E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Application of Machine Learning for Cycling Result Predictions

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Mikel Salvoch Vilches

Director: Mikel Sesma Sara

Pamplona/Iruña, 7 de junio de 2024

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Abstract and keywords

English

Abstract

This project aims to predict the results of road cycling races using Machine Learning techniques. This process includes the construction of a dataset which is done by first reviewing the existing literature to determine the key factors in the performance of a cyclist, followed by consulting various data sources to finally building web scraping scripts in order to recover the data.

With the dataset built, an experimental setup is designed in which different configurations of the dataset and the variables are considered, together with several Machine Learning regression models and their hyperparameters.

Finally, the results are evaluated in terms of numerical metrics to determine the most accurate approximations. Additionally, some conclusions are drawn based on these results and the completed work.

Keywords: *cycling, bicycle, machine learning, prediction, results.*

Euskara

Laburpena

Proiektu honek ikasketa automatikoko teknikak erabiliz errepideko txirrindularitza lasterketen emaitzak aurreikustea du helburu. Prozesuak *dataset* baten eraikuntza ere barne hartzen du. Horretarako, lehenik eta behin, dagoen literatura berrikusten da, txirrindulari baten errendimenduaren funtsezko faktoreak zehazteko. Ondoren, hainbat datu-iturri kontsultatzen dira, eta azkenik, *web scraping*-a egiteko *script* bat eraikitzen da aurkitutako datuak bildu ahal izateko.

Dataset-a behin eraikita dagoela, konfigurazio esperimental bat diseinatzen da, non *dataset*-aren eta aldagaien konfigurazio ezberdinak kontuan hartzen diren, ikasketa automatikoaren erregresio eredu batzuekin eta horien hiperparametroekin batera.

Azkenik, emaitzak metrika kuantitatiboen bidez ebaluatzen dira hurbilketa zehatzenak zehazteko. Gainera, ondorio batzuk atera dira emaitza horiek eta egindako lana oinarri harturik.

Gako-hitzak*: txirrindularitza, txirrindula, ikasketa automatikoa, aurreikusi, iragarpen.*

Castellano

Síntesis

El objetivo de este proyecto es predecir los resultados de carreras de ciclismo de ruta mediante técnicas de aprendizaje automático. Este proceso incluye la construcción de un conjunto de datos que se realiza, en primer lugar, revisando la literatura existente para determinar los factores clave en el rendimiento de un ciclista, seguido de la consulta de diversas fuentes de datos para, finalmente, construir *scripts* de *web scraping* con el fin de recopilar los datos.

Con el conjunto de datos construido, se diseña una configuración experimental en el que se consideran diferentes configuraciones del conjunto de datos y de las variables, junto con varios modelos de regresión de aprendizaje automático y los correspondientes hiperparámetros.

Finalmente, se evalúan los resultados en términos de métricas numéricas para determinar las aproximaciones más precisas. Adicionalmente, se extraen algunas conclusiones basadas en estos resultados y en el trabajo realizado.

Palabras clave*: ciclismo, bicicleta, aprendizaje automático, predicción, resultado.*

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Uncertainty and unpredictability are inherent characteristics of sport, and these aspects constitute the primary reasons for its widespread appeal. The consumption of professional sport has consistently been one of the main sources of entertainment in modern societies. The popularity of sports varies depending on the region, with some being more popular than others.

A key factor contributing to the success of a sport is its accessibility and universality. Sports that do not demand extensive infrastructure or complex equipment, such as football or certain athletics events, have gained popularity even in less developed countries [1]. These sports have a distinct advantage in reaching popularity. On the contrary, certain disciplines face challenges due to their resource-intensive nature. For example, skiing not only requires complex equipment but also specific climatic and geographical conditions that make it less accessible.

