681.0	42.195	2	170	52	1	5.376283	1.042123	1
11559	131.34	0.8	19	173.6	1	1	163	50	0	5.146454	1.116322	1
11559	131.34	0.8	19	265.8	1.5	0	163	50	0	5.146454	1.116322	1
11559	131.34	0.8	19	589.3	3	1	163	50	0	5.146454	1.116322	1

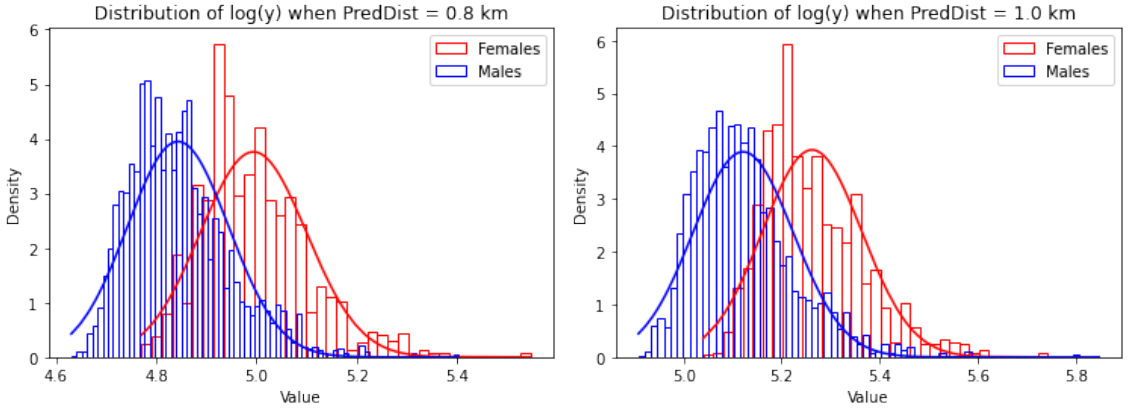
Table 2: Processed data set structure

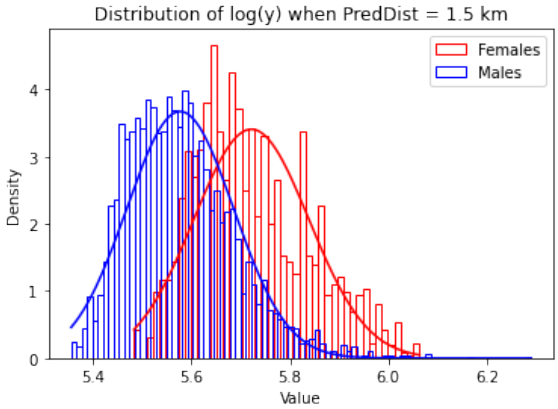
2 Data analysis

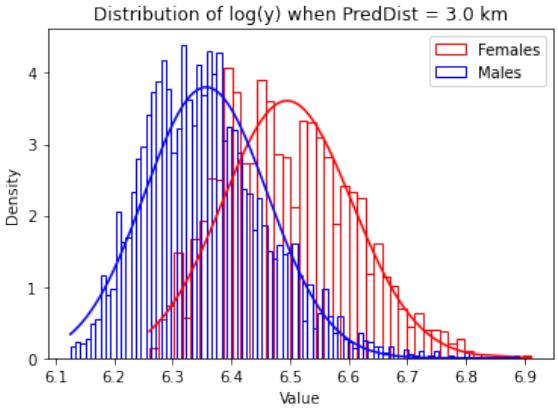
2.1 Exploratory data analysis

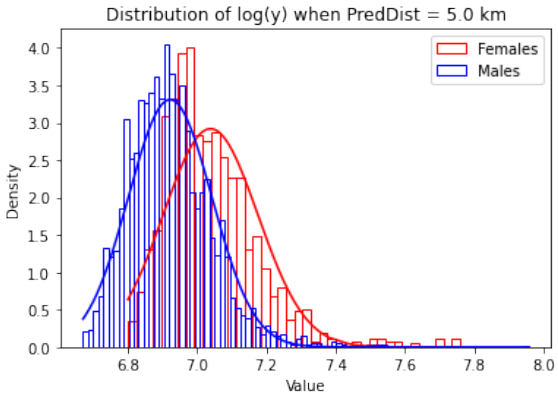
Before starting any analyses properly said, we decided to have a brief graphical look at how the data behaves, feature by feature.

