Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Department of Software Engineering
Specialization: Applications of Informatics in Natural Sciences

Využití regresní analýzy sportovního výkonu pro automatizované dolování dat

Using Regression Analysis of Sports Performance for Automated Data Mining

MASTER'S THESIS

Made by:	Bc. Vojtěch Zahradník
Supervisor:	Ing. Matej Mojzeš, Ph.D.
Year:	2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Zahradník**	Jméno: **Vojtěch**	Osobní číslo: **486308**
Fakulta/ústav:	**Fakulta jaderná a fyzikálně inženýrská**		
Zadávající katedra/ústav:	**Katedra softwarového inženýrství**		
Studijní program:	**Aplikace informatiky v přírodních vědách**		

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Využití regresní analýzy sportovního výkonu pro automatizované dolování dat

Název diplomové práce anglicky:

Using Regression Analysis of Sports Performance for Automated Data Mining

Pokyny pro vypracování:

1. Rozšiřte výzkumný úkol s důrazem na automatizaci a dolování dat relevantních pro zlepšování sportovního výkonu.
2. Proveďte rešerši literatury o využití regresní analýzy pro interpretaci záznamů o průběhu cyklistických a běžeckých tréninků či závodů za účelem zlepšování podaného výkonu.
3. Upravte vhodný datový model a příznakový prostor pro vlastní experimenty.
4. Zlepšete metody pro zpracování dat uložených ve formátu Flexible and Interoperable Data Transfer (FIT), nebo GPS Exchange Format (GPX), integrujte a zautomatizujte získávání dat od třetích stran.
5. Implementujte hledání optimálního modelu pro interpretaci dat s využitím heuristické optimalizace v jazyce Python s využitím vhodných balíčků, např. pandas, numpy, statsmodels a scikit-learn s důrazem na prevenci přetrénování modelu.
6. Implementujte rozhraní pro uživatelsky přívětivou interpretaci výsledků, umožňující identifikaci silných a slabých stránek výkonu daného atleta vůči širší populaci atletů.

Seznam doporučené literatury:

[1] ALLEN, H. a A COGGAN. Training and Racing with a Power Meter. VeloPress, 2010. ISBN 9781934030554.
[2] CHATTERJEE, S. a A. HADI. Regression Analysis by Example. 5. New York: Wiley, 2012. ISBN 9780470905845.
[3] KHOLKINE, L., T. SCHEPPER, T. VERDONCK a S. LATRÉ. Machine Learning Approach for Road Cycling Race Performance Prediction [online]. In: . MLSA, 2020 [cit. 2021-10-18]. ISBN 978-3-030-64912-8. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-030-64912-8_
9
[4] MCKINNEY, W. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2013. ISBN 9789351100065.
[5] MOJZEŠ, M. Integer Optimization Heuristics. Praha, 2017. Dizertační práce. ČVUT v Praze. Vedoucí práce Ing. Quang Van Tran, Ph.D.
[6] ZAHRADNÍK, V. Optimalizace modelů regresní analýzy sportovního výkonu. Praha, 2022. Výzkumný úkol. České vysoké učení technické v Praze. Vedoucí práce Ing. Matej Mojzeš, Ph.D.
[7] GPX: the GPS Exchange Format [online]. [cit. 2021-10-18]. Dostupné z: https://www.topografix.com/gpx.asp
[8] Garmin API [online]. Garmin [cit. 2022-10-12]. Dostupné z: https://developer.garmin.com/
[9] Flexible and Interoperable Data Transfer (FIT) Protocol Introduction [online]. [cit. 2021-10-18]. Dostupné z: https://developer.garmin.com/fit/protocol/
[10] Strava API [online]. [cit. 2022-10-12]. Dostupné z: https://developers.strava.com/

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Matej Mojzeš, Ph.D. katedra softwarového inženýrství FJFI

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **30.10.2023** Termín odevzdání diplomové práce: **10.05.2024**

Platnost zadání diplomové práce: **29.10.2025**

[illegible]	[illegible]	[illegible]
Ing. Matej Mojzeš, Ph.D.	doc. Ing. Radek Pučík, Ph.D.	doc. Ing. Václav Čuba, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

3.11.2023	[illegible]
Datum převzetí zadání	Podpis studenta

Declaration

I declare that I have prepared my Master's Thesis independently and have used only the materials (literature, projects, software, etc.) listed in the attached list.

In Prague 8. 1. 2024 .... [illegible]

Bc. Vojtěch Zahradník

Acknowledgement

I would like to thank my supervisor Ing. Matej Mojzeš, Ph.D. for his willingness, professionalism, language proofreading, for excellent communication during the problem and for the advice provided in the field of data analysis and machine learning.

Bc. Vojtěch Zahradník

Název práce:
Využití regresní analýzy sportovního výkonu pro automatizované dolování dat

Autor: Bc. Vojtěch Zahradník

Studijní program: Application of Natural Sciences
Obor: Applications of Informatics in Natural Sciences
Druh práce: Master's thesis

Vedoucí práce: Ing. Matej Mojzeš, Ph.D.
Department of Software Engineering, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague
Konzultant: –

Abstrakt: Cílem této práce bylo optimalizovat modely regresní analýzy, které by poskytly předpovědi celkových časů závodu na základě sportovních aktivit. Součástí této práce byla i rešerše podobných aplikací na trhu. Práce se opírá o data poskytovaná ve formátu FIT a GPX aplikacemi třetích stran. V práci je implementováno vyhledávání vhodných parametrů vstupujících do modelů pomocí heuristických algoritmů a automatizace zpracování dat a poskytování výsledků z modelů ve snadno interpretovatelné reportu. Finální aplikace je vyhodnocena na datech ze skutečných závodů.
Klíčová slova: Datová analýza, sportovní výkon, optimalizace, regresní analýza

Title:
Using Regression Analysis of Sports Performance for Automated Data Mining

Author: Bc. Vojtěch Zahradník

Abstract: The objective of this thesis was to optimize regression analysis models to provide predictions of overall race times based on sporting activities. As part of this thesis, the research of similar applications on the market. The thesis relies on data provided in FIT and GPX format by third party applications. We implemented a search for suitable parameters entering the models using heuristic algorithms and automated the data processing and the provision of results from the models in an easily interpretable report. The final application was then evaluated on data from real races.
Key words: Data analysis, sports performance, optimization, regression analysis

Contents

Introduction

Currently, the words machine learning, AI, and generative AI are very inflected words. These industries have recently received advances like no other industry before [25]. There are several reasons for this, advances on the hardware side, better marketing, but most importantly, the realization by companies that without machine learning and data collection, they will not be able to compete in the future. This has resulted in more investment in this type of innovation. Of course, progress is being slowed by concerns about data security, especially in an industry like banking.

Machine learning can be found in all industries, and sports are no different. In sports, we can find predictions of the results of individual matches or even more advanced methods where we recognize the position of players in the playing field using advanced computer vision [14] methods. In this thesis, data related to mostly running and cycling will be used. Based on this data, a machine learning model will be trained to predict whether a user's training is progressive.

The following output of the thesis will be the prediction of the final time of an individual race based on the trained model. For this, auxiliary models will predict the athlete's heart rate and cadence progression in the race. Knowing the possible outcome time is advantageous in racing to succeed. It allows the racer to know how to distribute energy, for example, where to accelerate or decelerate directly on the track.

Based on this situation, we decided to extend the research task [70], which involved investigating the behaviour of data from sports activities. Given these activities, we attempted to estimate the course of future races using regression analysis. Heuristic algorithms, mostly genetic optimization, were used to optimize the objective function. This thesis aims to extend this research and make the final prediction more accurate. Simultaneously, we would like to focus more on the automation of the whole application. All models and processed data will be packaged in the application.

Furthermore, we will be interested in comparing multiple athletes against each other based on their trained models, and we will try to interpret and display these results in the most readable way possible. An HTML report will display all the results, whether prediction results or athlete comparisons, so the application's output is easy to read for all users. The user will be offered the possibility to compare with individual reference sections for different distances. The main purpose of the report is to provide user with explained results extracted from model predictions. Therefore, the graphical appearance will be done in a simple yet readable form.

The bulk of the thesis will be conducted on the author's personal data, who will hereafter be referred to as Athlete V. Since many demands are placed on the data quality, from high granularity to accurate measurement, it is complicated to find multiple athletes with similar quality data. Therefore, most of the testing will be done on running data, but the entire application can be converted to cycling data without any problems or significant code changes required.

There are two main parts to the paper. A theoretical part, where we will extend the search from the research assignment, which will now focus more directly on a sample of similar applications offered by Garmin International[1], from now on, it's Garmin, and Stryd[2]. The market research chapter will end with a subsection about Strava[3] and why we decided not to use their API[4]. Next, we will describe the theoretical background of machine learning and time series in this thesis. The end of the theoretical part introduces the reader's use of heuristic algorithms, which are applied in this work and will be demonstrated by code examples in the practical part.

In the practical part, we will introduce the reader to tuning hyperparameters with heuristic algorithms. Next, we will focus on creating features using several approaches. In the implementation part, the exciting parts of the code will be shown, and the whole application will be described. The output of the whole application will be a short HTML report. The creation of the report will be described in the chapter on the user interface, where we will also describe the simple control of the application.

The paper concludes with a discussion about the overall contribution of our work and the presentation of the results obtained from several testing races while also testing the application on low-quality data. Possible extensions of the thesis will be discussed. Lastly, any potential shortcomings of the application will be addressed.

[1]A company interested in the production and development of civilian GPS receivers.
[2]A company that makes watt meters for running.
[3]Internet service for sharing sports activities.
[4]Application Programming Interface

Chapter 1

Machine Learning

Machine learning is a rapidly growing sub-field of artificial intelligence that focuses on creating algorithms and models. These models allow machines to form predictions for the future based on past experiences. Commonly, a machine learns the behaviour of a large volume of data and tries to predict the behaviour of newly arriving records based on this knowledge. Statistical methods are also used in machine learning to teach the machine various patterns in the data that are usually difficult to find.

In this chapter, We will describe the basic terms associated with machine learning so that the reader fully understands the theoretical workings of the application. The division of machine learning will be described based on the learning method and then describe the different models necessary to understand the whole application.

1.1 Division by way of learning

Machine learning can take several different approaches [59] based on how the models are used and what data are needed for the model input. In the following text, we will describe the basic categories.

1.1.1 Supervised learning

The first type is supervised learning. We need a training set with predictors in advance to train this type. The model will make predictions based on the learning in this training set, either classification type or regression type. On the other hand, the model needs to prepare a data set with the correct outputs that will be hidden from the model at the time of prediction. This set is usually called a test set and the performance of the newly trained model is verified.

This type also includes classification and regression, which is a further division of the model based on the type of task. Classification models use knowledge from the data to predict classes. On the other hand, regression models are used to predict numerical values. In this work, we use regression models and regression analysis. Thus, the model always predicts some numerical value.

This type of learning is very widespread nowadays, and it is also the easiest to apply and interpret. In the thesis, we used this type of learning.

Benefits:

- The prediction is based on previous experiences in the data.
- Because we have labelled data, we know exactly which class the object belongs to.

Disadvantages:

- The algorithms are very simple, so they are not very suitable for more complex tasks.
- Long learning process.
- If the test set is significantly different than the training set. The outputs will be very biased.

1.1.2 Unsupervised learning

No training set is needed for this type of learning. The model's training is done over unlabeled data; thus, the model makes predictions based on the actual data. The basic principle of this type of learning is to form clusters or sort the data by different types into several groups. This type of learning is very often used to find patterns in data.

Benefits:

- Can be used for complex tasks.
- The absence of labelled data makes data collection a bit easier.

Disadvantages:

- Outputs may be less accurate due to the lack of long training.
- Making models of this type is more difficult.
- Some abstraction in model interpretation.

1.1.3 Semi-supervised learning

This type of approach is a combination of unsupervised learning and supervised learning. Based on this information, we can infer that it is a learning approach where we have training data as input and based on this data the model learns to classify the data into different groups and subsets.

Benefits:

- Usually simple to understand.
- Useful for poorly balanced data.

Disadvantages:

- Comprehensive and long hyperparameter tuning.
- A very good model is required for high accuracy.

1.1.4 Reinforcement learning

This way of learning works on the basis of the agent receiving a reward for each of its activities. The agent learns over time which activities will give it the highest reward. Here, we have no input data to the model, and the agent learns only based on its decisions over time. Reinforcement learning is often seen in solving difficult games and has played a large role in creating large language models.

Benefits:

- Useful for solving real-world problems.
- Model can easily adapt to changing environments.

Disadvantages:

- Requires very powerful hardware.
- Understanding the model and the agent's decisions is very complex.

1.2 Decision Tree

Decision trees [18] are a type of model from the supervised learning category. It is one of the simplest but also one of the most used models. Because of its explainability, it is used in more advanced techniques. It is based on a tree structure, and each internal node places a condition on the input data. Each leaf then stands for the individual outputs of the model. The algorithm is based on recursively partitioning the training data into several subsets, as we can see in Figure 1.1

The main advantage of this type of model is its simplicity and, therefore, the ease of interpretation of the results. Another indisputable advantage is its simple application to the problem at hand.

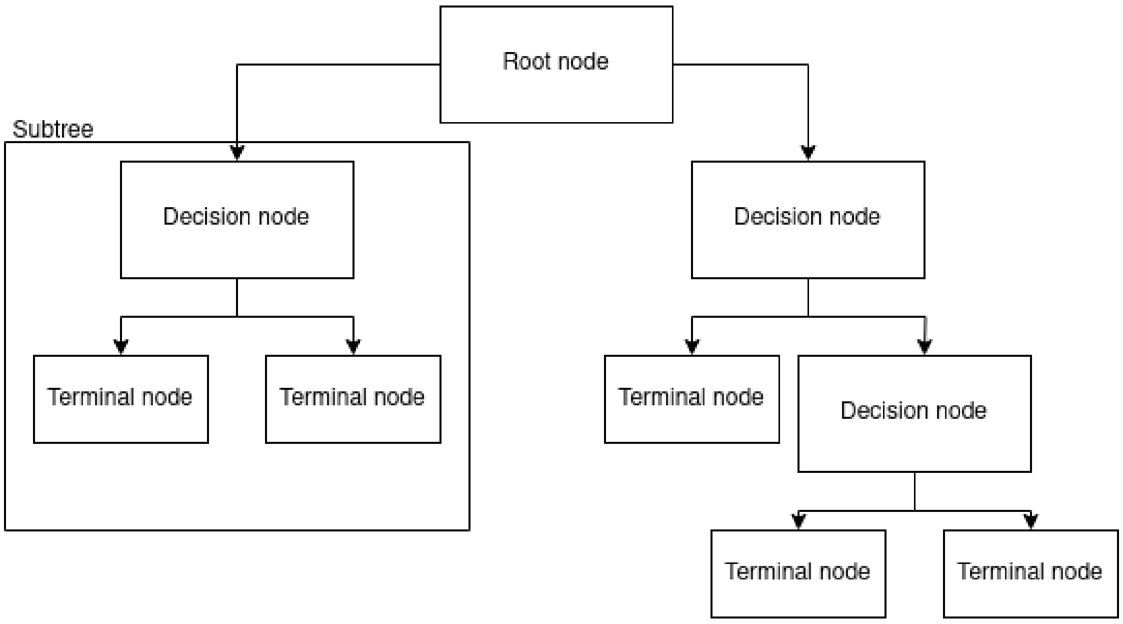


Figure 1.1: Decision tree structure

1.2.1 Impurity

Impurity [3] is a very important concept used by decision trees. As a metric, it tells us how much disordered data we have in a subset (sub-tree) relative to the predicted value. Based on this metric, the algorithm selects the predictor according to which the data will be partitioned to minimize the impurity as much as possible.

The two basic ways to measure impurity [19] are:

- **Entropy** is originally a physical metric that again determines disorder.

$$E = \sum_{i=1}^{\mathbf{n}} -p_i \log_2 p_i \tag{1.1}$$

  where E denotes the entropy, p_i denotes the probability of class i occurring in the input data and **n** denotes the number of classes.

- **Gini index** determines the probability that an input element to the tree will be incorrectly evaluated.

$$Gini = 1 - \sum_{i=1}^{\mathbf{n}} p_i^2 \tag{1.2}$$

where p_i tells us the proportion of elements in the data that belong to a class and $\mathbf{n}$ denotes the number of classes.

1.2.2 Information gain

Information gain measures the reduction in entropy after a node is split into two or more nodes using a certain criterion and a certain predictor. This measure is used in decision trees to determine a significant predictor for further branching to reduce the impurity concerning the predicted variable.

$$\mathrm{IG}(T, \mathbf{a}) = \mathrm{H}(T) - \mathrm{H}(T|\mathbf{a}) \tag{1.3}$$

where IG is the information gain, $\mathrm{H}(T|\mathbf{a})$ is then the entropy of the stack T after partitioning based on the attribute $\mathbf{a}$. $\mathrm{H}(T)$ denotes the entropy of the tree T before splitting into other sub-trees.

1.3 Ensemble learning

This type of learning [10] is based on combining results from different models. With this approach, we generally get more accurate results. We can grasp this method in many ways, but in general, there are four basic approaches, which we will also explain. These approaches are also called meta-learning, a subset of machine learning that means that algorithms learn how to combine predictions from other machine learning approaches. For tree-type models, decision trees are predominantly used.

In our thesis, we use decision tree-based models, and for more accuracy, the application's models are based on ensemble learning. In this category, we find many types of models, which we will talk about subsequently. These models are further divided into four basic categories according to how the different models are combined.

1.3.1 Bagging (Bootstrap Aggregating)

The key element of bagging [63] is learning multiple models that are independent of each other using the same algorithm, all on different training data. This method is usually used to reduce the variance over data with a large amount of noise.

First, random subsets are created from the training set. The elements in these sets need not be distinct. Models independent of each other are trained on these subsets, and then these models are used for prediction. The resulting predictions are then averaged to give the final prediction.

Random Forest

This technique [44] is one of the leading machine learning methods and extends simple decision trees. It is also one of the most widely used. It applies all the key steps of bagging and uses the randomness of individual predictors to generate many uncorrelated decision trees. This method is also sometimes referred to as feature bagging, which refers to creating random predictors that are then used in creating the decision tree during branching.

One of the main benefits of random forest is its robustness to overfitting, which comes from its definition and the fact that many independent decision trees are formed. However, the formation of many trees makes Random Forest somewhat hardware-intensive.

Bagged Decision Tress

The Bagged Trees [4] method is similar to Random Forest but does not use the feature bagging technique to form randomness predictors. While the method uses bootstrapping and results in many trained individual decision trees, each tree uses the same predictors to branch. While this method generally improves the accuracy of the decision tree predictions, it can easily overtrain the model.

1.3.2 Boosting

Boosting [64] is a type of ensemble learning that combines results from multiple models that perform worse over test data than the final model. The main idea of boosting is to train many models sequentially, where each successive model considers the errors (residuals) of the previous model and tries to account for them as much as possible. Since the method works sequentially, there is a reduced possibility of parallelization of training.