In an intermediate position, we find cycling, which, despite necessitating more sophisticated tools and accessories compared to a ball or a pair of trainers, does not require special infrastructure or a specific environment for its practice. All geographical conditions are suitable for cycling, as long as there is a road to ride on; high mountains as well as the most extensive plains are just as suitable. In this context, cycling has enjoyed great success, although it has mainly spread to more developed countries [2], it is a sport that is widely followed worldwide. Focusing on cycling, the *Tour de France* stands out as the main cycling event [3], the most recognised and demanding race where elite cyclists compete for the prestigious prize. Considering this event as the paramount representation of cycling, the *Tour de France* is one of the most followed sporting events globally each year. It competes for attention with other events that, *a priori*, might seem more successful, such as the NBA Finals in basketball or the Olympic Games [4], which encompass a greater amount of sports.

Based on this success, as can be seen in Figure 1, and keeping in mind the inherent unpredictability of results, it holds true that not all cyclists enter a race with equal chances of winning. Certain riders usually offer superior performances, influenced by numerous

Figure 1: *Fans attend the start of the second stage of the 2023 Tour de France in Vitoria-Gasteiz. Image by Pauline Ballet for A.S.O.*

factors such as physical conditions and capabilities, the route of the race or teammates. Some of them are unpredictable or challenging to discern, for instance the weather. One way to use these factors in order to predict the outcome of a cycling race is by using Machine Learning. In the purpose of using Machine Learning for this task, this project poses a main objective followed by other more detailed objectives.

1.2 Main objective

- To estimate cycling race outcomes as accurately as possible through the application of Machine Learning techniques.

1.3 Particular objectives

- To find reliable and well-structured data sources that constitute a considerable volume.
- To export the desired information from various sources by using web scraping techniques.
- To process the extracted information in order to structure it in a way that allows to work with the data easily, facilitating its manipulation and retrieval.

- To determine, by reviewing the existing literature, which factors are more decisive in the performance of a cyclist in a race.
- To design an experimental framework where various specific configurations of Machine Learning models are combined with certain data representing previously specified factors.
- To assess whether the models accurately predict the outcome of a cycling race, employing both individual and collective metrics.

Subsequent sections detail each step of the process to achieve the stated objectives as well as the mentioned techniques and metrics. Furthermore, the challenges encountered during these procedures and their corresponding proposed solutions are also detailed.

2 Preliminaries

2.1 About cycling

This subsection goes through the history of cycling, detailing its most special characteristics and finally making the most important considerations to be known about this sport. This very last part is written in order to better understand the vision of this project and its scope.

2.1.1 History

In the second decade of the nineteenth century, the first two-wheeled machines that used the power of the passenger to move began to appear. After the first experiments, in the 1960s the chain was introduced as an innovative element to transmit the pedalling force. It was then that the forerunners of modern bicycles were born and became more popular despite of their high price [5].

In addition to its more practical use as a means of transport, the use of bicycles as a sporting and leisure activity became increasingly important. In the 1860s, the first official races began to be organised by the written press, mostly in velodromes, and in the 1890s the first races were held in public streets. This phenomenon developed mainly in Europe and they were newspapers who organised the races [6].

In 1891, Pierre Griffard, editor of *"Le Petit Journal"*, organised for the first time the Paris-Brest-Paris, a race of about twelve hundred kilometres, which would take its champion, Charles Terront, more than seventy hours to complete [7]. Of the more than four hundred cyclists who started the race, including professionals and amateurs, only one hundred managed to complete this long race. The press covered these events as great feats that were attentively followed by the public. The uncertainty of knowing the outcome of the race encouraged the sale of the newspapers that covered it [8].

Following this trend, in 1903 the sports newspaper L'Auto organised the first edition of the *Tour de France* in order to increase its sales (see Figure 2). The success of this event, which is celebrating its 111^{th} edition in July 2024, goes without saying.


L'Auto

AUTOMOBILE — CYCLISME

LE TOUR DE FRANCE — LE DÉPART

Organisé par L'AUTO du 1er au 19 Juillet 1903

L'ITINÉRAIRE DU TOUR DE FRANCE

Figure 2: *Front page of L'Auto on 1st July 1903, the day of the start of the Tour de France's first edition.*

Things remained similar until after the 1950s. From the second half of the century, a real interest in making road cycling even more professional began to emerge. Fans demanded more coverage and showed great interest in the sport. As a reaction, the media put more effort into covering these events, and sponsors also wanted to show their brands to the general public. The first live broadcast of a cycling race was the ascent of the legendary Col d'Aubisque in the 1958 *Tour de France* [9]. This is another example of how the *Tour de France* has been for years a mass phenomenon and that is why it has remained as the highest exponent of professional road cycling. In this way and in this context, the professional environment that we know today began to develop.