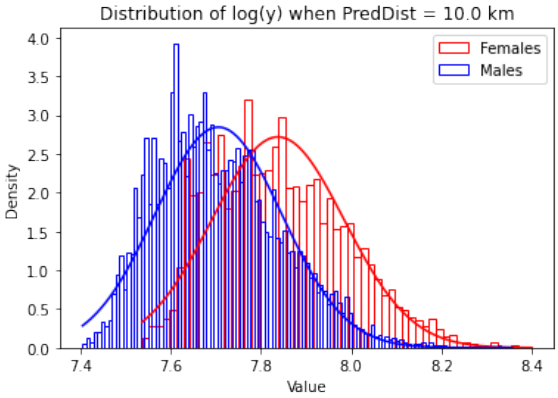


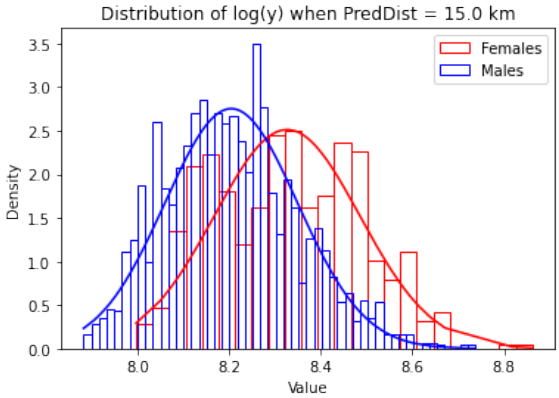


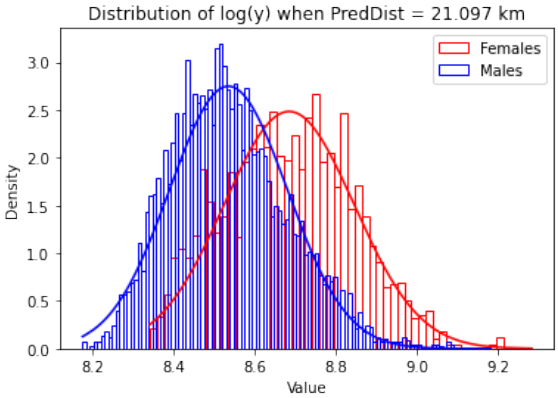

Distribution of log(y) when PredDist = 1.5 km
Females
Males
Density
Value

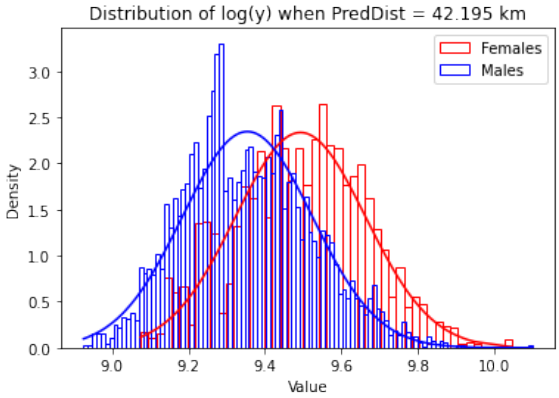

Distribution of log(y) when PredDist = 3.0 km
Females
Males
Density
Value

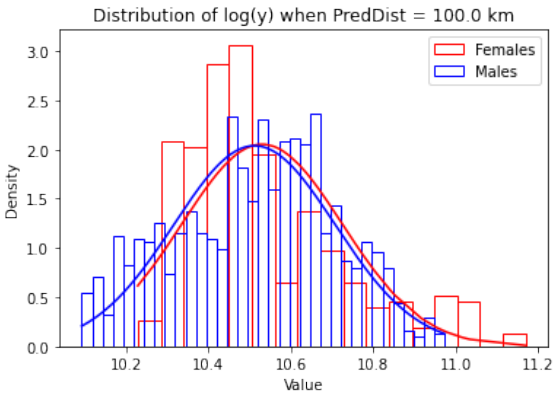

Distribution of log(y) when PredDist = 5.0 km
Females
Males
Density
Value

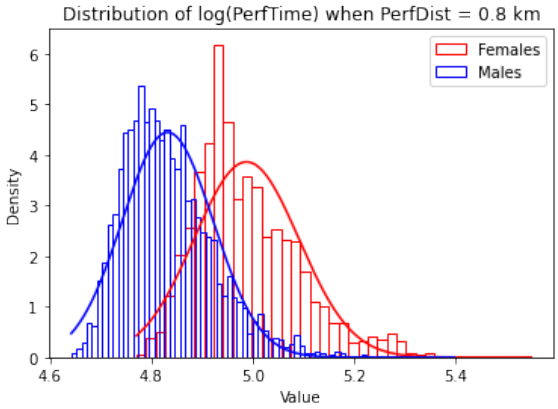

Distribution of log(y) when PredDist = 10.0 km
Females
Males
Density
Value

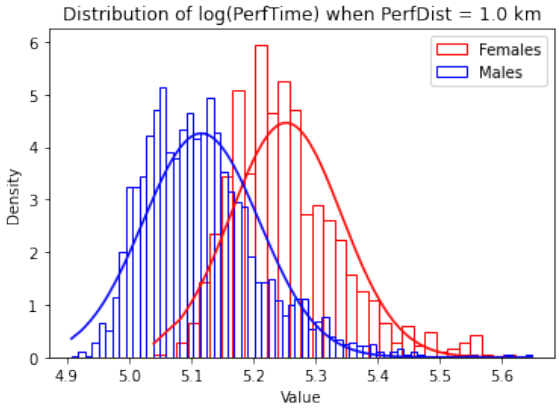

Distribution of log(y) when PredDist = 15.0 km
Females
Males
Density
Value

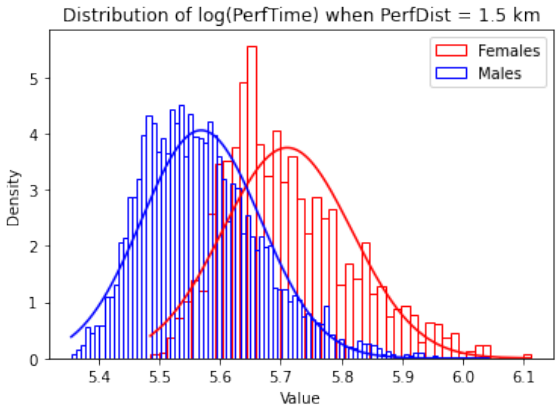

Distribution of log(y) when PredDist = 21.097 km
Females
Males
Density
Value

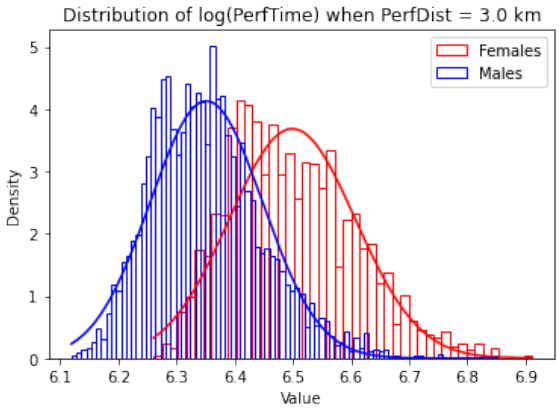

Distribution of log(y) when PredDist = 42.195 km
Females
Males
Density
Value

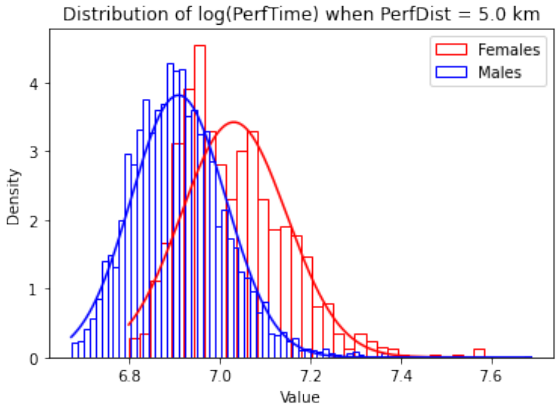