Important algorithm steps:

1. Each element of the training set is assigned the same weight, and the base model is trained.
2. Based on the evaluation of the prediction, the elements from the set with the larger error rate are assigned higher weights. This has the effect that the next model in the sequence will focus more on the erroneous predictions from the previous model. This principle is the reason for the name of the boosting method.
3. These weights are passed on to the next model
4. Steps 2 and 3 are repeated until the error rate is less than the set threshold.

This machine learning method is now widely used, and in general, it is shown to perform very often better than other ensemble learning methods. Its disadvantage is its sequential learning method.

Boosting can, therefore, be seen as minimizing the objective function

$$\mathrm{L}(\mathbf{x}, \mathrm{f}(\mathbf{x})) \tag{1.4}$$

where L is the objective function, x is the actual value and f($\mathbf{x}$) is the predicted value using the f model.

Gradient Boosting

Gradient Boosting [24] is the first method we will talk about with respect to boosting methods. It is a method that uses decision trees and boosting. In addition, it uses gradient logic and tries to minimize the objective function by moving just in the direction of the negative gradient of the objective function at each step. Hence the method is called Gradient Boosting. This process is very similar to the gradient descent optimization algorithm.

The base model, which denotes the first model of iteration $f_0(x)$ is then defined as

$$\mathrm{f}_0(\mathbf{x}) = \sum_{i=1}^{\mathbf{n}} \mathrm{L}(y_i, \hat{y_i}) \tag{1.5}$$

where $\mathbf{n}$ is the number of elements in the training set, $\mathbf{x}$ will denote the input data in all subsequent formulas, y_i is the real value for element i and $\hat{y_i}$ is the predicted value. The $\mathrm{f}_0(\mathbf{x})$ is usually referred to as the initialization model before any boosting. The L denotes the objective function comparing the actual and predicted values.

Each iteration adjusts the model using the output from the previous model, the hyperparameter, and the newly trained simple model, which is trained on the computed pseudo-residual.

$$\mathrm{f}_m(\mathbf{x}) = \mathrm{f}_{m-1}(\mathbf{x}) + \boldsymbol{\eta} \cdot \mathrm{h}_m(\mathbf{x}) \tag{1.6}$$

where m denotes the number of boosting iterations, $\boldsymbol{\eta}$ is the shrinkage parameter, which is a hyperparameter that determines the learning rate or how much impact a single model will have on the final result. Then, h_m denotes the simple trained model from which the outputs are used to refine the prediction at each step.

Pseudo-residua is a term that denotes the negative gradient of the objective function concerning the model f at a given point $\mathbf{x}$ and can thus be calculated as

$$r_{im} = -\left[\frac{\partial(\mathrm{L}(y_i, \mathrm{f}(x_i)))}{\partial \mathrm{f}(x_i)}\right]_{\mathrm{f}(\mathbf{x})=\mathrm{f}_{m-1(\mathbf{x})}} \tag{1.7}$$

where r_{im} is the pseudo residual for the i training example at the m, step of the algorithm, $\frac{\partial}{\partial \mathrm{f}(x_i)}$ then denotes the partial derivative of the objective function concerning the previous model.

These pseudo-residuals are then used to train a new simple model $\mathrm{h}_i(\mathbf{x})$ to direct the model to predict the pseudo-residuals from the previous model rather than just trying to mimic the target variable. The model is trained on the set:

$$(x_i, r_{im})_{i=1}^{\mathbf{l}} \tag{1.8}$$

where **l** is the number of models to be trained.

In this way, we get a powerful model that can handle many machine learning problems, whether regression or classification problems. This method also gives rise to other models we will discuss, such as XGBoost, which only differ in the way the objective function is optimized.

Extreme Gradient Boosting (XGBoost)

XGBoost [5] is based on the previous Gradient Boosting method, i.e. it also tries to optimise the objective function using a gradient. It is currently one of the best models that win most of the transformation tests [49], both in academia and elsewhere. In the thesis, the work uses this model.

The main difference between XGBoost and Gradient Boosting is the regularisation that XGBoost [46] uses. This technique makes XGBoost considerably faster than Gradient boosting while achieving better results. Regularization also results in a model that is more resistant to overtraining. It also offers benefits such as missing value handling and better. More flexible hyperparameter tuning.

The **Regularization**, denoted Ω, of the objective function, can be seen as penalizing the model for its complexity. This has the effect of protecting against overtraining the model on specific data and increasing its generality. We can write the regularization as a combination of the L1 and L2 regularizations and define it as

$$\Omega(\mathrm{f}) = \boldsymbol{\lambda} \cdot \Omega_1(\mathrm{f}) + 0.5 \cdot \boldsymbol{\gamma} \cdot \Omega_2(\mathrm{f}) \tag{1.9}$$

where $\Omega_1(\mathrm{f})$ and $\Omega_2(\mathrm{f})$ are the L1 and L2 regularizations and $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma}$ are the regularization parameters.

L1 and **L2** regularization in supervised machine learning refers to techniques that penalize the model for many predictors and thus prevent overtraining. Regression models that use L1 regularization are called Lasso [33] regression, then regression models with L2 regularization are called Ridge [55] regression.

The main difference between these two types is how the purpose function is penalized. In Lasso regression, we penalize the objective function by the first power of the norm of the weight vector, so if we understand the objective function as $\mathrm{L}(y, \hat{y})$, where L is the objective function then, $\mathrm{L}(y, \hat{y})$ is the difference between the actual value and the predicted value. Then, we can think of a regularization of type L1 as

$$\mathrm{L}(y, \hat{y}) + \boldsymbol{\lambda} \sum_{i=1}^{\mathbf{n}} |w_i| \tag{1.10}$$

where $\boldsymbol{\lambda}$ is a regularization parameter specifying how much the model should be penalized for a high number of features. The vector w is then a vector of weights in absolute value.

L2 regularization is then defined as

$$\mathrm{L}(y, \hat{y}) + \boldsymbol{\lambda} \sum_{i=1}^{\mathbf{n}} |w_i|^2 \tag{1.11}$$

We get a very powerful ensemble model by optimizing the objective function in this way and using many simple models.

Adaptive Boosting (AdaBoost)

Adaptive Boosting [11], sometimes also called AdaBoost, is another boosting model that has the great advantage that it can be deployed as an extension to other boosting models to refine already very good predictions.

It is based on the use of Decision Stumps [34], which means that it does not use full decision trees with a root, several nodes and leaves, like Random Forest, but only subsets of them, which take the form of only one node and two leaves for each predictor (see Figure 1.2).

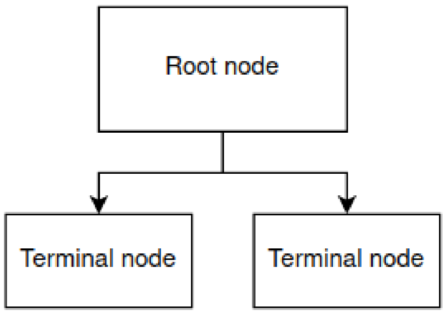


Figure 1.2: Structure of Decision Stumps

Light Gradient Boosting Method (LightGBM)

This method of learning [37] is relatively new, so there is not much information on how to choose its hyperparameters, but like the previous methods, it is based on a tree structure. The main difference from the previously mentioned methods is that this technique [54] uses tree building based on the leaf-wise [53] technique see Figure 1.3

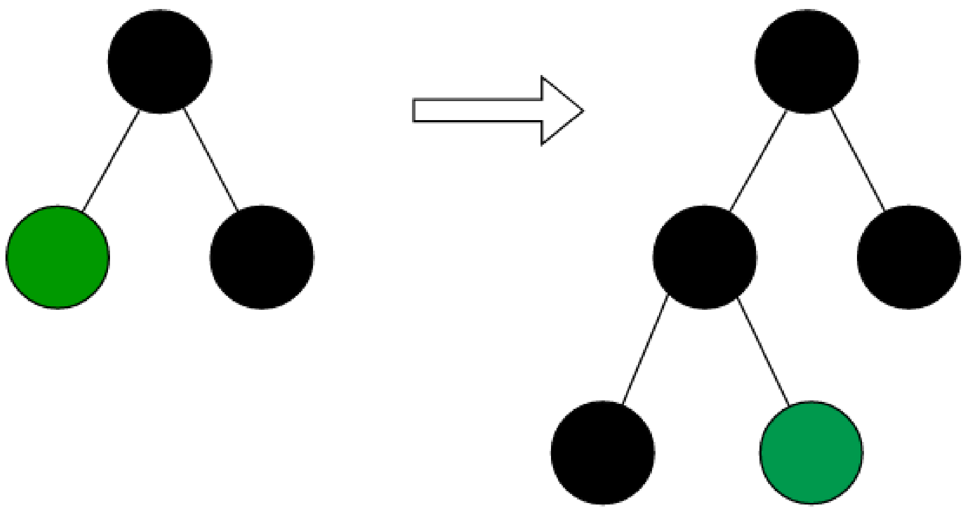


Figure 1.3: How to build trees in LightGBM using the leaf-wise technique

In tree-based methods, we can build trees in two ways:

- **Leaf-wise** – Building the tree from the node that has the biggest impact on the purpose function. This is the method used by LightGBM. This method increases the complexity of the model but has the advantage of greater optimization of the objective function.
- **Level-wise** – Splitting the tree by nodes. Each node is split into other nodes until only leaves remain. The tree is then positioned relative to its root. Other boosting models use this method.

Categorical Boosting (CatBoost)

If we process data sets with categorical predictors with other techniques, we first have to transform them with techniques [68] like One-hot encoding or Label encoding. In this way, we can run into the problem of getting a sparse matrix and overtraining the model.

This is not the case with CatBoost [40] because this model can handle categorical predictors automatically by itself and thus does not need any feature encoding. It also automatically handles null values.

Other advantages of CatBoost:

- L1 and L2 regularization
- Internal cross-validation built directly into the model
- Automatic feature scaling – Converts all columns to the same normalized scale.

1.3.3 Stacked Generalization

Another way of learning from the viewpoint of ensemble learning is Stacked Generalization [10], sometimes shortly Stacking. Like the previous methods, stacking combines multiple machine learning models learned on the same data.

However, in this type of method, we do not use only one type of model, as in bagging and boosting, but we try to use multiple model types from which predictions are then combined. The base models are usually called base models, and the model that combines their predictions to form the final prediction is called meta-model [51]. The base models are usually linear, logistic regression or even simple neural networks.

1.3.4 Majority Voting Ensembles

Voting [61] is a technique to help the data scientist choose between two more good-performing models. These models may arise in stochastic learning or hyperparameter tuning.

Thus, the output is a class determining the best-performing model on a given data set. Thus, this technique can be built on top of the previously mentioned methods from categories such as boosting or bagging.

Chapter 2

Market research

In this chapter, we would like to present research based on the findings from our research assignment [70], where we described the types of relationships found between future predictors, both positive and negative. Previous researches on a similar topic from abroad were described, which we will mention in this chapter.

In the research assignment, we have described the solutions that Garmin [23] and Strava [16] have. Currently, we would rather describe the company Garmin and Stryd [47], which provides a sensor for measuring watts. Strava will briefly mention and give reasons why we have decided not to continue using data from the company. Stryd uses the sensor's measured watts for predictions, and we assume these predictions are more accurate. Garmin also records watts, but only through calculation. So it does not consider the weight of the athlete and other important variables, which makes the calculated values mere estimates, which are often very inaccurate.

This thesis research will focus more on commercial use and point out Garmin and Stryd's inaccuracies. We will also discuss the performance of the predictive models behind the predictions from these companies and explain why these predictions are generally not very accurate, even though these companies have huge amounts of data because users can use their services for free.

Of course, it is impossible to know the exact workings of these algorithms because the companies in question are very guarded about this, so the following subsections on discussing company services performances will be derived from the observed experiences of the general public, including ourselves.

2.1 Garmin International

Garmin states in its documentation [26] that it uses personal data input from the user (e.g., gender, age) and the user's fitness level, which is based on the user's activities, to make predictions. Along with this information, it additionally uses Vo2Max [28], which is a value that determines the maximum amount of oxygen the athlete's body can use during activity. Based on this information, it appears that to make accurate predictions, it is necessary to use their application for at least a few weeks. Of course, as with our application, the more historical data, the more accurate the prediction will be.

However, the outputs from this app tend to fall into two categories: either Garmin overestimates your capabilities, meaning the predicted times are unrealistically challenging, or it underestimates them, leading to times that are too easy. A significant downside of the Garmin app is its tendency to overestimate for most users. This can lead to demotivation among athletes who struggle to meet these ambitious predictions, often finding them unattainable.

Garmin's general predictions are calculated on a flat course [38] and, let's say, under ideal conditions. This means suitable weather conditions for the activity and the road route. Based on athlete V's data, Figure 2.1 is an example of time predictions for 5 km, 10 km, half marathon, and marathon races. If we compare the real-time for 10 km of athlete V with Garmin's prediction, Garmin is about four minutes worse.

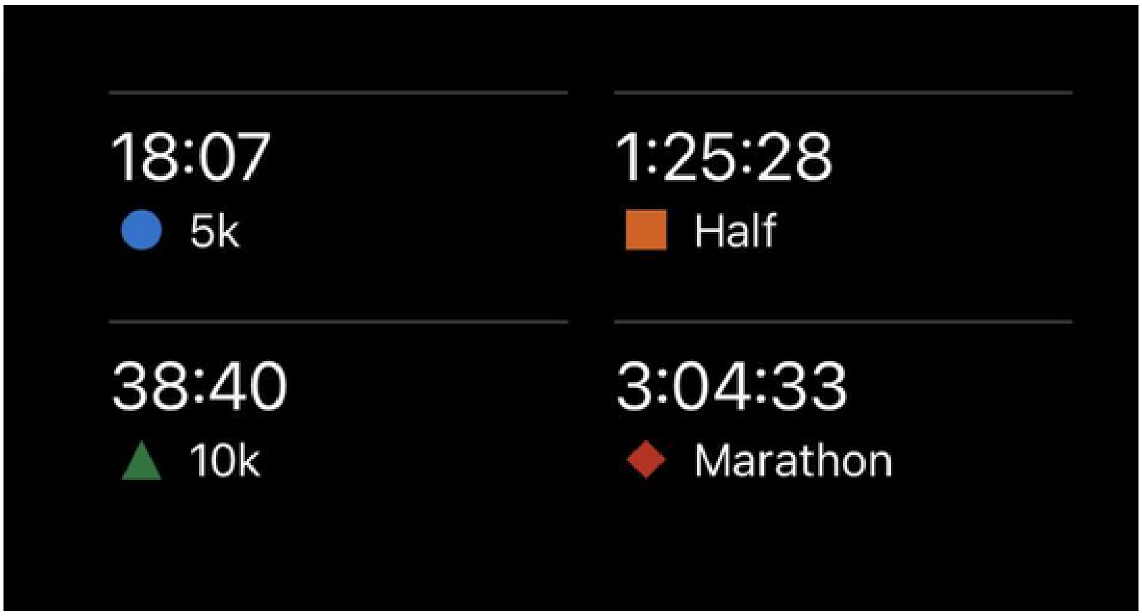


Figure 2.1: Sample forecasts from Garmin Race Predictor

Maximizing the accuracy of Garmin outputs

This section will describe tips that athletes should follow to make predictions more accurate. Our research assignment also mentioned these tips about the behaviour towards the data entering this application because all machine learning-based applications need the most accurate data possible.

Heart rate measurements are needed using a chest strap to make prediction more accurate. Wrist heart rate measurements can be accurate, but at lower temperatures, the accuracy of these sensors decreases. If the user is using multiple Garmin devices, activity and fitness level synchronization between these devices needs to be enabled.

The high granularity of the data will also aid in quality prediction and activities that are longer than half an hour.

2.1.1 Possible problem with Vo2Max

A possible problem with inaccuracy may lie in too much weight on the Vo2Max value, which the watch tends to estimate. Table 2.1 shows the Vo2Max values and the predicted times at different distances. We obtained the values from the article [66]. To determine this value accurately requires the supervision of an expert and special equipment typically found in medical facilities or training centres.

When comparing the Vo2Max value of athlete V, which was 65, there is a noticeable divergence from the tabulated values, particularly as the distance increases. In the developed application, We intentionally exclude the Vo2Max value from consideration. This decision stems from the assumption that the Vo2Max value might be highly inaccurate and could potentially skew the final prediction.

Vo2Max [ml/kg/min]	**5K**	**10K**	**21K**
30	37:00	1:17:41	2:52:42
40	27:45	58:16	2:09:31
50	22:12	46:36	1:43:37
60	18:30	38:50	1:26:21
70	15:51	33:17	1:13:59

Table 2.1: Vo2Max values and the predicted times from article

Garmin estimates the Vo2Max value incorrectly within units, which can result in large prediction inaccuracies. It is also important not to confuse cycling Vo2Max with running Vo2Max. The two values can be different.

2.1.2 PacePro

In addition, Garmin offers a pace layout feature on a pre-loaded route. This feature is called PacePro. The output of PacePro is a graph that tells you at what distance you should run or bike. The downside of this app is you must enter a final time on the input. Thus, the app does not solve the question "How fast are athletes able to run a given route?" for you but only schedules the pace based on the elevation difficulty of each section so that the finish time matches the input time.

2.1.3 Summary

To conclude the subsection on Garmin, we would like to mention that, in our opinion, Garmin usually predicts better results than an athlete can achieve, which can have negative psychological consequences for the athlete. This may be due to too much weight on the estimated Vo2Max value.

Another problem may be that the athlete belongs to a lower percentage of the population in his performance if we look at Figure 2.2, which depicts athlete V's Vo2Max value compared to the broader population. We can see that only 2 % of athletes are similar to him, which may result in worse predictions. The models that make the predictions may have more weight and representation, i.e. athletes with Vo2Max equal to 52, for example. From Garmin's side, it would be logical that they would target models for the most represented population. One of the possible solutions for this problem might be segmenting the data, which could help solve this problem.

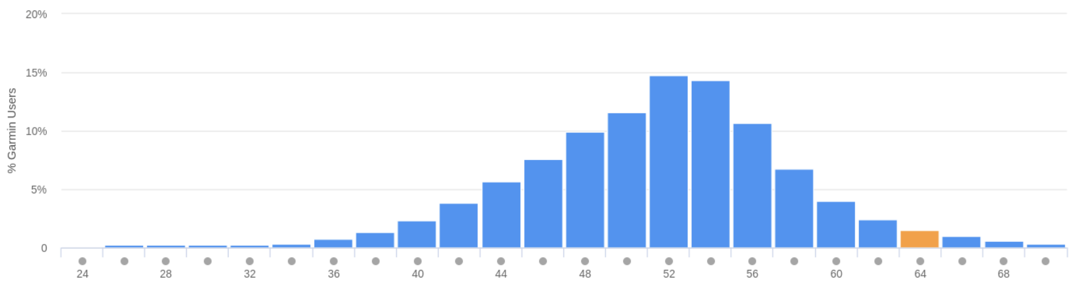


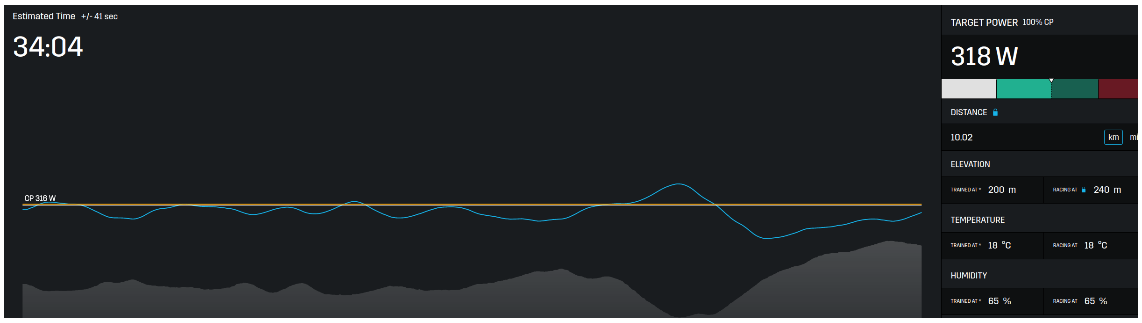
Figure 2.2: Distribution of running Vo2Max values compared to the wider population

2.2 Stryd

Stryd offers a sensor to measure watts while providing athletes with a user-friendly environment to manage their data. The company focuses on the aforementioned watts and thus lists watts rather than average pace. Watts represents how much power an athlete puts into each step or each pedalling stroke in the case of cycling. This is one of the most accurate ways to measure power because wind speed, terrain and other variables will also be reflected in this value. Currently, this is the only running sensor and data company. Many users find the interpretation of watts quite difficult, so this system of watt-based training is not that common.