Going back in time, in 1900 the *Union Cycliste Internationale* (UCI) was created in Paris (France) by the Belgian, French, Italian, Swiss and United States National Federations. This body was created to manage and promote cycling as a professional sport. In 1965, it took an important step by creating the International Amateur Cycling Federation (FIAC) and the International Professional Cycling Federation (FICP), the UCI would be in charge of managing both bodies. Today, it is the governing body for cycling, with its status recognised by the International Olympic Committee (IOC) and the 203 state federations associated with it [10].

UCI UNION CYCLISTE INTERNATIONALE			
Road	Track	Mountain Bike	BMX Freestyle
Indoor Cycling	Para-cycling	Gravel	BMX Racing
Trials	Cyclo-cross	Cycling Esport	

Figure 3: *Representation of the UCI disciplines.*

2.1.2 Differential characteristics

As every sport, cycling has characteristics that differentiate it from the rest. The main element that makes the difference with respect to many other sports and that constitutes the main element of cycling is the bicycle. This tool is the most important in the practice of this sport, without forgetting, of course, the rider. It is necessary that it functions correctly throughout the entire course of the race, as a failure could make the difference between raising your arms at the finish line and suffering defeat. In this sense, technological advances within the sport have always been focused on improving the bike, focusing mainly on its reliability, lightness, design and comfort, always with speed gain in mind [11]. The same applies to other elements such as helmets, clothing and footwear. In short, these are the protagonists of any event in this discipline: we have the person, and the tool that makes the person a cyclist, the bicycle.

Cycling covers different disciplines, in Figure 3 we can see those recognised by the UCI, which vary depending mainly on the terrain but also on the type of bicycle, but the latter is really conditioned by the terrain in which it is practised and the nature of the discipline. This project focuses on road cycling.

However, the cyclist does not compete individually against other cyclists, but has a team to accompany him/her. In this sense, it can be confusing for the beginner who is just starting to follow this sport, because despite the existence of teamwork and the importance of it, on most occasions races end up being a battle between a few cyclists, and it is a single rider who takes home the title and not the team as a whole. This idea is developed further in the following section, as this last characteristic is where the figure of the team leader is born.

2.1.3 Important considerations for the project

This section aims to summarise the most important elements to be considered about cycling as well as to give some definitions in order to provide more clarity about the project.

- Within professional cycling there are different categories. There are categories both for races and teams. For this project we focus on the highest category in both cases. For the races, we take those that belong to the UCI World Tour (UWT). For the teams, those belonging to UCI WorldTeams. In the UWT races, some teams from the UCI ProTeams category can participate as guests, so data from riders belonging to these teams are also in the dataset.

- This project focuses on professional male road cycling. This decision is due to the volume of available data regarding the female counterpart. Unfortunately, there are not as many women's races, and the races are not organised in the same way. For example, the women's *Tour de France* consists of 8 stages compared to the 21 stages in the men's race.

- Road cycling races can either be one-day events where the first to finish wins, or multiple-day races (composed by several stages). In the latter, the main attraction is winning the overall classification, but winning individual stages as well as other complementary classifications (such as points classification, team classification, or mountain leader) is also significant.

- Since not all teams have equal opportunities to win the overall classification, and depending on how well they fare in pursuing this goal, some teams may choose to target stage victories or win complementary classifications.

- The stages can be seen as one-day races except that the stages, unlike one-day races, can be individual time trials (ITT) or team time trials (TTT).

- Despite the existence of teams, except in certain events such as team time trials (TTT), professional races have a single winner, so each team usually has one or more leaders for whom the team rides. This figure is very obvious in some teams and more diffuse in others, since the leader may vary depending on the type of race or objective.