Distribution of log(y) when PredDist = 100.0 km
Females
Males
Density
Value

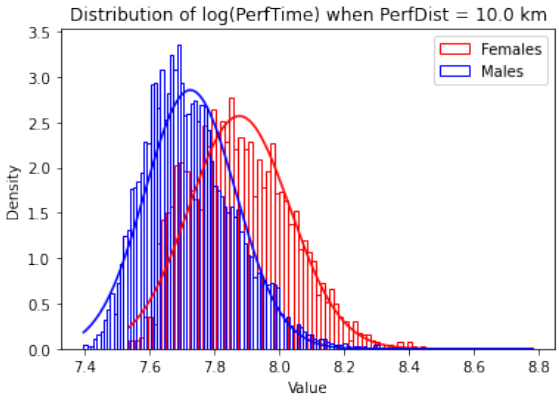

Distribution of log(PerfTime) when PerfDist = 0.8 km
Females
Males
Density
Value

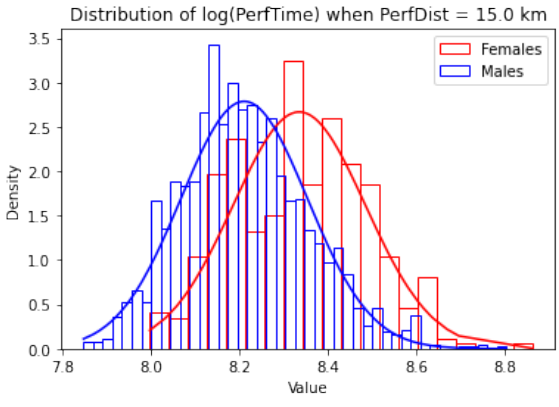

Distribution of log(PerfTime) when PerfDist = 1.0 km
Females
Males
Density
Value

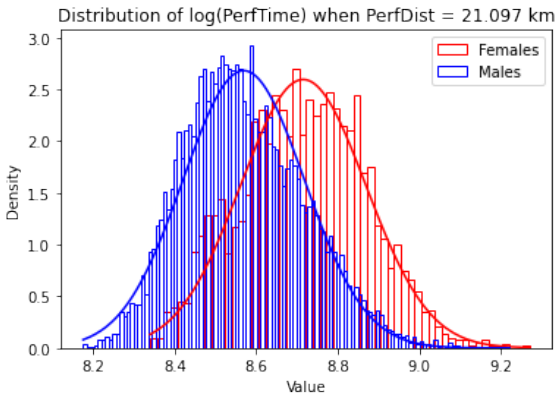

Distribution of log(PerfTime) when PerfDist = 1.5 km
Females
Males
Density
Value

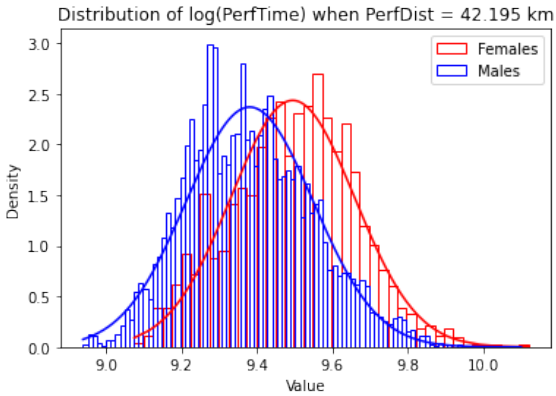

Distribution of log(PerfTime) when PerfDist = 3.0 km
Females
Males
Density
Value

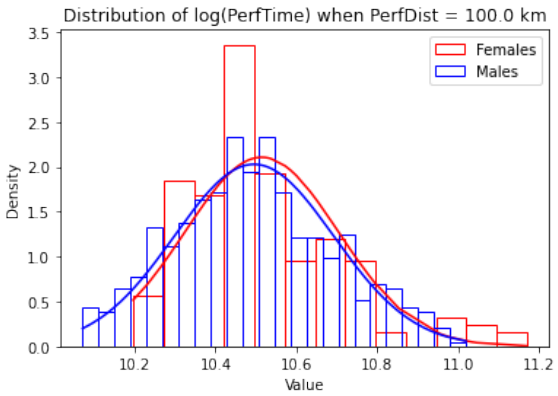

Distribution of log(PerfTime) when PerfDist = 5.0 km
Females
Males
Density
Value

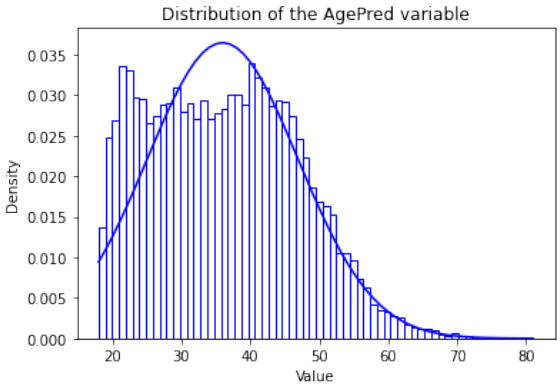

Distribution of log(PerfTime) when PerfDist = 10.0 km
Females
Males
Density
Value

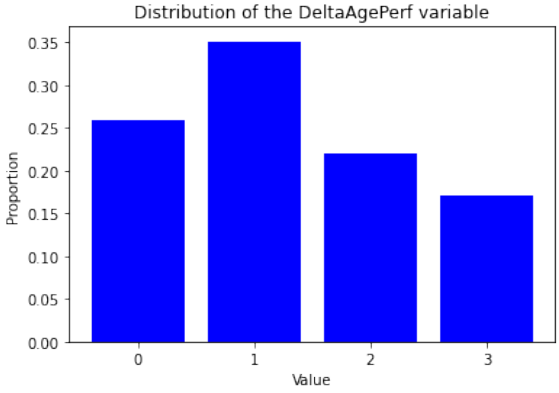

Distribution of log(PerfTime) when PerfDist = 15.0 km
Females
Males
Density
Value