2.2.1 Race Calculator

Unlike Garmin's PacePro, Stryd's Race Calculator [56] directly offers distance input and, based on your activities, writes an estimated time you should be able to run the distance. Stryd's Race Calculator considers features like temperature, activity altitude and humidity, all of our app's predictors. This application also offers the possibility to input a track and predicts the final time and pace based on your activities, as we can see in Figure 2.3


Estimated Time +/- 41 sec
34:04

TARGET POWER 100% CP
318 W

DISTANCE
10.02 km

ELEVATION
TRAINED AT * 200 m
RACING AT 240 m

TEMPERATURE
TRAINED AT * 18 °C
RACING AT 18 °C

HUMIDITY
TRAINED AT * 65 %
RACING AT 65 %

Figure 2.3: Sample of the Stryd prediction web application

The inserted route is from the race Běchovice Praha [6], which our application will be tested on because it is standardized directly by the Czech Athletics Federation. We can see that the prediction is 34:04 minutes and will be later used as the baseline threshold for this thesis model's performance evaluation.

Race Calculator uses a Stryd feature called Critical Power to determine the final race time. This value determines how many watts we can sustain over a distance or track. In the aforementioned Figure 2.3, athlete V's goal is 316 watts at a Critical Power of 318 watts. Another important value that greatly impacts the prediction is the Power Duration Curve, which represents how long a given watt can last. This curve is downward sloping as a function of time.

Notably, Stryd uses the last 90 days of training for its predictions. This reduces the likelihood of major dropouts in the data and significantly reduces the training set. Given that Stryd can use watts while using the sensor to have more accurate data than most athletes do with the aforementioned Garmin, the prediction of their app is very much above expectations. It performs much better than Garmin, where that data is often less accurate.

2.2.2 Summary

Stryd offers a similar application, which inspires this thesis application. However, Stryd has secured more accurate data due to the assumption of using their sensor. Additionally, when purchasing two of these sensors, they offer to extend the data by detecting running techniques when attaching the sensors to each shoe. This opens up another feature creation dimension and could lead to more accurate predictions based on the athlete's running style.

2.3 Strava, Inc

In this subsection, we will only briefly mention an application from this company and explain why we chose not to use this application as a data source for our work.

One of the reasons is that Strava does not offer predictions for your chosen section. Thus, comparing the results with our work would not be possible. At the same time, it's only a consumer of data from the manufacturer of your sports trackers (e.g. Garmin), so it doesn't even get all the data your sports tracker measures. Last but not least, the unfriendly API must be mentioned. API returns very aggregated activities, and thus, the application could not get directly to the measured FIT files.

2.4 Previous research on similar topics

After market research, we will follow with suitable approaches for sport activity data analysis. In this paper section, a summarization of academic papers and articles on a similar topic will be made. The papers are from abroad only.

2.4.1 Regression Analysis of Pacing When Running a Marathon

A 2021 paper by Hawkin Starke [52], from the University of Arkansas at Fayetteville, discusses using regression analysis when running a marathon. The paper focuses on one specific marathon, namely *Little Rock Marathon.* The author wants to predict the final time based on the intermediate times on each lap. Thus, in the regression model, we find the following four variables:

- Time from the start line to 10 kilometres
- Time from 10 kilometres to the half marathon line
- Time from half marathon to 33.8 kilometres (21 miles)
- Time from 33.8 kilometres to finish

An interesting takeaway from this research was that the time in the first 10 kilometres is the feature with the highest importance in marathon time prediction. In addition, the regression model is enriched with the age and gender of the athlete.

Starke's research differs from the aim of the thesis in selecting explanatory variables derived from past races of a specific type. Thus, using his work on other races would be difficult, and the research would have to be done again. This thesis is interested in prediction based primarily on the physical condition of the athlete in question and training sessions completed over the past year.

Results

It was necessary to split the data sets into faster and slower runners for the first ten kilometres due to the significance of the variables. This paper also finds the models evaluated using the R^2 metric. As a final result, the author states that the regression model can explain 80 % of the finish times of the participants. The most significant variable was the time between ten kilometres and the half marathon, as expected by the author.

2.4.2 A Comparison of Multiple Regression Models to Help Predict Road Race Performance

The goal of Lynn Brown's thesis [9] is very similar to mine. To predict a runner's time in a 5k race to be run in a week. It also deals with comparing two predictions of different runners. The predicted value is then compared to the actual value after the race.

The author had only two data sets of the runners' training on which he then learned the models. The final models are then compared with each other. The author also discusses the significance of the variables in the paper. He tests the significance itself using the well-known *t-test*. After learning the model, he tests the multi-collinearity problem between the variables. As far as the results are concerned, in the paper, we can find the plots of the residuals and all the basic statistics, such as *t-test* and *F-test*.

Chapter 3

Time Series

3.1 Introduction

Time series [42] are a sequence of elements x_t that are chronologically ordered according to the time dimension at some time point t. Typically, time series are measured at a time interval of, for example, one year. They differ from known cross-sectional data by the dependence of each record on the records preceding it. Although cross-sectional data seem to be the opposite of time series, the two types are often used together. One of the simplest examples of time series is the stock market, but they can be found in any industry.

Time series can be divided according to the distance between two records into:

- **Equidistant** – These are time series where the intervals between two observations are constant. These time series are most often found in physics when making measurements on machines. In the real world, most of the time, we encounter a non-constant step.
- **Non-Equidistant** – These are time series where the intervals between observations are not constant. This happens by collecting data in irregular steps. We often encounter time series of this type, and a non-equidistant series will be used in this thesis.

Another division we would like to mention is by the number of values measured at time t:

- **Univariate** – Only one value is recorded simultaneously.
- **Multivariate** – Many values are recorded simultaneously.

3.2 Concepts bound with time series

Now, let's review a few concepts directly related to time series.

3.2.1 Trend

Trend [62] is a pattern in the data. This pattern shows us which direction the data will change in the next steps. Trends can be:

- **Upward** – Data will grow over time
- **Descending** – With further measurement, the data will decrease
- **Stationary** – The time dimension does not affect the direction of the data.

Trend detection is often key in time series analysis and is also the first analysis that an analyst performs. The simplest way to detect a trend is by visualization, or a more advanced way is by decomposition. Approximation methods and straight-line interleaving of the data are used to determine how the trend will behave in future steps. A typical trend in the data can be seen in Figure 3.1

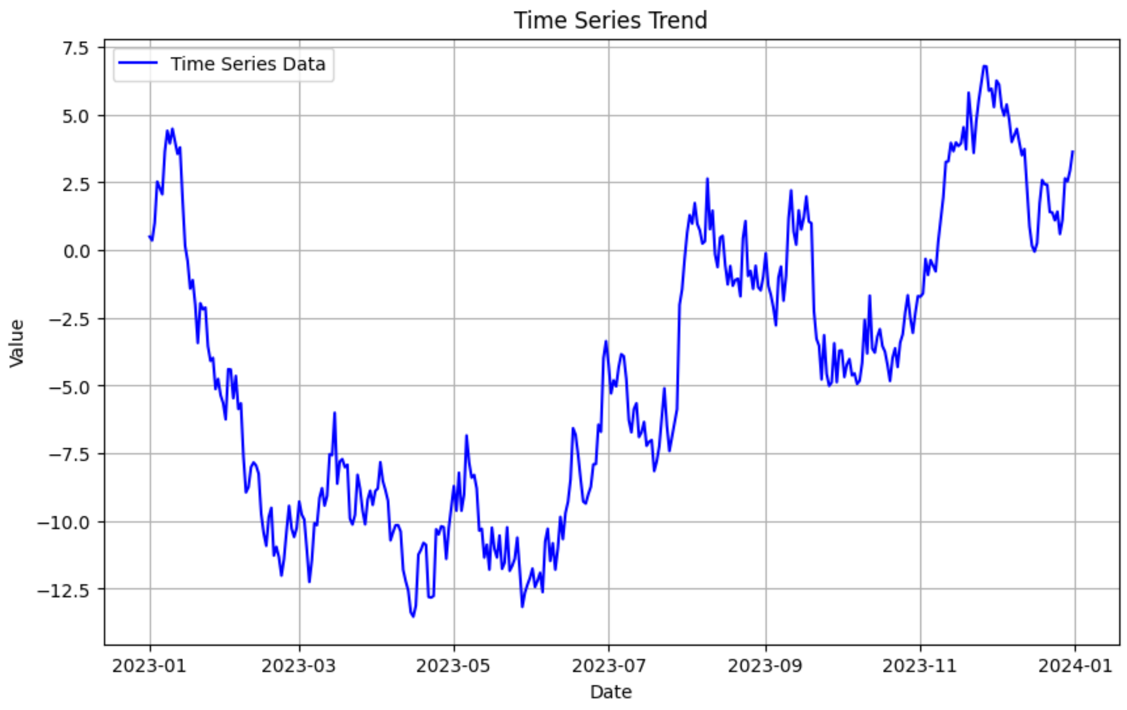


Figure 3.1: Upward trend in time series

3.2.2 Seasonality

Seasonality means a deviation from the trend within a fixed time step in the time series. This can occur, for example, annually or even daily. A typical example of seasonality might be the demand for Christmas trees. This curve will be significantly skewed from the standard at Christmas time throughout the year.

3.2.3 Delayed variables

Sometimes, the term lagged variables is also used. In both cases, these are variables that point to the values of one variable in previous time steps. These lagged values can significantly impact the accuracy of the prediction because they provide the model with information about the presence of a trend. At the same time, they hide the danger of overfitting the future model.

A key decision in time series analyses is how large a lag to choose. We usually arrive at the right number by several experiments and statistical techniques such as the Akaike criterion [8].

3.2.4 Moving Averages

This is a sequence of averages [45] calculated on a given interval from the past, for example, the last 12 months. The calculation then forms a moving window on which the values are counted. Moving averages can be calculated on any period. Like lagged variables, they give the data scientist examining the series some ability to insert a long-term trend into the model, thus influencing the series.

Three basic types of moving averages [57]:

- **Simple moving average** – Calculation of the standard arithmetic mean over a given time interval.

$$y_t = \frac{\sum\limits_{i=0}^{t} x_{t-i}}{t} \tag{3.1}$$

  where t is the size of the interval for the calculation and x_{t-i} is the value of the element in the time interval $t - 1$

- **Weighted moving average** – Adds the logic of adding weights to each element in the series to the simple moving average, with the last element in the interval having the largest weight.

$$y_t = \frac{\sum\limits_{i=0}^{t} w_i x_{t-i}}{\sum\limits_{i=0}^{t} w_i} \tag{3.2}$$

where w are the weights

- **Exponential moving average** – In addition to the previous approach, it adds a small exponential weight to the older elements in the series.

$$y_t = \alpha x_t + (1 - \alpha) y_{t-1} \tag{3.3}$$

where α is the smoothing factor that determines what weight is given to the most recent element and y_{t-1} is the exponential moving average at time $t - 1$

3.2.5 Autocorrelation and partial autocorrelation

These two concepts [1] help us to understand what dependencies (correlations) we have between the values in the series, and at the same time, we try to use these techniques to figure out the values over time that will have the biggest impact on the final prediction.

For correlation, we can examine several types. A positive correlation means dependency. If one value increases, the other value will also increase. With a negative correlation, the opposite is true. Correlation is defined on the interval $\langle -1.1 \rangle$, where 0 means that the two values are not dependent on each other and, therefore, do not affect each other.

In the case of autocorrelation (ACF), it is the examination of correlations between the current element and an element lagged by k steps. If we get a high autocorrelation, it means that the elements lagged by k have a large influence on our acute element, indicating the presence of a trend or seasonality. Figure 3.2 shows a simple example of autocorrelation.

This plot shows us the dependence of one variable on their lagged variables with step k. The blue part of the graph shows us the confidence interval, which gives us the interval in which the statistically insignificant variables are expected to decrease towards 0.

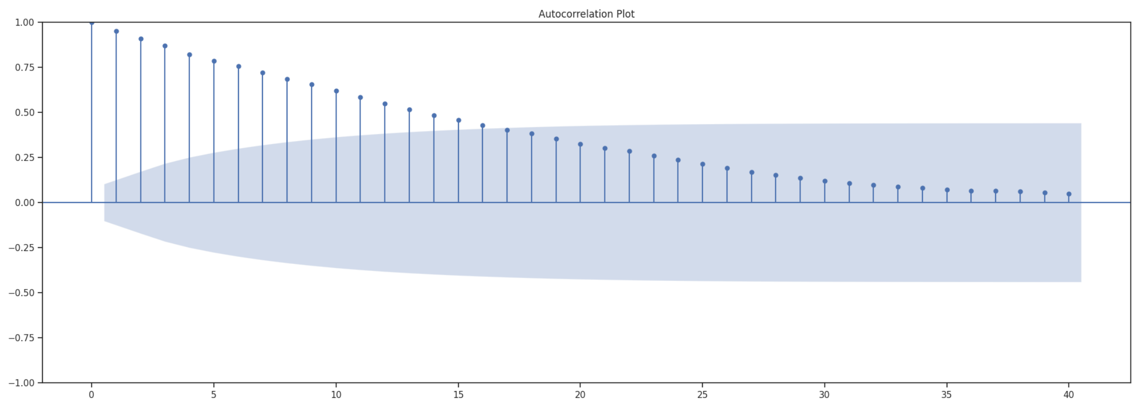


Figure 3.2: Autocorrelation in time series

The partial autocorrelation (PACF) then gives us the dependence of two variables x_t and x_k, where k is several steps into the past, adjusted for the effects of any intervening lags, and t is a random time interval. In other words, the PACF tells us the dependence of two values adjusted for dependence relationships with their previous values.

3.3 Metrics for time series

Metrics are used to evaluate the quality of a time series prediction. Metrics allow us to compare different models with each other, and thus, we can indicate better and worse model performance. The choice of the appropriate metric depends mainly on the nature of the analysis. We will list here some basic metrics.

3.3.1 Coefficient of determination

It tells us the ratio of the variability of the dependent variables that the model is able to explain. This metric [15] can only take values on the interval $\langle 0, 1 \rangle$, where 0 is the worst case, and thus the model explains nothing, and 1 is the best case, where the model explains 100 % of the data. We denote this metric by R^2, and it is one of the most widely used metrics due to its ease of interpretation, but this version cannot compare models with different numbers of variables.

$$R^2 = 1 - \frac{\sum_{i=0}^{\mathbf{n}} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{\mathbf{n}} (y_i - \bar{\mathbf{y}})^2} \tag{3.4}$$

where $\mathbf{n}$ is the total number of observations, y_i is the real value at time i, $\hat{y}_i$ is the predicted value at time i, and $\bar{\mathbf{y}}$ is the average of the real values.

3.3.2 Adjusted R^2

This is a modified version of the coefficient of determination. The original version requires that the compared models have the same number of predictors. This condition is often not met in reality, so the adjusted coefficient of determination does not consider this condition. Another disadvantage of the unadjusted version is that its value tends to increase as the number of predictors increases. In this case, we may get a negative number, which is then interpreted as 0, i.e. the model does not explain any data.

$$\bar{R}^2 = 1 - \frac{(1 - \mathbf{R^2})(\mathbf{n} - 1)}{\mathbf{n} - \mathbf{k} - 1} \tag{3.5}$$

where $\mathbf{R^2}$ is the coefficient of determination, $\mathbf{n}$ is the number of observations and $\mathbf{k}$ is the number of predictors.

3.3.3 MAE

In the case of *Mean Absolute Error*, this metric gives the average absolute error over the entire data sample. It can be interpreted as the average absolute error of the model, for example if our model has MAE = 100, it means that the model misses on average by this number.

This metric is widely used because of its ease of interpretation and also because of the small impact of outliers on the resulting metric, for example, compared to MSE. It is recommended to use it to compare models trained only on the same data set. We will want to minimize all metrics based on calculating the difference between actual and predicted values.

$$MAE = \frac{\sum_{i=1}^{\mathbf{n}} |y_i - \hat{y}_i|}{\mathbf{n}} \tag{3.6}$$

where $\mathbf{n}$ is the total number of measurements, y_i is the actual value at time i and $\hat{y}_i$ is the predicted value at time i.

3.3.4 MSE

This metric extends the MAE and calculates the mean squared error of the model. This results in the metric giving more weight to large errors over smaller errors. A major drawback with this metric is the higher sensitivity to outliers compared to, for example, MAE.

$$MSE = \frac{\sum_{i=1}^{\mathbf{n}} (y_i - \hat{y}_i)^2}{\mathbf{n}} \tag{3.7}$$

3.3.5 RMSE

One of the most widely used metrics is RMSE, which stands for *root mean squared error*. It only takes positive values, and we want to minimise this value like the previous metrics, MAE and MSE.

$$RMSE = \sqrt{\frac{\sum_{i=0}^{\mathbf{n}} (y_i - \hat{y}_i)^2}{\mathbf{n}}} \tag{3.8}$$

This metric is not as easy to interpret as the previous metrics, but in this thesis, We decided to use it for model comparison, mostly because of its accuracy. As with MSE, it has the disadvantage of a high sensitivity to outliers.

3.4 Data preprocessing

With time series, we can encounter many problems related to data quality. One of the very important parts of time series analysis is usually its preprocessing, which can affect the future quality of the model and, subsequently, the output predictions. We would divide preprocessing into two notional parts. Cleaning the data set and refining it.

Cleaning refers to removing noise or filtering outliers, which, for example, can occur through instrument or human error. Data set enhancement can be understood as, for example, combining different quantities to derive new quantities. It also positively affects the model's quality if the data are of the same scale or if we perform normalisation.

3.4.1 Missing values handling

Very often, time series contain empty values. In practice, it may happen that the machine that records the data had a failure, or the data was lost due to some other influence. This data cannot enter the model and must be processed somehow.