- The types of routes in a race can vary, with some being flatter, others featuring

more elevation and mountainous climbs, and others being longer (typically those with less elevation).

- One-day races usually combine long routes with a multitude of ascents, not very prolonged or demanding, but avoiding extensive flat sections.
- There can be several complementary classifications for multiple-day races:
    - *Stage classification* represents the finishing order and times of the cyclists for that day. If a rider does not finish a stage, he or she must abandon the race.
    - *General classification* is the classification resulting from adding the times accumulated by each rider in each stage.
    - *Youth classification* is a reduced version of the general classification but only applied to young riders. The age range varies, being for riders under 25 in the *Tour de France* [12].
    - *Team classification* is calculated by adding the times of the best n riders of a team on each stage. The number of riders to be considered may vary and is currently 3 for the *Tour de France* [12].
    - *Mountain classification* represents the performing on mountain. When there is a mountain pass in a stage, points are awarded to the first ones to cross this pass, in addition, bonus seconds can also be awarded which are deducted in the general classification.
    - *Points classification* works similarly to the mountain classification. In certain sectors of the race, points are awarded to the first to arrive, it is also common to award these points to the first to cross the finish line (stage top riders). They are usually accompanied by time bonuses.

2.2 Machine Learning

This section delves into Machine Learning, focusing specifically on the employed techniques as well as the metrics that allow us to measure the performance of our work.

2.2.1 Programming

For the construction and subsequent evaluation of the Machine Learning techniques and models that are implemented in this project, Python is used as programming language. Some of the most popular libraries for working with Machine Learning in Python are as well used, such as: *NumPy*, *pandas* or *scikit-learn*. Additionally, other libraries are used for some specific cases but are omitted as they are not part of the core implementation.

NumPy It is an open-source and high level syntax Python library for numerical computing that provides support for arrays, matrices, and mathematical functions. It offers well-optimised data structures and mathematical functions for handling large arrays and matrices [13]. NumPy is essential for scientific computing and data analysis tasks due to its speed and versatility.

Pandas It is an open-source Python library for data manipulation and analysis, featuring a fast DataFrame object. It offers tools for reading/writing data from various formats, intelligent handling of missing data, flexible reshaping, and slicing. It supports column operations, aggregation, and transformation, along with high-performance merging and joining [14].

Scikit-learn It is an open-source Machine Learning library for Python. It is accessible to everyone and can be reused in various contexts. Built on *NumPy*, *SciPy*, and *matplotlib*, it provides a wide variety of machine learning algorithms for classification, regression, clustering, and dimensionality reduction [15].

2.2.2 Models

As we are facing a regression problem in which we aim to determine the time in which a cyclist completes a certain race, the following section presents the different Machine Learning regression models that we are going to use to achieve this objective.

For clarity we define a generic way to refer to our dataset as well as the entries and features that compose it. Let us suppose we have a dataset with m different values $X = x_1, x_2, \ldots, x_m$. For each x_i we have that it is composed of n different features $x_i = x_{i1}, x_{i2}, \ldots, x_{in}$ (they may also be refereed as attributes or variables).

Following this notation, we now show all the steps that have to be made for the application of each of the models.

Linear Regression Regression in the field of machine learning refers to the study of dependence. In this regard, Linear Regression is nothing more than an instance of this methodology [16]. In the application of Linear Regression, we are interested in formulating a hypothesis of dependence between a target variable and the remaining values. This is done with the aim of being able to explain the target variable in terms of the rest [17].

Taking our dataset X, we could consider that there exists a relationship between some variables, specifically a linear relationship. We could define that a variable denoted by y is obtained by a linear combination of the rest of the values:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n.$$

We would need to collect data on these variables. Then, using techniques such as Ordinary Least Squares (OLS) [17], we can estimate the coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ that best fit the data.