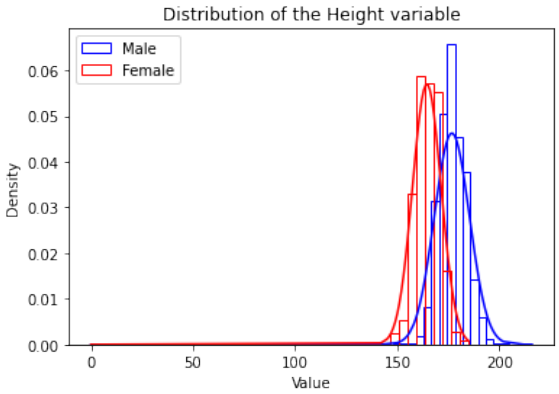

Distribution of log(PerfTime) when PerfDist = 21.097 km
Females
Males
Density
Value

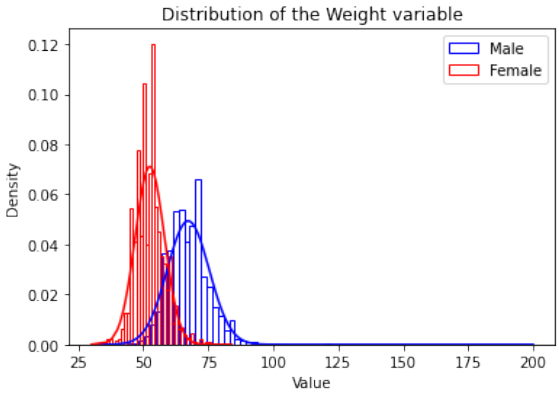

Distribution of log(PerfTime) when PerfDist = 42.195 km
Females
Males
Density
Value

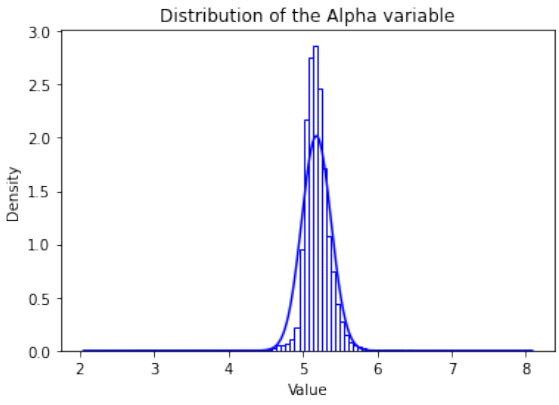

Distribution of log(PerfTime) when PerfDist = 100.0 km
Females
Males
Density
Value

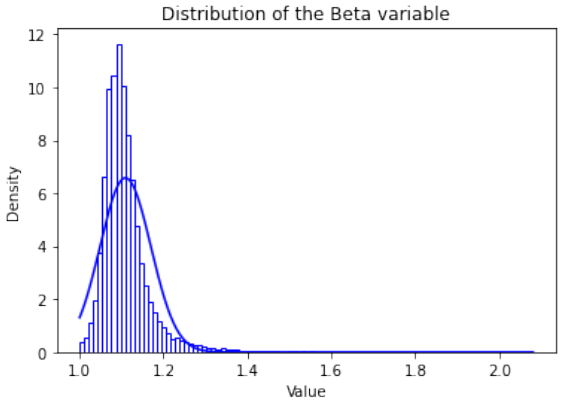

Distribution of the AgePred variable
Density
Value


Distribution of the DeltaAgePerf variable
Proportion
Value


Distribution of the Height variable
Male
Female
Density
Value


Distribution of the Weight variable
Male
Female
Density
Value


Distribution of the Alpha variable
Density
Value


Distribution of the Beta variable
Density
Value

To better illustrate the purpose of our analysis, we also plotted a graph showing the correlation between personal bests on two different distances (here, between marathon and half-marathon). Unsurprisingly, the correlation is strongly positive (see Figure 1).

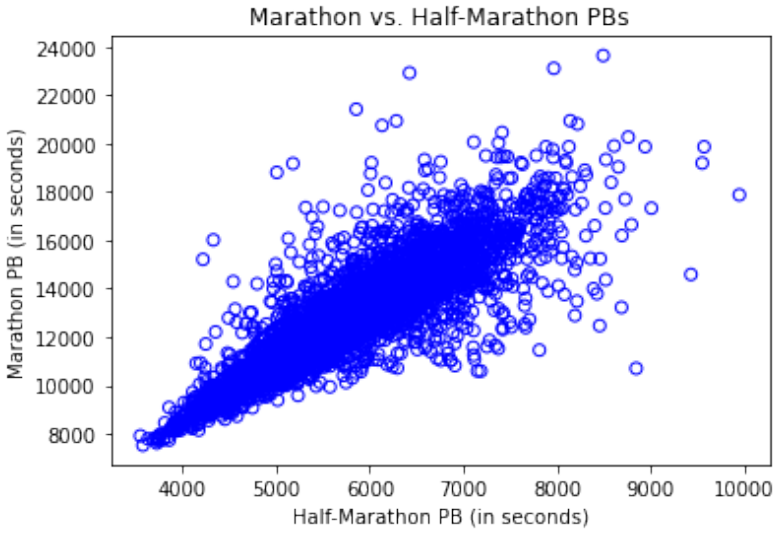


Figure 1: Correlation between half-marathon and marathon personal bests

2.2 Removing outliers

As shown by the above graphs, each feature could be roughly summarized by a normal or a log-normal distribution, except $DeltaAgePerf$ which follows a discrete distribution bounded between 0 and 3, and $Beta$, which is supposed to be bounded on the left by 1 (since $\beta \leq 1$ would mean that the athlete runs faster on longer distances, which makes no sense).

Relying on these assumed distributions we computed a score for each data point, equal to the probability it had to take its own value for a given feature.

$$Score_{ij} = \min(F_j(x_{ij}), 1 - F_j(x_{ij}))$$

Where x_{ij} denotes the value of feature j for data point i, and F_j denotes the assumed cumulative distribution function of feature j

Each time a score was inferior or equal to 0.1%, we considered the whole data point (the whole row) as an outlier.

2.3 Main tools

To conduct our analysis, we decided to use Python and relied more specifically on the scikit-learn and keras packages.