This problem can be solved in several ways:

- Empty values are removed from the data set.
- Replace empty values with mean or median, for example.
- Use advanced methods to replace these values (interpolation).

3.4.2 Outlier detection

To outlying values [7] can occur as with missing values, but here, the instrument has recorded the value. However, this value differs greatly from the rest and does not copy the observed trend. Outliers negatively affect models built over time series, especially when using metrics with high sensitivity to this problem in the data.

Simple statistical techniques, such as Z-scores or visualization, can be used to detect them. For more advanced detections, we can use machine learning techniques called anomaly detection [35], specifically the decision tree-based isolation forest method, or k-means clustering, which is a method from unsupervised learning and partitions the data into k clusters. If an element is more than a specified distance from the boundary, we consider it an outlier.

3.4.3 Feature engineering

This concept is key in the development of more complex time series-based models. We can think of it as inferring new attributes based on other attributes, for example, time dimension, or creating lagged variables or moving averages, which is also feature engineering.

In this case, we have unlimited possibilities to enrich our model. However, we must be careful about the correlation between the created and original attributes. Of course, by increasing the number of attributes, we create more complex models, and the time needed to train such models will increase. When creating new attributes, we have to look at the called feature importance [2], which gives us the given attributes' importance on the model's final predictions.

3.4.4 Normalization

Some time series require that all the data have the same scale or distribution at the input to the model. Otherwise, we will not get a good quality model. This requirement makes it necessary to normalize the data. This step is particularly important for machine learning methods based on distance between features.

Types of normalization:

- **Min-max scaling** – Data transformation to scale on interval$\langle 0, 1\rangle$.

$$y_{\text{new}} = \frac{\mathbf{y} - \min(\mathbf{y})}{\max(\mathbf{y}) - \min(\mathbf{y})} \tag{3.9}$$

  where y_{new} is the new normalized value, $\mathbf{y}$ is the real value.

- **Z-score normalization** – Transform data to scale on the interval $\langle 0, 1\rangle$ with standard deviation equal to 1.

$$y_{\text{new}} = \frac{\mathbf{y} - \bar{\mathbf{y}}}{\sigma(\mathbf{y})} \tag{3.10}$$

  where $\bar{\mathbf{y}}$ is the average of the real values and $\sigma(\mathbf{y})$ denotes the standard deviation of the real values.

3.5 Time series analysis

Time series analysis [12] can be performed for several purposes. Usually, the basic reason is to understand the functioning of a given time series and find the principles already mentioned in the series. This helps us to estimate possible future behaviour, and we can adjust the behaviour of our investments accordingly.

3.5.1 Decomposition

Decomposition is a technique used to describe trends and seasonality in time series. More advanced decomposition can even point out days of the week where there may be reduced demand in the case of the weekends or holidays. The main reason for decomposition is to clean the data of the aforementioned negative effects in the series.

Two models are used for decomposition:

1. **Additive model**

   Used when seasonality is constant over time.

   $$y_t = \mathbf{T} + \mathbf{S} + u \tag{3.11}$$

   where y_t is the real value at time t, $\mathbf{T}$ is the trend, $\mathbf{S}$ denotes seasonality and u is the random component.

2. **Multiplicative model**

   If seasonality increases over time, a multiplicative model is more suitable for decomposition.

   $$y_t = \mathbf{T} \cdot \mathbf{S} \cdot u \tag{3.12}$$

3.5.2 Models to explore time series

In this subsection, we will talk about models commonly used for time series analysis. These models are used to predict values at time $t + 1$ and are trained on historical data. Iteratively, we can generate **n** additional values, where **n** gives us the number of steps into the future that we will predict.

The two basic models on which the other models mentioned are based are the **AR (Autoregressive model)** and **MA (Moving averages model)** models. The AR model captures correlations between historical values and their linear combinations. The MA model, on the other hand, highlights the correlation between the historical data and the linear combination of their residual errors.

ARMA and ARIMA

We first describe the ARMA model. It is a combination of the AR and MA models. In this model, knowledge of the historical values and also their residuals is used. In addition, an integrated factor is added to the ARIMA model, which gives the order to make the time series stationary. This factor implies subtracting the previous value from the current value, which can eliminate trends.

3.5.3 Transformers

Transformers [48] is a very advanced architecture introduced in 2017 by Ashish Vaswani in "Attention is All You Need" [60]. The main components of this architecture are the encoder and decoder. This new architecture is based solely on the attention layer and does not rely on recurrent layers, which we can find in LSTM neural networks. We mention Transformers only as an example of a very advanced approach to time series analysis.

Its main purpose is to be used on NLP[1] tasks, but it can also be used on time series after modifications to the data representation. With these modifications, we obtain high-performance models that can perform better than boosting algorithms. One of the first applications can be found in 2021, where this method performed even better than other methods.

To use transformers, it is necessary to convert numerical values of time series into vectors. This is done with embeddings[2], in the context of NLP, techniques such as word2vec [67] are used for this. In time series, similar embeddings must be used. More information on how to use transformers to predict time series can be found in an article on HuggingFace [50]

[1]Natural Language Processing

[2]Converting categorical variables into continuous vectors.

Chapter 4

Heuristic algorithms

The term heuristic algorithm [13] is usually used for algorithms that do not guarantee the quality of the solution. They are mostly used to solve complex problems involving functions with many parameters or extremes.

In contrast to our previous research [70] where heuristic algorithms were used for feature selection. In other words, objective function was optimized using genetic optimization, where the input was an initial population generated by random shooting. The objective function was refined after genetic optimization using the steepest descent methodology.

We have now decided to use heuristic algorithms to refine the model, i.e. we will try to find the optimal hyperparameters. Refining the objective function using heuristics started to appear inefficient as the number of features increased, mostly due to the high time consumption.

4.1 Heuristics

This term can be understood as a solution to a problem for which we do not know the algorithm or the exact solution procedure. However, using heuristics [39], we only get an approximate solution. The accuracy of the solution then depends on many factors, primarily the time we put into the calculation. Usually, we start with a first estimate, which is usually very inaccurate. Over time, using heuristics, we arrive at the optimal solution. The heuristic procedure is universal.

We can often talk about the fact that heuristics quickly gives an exact solution. However, this claim cannot be proven. The use of heuristics is usually under the assumption that we do not know a better algorithm to solve the problem or for time reasons.

4.2 Hyperparameter tuning

In this case [29], we will discuss the process where we try to find the right combination of parameters at the input of the model to maximize the accuracy of the model in prediction. This process has, of course, unlimited possibilities and plays a key role in model building. It is also one of the most time-consuming parts of model building and machine learning.

4.2.1 Concepts related to hyperparameter tuning

Hyperparameter

This parameter enters the model when it is created before training. The values of these parameters are determined by the developer and greatly affect the training of the model. The hyperparameters may vary for different data sets. In the case of XGBoost, we can talk about parameters such as the maximum tree depth or the minimum sum of weights in each sub-tree.

Search Space

When searching for suitable hyperparameters, we must first define the search space. We can do this in several ways. A common approach is a geometric approach, where we define this space as n-dimensional, where each hyperparameter is a single dimension, and the scale dimensions give us the values of the parameters. We can specify the parameters as integers or even categorical values.

Cross Validation

The data is randomly divided into several folds. The model is then trained on each of the folds and tested on the remaining fold. This is the general approach for cross-validation [41]. For time series, we cannot use this approach because of the time dimension and dependencies between the data. For time series, we use a rolling window to maintain temporal ordering, where we start with a model trained on a small number of records and tested on the subsequent data segment. After each training of the model, the rolling window is moved forward to include more data in the training set. This approach ensures that the model is trained on past data and tested on future data.

Objective Function

This is a function that is performed to evaluate the newly created solution using heuristics. This function has the hyperparameters as input and the value of the selected metric as output. Thus, this function is called in each iteration of the heuristic and is used to evaluate the quality of the model. We then use this function to determine how much the model has improved through hyperparameter tuning.

4.2.2 Possible hyperparameters for tuning

In the selection phase of the appropriate model for the machine learning part of the thesis, it was decided to use XGBoost. This model is renowned for its performance and training speed. It provides a robust solution to handle missing data values and outliers. Other benefits of this model are L1 and L2 regularizations, mentioned above, and the friendly hyperparameter tuning process. All of these features were taken into account when deciding on the suitability of the model.

Since we use the XGBoost model, the basic hyperparameters that the thesis can tweak are:

Learning Rate

A factor that controls the step size of the optimization. It influences the overall effect of each tree on the final prediction. Lower values of learning rate require more rounds of boosting, but usually, the model is then more generalized.

Number of Boosting Rounds

Number of boost rounds. This number can be considered the total number of trees in the set. With this hyperparameter, we can influence the overtraining of the model.

Maximum Depth of a Tree

It is used to check the maximum depth of each tree in each set. Deeper trees can support more complex data but risk overfitting.

Minimum Sum of Instance Weight needed in a Child

Used to check the minimum amount of data needed to create a new node or leaf in the decision trees in one boost round.

Other possible hyperparameters to tune are, for example, subsample, fraction of features used for training or gamma. Taking into account the time complexity with most search spaces, we decided on tuning the parameters' learning rate, max depth, and minimum number of data in each leaf, which generally have the biggest impact on model performance.

4.3 Random Shooting

This is a very simple heuristic approach that is usually used for the initial optimization of the problem. The result from Random Shooting is then further optimized using more advanced methods. This method is based on the random selection of hyperparameters from the search space and then testing them on the model. In contrast to, for example, Grid Search, this method does not proceed systematically but randomly.

The main advantage of this method is its simple optimization and low computational effort. As a disadvantage, the condition where the hyperparameters are somehow correlated can be a drawback. This situation will not be detected by Random Shooting and will be ignored.

4.4 Simulated Annealing

Simulated annealing [65] is a heuristic based on probabilistic optimization. The original idea is from the metallurgy industry, where the material is heated and then gradually cooled to remove defects from the material. We can easily transfer this process to our hyperparameter problem. In this context, simulated annealing is used to search the hyperparameter space to find the optimal solution.

Steps:

1. Setting the initial hyperparameters. Usually, default values for the model.
2. Definition of the objective function, which is used to quantify the performance of the model with the selected hyperparameters. Often based on metrics such as RMSE or MAE.
3. Set temperature parameter that influences the adoption of a worse solution. Initially, the temperature is high at the start of the algorithm run, allowing the algorithm to explore a wider space of hyperparameters.
4. Perturbation of the solution, which is a small random change to the current solution, for example, adding a small value to one hyperparameter in order to find a neighbouring solution that will be the next possible optimal solution.
5. Evaluating a new solution using an objective function. If the quality of the model is better than in the previous step, the new parameters are stored. If not, then with some probability, which is calculated using temperature, we can accept this worse solution as well. This principle helps the algorithm to leave the local minimum.
6. We reduce the temperature parameter using the cooling parameter and repeat the process until the metric value satisfies some predetermined threshold.

4.5 Genetic Optimization

Genetic Optimization [31] (GO) is a set of algorithms inspired by natural selection and genetics principles. They are used to optimize complex problems by generating an initial population that is iteratively improved through a process of evolution. The basic operation is based on strings with binary numbers.

Population is an elementary component of GO. The population can be understood as the set of potential optimal solutions to the problem. Each individual, sometimes also a chromosome, in the population represents an individual solution with genes. The genes represent the binary values mentioned earlier that define the characteristics of each individual. The initial population is generated randomly using algorithms like Random Shooting. When the genetic algorithm runs through the entire population, the state is called an epoch. The overall structure of the population is shown in Figure 4.1.

The next component is the selection phase, where individuals are shortlisted based on a fitness function that quantifies individuals based on their characteristics in order to solve a given problem. If an individual has a higher fitness function than another, it has a higher probability of being selected. Of course, if the fitness function is maximized. The selected individuals are also sometimes referred to as the parents of the next population.

After the parents are selected, crossover occurs. Crossover is a term that refers to the process of exchanging traits between two parents to form the next population that will enter the next iteration of the algorithm.

The last basic component of GO is mutation. Mutation occurs under a certain probability, which is given by a parameter. If mutation occurs, we are talking about a small change in the characteristics of individuals in the population. Similar to simulated annealing, this helps to traverse a wider space and ensures diversity of individuals.

Steps of GO:

1. Initial generation of population
2. Evaluating each individual in the population using a fitness function.
3. Selection of parents from the whole population using the value of the fitness function.
4. Controlling the characteristics of the parents. How much crossover should occur is determined by the parameter.
5. We mutate the parents for the ability to traverse a wider space.
6. Generate a new population using the parents and repeat the process until we find the optimal solution.

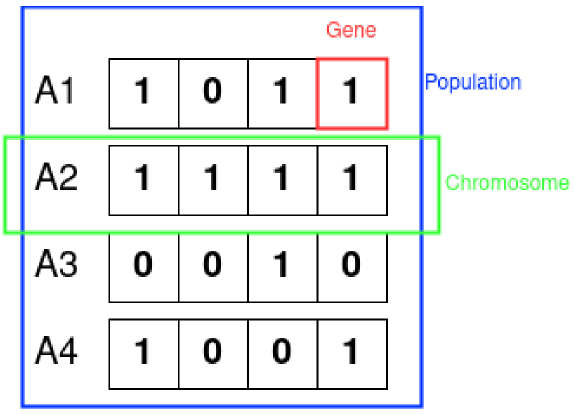


Figure 4.1: Components of Genetic Optimization

Chapter 5

Implementation

In the chapter on implementation, we would like to mention the technical side of the thesis, point out interesting parts of the code, and mention the technologies used throughout the thesis. Programming was mostly in Jupyter Notebooks, and then the code was converted into scripts that acted as modules for better clarity and the ability to step the code.

The whole application has documented code using docstrings, so if individual parts of the code are not described here, it is possible to track them down in the attached code on Git [69] and read their documentation.

5.1 Used technology

This chapter describes the technologies and programming languages used in the thesis's implementation.

The Python programming language belongs to standard data analysis and machine learning [36]. In the case of parallel processing on multiple devices, we would use Spark. As a development environment, we used PyCharm from JetBrains, which also provided us with a local server for Jupyter, where most of the research on the subject took place.

5.1.1 Python

Python is an object-oriented programming language that is one of the most widely used programming languages [58] today. From artificial intelligence research to automation. Python is a multi-paradigm language and also supports a form of functional programming.

Today, it is the programming language that best suits the needs of data analysis. Alternatives to Python for data analysis are the programming languages R and MATLAB. From our point of view, none of these programming languages offers such an extensive range of libraries and solutions as Python.

5.1.2 JetBrains

JetBrains s.r.o. [27] is a company developing IDE for programmers. In 2000, it began offering a programming environment for the Java programming language. Currently, it offers environments for more than ten programming languages.

In this thesis we used the PyCharm environment to program in Python to run a local server for Jupyter.

5.1.3 Jupyter

Jupyter [30] is a web interface that initially supported programming in three main programming languages, namely R, Julia, and Python, but nowadays, the range is a bit wider. It is a web application that, among other things, supports TeX embedding and UML diagrams. The main advantage is the possibility of sharing code in the form of notebooks and a very friendly environment for analysis. The architecture of the environment allows us to run the code in parts, so it will speed up, e.g. the loading of large amounts of data, which we can only load once after starting Jupyter.

The initial development of the application took place in Jupyter Notebooks. All methods were written and tested in Jupyter and then rewritten into Python modules, from where they are called standard methods.

5.1.4 HTML

Hypertext Markup Language is a markup language used to create web pages that are linked by hyperlinks. HTML itself allows the creation of static web pages; for dynamic web pages, it is necessary to use other technologies.

Our application uses this technology to generate the athlete report. To display predicted values and various useful graphs. The report is then taken as the main output of the work.

5.2 Acquisition

Data Acquisition is the collective name for the following subsection. It is mostly about data collection and subsequent processing. In our data acquisition section, we will describe how the data collection was done, what format had to be loaded, and how the data was cleaned from, e.g. *outliers*, wrong measurements and *null* values.

5.2.1 Data collection

The data collection is done by querying the API directly from Garmin using the *garmin* Python library. This library creates an object to which you pass basic parameters that determine, for example, the time interval of activities or your login information.

The library then downloads all the user's activities in the given time interval in the **FIT** format, the standard for sports activities. However, these files are zipped, so the *pyunpack* library is used to unzip them automatically. These files are then sorted based on the type of activity, for example, cycling and running. Furthermore, the application only considers the activity type that the user selects. Various data, such as latitude and longitude, are then parsed from each FIT file and used to calculate features. The *fitparse* library is used for parsing.

Unfortunately, we could not fully automate the data download due to the limitations of Garmin's developer license, which is for business use only. So, for a long time, we downloaded data by entering login credentials into the application configuration file. This is not very secure, but it was usually only a one-time data download. If we were discussing productizing the application, this solution is not an option. In addition, in the last months of work, Garmin has disabled downloading data via the API without charge, so there is a problem with downloading data via the API even when providing a password. So, currently, manual export on Garmin's website is used.

5.2.2 Data requirements

The data should, of course, be of the best quality we can collect. Much emphasis is placed on either the technological elements or the volume of data.

- **Technological requirements**

  One of the main explanatory variables is the heart rate per minute. This variable should, therefore, be as accurate as possible, as it has a great impact on the subsequent prediction. Heart rate is, therefore, best measured using a chest strap rather than a sensor on the hand. Today's sports testers already have this sensor accurate enough, but especially in a race, it can happen that due to the delay, this value can vary. The sensor on the wrist is mostly bothered by sweat and violent handshaking, which is commonplace when running. In cycling, the measurement tends to be more accurate due to the small jolts in the case of road cycling.

- **Data volume requirements**

  The volume requirement is also very important. If an athlete has only one activity in the past period that we want to use as a training data set, the model will not have a chance to predict the correct results. Therefore, the minimum period we would use for prediction is one year. The number of activities is up to each individual, but the athlete should have a regular batch of training activities and thus collect data regularly. For activities from the athlete's last year, it is assumed that the resulting data set will have more than a million rows.

If we have a data set with data only from the first month of the year and the last month, the application will have no chance of predicting correct results. Thus, regularity of activities is also an important requirement for an athlete.

5.2.3 Data format

The format of the downloaded data is FIT[1]. It is one of the most used formats for storing sports activity, and another format can be GPX[2]. Both formats are used in our application, but each has different functionality.

FIT

This format is designed specifically for storing and sharing data regarding sports and health. The FIT format defines a set of data stores (FIT messages) that we can use to store data such as activity or user information. The advantage of the FIT format is that any device that supports this format can display it. A more extensive description of this format can be found in the documentation [20] for developers.

Typical use case:

- ANT+[3] the sensor measures the quantity being monitored
- Data is disseminated in real-time
- Data is stored in a FIT file and displayed on the device
- FIT file is transferred from the device via WiFi or Bluetooth to external devices and further to third-party applications.

GPX

It is an XML[4] data format for transferring GPS data between an application and a web service. Because GPX is based on XML, it has all of its properties. The file itself contains longitude[5] and latitude[6] data for each recorded point. It is mainly used as a standard for exchanging GPS data between devices. The GPX file contains:

- Waypoint – Recorded point

[1]Flexible and Interoperable Data Transfer
[2]GPS Exchange Format
[3]Method of communication between sensor and sport tester, similar to Bluetooth
[4]Extensible Markup Language
[5]Refers to the distance north or south of the equator
[6]Refers to the distance east or west of the prime meridian

- Route – List of points that have changed the direction of the route.
- Track – List of points that make up the route

This format stores data about the position where the record was located. This format, therefore, will be used to insert a route for future prediction. The route will need to be created by the athlete on a suitable web application or is usually available directly on the race website in GPX format.