After a long but no less crucial data pre-processing phase, the objective was to apply a certain number of machine learning algorithms and to identify the most efficient one using 5-fold cross validation combined with relevant evaluation metrics (more details below).

Below are the algorithms we used in our analysis:

- Linear regression [2]
- K-nearest neighbors regression [3]
- Regression trees [4]

- Random forests [5]
- Extreme gradient boosting (XGBoost) [6] [7]
- Artificial neural networks [8]

The two scoring metrics we used to assess the performance of each algorithm are $MAPE$ (mean average percentage erorr) and $MedAPE$ (median average percentage error). Indeed, it seemed logical that the higher the predicted time, the higher the absolute prediction uncertainty. Therefore, metrics such as mean squared error and median squared error would have proved irrelevant, as they would have tended to over-weigh larger distances while under-weighing the shorter ones.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y_i}}{y_i}\right|$$

$$MedAPE = median\left\{\left|\frac{y_i - \hat{y_i}}{y_i}\right|, 1 \leq i \leq N\right\}$$

Where y_i denotes the true response value and $\hat{y_i}$ the predicted one for data point i

Our goal was finally to come up with a reasonable confidence interval in order for the runner to have a sufficiently accurate idea of his potential to properly calibrate his/her training plan.

3 Model selection

3.1 Multiple linear regression

Before performing any model fitting, we had first to slightly transform our data set in order to make it usable for linear regression. The $PredDist$ and $PerfDist$ variables were thus transformed into dummy variables.

3.1.1 Using the untransformed data set

The first machine learning model that comes to mind to explain the relationship between X and y variables is the simple linear regression.

$$\forall 1 \leq i \leq N,\ y_i = X_i\beta + \epsilon_i$$

The ordinary least square methodology solves for the above equation by minimizing the following loss function (sum of squared errors):

$$SSE(\beta) = \sum_{i=1}^{N}(y_i - X_i\beta)^2 = ||y - X\beta||^2$$

Table 3 below shows the model's cross-validated level of performance.

MAPE		**MedAPE**	
Avg	*Std*	*Avg*	*Std*
54.4%	1.64%	14.9%	0.15%

Table 3: Model's predictive performance

We can observe that the classical linear regression model has an average level of precision of 54.4% with standard deviation of 1.64% and a median level of precision of 14.9% with standard deviation of 0.15%. This means that the true performance potential of the median athlete will be within about ±15% of the prediction made by the model. For example, if the model returns a performance prediction of 40 minutes on 10km, the athlete's true potential over this distance will actually be between 34 and 46 minutes. This range is much too wide to be relevant.

3.1.2 Using a transformed data set

As explained in section 3.3, the main issue of OLS is that it tries to minimize the sum of squared residuals regardless of the order of magnitude of the response variable. Instead, we wished to fit a model that was able to minimize the following loss function (sum of squared percentage errors):

$$SSPE(\beta) = \sum_{i=1}^{N} \left(\frac{y_i - X_i\beta}{y_i} \right)^2 = \sum_{i=1}^{N} \left(1 - \frac{X_i}{y_i}\beta \right)^2 = ||y_{mod} - X_{mod}\beta||^2$$

$$\text{Where } y_{mod} = \begin{pmatrix} 1 \\ ... \\ 1 \end{pmatrix} \text{ and } X_{mod} = \begin{pmatrix} \frac{X_1}{y_1} \\ ... \\ \frac{X_N}{y_N} \end{pmatrix} = \begin{pmatrix} \frac{x_{11}}{y_1} & ... & \frac{x_{1p}}{y_1} \\ ... & ... & ... \\ \frac{x_{N1}}{y_N} & ... & \frac{x_{Np}}{y_p} \end{pmatrix}$$

Therefore, to minimize $SSPE$, we only had to transform our data set in the above way and then solve the ordinary least squares optimization problem.

Table 4 below shows a clear improvement in terms of prediction performance.

MAPE		**MedAPE**	
Avg	*Std*	*Avg*	*Std*
9.0%	0.07%	7.0%	0.12%

Table 4: Model's predictive performance

3.1.3 Adding squared and interaction variables

One of the main weaknesses of the two previous models is that they do not take into account possible interactions between explanatory variables. For instance, older athletes tend to perform relatively better over long distances compared to younger athletes, all else remaining equal. This means that the interaction between the variables *AgePred* and *DistPred* should have a significant impact on the predicted level of performance.

Therefore, in order for our model to be able to understand such patterns, we decided to transform the set of explanatory variables applying the $PolynomialFeatures(degree = 2)$ method available in scikit-learn (the method returns simple features, squared features, as well as interaction terms).

Unsurprisingly, Table 5 below shows again clear improvements in terms of prediction performance.

MAPE		**MedAPE**	
Avg	*Std*	*Avg*	*Std*
5.1%	0.05%	3.7%	0.07%

Table 5: Model's predictive performance

3.2 K-nearest neighbors regression

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and make a prediction from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice.

In our analysis, we decided to fit a $KNeighborsRegressor$ model with the following parameters:

- Number of neighbors to consider: 1, 5 and 10 ($n_{neighbors} = 1, 5, 10$)
- Distance: euclidian ($metric =$ "$euclidian$")

$$Dist((a_1, ..., a_N), (b_1, ..., b_N)) = \sqrt{\sum_{i=1}^{N} (a_i - b_i)^2}$$

Note: in order for euclidian distances not to be corrupted by the varying scale of the features' values, we needed to perform feature scaling. In that case, we chose to standardize the explanatory variables (subtract mean and divide by standard deviation).

See in Table 6 below the model's cross-validated predictive performance.

	MAPE		**MedAPE**	
	Avg	*Std*	*Avg*	*Std*
$n_{neighbors} = 1$	8.9%	0.08%	7.0%	0.06%
$n_{neighbors} = 5$	7.3%	0.11%	5.7%	0.04%
$n_{neighbors} = 10$	7.2%	0.13%	5.6%	0.07%

Table 6: Models' predictive performance

3.3 Regression tree

Decision trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piece-wise constant approximation.

Given training vectors $x_i \in R^n, i = 1, ..., l$ and a label vector $y \in R^l$, a decision tree recursively partitions the feature space such that the samples with the same labels or similar target values are grouped together.

Let the data at node m be represented by Q_m with N_m samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, partition the data into $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ subsets.

$$Q_m^{left}(\theta) = \{(x, y)|x_j \leq t_m\}$$
$$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta)$$

The quality of a candidate split of node m is then computed using an impurity function or loss function $H()$.

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} H(Q_m^{left}(\theta)) \frac{N_m^{right}}{N_m} H(Q_m^{right}(\theta))$$

The parameters that minimize impurity are then selected.

$$\theta^* = \text{argmin}_\theta \, G(Q_m, \theta)$$

The algorithm then recurses for subsets $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ until the maximum allowable depth is reached, $N_m < \min_{samples}$ or $N_m = 1$, depending on the selected parameters.

We decided to implement the *DecisionTreeRegressor* model using the following parameters:

- Loss function $H()$: mean squared error ($criterion =$ "*squared error*")
- Minimum number of samples in a leaf node: 0.01%, 0.1%, 1% of the total number of rows ($min_{samples_{leaf}} = 0.0001, 0.001, 0.01$), in order to avoid over-fitting.

See below (Table 7) the predictive performance of the implemented decision tree regressors.

	MAPE		MedAPE	
	Avg	*Std*	*Avg*	*Std*
$min_{samples_{leaf}} = \text{None}$	6.1%	0.06%	4.0%	0.08%
$min_{samples_{leaf}} = 0.0001$	5.0%	0.10%	3.4%	0.07%
$min_{samples_{leaf}} = 0.001$	5.4%	0.08%	3.9%	0.07%
$min_{samples_{leaf}} = 0.01$	7.5%	0.08%	5.9%	0.13%

Table 7: Models' predictive performance

3.4 Random forest

A random forest, as its name indicates, is a tree method that introduces randomness in the regressor construction, the prediction of the ensemble being given as the averaged prediction of individual regressors. In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set.

The purpose of this randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to over-fit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias.

For the purpose of our analysis, we chose to build random forests comprising of 100 decision trees each ($RandomForestRegressor(n_{estimators} = 100)$).

See below (Table 8) the predictive performance of the implemented random forest regressors.

MAPE		MedAPE	
Avg	*Std*	*Avg*	*Std*
4.4%	0.04%	3.0%	0.03%

Table 8: Model's predictive performance

3.5 Extreme gradient boosting

Boosting is a tree method that combines a series of weak learners to achieve a stronger one. The idea behind boosted trees is, at each step, to fit a new decision tree to the errors made by the previous one.

It works the following way:

- Step 1: a first decision tree is fitted according to some predefined loss function (in our case, MSE), and residuals are kept in memory:

$$\epsilon_1 = f_1(X) - y$$

Where f_1 denotes the first tree's prediction function

- Step 2: a second decision tree is fitted on the residuals computed in the first step, the new residuals being kept in memory:

$$\epsilon_2 = f_2(\epsilon_1) - \epsilon_1$$

- Step 3 to $n_{estimators}$: a series of decision trees is fitted on the residuals of the previous step.
- Last step: all the weak learners are combined to make a final prediction:

$$F(X) = f_1(X) + \alpha f_2(\epsilon_1) + ... + \alpha f_{n_{estimators}}(\epsilon_{n_{estimators}-1})$$

Where α denotes the boosted tree's learning rate, which determines the speed at which the algorithm will learn the data's patterns

Extreme gradient boosting (XGBoost) is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best tree model. The method is more sophisticated than simple gradient boosting, for two main reasons:

- It uses second-order gradients, i.e. second partial derivatives of the loss function (similar to Newton's method), which provides more information about the direction of gradients and how to get to the minimum of our loss function.
- It also uses advanced regularization methods (L1 and L2) to improve model generalization.

For the purpose of our analysis, we chose the following parameters:

- Learning rate: 0.3 ($eta = 0.3$)
- L1 parameter: 1 ($reg_{lambda} = 1$)
- L2 parameter: 0 ($alpha = 0$)

See below (Table 9) the results we obtained using extreme gradient boosting.

MAPE		**MedAPE**	
Avg	*Std*	*Avg*	*Std*
17.4%	0.30%	8.2%	0.22%

Table 9: Model's predictive performance

Surprisingly, it appears that XGBoost did not manage to capture finely the patterns of the data set. Even by making the learning rate vary, we were not able to obtain more satisfactory results.

3.6 Artificial neural networks

Artificial neural networks are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks are composed of several layers: an input layer (composed of as many neurons as features), one or more hidden layers, and an output layer (see Figure 2). Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

At each node k (located in layer $j > 1$), a weighted sum is computed and then passed to an activation function:

$$f\left(\sum_{i=1}^{n_j} w_{ki} x_{in_j}\right)$$

Where n_j denotes the number of neurons in layer $j-1$

At the output layer, a final weighted sum is computed, giving the value of the prediction. Therefore, the goal of the algorithm is to find the best sequence of weights that minimizes a given loss function at the output layer.

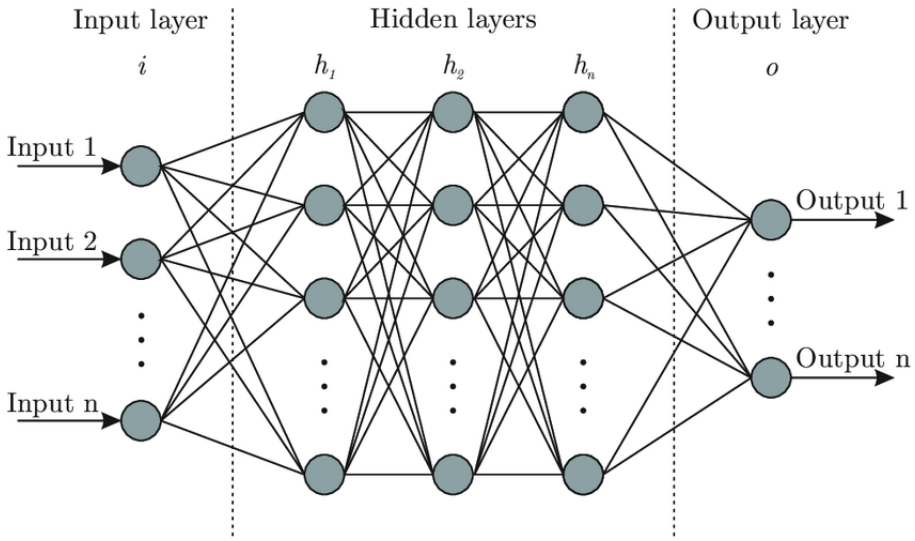


Figure 2: Architecture of an artificial neural network

For the purpose of our analysis, we chose to take the following parameters:

- Architecture: one network composed of 1 hidden layer of 20 nodes, and another one composed 2 hidden layers of 10 nodes each
- Activation function: ReLU ($ReLU(x) = \max(0, x)$)
- Loss function: mean absolute percentage error (MAPE)

See below (Table 10) the results we obtained using neural networks.