5.3 Data Preprocessing

5.3.1 Data sorting

After downloading the *raw data* from the Garmin web app, it was necessary to extract the data and sort it by activity type. Thanks to the definition of the FIT format, sorting is very simple, and you only need to query the type of activity the FIT file holds. We can see the inside of the algorithm in the code Listing 5.1

Many new directories are created in the athlete directory with the activity type as the name. Some activities, such as walking or swimming, are not used, but the application accounts for them and sorts them into the appropriate directory.

```python
for record in fitfile.get_messages('session'):
    if (
        record.get_value('sub_sport') != 'indoor_cycling'
        and record.get_value('sub_sport') != 'treadmill'
    ):
        if record.get_value('sport') not in os.listdir(
            path_to_save + athlete
        ):
            os.mkdir(
                os.path.join(
                    path_to_save,
                    athlete,
                    record.get_value('sport'),
                )
            )
        shutil.copyfile(
            os.path.join(path_to_save, files[x]),
            os.path.join(
                path_to_save,
                athlete,
                record.get_value('sport'),
                files[x],
            ),
        )
```

Listing 5.1: Sorting of activities based on type

5.3.2 Data cleanup

Cleaning was already more complicated because when collecting real data, we collected a lot of bad measurements. These can be problems like *outlier* or the problem can be in the device. All this data were filtered out to not unnecessarily bias future predictions. The data also needed to be filtered of duplicate records and *Null* values that would only slow down or make the final calculation impossible.

We used basic filtering using predefined thresholds to clean the data from *outliers*. The filtering is done in the sense that if the value is greater than the upper bound, it is discarded. If it is less, it is similar. The thresholds for filtering can be seen in the Table 5.1. This table tells us the lower and upper boundary of filters. Everything that is not in this interval is dropped. It was also necessary to determine what incline the athlete can climb and when the activity is more likely to be transferred to walking. For uphill and downhill, the threshold is set to a difference of 10 meters in elevation from the last point.

All thresholds were determined based on the author's experience with certain dimensions. More extensive determination of thresholds is not currently needed.

	Lower boundary	**Upper boundary**
Cycling speed	3	100
Running speed	3	25
Cycling cadence	30	180
Running cadence	50	210
Heart rate cycling	60	210
Heart rate running	50	210
Slope	45	45
Ascent and descent	10	10

Table 5.1: Threshold for filtration

5.3.3 Training set segmentation

It made sense to split the training set into more parts to make the results more accurate. The division was based on the total distance of each activity, and only one set entered the model's training. The selection was decided based on the length of the test sample. This reduces the training set by a significant fraction of records. Still, due to the sufficient granularity and diversity of the set, it did not have a large impact on the prediction impact. For small samples, in terms of the number of activities, this impact could be larger. The training data set is divided into:

- The set of shorter-range activities
- For the longer range activity set

Based on our experience with running training, we came up with the solution of dividing into activities where the distance is greater than 5 km and activities where the distance is less than 5 km. If we wanted to consider cycling activities, we would have to increase the dividing line.

Results:

The results were compared on a 10 km track, so the model should use a set with longer distances. As we can see in Table 5.2 the results improved due to this segmentation. In this table, we can see predictions based on real data, with or without added features. The difference between the use of a high-distance set and a low-distance set we can see also.

Furthermore, without using this segmentation, we can see that the model overestimates and the final time is much faster than it actually was.

	Reality	**Without feature**	**Low dist.**	**High dist.**
Cadence[spm]	93.1	92.5	92.7	92.2
Heart rate[bpm]	183.0	178.6	180.7	177.9
Final time[mm:ss]	35:02	34:42	34:12	35:11

Table 5.2: Comparison of results after adding segmentation

5.4 Hyperparameter tuning

For hyperparameter optimization, three heuristic approaches, random shooting, simulated annealing, and genetic optimization, was used to compare multiple methods. In the following subsections, we describe the operation of simulated annealing and genetic optimization in more detail. Then, the conclusion with the results for each algorithm will be mentioned. Hyperparameter tuning in the application is not automated and is done once in Jupyter Notebook. We decided to do this because of the long run time of the application when optimizing the hyperparameters for each model. The optimization algorithms were focused on finding the optimal solution for the combination of *learning_rate*, *max_depth* and *min_child_weight* parameters.

Definition of objective function:

This function, whose code can be seen in Listing 5.2, as mentioned in the theoretical part, indicates how good the hyperparameters the algorithm has chosen. The input to the function has the parameters we want to optimize. These parameters are then used to train the model and evaluate its prediction accuracy using the MAE metric. We try to minimize this metric in the objective function.

```
# Objective function for XGBoost regression
def objective_function(params) -> int:
    # Extract hyperparameters
    learning_rate, max_depth, min_child_weight = params

    print(f"Training: {params}")

    # Train XGBoost model
    model = xgb.XGBRegressor(learning_rate=
    abs(learning_rate),
    max_depth=round(max_depth),
    min_child_weight=min_child_weight,
    random_state=42)

    model.fit(X_train, y_train)

    # Predict on the validation set
    y_pred = model.predict(X_valid)

    # Calculate mean squared error (you can replace this with your
   own evaluation metric)
    mae = mean_absolute_error(y_valid, y_pred)

    # Return mae since we are maximizing the objective
    return -mae
```

Listing 5.2: Objective function to optimalize

5.4.1 Simulated Annealing

In implementing simulated annealing to find optimal hyperparameters, the basic idea was to browse the space of hyperparameters to find the optimal combination. In each iteration of the algorithm, a new possible optimal solution was generated either by shifting to the nearest neighbour or by other mechanisms. The objective function is evaluated using the newly found solution, and a decision is made as to whether or not an improvement has been made.

In the code Listing 5.3, we can see the implementation of the mentioned idea. At the function's input, we have the initial parameters and the objective function we are trying to optimize. Then, the temperature and the cooling parameters. Lastly, the number of iterations of the algorithm.

New parameters are selected based on the random direction of the shift in the parameter space, and then the new solution is evaluated. If the new solution is better than the previous one, it is marked as the new best solution. At the same time, the idea of using temperature to help leave local minima is implemented. The temperature is then multiplied by a cooling factor to reduce the probability of selecting a worse solution in a given iteration. The function then returns the best parameters and the lowest value of the objective function from all iterations.

```python
def simulated_annealing(initial_params, objective_function,
    temperature, cooling_rate, num_iterations):
     current_params = np.array(initial_params)
     current_cost = objective_function(current_params)

     best_params = current_params
     best_cost = current_cost

     for iteration in tqdm(range(num_iterations)):
         # Generate a neighboring solution
         new_params = current_params +
         np.random.uniform(-0.1, 0.1,
         size=len(current_params))

         # Evaluate the objective function for the new solution
         new_cost = objective_function(new_params)

         # Accept the new solution if it's better or with a certain
    probability if it's worse
         if new_cost < current_cost or
         random.uniform(0, 1) < math.exp((current_cost - new_cost) /
     temperature):
             current_params = new_params
             current_cost = new_cost

         # Update the best solution if needed
         if current_cost < best_cost:
             best_params = current_params
             best_cost = current_cost

         # Reduce the temperature
         temperature *= cooling_rate

     return best_params, best_cost
```

Listing 5.3: Simulated annealing in context of hyperparameter tuning

5.4.2 Genetic Optimalization

Genetic optimisation is The most complex method we will introduce to the reader in our work. To speed up the optimization process, we used of the DEAP[7] [17] library, a library for creating experiments based on evolutionary optimization algorithms. The whole function is presented in the Listing 5.4.

Population and Individuals

Based on the input parameters for population size, number of iterations and search space, the toolbox from the DEAP library is initialized. The toolbox can be considered a container of registered functions with individual parameters. We specify

[7]Distributed Evolutionary Algorithms in Python

a fitness function to the DEAP library and register the class *Individual* for individuals in the population. Each object from this class denotes a different combination of hyperparameters from the search space on the input to the function. Next, we register the individual attributes as parameters entering the optimization.

Genetic Operators

Next, the genetic operators that are determined before the optimization starts need to be defined. These are the mutation parameter, crossover, and also need to determine how the parents will be selected to create the next population. In this case, application using tournament selection [32]. The selection is based on the fitness function value of each individual.

Main Genetic loop

An objective function is used to evaluate the fitness value of each individual. Based on these values, the algorithm proceeds further. The main optimization cycle is run as many times as there are inputs to the function. Each iteration creates a population with offspring through crossover and mutation. Another population is created using the selection and fitness value of individuals. The output of the function after several iterations are the best found combination of parameters and the value of the metric for the trained model.

```
def genetic_opt(pop_size: int, genetic_iterations: int,
    search_space: list):
     # Define individual and population
     creator.create("FitnessMax", base.Fitness, weights=(1.0,))
     creator.create("Individual", list, fitness=creator.FitnessMax)

     toolbox = base.Toolbox()
     toolbox.register("attr_float", np.random.uniform, abs(
    search_space[0][0]), abs(search_space[0][1]))
     toolbox.register("attr_int1", np.random.randint, search_space
    [1][0], search_space[1][1])
     toolbox.register("attr_int2", np.random.randint, search_space
    [2][0], search_space[2][1])
     toolbox.register("individual", tools.initCycle, creator.
    Individual, attr_set, n=1)
     toolbox.register("population", tools.initRepeat, list, toolbox.
    individual)

     # Register genetic operators
     toolbox.register("mate", tools.cxBlend, alpha=0.5)
     toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=1,
    indpb=0.2)
     toolbox.register("select", tools.selTournament, tournsize=3)

     # Evaluation function
     toolbox.register("evaluate", objective_function)

     # Main genetic algorithm loop
     population = toolbox.population(n=pop_size)

     for gen in tqdm(range(genetic_iterations)):
         offspring = algorithms.varAnd(population, toolbox, cxpb
    =0.7, mutpb=0.2)
         fits = toolbox.map(toolbox.evaluate, offspring)

         for ind, fit in zip(offspring, fits):
             ind.fitness.values = [fit]

         population = toolbox.select(offspring, k=len(population))
     best_individual = tools.selBest(population, k=1)[0]
     return tools.selBest(population, k=1)[0], best_individual[0]
```

Listing 5.4: Genetic optimalization with DEAP

5.4.3 Results of hyperparameter optimization

The results are displayed in individual tables for each method. The number of iterations and the search space were based on the time available for each method. For example, for genetic optimization, the number of iterations had to be lower than for random shooting due to the time required for each iteration. The result table presents our results from each model for cadence, heart rate and enhanced speed. In the first two rows, we can see the difference between the MAE of the model using default hyperparameters and the MAE of the model using tuned hyperparameters. After this, we have used the hyperparameters listed.

Random Shooting:

The random shooting was calculated for 100 iterations. We can see from the results in Table 5.3 that even using a simple method like Random Shooting, we get better hyperparameters than the standard XGBoost. The search space for learning rate was in the interval $\langle 0.001, 0.2 \rangle$, $\langle 3, 10 \rangle$ for max depth and $\langle 1, 5 \rangle$ for child weight sum.

Parameter	**Cadence**	**Heart rate**	**Enhanced Speed**
Default params. MAE	1.09	4.14	0.68
Tuned params. MAE	0.68	3.70	0.52
Learning rate after tuning	0.03	0.10	0.18
Max depth after tuning	5.00	9.00	3.00
Min child weight after tuning	3.20	1.43	3.47

Table 5.3: Results after hyperparameter tuning using Random Shooting

In the attached graph, see Figure 5.1 we can notice the evolution of the MAE value over time using the Random Shooting optimization method. MAE, as we can see, is only in negative values. The main idea of negative MAE is to keep the optimization logic consistent and that we are trying to maximize a given metric. But in the case of MAE, we are trying to minimize the metric, so it is converted to a negative value in the algorithm. If we want to interpret the values correctly, converting them to a positive value using an absolute value is not a problem.

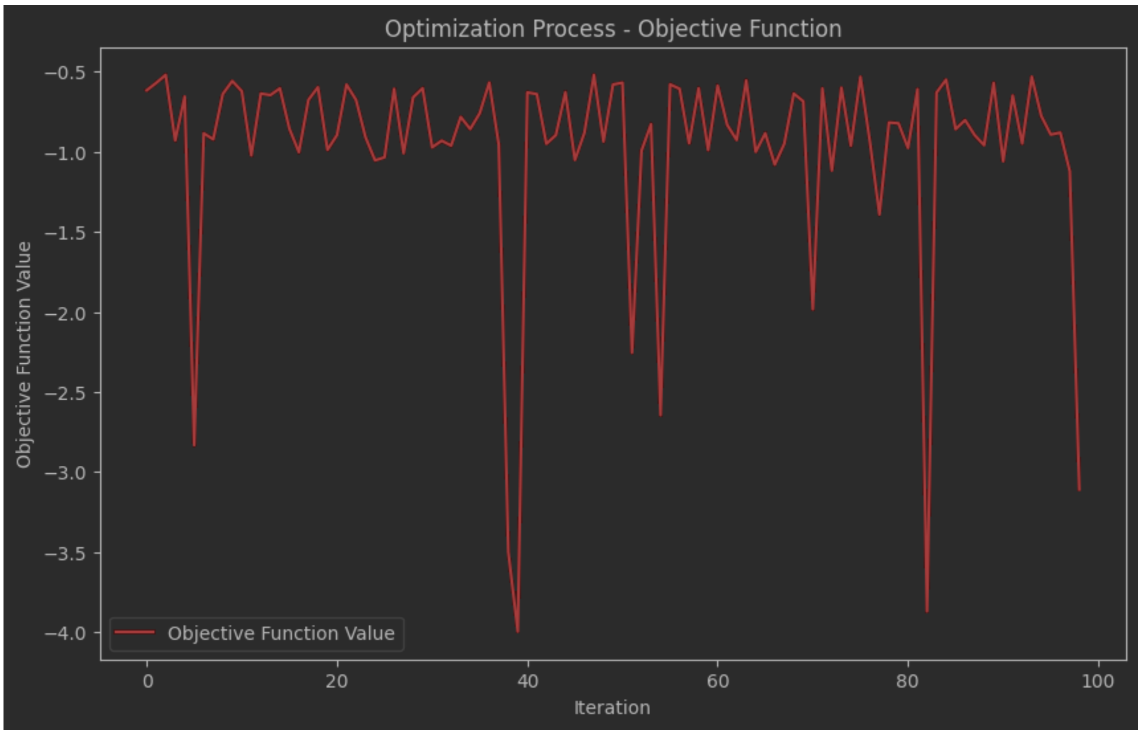


Figure 5.1: Loss function in Random Shooting optimization

Simulated Annealing:

In the case of simulated annealing, it is a heuristic algorithm that is very sensitive to the input parameters. As with random shooting, we had the algorithm set to 100 iterations. The temperature value was chosen to be 1.0, and the cooling parameter to be 0.95. The combination was chosen based on the author's experience so far. Unfortunately, the method did not yield optimal results even after several attempts, as shown in Table 5.4. One of the reasons why the optimization did not lead to optimal results may be poorly chosen initial parameters at the method's input.

Parameter	**Cadence**	**Heart rate**	**Enhanced Speed**
Default params. MAE	1.09	4.14	0.68
Tuned params. MAE	1.10	6.15	1.00
Learning rate after tuning	0.09	0.11	0.13
Max depth after tuning	7.00	10.00	8.00
Min child weight after tuning	1.80	6.72	6.99

Table 5.4: Results after hyperparameter tuning using Simulated Annealing

Genetic Optimalization:

From our point of view, genetic optimization is the most advanced heuristic algorithm that will be present here. At the same time, we have attempted a distributed computation for this algorithm that could speed up the optimization process. With a population size of twenty individuals and several generations of ten, we got the best results of the mentioned algorithms, and these resulting hyperparameters are used in the application. We can see the results in the Table 5.5.

Parameter	**Cadence**	**Heart rate**	**Enhanced Speed**
Default params. MAE	1.09	4.14	0.68
Tuned params. MAE	0.66	3.05	0.45
Learning rate after tuning	0.07	0.146	0.137
Max depth after tuning	2.00	8.00	2.00
Min child weight after tuning	1.23	4.06	2.40

Table 5.5: Results after hyperparameter tuning using Genetic Optimization

5.5 Feature Engineering

Feature engineering is one of the most important parts of preparing data for machine learning models. From a few features, you can produce many more using different approaches. This makes the base model more accurate but makes it more complex and risks overtraining the model. When producing features, we must also consider the correlation and significance of different variables. Introducing statistically insignificant variables or correlated features into the model would not make sense and could harm the model.

Feature engineering can be divided according to the approach of creation:

- Numerical features – Use of statistical functions such as mean, median standard deviation
- Categorical features – Transforming categorical variables into numerical variables because most models only work with numerical values
- Time-based features – Creating new features based on the time dimension. For example, whether it was a weekend or a holiday.
- Text-based – Creating new predictors based on text analytics.

5.5.1 Statistical approach to creating features

The statistical approach is one of the simplest and most basic approaches to creating new variables in a model. It usually uses basic statistical functions such as min, max, mean, etc. These functions cover basic information regarding the relationships between variables in the data. In this thesis, we used these functions to generate new features:

- Mean – The average values at a certain time interval.

- Max – The maximum value in the data set for a particular predictor. It gives us information about the maximum value measured.
- Min – The minimum value in the data set.
- Sum – Gives us the total sum of a given variable and can also give us the cumulative effect of a given variable over a specific time interval.

5.5.2 Creating features based on time dimension

Feature Engine [22], a Python library, was used to create features from the time dimension based on the time at which the values were measured. We used it because it is easy to implement directly in the application's code and offers other possible feature engineering techniques.

Types of new features based on time column:

- Month of the year
- Week of the year
- Hour
- Minute
- Second

Given the nature of the data, where time greatly affects our performance, these predictors positively affect the accuracy of the final model. Of course, as we reduce the time interval, the significance decreases. The other predictors tested based on the time dimension were the day of the week, weekend, and various combinations of quarters. Still, the analyses showed that these variables are correlated with the others and do not have sufficient significance in the model.

5.5.3 Creating features based on heart rate zones

A feature called the heart rate zone is very important in race prediction. It is a variable that tells us what heart rate zone we are currently in based on our heart rate. It can only take values from the interval $\langle 1, 5 \rangle$, where 5 is the highest zone and should always mean race. Of course, the data is subject to external effects, and we may reach such a heart rate zone during training. This is where the absence of a chest strap and wrist heart rate measurements with sports trackers become apparent.

Table 5.6 shows Athlete V's heart rate zones that enter the model as an example of how the values can look. For the final prediction, we assume that we will run the race in zone 5, so the model will predict based on higher intensity. We used this variable to solve a problem that permeated all the models we tried, which was that the model under-predicted the final prediction and over-predicted the final time.