	MAPE		**MedAPE**	
	Avg	*Std*	*Avg*	*Std*
1x20-node hidden layer	7.4%	0.66%	4.4%	0.06%
2x10-node hidden layers	5.9%	0.49%	4.2%	0.27%

Table 10: Models' predictive performance

Not surprisingly, neural networks do good in terms of cross-validated predictive performance. Nevertheless, they are still not the best and take a considerable amount time to learn the patterns of the training data.

3.7 Summary

After having fitted all the models, we then ranked them in increasing order of MedAPE in order to decide which one we would retain for out-of-sample final evaluation.

Even taking very conservative confidence intervals around MAPE and MedAPE, random forest appeared to clearly outperform all other algorithms.

	MAPE		MedAPE	
	Avg	*Std*	*Avg*	*Std*
Random forest	4.4%	0.04%	3.0%	0.03%
Decision tree ($min_{samples_{leaf}} = 0.0001$)	5.0%	0.10%	3.4%	0.07%
Linear regression (using interaction terms)	5.1%	0.05%	3.7%	0.07%
Decision tree ($min_{samples_{leaf}} = 0.001$)	5.4%	0.08%	3.9%	0.07%
Decision tree ($min_{samples_{leaf}} = \text{None}$)	6.1%	0.06%	4.0%	0.08%
Neural network (2x10-node hidden layers)	5.9%	0.49%	4.2%	0.27%
Neural network (1x20-node hidden layer)	7.4%	0.66%	4.4%	0.06%
KNN ($n_{neighbors} = 10$)	7.2%	0.13%	5.6%	0.07%
KNN ($n_{neighbors} = 5$)	7.3%	0.11%	5.7%	0.04%
Decision tree ($min_{samples_{leaf}} = 0.01$)	7.5%	0.08%	5.9%	0.13%
KNN ($n_{neighbors} = 1$)	8.9%	0.08%	7.0%	0.06%
Linear regression (transformed data set)	9.0%	0.07%	7.0%	0.12%
XGB Regressor	17.4%	0.30%	8.2%	0.22%
Linear regression (untransformed data set)	54.4%	1.64%	14.9%	0.15%

Table 11: Predictive performance summary (ranked by average MedAPE)

4 Performance prediction

4.1 Performance over the test sample and comparison with the French Athletics Federation scale

At the very beginning of our study, we isolated 25% of the scrapped data for testing purposes. We chose to proceed in two steps:

- Fit a 100 trees random forest to the whole training data and then assess the absolute predictive performance over the test sample.
- And then make a comparison with the predictive performance of the French Athletics Federation official scale (see Tables 12 and 13). [9] [10]

Level	800m	1000m	1500m	3000m	5km	10km	15km	HalfMarathon	Marathon	100km
IA	105.54	136	214	464	790	1670	2610	3690	7800	23700
IB	106.54	137.5	217	470	805	1695	2640	3735	7950	24780
N1	108	139	222	480	825	1740	2670	3840	8100	25800
N2	110	141.5	226	490	840	1785	2740	3930	8400	26700
N3	112	144	230	500	860	1830	2810	4020	8640	27600
N4	114	146.5	234	510	880	1875	2880	4110	8880	28500
IR1	117	150	239	520	900	1920	2950	4230	9120	29400
IR2	119	152.5	244	530	920	1965	3020	4350	9360	30300
IR3	121	155	249	540	940	2010	3090	4440	9600	31200
IR4	123	158	254	550	960	2055	3160	4530	9840	31800
R1	125	161	259	560	980	2100	3230	4650	10080	32400
R2	127	164	264	570	1000	2145	3300	4740	10380	33000
R3	129	167	269	580	1020	2190	3370	4830	10680	33600
R4	131	170	274	590	1040	2235	3440	4950	10980	34200
R5	134	173	279	600	1060	2280	3510	5040	11280	34800
R6	137	177	284	610	1080	2325	3585	5220	11640	35400
D1	140	181	289	620	1100	2370	3660	5400	12000	36000
D2	142	184	294	630	1120	2430	3780	5640	12360	37200
D3	144	188	299	640	1150	2490	3900	5820	12840	38400
D4	147	192	304	660	1180	2580	4020	6060	13320	39600
D5	150	196	312	675	1210	2700	4260	6300	13800	40800
D6	153	200	320	690	1260	2820	4440	6600	14400	42000
D7	156	205	328	705	1320	3000	4680	6900	15000	43200

Table 12: French Athletics Federation scale - Men (performances in seconds)

Level	800m	1000m	1500m	3000m	5km	10km	15km	HalfMarathon	Marathon	100km
IA	119.84	155.7	246	533	910	1900	3000	4200	8940	29100
IB	121.84	158.5	251	545	930	1960	3045	4320	9120	30300
N1	126	162	261	560	990	2040	3150	4470	9600	31500
N2	130	166	268	575	1020	2100	3240	4620	10080	32400
N3	133	170	275	590	1040	2160	3330	4770	10560	33300
N4	137	174	281	605	1060	2220	3420	4920	11040	34200
IR1	140	178	287	620	1080	2280	3510	5070	11520	35100
IR2	142.5	181	295	640	1100	2340	3600	5190	11880	3600
IR3	145	184	300	650	1120	2400	3690	5310	12240	4500
IR4	148	188	305	660	1140	2460	3780	5430	12720	5400
R1	151	192	310	670	1160	2520	3870	5550	13080	6300
R2	154	196	315	680	1180	2580	3960	5670	13440	3600
R3	157	199	320	690	1200	2640	4050	5790	13800	4500
R4	160	203	325	705	1220	2700	4140	5910	14160	5400
R5	164	209	330	720	1240	2760	4230	6030	14520	6300
R6	166	212	340	735	1260	2820	4320	6150	14880	3600
D1	168	215	350	750	1290	2880	4440	6300	15240	4500
D2	170	218	360	765	1320	2940	4530	6450	15600	6000
D3	173	222	370	780	1365	3000	4620	6600	15960	3900
D4	176	226	380	800	1410	3120	4710	6900	16320	5400
D5	179	230	390	820	1440	3240	4800	7200	16800	6900
D6	183	235	400	850	1500	3360	4980	7500	17400	4800
D7	188	242	410	880	1560	3600	5160	7800	18000	3600