Zone	Min. HR	Max. HR
1	50	140
2	141	156
3	156	166
4	167	176
5	177	210

Table 5.6: Heart rate zones

5.5.4 Features based on weather conditions

To find out the weather at the time of the activity at the exact location, the *meteostat* [43] library was used, which returns the values we specify from the nearest weather station. We use variables like temperature, humidity, rain, snow and wind speed. All of these variables enter the model and affect the prediction. If the library does not return any value, then this activity is discarded and not in the training set.

Only the start location is currently used to determine the venue, so this approach could be inaccurate for long activities where the athlete moves a large distance away. However, calculating the temperature at each point in the activity would be too time-consuming and would not have as much benefit. Longitude and latitude are used to represent the starting point.

5.5.5 Ascent, descent and slope of the hill

Other derived variables include the indication of ascent and descent within the elevation. The calculation works based on the difference of two consecutive elevations, and then we just ask whether the difference is greater than zero or less.

The formula used to calculate the slope of the elevation is

$$\frac{\text{elevation}}{\text{distance}} \cdot 100 \tag{5.1}$$

which returns the slope of the elevation as a percentage. From those derived variables, other variables are then derived based on lagged variables and moving averages.

5.5.6 Hills Segmentation

Due to increasing fatigue on longer hills, it was necessary to indicate in the data that this is a longer hill. So, we developed an algorithm that takes advantage of the monotonicity of the function and can tell when it is a long hill and a short hill. We can then find this information in the data, and the model can consider this.

In the Figure see 5.2, we can see the start and end of each hill in one activity. In the data, we can then find this information represented ternary. That is one for ascent, 0 for level and -1 for descent. In Table 5.7, we see improvements before and after adding the newly created variable. However, this is not the same test case in the figure and the table. Since the activity used to compare the results was not as diverse in elevation, the display would not be very telling.

	Reality	**Before change**	**After change**
Cadence[spm]	93.1	92.3	92.3
Heart rate[bpm]	183.0	177.8	177.9
Final time[mm:ss]	35:02	35:13	35:11

Table 5.7: Results using the gradient segmentation feature

Implementation:

- Segments are created where the altitude either rises or falls using the monotonicity function. So far, without identifying the ascent and descent.
- For each segment, the sign of the difference between the first and last point is found. The sign is used to determine whether it is a rise or a fall.

Results:

Testing on an activity that had a diverse profile. On a planar activity, of course, the feature does not have such an effect, and the difference would not be so marked.

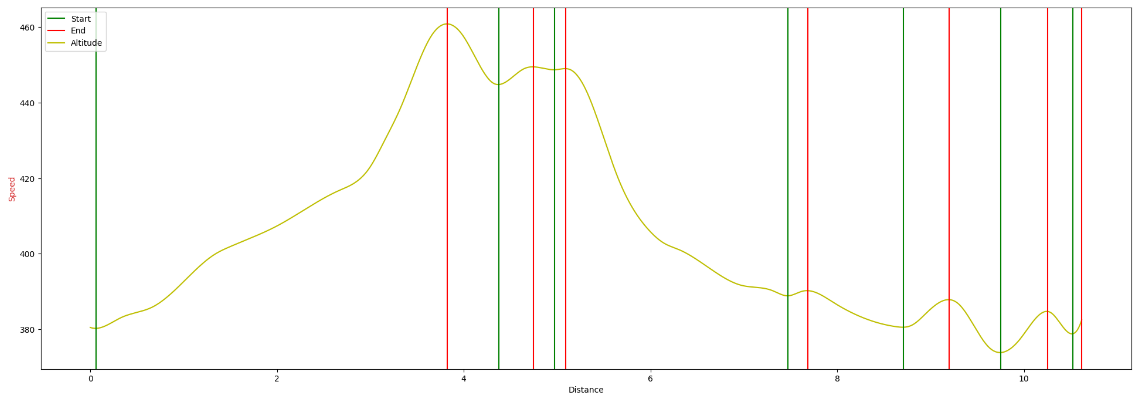


Figure 5.2: Segmentation of elevation

5.6 Model Training

This thesis uses three models for predicting athletes' inserted race and three for reference tracks. The three basic models predict cadence, heart rate, and movement speed. Other trained models are used to predict reference sections and compare multiple athletes.

Since there is a lot of automatic loading and saving of models in the application, it is necessary to have the training and testing logic implemented correctly, especially on time series, where we cannot randomly shuffle the data due to the time dimension.

One of the big problems that arises in training is that the models depend on each other sequentially. This has implications for the need for sequential training rather than parallel training. This dependency slows down the running of the application, and the user, thus, has to wait longer for his report.

Another problem that arises due to the models' dependency is the prediction's inaccuracy, which is carried by the whole application and is constantly increasing. What is meant by this is that the first model that predicts the cadence has some definite prediction error. This prediction error will be reflected in the heart rate model and increase even more. In addition, this error is propagated by derivative attributes such as moving averages for heart rate.

5.7 Map generation

Generating maps from a GPX[8] file is one of the features that the report offers users. For this purpose, was used the *Folium* [21] library, which can display your data in Python in called JavaScript-based leaflet maps

The Listing 5.5 is an example of a function that creates an interactive map based on the data in your GPX file. The *gen_map* function takes the name of the GPX file as input and outputs the generated HTML file in your folder. The *gpxpy* library extracts the geographic coordinates from the file.

[8]GPS Exchange Format

```python
def gen_map(track_name: str) -> None:
    # Read the GPX file and parse coordinates
    gpx_file = f"tracks/{track_name}.gpx"
    gpx_data = open(gpx_file, "r").read()
    gpx_parser = gpxpy.parse(gpx_data)

    # Extract coordinates from GPX data
    coordinates = []
    for track in gpx_parser.tracks:
        for segment in track.segments:
            for point in segment.points:
                coordinates.append(
                [
                point.latitude,
                point.longitude
                ])

    # Create a folium map centered around the first coordinate
    map_center = [coordinates[0][0], coordinates[0][1]]
    mymap = folium.Map(location=map_center, zoom_start=14)

    # Add polyline to the map
    folium.PolyLine(
    locations=coordinates,
    color="blue").add_to(mymap)

    # Save the map as an HTML file
    mymap.save(f"maps/map_{track_name}.html")
```
Listing 5.5: Generating map by Folium library

Chapter 6

Athletes comparison

This chapter focuses on comparing multiple athletes based on several metrics. The first metric is the benchmark stretches that were selected. The first section is a 10k race, the second is a half marathon, and the last section is a marathon. Based on the resulting predicted times, we are able to determine which of the two athletes would do better on that course. At the same time, the user will be able to see how he or she would have done on the reference leg.

The second comparison is based on feature importances[1], where we point out the strengths and weaknesses of each athlete based on the trained model. This gives the user the opportunity to compare themselves to a semi-professional runner and see which category they fit into more. Another output of this analysis is that the user can see where he is better than others or, on the contrary, where he suffers. Appropriate interpretation is, of course, absolutely crucial here.

Both comparisons can also be found in the output report for the general public. The presentation is in the form of simple and easy-to-read graphs. The chart for comparing multiple athletes is more difficult to interpret. If we had data for the general public, it would not be a problem to load multiple athletes into the chart.

6.1 Comparison of athlete with reference sections

Trained models were used to predict the times on the reference sections. Each section is standardized and its distance therefore corresponds to reality. In the first case, it is the Czech Republic Championship in Běchovice Praha. The second half marathon is a flat course in Hradec Králové, and the last marathon is one of the biggest races in the world, the Boston Marathon.

All three sections are also available in the final report for an interesting demonstration of how a given user would do on standard courses. The multi-athlete comparison will, of course, work for cycling as well, and if we put in a cycling race as a reference section, we can easily get predictions for that route. Of course, the data of

[1]Numerical value of features based on how useful they are at predicting a target variable.

the reference athlete is running.

To clearly understand the graphs listed below, we need to know that we will compare the speed of the movement of each athlete in kilometres per hour. The curve for each of the athletes shows speed at one point in time. On the second axis of the graph, we can see the altitude at which the athletes are moving. The x-axis then represents the distance variable.

6.1.1 10 kilometres

On the 10 km track, we can see Figure 6.1 that the difference between the athletes is not so noticeable, because the difference in predictions is only 20 seconds in final time. This is due to the fact that the inserted athletes are very similar in terms of performance. At the same time, insufficient data from earlier this year enters the model for athlete 1, but we will discuss this further in subsection 6.3.

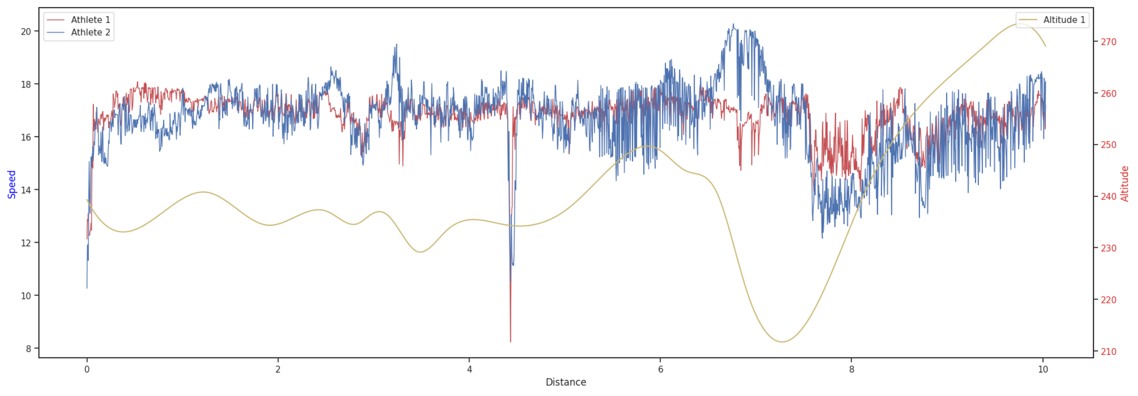


Figure 6.1: Comparison of two athletes on the track in Běchovice Praha

6.1.2 Half marathon

In the half marathon distance, we can already see a bigger difference in the final times. Athlete 1 did this distance in 1:15:48 and athlete 2 in 1:18:28, although from the graph, see Figure 6.2, it is not very noticeable, this difference is already noticeable. This difference confirms the assumption that Athlete 1 should complete the half marathon faster, which we made based on our knowledge of the data for both athletes. In reality, this is indeed the case and is based on the training direction of both of the athletes.

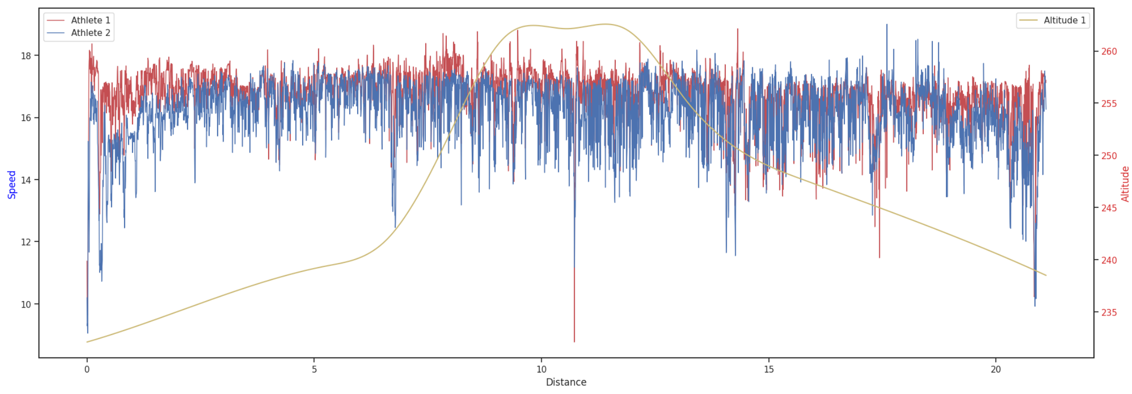


Figure 6.2: Comparison of two athletes on the Hradec Králové half marathon course

6.1.3 Marathon

Here, we come to the section where the reference section is only of interest to most users, given that not so many athletes run the marathon distance. None of the reference athletes have run this distance, so we are only estimating the accuracy of the prediction based on our knowledge of the subject. The final times will be very influenced by the fact that neither of the athletes trains for the required distance, but even so the predictions are very telling. The predictions can be seen in Figure 6.3

Athlete 1 should complete the marathon in 2:34:29, and Athlete 2 should complete the marathon in 3:14:55. As we mentioned before, we don't currently have anything to compare the accuracy to, so the benchmark distance is mostly to show that there is no specific geared to for the distance the athlete is training for. Still, Athlete 1 will do better than Athlete 2 because of the direction of his training.

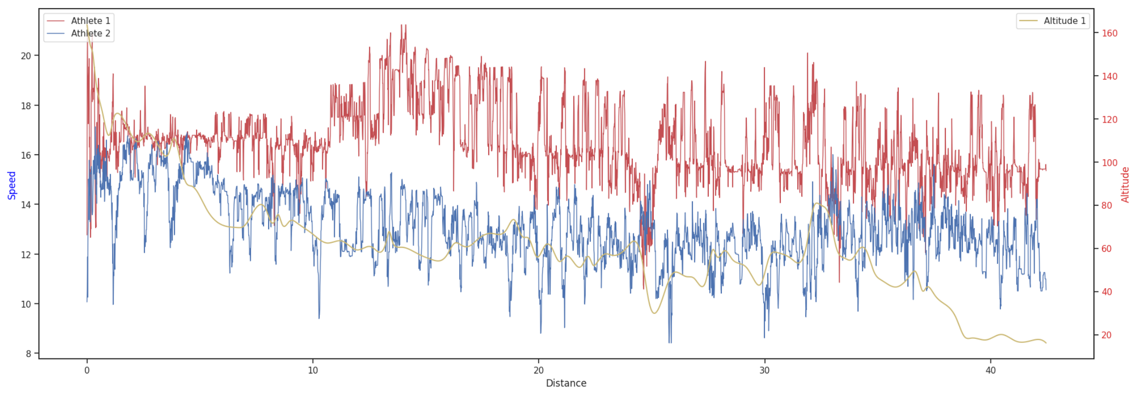


Figure 6.3: Comparison of two athletes on the Boston Marathon course

6.2 Example of athlete's strengths and weaknesses

To show which athlete is better at what, we chose only non-derived attributes because derived attributes are often quite abstract, and their interpretation can be very difficult, for example, with moving averages. There were other non-derived predictors, such as a measure of climb length or heart rate zone, but we chose not to show them in the graph because of their specific definitions. Anyway, the heart rate zone indicator is very significant. The climb length indicator is not, although it does make the predictions more accurate by a few seconds within half an hour. Radar-type graph was used to clearly show the strengths and weaknesses of the athlete.

6.2.1 Interpretation

In Figure 6.4 we can see some basic features, which we will now try to interpret so that the graph can be understood correctly. The further the curve is from the centre of the graph, the more it affects the behaviour of the model for a given athlete. The values present in the graph can be interpreted as feature importance scores.

- heart_rate – Heart rate. In this case, we can see that the heart rate of Athlete 2 will have a much greater impact on his performance. Together with the predictor of temperature (temp), this is the most explanatory variable in the Athlete 2 model.
- rain – Rain. We can see that rain has similar significance for both athletes.
- temp – Temperature. In the case of Athlete 1, temperature does not have much effect on his performance, but in Athlete 2 we can see that it is the second most significant variable.
- cadence – Cadence. Since cadence is very similar for most runners, the significance of this variable is almost the same for athletes.
- slope_ascent – Slope on the track. From the graph we can see that the slope on the track does not affect the performance of both athletes. Given that both athletes are more likely to be road runners, this behaviour was expected.
- slope_descent – Descent on the route. Like the ascent, the descent on the trail has little effect on the performance of either athlete.
- enhanced_altitude – Altitude. Although it would seem that altitude would not have much effect on performance, we can see from the graph that Athlete 1 is very much affected by this variable.

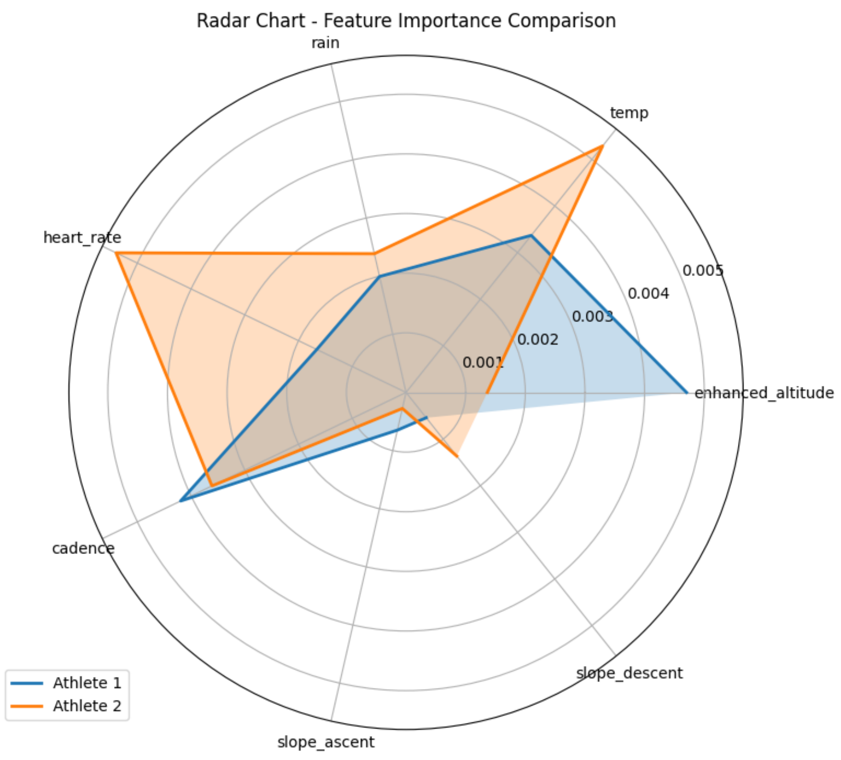


Figure 6.4: Demonstration of the strengths and weaknesses of two athletes

6.3 Data quality based comparison

This approach is no longer shown in the report because it is more complex information and requires deeper insight into the models, which is assumed the general user won't have. This subsection will point out the split of the data from both data sets. By doing so, we can reveal time intervals, where the athlete was not doing sport that may affect the accuracy of the prediction. Another indication will be the time spent in intensity, i.e. in higher heart rate zones, which have a greater impact on training. This will be able to determine how much a given athlete is dedicated to training.

In Figure 6.5, we can see the difference between the data granularity of Athlete 1 and Athlete 2. We can understand that over the last twelve months, Athlete 1 has a higher granularity of data and is predicted to have more accurate predictions, which was confirmed in the reference sections. Athlete 2 has sufficient data for up to the last six months. Since the last year of data enters the model, the amount of data at the beginning of 2023 for Athlete 2 will likely have a negative impact on his predictions.

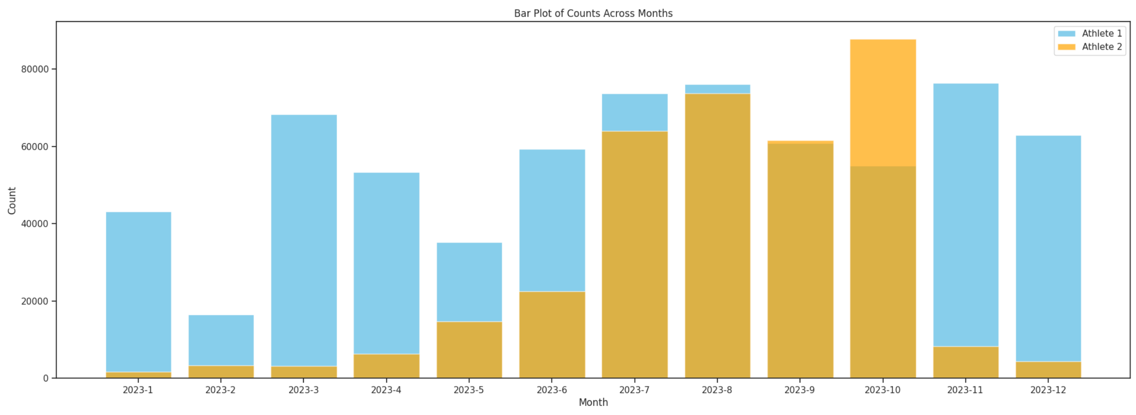


Figure 6.5: Sample number of data for each month

Chapter 7

User interface

7.1 Application Guide

The application can be controlled using the created Jupyter notebooks or by using CLI[1]. To use Jupyter it is of course necessary to have a prepared environment, so this option is more for advanced users.

- Jupyter notebook – In the root folder is created a notebook *manage.ipynb*, which contains all important steps in each cell and if the user enters the necessary parameters, the notebook will generate a report.
- CLI – The application can also be controlled using the command line. Just call Python on the *main.py* file using a virtual environment that has all the necessary libraries installed, which are contained in the *requirements.txt* file.

To successfully run an application using the CLI, you must call the application with two mandatory parameters:

- gpx_file - The name of the GPX file that is inserted into the *tracks* folder. race_day - Race time in *YYYY-mm-dd-HH-MM* format

The individual steps are logged and their execution is recorded, either as cell output in the notebook or in the command line. In addition, the entire background run is logged to an external file that can be found in the root directory.

Most parameters, such as the name of the athlete or the type of activities, are requested by the application at the beginning of the run, but some are stored in a configuration file that is YAML formatted. This file contains the necessary file paths and/or other necessary parameters that are not specified by the user.

[1]Command Line Interface

7.2 Report

Once the models have been successfully trained and predictions made, a report is automatically generated from the application in HTML format. This report can be viewed in any browser and gives the user an instant overview of their future performance.

The first graphs in the report, see Figure 7.1 show how the user will perform on the inserted track. We find three graphs, the first for cadence, the next for heart rate and the last for speed. The order corresponds to the training of each model. In addition, each graph shows the average value and the elevation on the track. On the x-axis we find the distance of the given track and on the y-axis the value of the predicted variable.

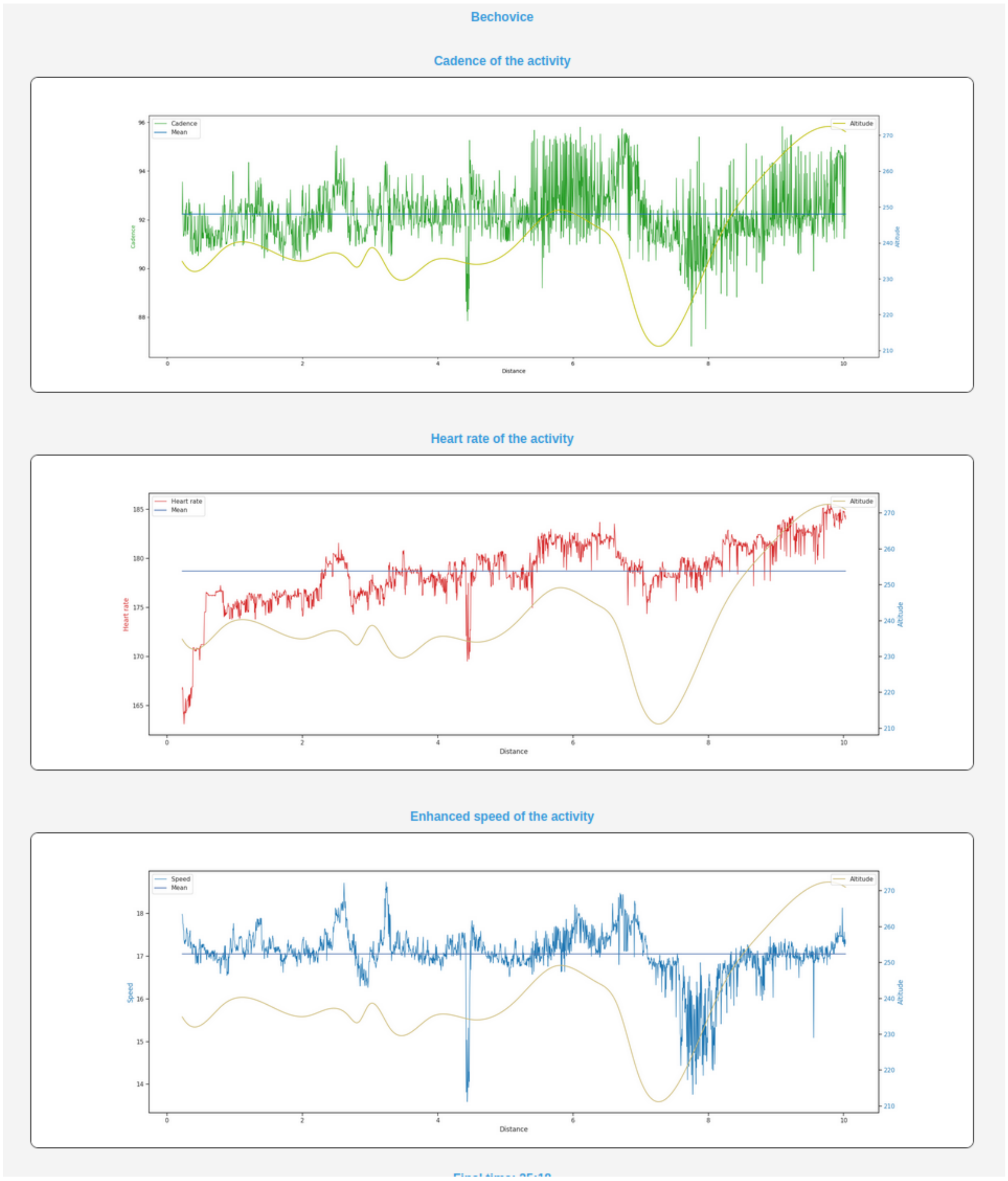
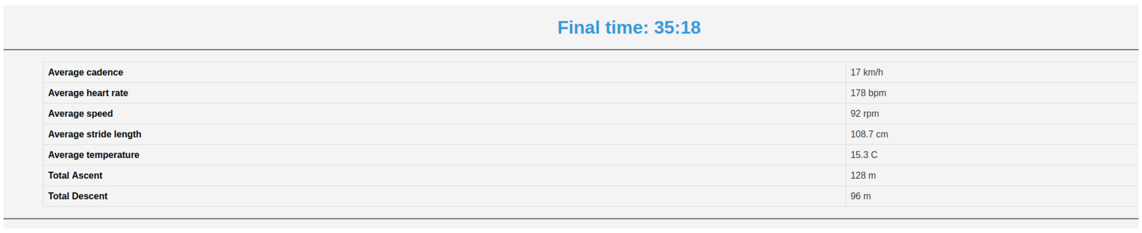


Figure 7.1: First three graphs of report

Another source of information in the report is a table (see Figure 7.2) that shows the average values from the predictions and also the predicted final time, which is calculated from the speed prediction. In the table, we also find the maximum values of all three predicted quantities, plus other interesting values, such as the average step length and the average temperature.


Final time: 35:18

Average cadence	17 km/h
Average heart rate	178 bpm
Average speed	92 rpm
Average stride length	108.7 cm
Average temperature	15.3 C
Total Ascent	128 m
Total Descent	96 m

Figure 7.2: Table with additional information

One of the last sections, see Figure 7.3, which is offered to the user, is a section where the user can compare himself with other athletes using a trained model. The athlete is presented with a graph where it is easy to see in which ways he is better than the reference runner and in which ways he is not. At the same time, he is offered predictions on several well-known courses around the world from 10 kilometres marathon as well as a track view. The user may never have run these distances, although, of course, the model can get very confusing as the distance increases if the athlete focuses mostly on shorter distances.

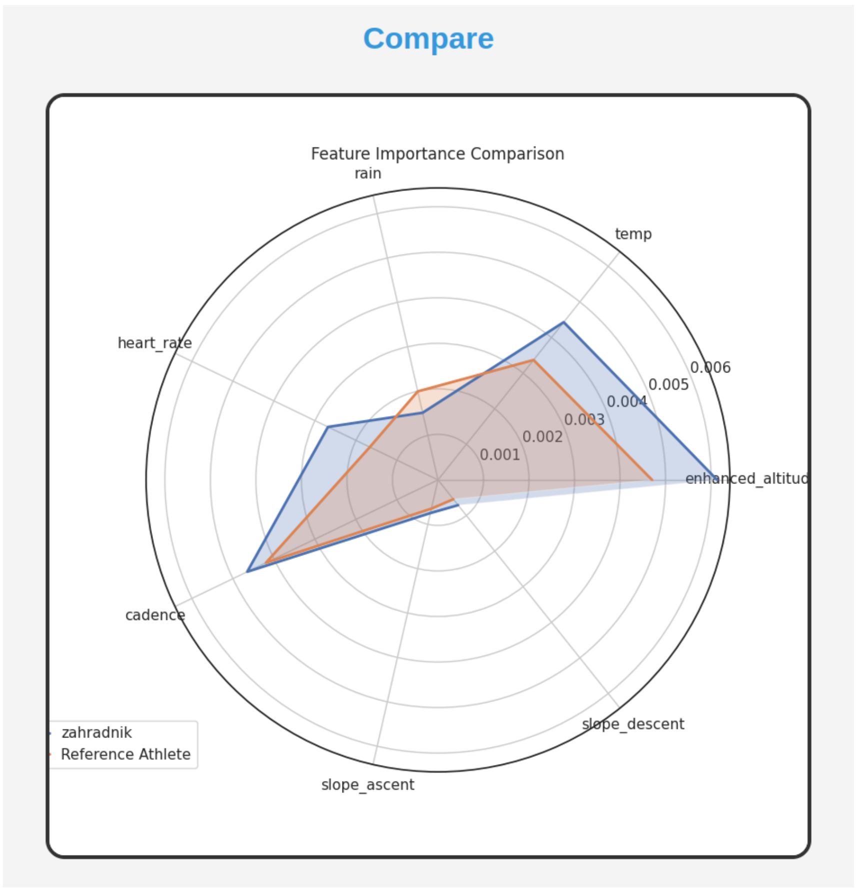


Figure 7.3: User versus reference athlete in Report

In the last row, the user is shown an activity map. The display is interactive, and the user can move around the map freely. You can see the map in Figure 7.4

Map of the Race

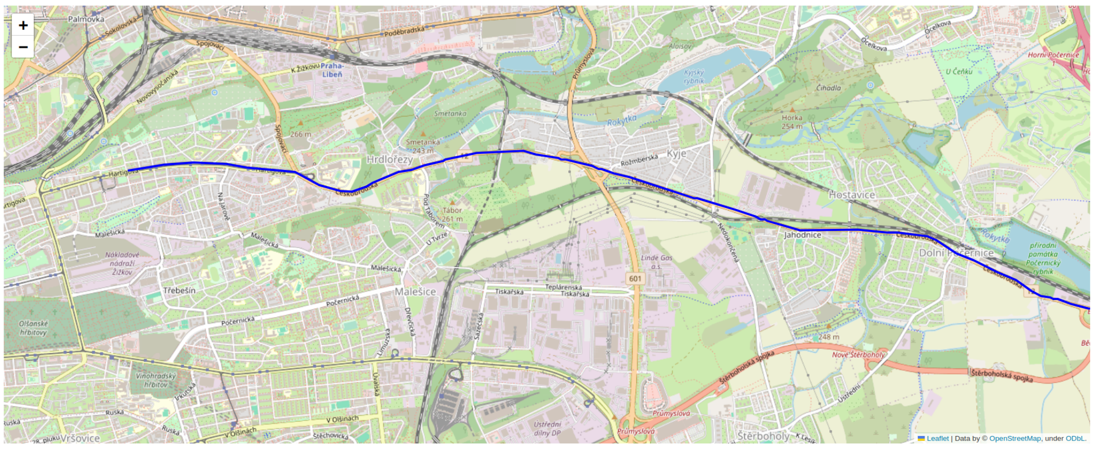


Figure 7.4: Map of the inserted race in report

Chapter 8

Testing

In this chapter, a few races will be mentioned and where we participated to test our work. The launch of the application always took place a few days before the actual race to ensure that the data collected from third-party applications was as accurate as possible. Mostly it is weather data.

In the graphs, we see variables such as the speed of the movement of the athlete, called *enhanced_speed* and altitude, which stands for altitude on the track on exact latitude and longitude. Also, the mean of the speed of the athlete is presented.

8.1 Birell Grand Prix

The application was tested at different distances and on different tracks. The first testing was at the Prague Birell Grand Prix, which was a 10 kilometres course that was mostly flat, but with a few switchbacks to slow the athlete down. Even this reversal of direction showed up in the final prediction. So we can say that the model took this into account as well.

Table 8.1 points out the differences between the real measured values by the sport tracker and the predicted values a few days before the race. We can see that the measured and predicted values do not differ much and we consider the testing very successful. The main metric is the Final time, which is the final time of the race.

	Reality	**Prediction**
Cadence[spm]	93	92
Heart rate[bpm]	184	175
Final time[mm:ss]	34:18	33:50

Table 8.1: Predicted and real data on Birell Grand Prix

In Figure 8.1 we can see how the model predicts the speed of the race. Since the GPX file with only zero elevation was provided to us, the yellow curve is constant, and the model does not take elevation into account. Which may affect its accuracy, but unfortunately a better GPX file could not be obtained.

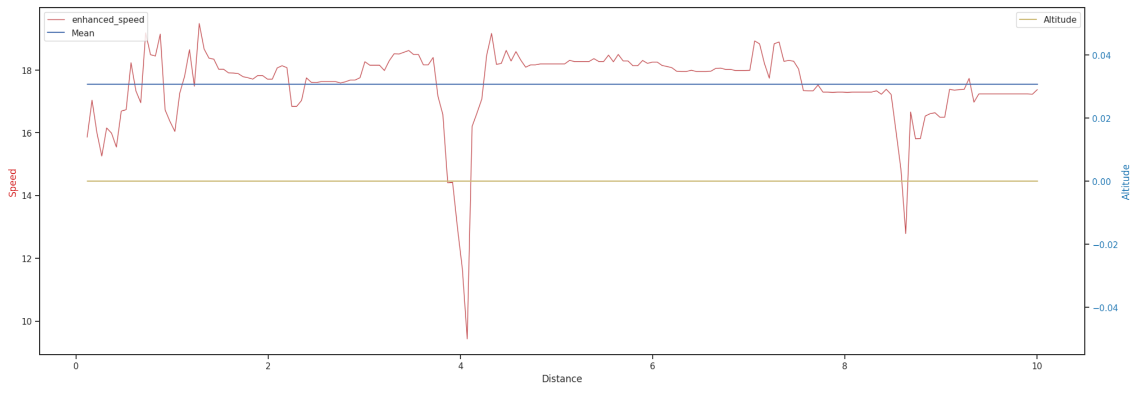


Figure 8.1: Birell Grand Prix speed chart

In Figure 8.2 we can see the importance of each predictor. In the current case, it is clear that the most important variables were distance and temperature. Altitude and humidity were also very important for the model.

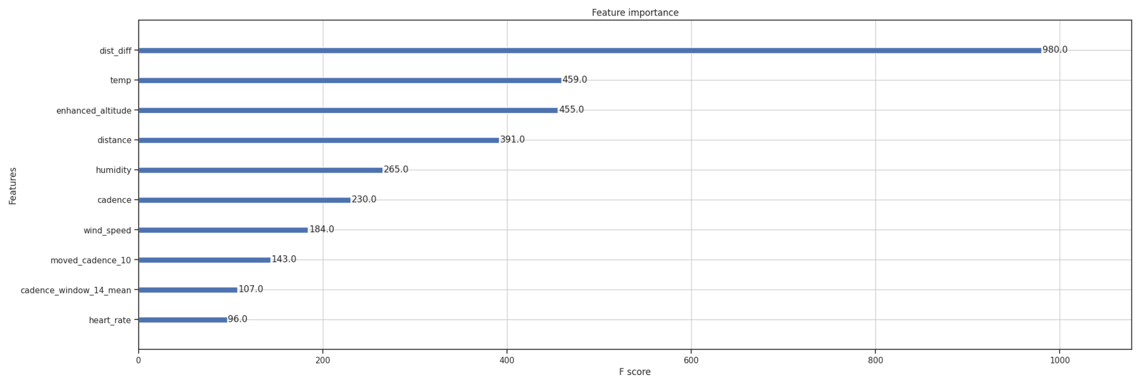


Figure 8.2: Feature importances for Birell Grand Prix

As a final evaluation of this testing on a track that measured ten kilometres, we can say that the final speed prediction model was very accurate and the testing was successful.

8.2 Běchovice Praha

The second testing was on the road race at Běchovice Praha, where the track is also somewhat rugged, which makes the resulting models more accurate, as we can see in the Table 8.2. If we compare the test with the previous race, the prediction is even more accurate than in the previous testing.

	Reality	**Prediction**
Cadence[spm]	93.0	92.3
Heart rate[bpm]	184.0	177.9
Final time[mm:ss]	35:02	35:11

Table 8.2: Predicted and real data on Běchovice Praha

If we look at Figure 8.3 we can see that the trained model respects the elevation at the end of the track and reacts quickly to the current hill.

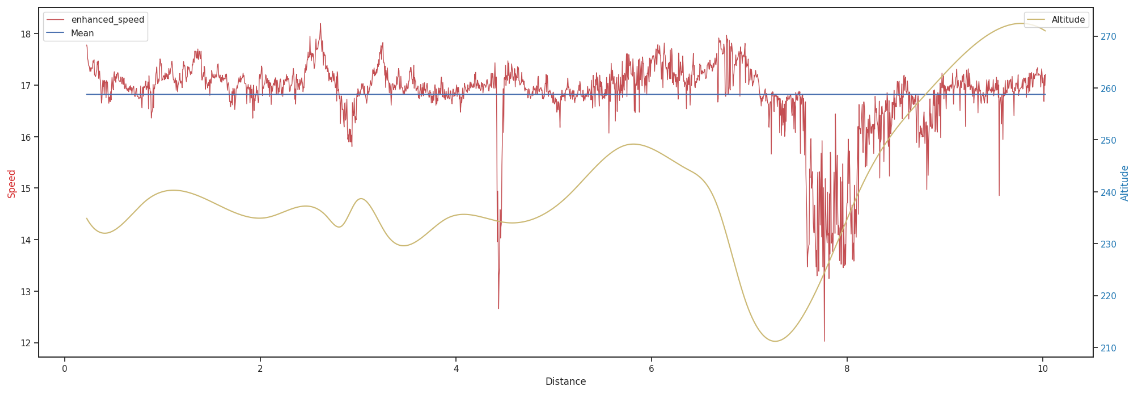


Figure 8.3: Běchovice Praha speed chart

8.3 Half marathon Hradec Králové

The third and last test was on a longer course than the previous tests. It was the half marathon Hradec Králové, a distance of over 21 kilometres. For this distance, the model was expected to perform rather worse (underestimate the athlete) and predict slower times, given that athlete V did not have focused training for this type of exercise. Which, of course, affects the accuracy of the prediction.