Table 13: French Athletics Federation scale - Women (performances in seconds)

See just below (Table 14, Figures 3 and 4) the results we obtained. The random forest significantly outperformed the FFA official scale.

	MAPE	MedAPE
Random forest	4.30%	2.83%
FFA scale	7.55%	3.90%

Table 14: Predictive performances on the test sample

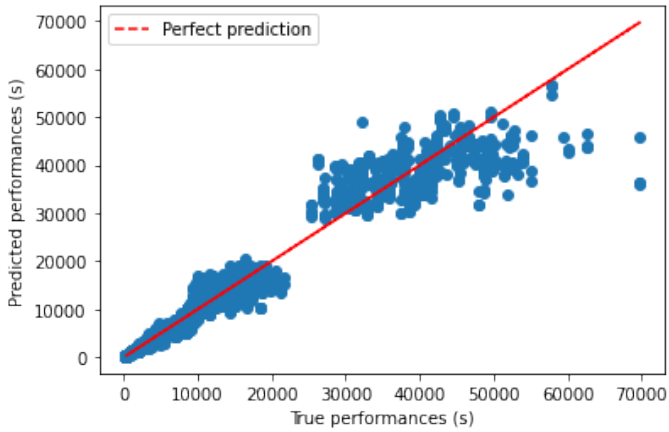


Figure 3: Predicted vs. True performances - Random forest

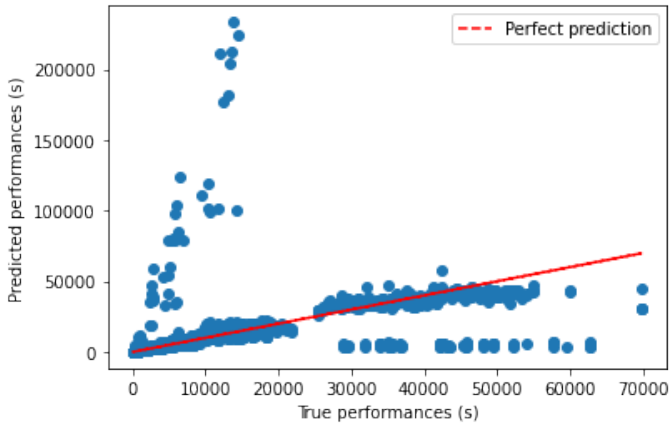


Figure 4: Predicted vs. True performances - FFA scale

4.2 User interface for new predictions

Finally, we thought that a performance prediction algorithm would not be of much interest if it did not have a user interface. The whole point of our analysis was to be able to make new predictions. This is why we created a simple interface using the streamlit package of Python. In order to use it, you just have to:

- Download and save in an identified location the *performance_ predictor.py* and *rf_ regressor.pckl* files.
- Install streamlit on your computer, running the following instruction in your command prompt: "pip install streamlit".
- Indicate the location of the two relevant files to the command prompt, running the command "cd" + location of the file (both files should be located in the same place).
- Finally, run the command "streamlit run performance_predictor.py" and wait for the new tab to open.

You can now interact with the performance predictor. Please recall that the displayed prediction carries some uncertainty. We estimated in our analysis that, for 50% of the runners, the random forest prediction was less than 2.83% off the exact value. Therefore, we advise you take the prediction result with care and add a margin of about 3% around the indicated value.

5 Conclusion

The results of our analysis are encouraging but call for a certain number of improvements, both in terms of data collection and data analysis.

For instance, in order to limit computation time, we voluntarily chose to limit the quantity of data collected. For each individual, we only considered his/her whole time personal bests, whereas considering personal bests localized in time (best times made on two-year windows for example) would have perhaps allowed us to capture some patterns more finely.

It would also have been possible to spend more time on tuning the hyper-parameters of our machine learning models. We are thinking in particular of XGBoost for which a more thorough sensitivity study on the learning rate and the number of estimators would probably have allowed us to get more satisficatory results.

And last but not least, we were missing essential factors of performance but which are unfortunately much more difficult to access and measure, such as the shape of the athlete on the day of the competition, the quality of his/her training, diet or sleep... This could be the project of a future study.

References

[1] F. française d'Athlétisme, "Tous les bilans." https://www.athle.fr/asp.net/main.html/html.aspx?htmlid=5268, 2019.

[2] scikit learn, "Linear models." https://scikit-learn.org/stable/modules/linear_model.html.

[3] scikit learn, "Nearest neighbors." https://scikit-learn.org/stable/modules/neighbors.html.

[4] scikit learn, "Decision trees." https://scikit-learn.org/stable/modules/tree.html.

[5] scikit learn, "Ensemble methods." https://scikit-learn.org/stable/modules/ensemble.html.

[6] T. D. Science, "Gradient boosted decision trees." https://towardsdatascience.com/gradient-boosted-decision-trees-explained-9259bd8205af.

[7] S. playgRound, "Machine learning basics - gradient boosting & xgboost." https://shirinsplayground.netlify.app/2018/11/ml_basics_gbm/.

[8] keras, "Artificial neural networks." keras.io.

[9] F. française d'Athlétisme, "Barèmes et tables de cotation (m)." https://www.athle.fr/asp.net/main.html/html.aspx?htmlid=125.

[10] F. française d'Athlétisme, "Barèmes et tables de cotation (f)." https://www.athle.fr/asp.net/main.html/html.aspx?htmlid=122.