In the final Table 8.3 we can see that the model was very accurate and its final time prediction differed by only 21 seconds, which on times within the clock is a very small deviation.

The overall speed in the race can be seen in Figure 8.4. The actual speeds are very close to the average, which is usually the case in endurance distance running unless it is a rugged course.

	Reality	**Prediction**
Cadence[spm]	92.6	93.0
Heart rate[bpm]	178.0	182.45
Final time[mm:ss]	1:13:53	1:13:32

Table 8.3: Predicted and real data on half marathon Hradec Králové

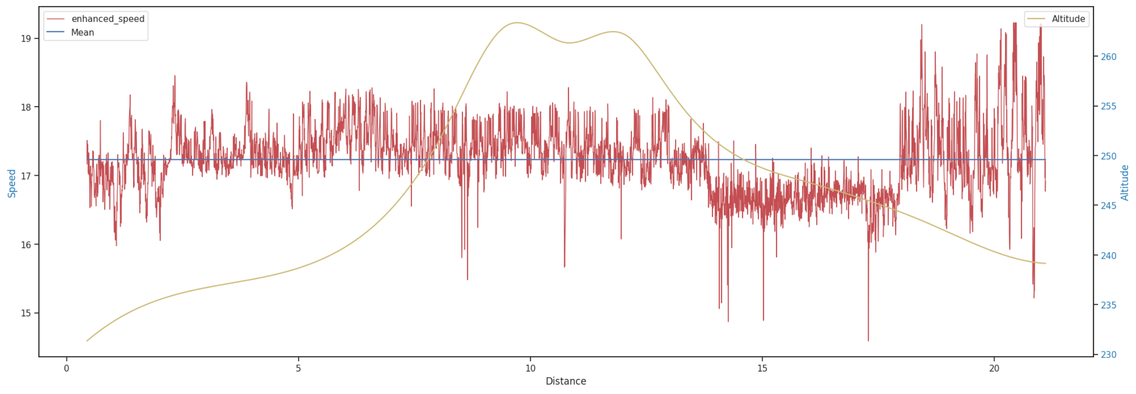


Figure 8.4: Speed chart for Hradec Králové half marathon

Conclusion

The aim of this master thesis was to extend our previous research on a similar topic of sports data analysis with more weight on automation and optimization of the resulting models using heuristic optimization. At the same time, there was a need to focus on a user-friendly interpretation of the results in a simple report with the possibility of user comparison with the wider community.

In this thesis, you can find research where we focused on two companies that offer applications working with sports data. We compared these two applications with this solution and also described the possible functioning of their applications. We also pointed out the shortcomings and sources of inaccuracies of these applications. The reasons why not not using Strava's data were also mentioned.

If we had to give a final verdict between the applications implemented and the apps mentioned in the market research chapter, our work would generally perform better than Garmin's. This is, due to the assumption of greater data accuracy and not using the estimated Vo2Max value. On the other hand, Stryd, which uses very accurate data based on watts, has predictions far more accurate than the app we created. As we mentioned before, the reason is mostly due to the use of watts in the predictions.

In the next two chapters of the thesis, we described the theoretical foundations of machine learning and time series analysis. For machine learning, we introduced the reader to the basic approaches to learning and then described the algorithms that are commonly used today. All of these algorithms resulted in the model application used, which is XGBoost. We considered knowledge of the previous models to be important, so we decided to describe them as well.

For the time series analyses, we briefly described the issues and basic concepts. We also introduced the metrics that are used to compare models based not only on time series but also on tabular data. We then focused on data preparation for the purpose of the analyses and listed the possible types of time series analyses.

In the last chapter of the theoretical part of the thesis, we described heuristic algorithms and their use for the purpose of hyperparameter tuning. In this chapter, we described all the necessary terms related to this problem and mention the heuristic algorithms we used to find the optimal solution for these parameters. The exact solution can be found in the implementation section of the thesis, along with the results.

The practical part started with a description of the automation of data collection and preprocessing, which has been enriched with extensive feature engineering compared to our previous research. Next, we presented the reader with the interesting parts of the code that dealt with hyperparameter tuning and feature engineering. The feature engineering itself is extensively described in its subsection, where we explain the approaches we used to create the new predictors, and also show the results before and after adding each predictor.

As for comparing athletes to the general public, this section has been extended from the research task to include new predictors, offering the user the opportunity to compare themselves to a reference athlete from a broader perspective. At the same time, reference sections were added for each running endurance distance. These are all outputs the athlete will find in the output report. Additionally, we compared the distribution of each athlete's data as part of the research.

The output report displays everything the user might be interested in using simple tables and graphs. The user does not need to have extensive knowledge of the subject. As a shortcoming of this work, we consider the user interface not graphical, but only text-based via the command line, which can be difficult for some users.

The major problem that we had was access to Garmin's data. Most of the data we had was from this company, and to completely automate it so that the application didn't have to know users' passwords from their Garmin account, we would need a developer license, which is contingent on registration to the company, as was mentioned in the implementation chapter. Additionally, in recent months Garmin has tightened their rules and data can no longer be downloaded even with a password via the API without a license. So, the future extension is clear, and that is to link the app with a developer license from Garmin for automated data collection. Unfortunately, we are currently dependent on manual export again.

If we discuss the benefits of this work in the industry, we should mention that we have been able to get similar or sometimes even more accurate race predictions than Garmin has in general and unified the functions offered by the companies mentioned in the research in their applications. Although they have a very large amount of data available, their application expansion has been slow, which we gather from having used their services for several years.

Among other possible extensions, we would definitely mention the possibility of improving the user interface with graphical access and not having access only through the command line. Of course, another improvement would be to get more data from a wider population, which is a conditional requirement from Garmin. If we were to give some of our ideas for possible uses of this work, since the application was trained as a model based on the activities of an athlete, the model can be used in many ways. For example, in detecting segments[1] that the athlete has a chance to win, or for creating workouts that would result in improving the user's fitness. Lastly, we would mention the ability to upload the app to a Cloud service and the ability to deliver the app online.

Overall, we would evaluate this work positively, although we faced the aforemen-

[1]Virtual race routes created by the running and cycling community where users can compare

tioned problems, such as a lack of data or problems in automating data collection. We offered the user features that can be used by a novice athlete or even a professional. The overall run time of the application is within a few minutes, which, considering training several models, is a very good performance. Of course, this statement is conditional on the device on which the application runs given that it is a local application.

Literatura

[1] *Autocorrelation and partial autocorrelation functions.* 2018. URL: https://www.ibm.com/docs/en/spss-modeler/18.2.0?topic=data-autocorrelation-partial-autocorrelation-functions (visited on 11/17/2023).

[2] N. AZARIA. *Feature Importance.* 2023. URL: https://www.aporia.com/learn/feature-importance/feature-importance-7-methods-and-a-quick-tutorial/ (visited on 12/10/2023).

[3] E. BAELDUNG. *Node Impurity in Decision Trees.* 2022. URL: https://www.baeldung.com/cs/impurity-entropy-gini-index (visited on 11/17/2023).

[4] *Bagged Trees.* 2016. URL: https://bookdown.org/mpfoley1973/data-sci/bagged-trees.html (visited on 11/17/2023).

[5] V. BAGHEL. *Math behind GBM and XGBoost.* 2019. URL: https://medium.com/analytics-vidhya/math-behind-gbm-and-xgboost-d00e8536b7de (visited on 11/17/2023).

[6] *Běchovice - Praha 2024.* URL: https://www.bechovice-praha.cz/ (visited on 01/07/2024).

[7] N. BENATTI and A. MAURIN. *Time series outlier detection, a data-driven approach.* 2020. URL: https://www.bis.org/ifc/publ/ifcb57˙07.pdf (visited on 11/17/2023).

[8] R. BEVANS. *Akaike Information Criterion.* 2023. URL: https://www.scribbr.com/statistics/akaike-information-criterion/ (visited on 12/10/2023).

[9] Lynn Brown. "A Comparison of Multiple Regression Models to Help Predict Road Race Performance for Two Runners". Florida: University of North Florida, 2003. URL: https://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1101%5C&context=ojii˙volumes (visited on 06/23/2022).

[10] J. BROWNLEE. *A Gentle Introduction to Ensemble Learning Algorithms.* 2021. URL: https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/ (visited on 11/17/2023).

[11] B. BRUGGER. *Adaptive Boosting: A stepwise Explanation of the Algorithm.* 2021. URL: https://towardsdatascience.com/adaptive-boosting-a-stepwise-explanation-of-the-algorithm-50b75c3729c1 (visited on 11/17/2023).

[12] M. BURGER. *Beginning Time Series Analysis and Forecasting with R.* URL: https://www.pluralsight.com/courses/r-time-series-analysis-forecasting (visited on 11/17/2023).

[13] K. Cherry. *What Are Heuristics?* c2022. URL: https://www.verywellmind.com/what-is-a-heuristic-2795235 (visited on 06/23/2022).

[14] *Computer Vision - What it is and why it matters.* 2023. URL: https://www.sas.com/cs˙cz/insights/analytics/computer-vision.html (visited on 12/10/2023).

[15] D. COTE. *RdR score metric for evaluating time series forecasting models.* 2022. URL: https://medium.com/@dave.cote.msc/rdr-score-metric-for-evaluating-time-series-forecasting-models-1c23f92f80e7 (visited on 11/17/2023).

[16] *Cycling, Running and Hiking app.* 2009. URL: https://www.strava.com/ (visited on 12/10/2023).

[17] *DEAP documentation.* 2009. URL: https://deap.readthedocs.io/en/master/ (visited on 01/07/2024).

[18] *Decision Trees.* URL: https://scikit-learn.org/stable/modules/tree.html (visited on 01/08/2024).

[19] *Entropy: How Decision Trees Make Decisions.* 2019. URL: http://tinyurl.com/bdfycfxn (visited on 11/17/2023).

[20] *Flexible and Interoperable Data Transfer (FIT) Protocol.* URL: https://developer.garmin.com/fit/protocol/ (visited on 01/07/2024).

[21] *Folium.* 2013. URL: https://python-visualization.github.io/folium/latest/%5C# (visited on 12/08/2023).

[22] S. GALLI and M. SELL. *A Python library for Feature Engineering and Selection.* 2018. URL: A%20Python%20library%20for%20Feature%20Engineering%20and%20Selection (visited on 11/17/2023).

[23] *Garmin International.* 2021. URL: https://www.garmin.com/en-US/ (visited on 12/10/2023).

[24] *Gradient Boosting in ML.* 2023. URL: https://www.geeksforgeeks.org/ml-gradient-boosting/ (visited on 11/17/2023).

[25] W. HENSHALL. *4 Charts That Show Why AI Progress Is Unlikely to Slow Down.* URL: https://time.com/6300942/ai-progress-charts/ (visited on 12/10/2023).

[26] *How Does the Race Predictor Feature Work on My Watch?* 2023. URL: https://support.garmin.com/en-US/?faq=HUB4yrzJkg1BbgmozWkBm7 (visited on 11/18/2023).

[27] *JetBrains.* 2022. URL: https://www.jetbrains.com/ (visited on 07/06/2022).

[28] T. JEWELL. *Everything to Know About VO Max.* 2023. URL: https://www.healthline.com/health/vo2-max (visited on 11/18/2023).

[29] J: JORDAN. *Hyperparameter tuning for machine learning models.* URL: https://www.jeremyjordan.me/hyperparameter-tuning/ (visited on 01/02/2024).

[30] *Jupyter.* 2022. URL: https://jupyter.org/ (visited on 07/06/2022).

[31] V. KANADE. *What Are Genetic Algorithms? Working, Applications, and Examples.* 2006. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-are-genetic-algorithms/ (visited on 12/08/2023).

[32] T. KRISHNAMOHAN. *Tournament Selection in Genetic Algorithms.* URL: https://www.thearmchaircritic.org/mansplainings/tournament-selection-in-genetic-algorithms (visited on 01/02/2024).

[33] D. KUMAR. *A Complete understanding of LASSO Regression.* 2023. URL: https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/ (visited on 12/10/2023).

[34] V. KURAMA. *A Guide to AdaBoost: Boosting To Save The Day.* 2019. URL: https://blog.paperspace.com/adaboost-optimizer/ (visited on 11/17/2023).

[35] G. LAWTON. *What is Anomaly Detection?* 2022. URL: https://www.techtarget.com/searchenterpriseai/definition/anomaly-detection (visited on 12/10/2023).

[36] *Learning Python for Data Analysis.* URL: https://bootcamp.cvn.columbia.edu/blog/learning-python-for-data-analysis/ (visited on 01/07/2024).

[37] P. MANDROT. *What is LightGBM, How to implement it? How to fine tune the parameters?* 2017. URL: https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc (visited on 11/17/2023).

[38] M. MASLAKOVIC. *Garmin Race Predictor: how does it work, is it accurate?* 2022. URL: https://gadgetsandwearables.com/2022/09/19/garmin-race-predictor-accuracy/ (visited on 11/18/2023).

[39] M. MOJZEŠ. “Integer Optimization Heuristics”. Dizertační práce. Praha: ČVUT v Praze, 2017.

[40] A. OPPERMANN. *What Is CatBoost?* 2023. URL: https://builtin.com/machine-learning/catboost (visited on 12/10/2023).

[41] S. PANDIAN. *K-Fold Cross Validation Technique and its Essentials.* 2023. URL: https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/ (visited on 11/17/2023).

[42] S. Pandian. *Time Series Analysis and Forecasting — Data-Driven Insights.* 2023. URL: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/ (visited on 11/17/2023).

[43] *Python library — Meteostat Developers.* 2022. URL: https://dev.meteostat.net (visited on 11/17/2023).

[44] *Random Forests.* 2023. URL: https://bookdown.org/mpfoley1973/data-sci/random-forests.html (visited on 12/10/2023).

[45] P. RAVAL. *A Practical Introduction to Moving Average Time Series Model.* 2023. URL: https://www.projectpro.io/article/moving-average-time-series-model/716 (visited on 11/17/2023).

[46] S. RAVICHANDRAN. *Boost Your Machine Learning with XGBoost.* 2023. URL: https://www.linkedin.com/pulse/boost-your-machine-learning-xgboost-santhiya-r (visited on 11/17/2023).

[47] *Run with Power.* 2017. URL: https://www.stryd.com/eu/en (visited on 12/10/2023).

[48] E. SAMHAYEV, K. RASUL, and N. ROGGE. *Yes, Transformers are Effective for Time Series Forecasting.* 2023. URL: https://huggingface.co/blog/autoformer (visited on 11/17/2023).

[49] K. SHUBMAN. *Kaggle winners algorithm xgboost.* URL: https://medium.com/@MrBam44/kaggle-winners-algorithm-xgboost-87819eb300ae (visited on 01/07/2024).

[50] E. SIMHAYEV, K. RASUL, and N. ROGGE. *Yes, Transformers are Effective for Time Series Forecasting.* URL: https://huggingface.co/blog/autoformer (visited on 01/07/2024).

[51] B. SONI. *Stacking to Improve Model Performance: A Comprehensive Guide on Ensemble Learning in Python.* 2023. URL: https://medium.com/@brijesh˙soni/stacking-to-improve-model-performance-a-comprehensive-guide-on-ensemble-learning-in-python-9ed53c93ce28 (visited on 11/17/2023).

[52] H. Starke. "Regression Analysis of Pacing When Running a Marathon". Undergraduate Honors Theses. Fayetteville, Arkansas: University of Arkansas, 2021. URL: https://scholarworks.uark.edu/cgi/viewcontent.cgi?article=1077%5C&context=ineguht (visited on 06/23/2022).

[53] R. SWANSON. *Leaf-wise (best-first) and level-wise traversal of decision trees.* 2023. URL: https://copyprogramming.com/howto/decision-trees-leaf-wise-best-first-and-level-wise-tree-traverse (visited on 11/17/2023).

[54] TASHMIT. *LightGBM.* 2023. URL: https://www.codingninjas.com/studio/library/lightgbm (visited on 11/17/2023).

[55] CFI Team. *What is a Ridge?* 2023. URL: https://corporatefinanceinstitute.com/resources/data-science/ridge/ (visited on 12/10/2023).

[56] Stryd Team. *How to Predict the Finishing Time of Your Next Race with Stryd.* 2023. URL: https://blog.stryd.com/2023/07/13/how-to-predict-finishing-time-of-your-next-race/ (visited on 11/18/2023).

[57] *Time series and moving averages.* 2022. URL: https://www.accaglobal.com/gb/en/student/exam-support-resources/fundamentals-exams-study-resources/f5/technical-articles/time-series.html (visited on 11/17/2023).

[58] *Top Computer Languages.* 2021. URL: https://statisticstimes.com/tech/top-computer-languages.php (visited on 08/26/2022).

[59] *Types of Machine Learning.* 2011. URL: https://www.javatpoint.com/types-of-machine-learning (visited on 11/17/2023).

[60] A. VASWANI. *Attention Is All You Need.* Neural Information Processing Systems, 2017. URL: https://proceedings.neurips.cc/paper˙files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (visited on 12/10/2023).

[61] *Voting Classifier using Sklearn.* 2019. URL: https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/ (visited on 11/17/2023).

[62] *What is a trend in time series?* 2022. URL: https://www.geeksforgeeks.org/what-is-a-trend-in-time-series/ (visited on 11/17/2023).

[63] *What is Bagging?* 2023. URL: https://www.ibm.com/topics/bagging (visited on 11/17/2023).

[64] *What is Boosting?* 2023. URL: https://aws.amazon.com/what-is/boosting/ (visited on 11/17/2023).

[65] *What Is Stimulated Annealing?* 2020. URL: https://www.mathworks.com/help/gads/what-is-simulated-annealing.html (visited on 12/08/2023).

[66] *What is wrong with the Garmin Race Predictor?* 2020. URL: http://tinyurl.com/3euxwj44 (visited on 11/18/2023).

[67] *Word2vec.* URL: https://www.tensorflow.org/text/tutorials/word2vec (visited on 01/07/2024).

[68] ZACH. *Label Encoding vs. One Hot Encoding: What's the Difference?* 2022. URL: https://www.statology.org/label-encoding-vs-one-hot-encoding/ (visited on 12/10/2023).

[69] V. ZAHRADNÍK. *Master thesis.* 2023. URL: https://github.com/VojtaZahradnik/master˙thesis (visited on 12/29/2023).

[70] V. ZAHRADNÍK. "Optimalizace modelů regresní analýzy sportovního výkonu". Výzkumný úkol. Praha: České vysoké učení technické v Praze, 2022.