This method is used to find the coefficients of a Linear Regression model that minimises the sum of squared differences between real and predicted values. We have the sum of squared differences that can also be denoted as the Residual Sum of Squares:

$$\text{RSS} = \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{n} x_{ij}\beta_j \right)^2$$

and we aim to minimise the RSS function

$$\hat{\beta}_{\text{OLS}} = \underset{\beta_0, \beta_1, \ldots, \beta_n}{\arg\min} \{RSS\}$$

where:

- y_i is the real value of the dependent variable for the i^{th} example.
- x_{ij} is the value of the j^{th} independent variable for the i^{th} example.
- $\hat{\beta}$ is the set of the coefficients $\beta_0, \beta_1, \ldots, \beta_n$ to be estimated.
- N is the total number of data points.

This way, OLS estimates the coefficients that define the linear relationship between variables in a regression model. Additionally, regularisation can be introduced to avoid overfitting the model to the training data. Regularisation solves this by penalising some of the coefficients. We present two regularisation strategies which are Ridge:

$$\hat{\beta}_{\text{ridge}} = \underset{\beta_0,\beta_1,\ldots,\beta_n}{\arg\min} \left\{ RSS + \alpha \sum_{j=1}^{n} \beta_j^2 \right\} \tag{1}$$

and Lasso:

$$\hat{\beta}_{\text{lasso}} = \underset{\beta_0,\beta_1,\ldots,\beta_n}{\arg\min} \left\{ RSS + \alpha \sum_{j=1}^{n} |\beta_j| \right\} \tag{2}$$

where:

- α is the regularisation strength.

However, when using Linear Regression to build a model, some assumptions are done [18]:

- The existence of a continuous dependent variable and independent variables that are either quantitative or categorical.
- The existence of a linear relationship between dependent and independent variables.
- The existence of homoscedastic residuals, this means that the difference between the predicted and the real values remains constant.
- The existence of normally distributed residuals, this means that most residuals are around zero and fewer residuals further away from zero.
- The absence of multicollinearity. Multicollinearity occurs when some of the dependent variables have strong correlation.
- The absence of external variables with strong relationships with the dependent variable. We assume there are no omitted variables with strong associations with the outcome.
- The existence of independent observations. One observation is not related or influenced by the another observation.
- The existence of independent errors. Similarly to the previous assumption, the error of one observation is not related or influenced by the error of another observation.

Note that the violation of most assumptions does not mean Linear Regression cannot be used, but it must be known to assess the performance.

K-Nearest Neighbours (KNN) It is a non-parametric Machine Learning method used for classification and regression tasks [19]. Let us suppose we have some element x on which we want to do a prediction. For this given x, we want to classify it somewhere or predict a target value. For this purpose, we calculate the k elements with the greatest similarity (nearest ones) and, based on these, we decide the label or value to assign to our x.

This method is based on the idea that closest data points in space are likely to share similar target values. This principle is applied to predict the target value of our new data point x based on the target values of its k-nearest neighbours.

Based on this intuition, there are multiple ways of performing this task. In any case and in general, taking our dataset X, we chose a function that defines the distance between two points, for instance we could take the Euclidean distance:

$$d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

where:

- d_{ij} is the distance between points x_i and x_j.
- n is the number of features for each data point.

Once we compute this distance for every pair of points x_i and x_j where $x \neq j$, for a new data point x the k nearest neighbours are identified based on the chosen distance metric. In regression problems, the predicted value for the new data point is usually calculated averaging the target variable values from these neighbours. In classification problems, the most frequent class among these neighbours is assigned as the predicted class.

Notice that a very important hyperparameter in this method is the number of neighbours, denoted by k. This value determines how many of the closest data points are considered for prediction.

Random Forest First introduced by Ho in [20] and later extended by Breiman in [21], *Random Forest* refers to the ensemble by which we can develop both regression and

classification tasks mainly based on a set of decision trees. Figure 4 shows a commonly used graphical representation for Random Forests.

The method begins by generating M samples from the original dataset. This sampling consists in the replacement of randomly selected data points from the original dataset.

For each $m = 1, \ldots, M$:

1. A new sample X_i is generated with replacement. This sampling process is known as bootstrap.
2. Taking X_i a decision tree T_i is trained.
3. At each node of the tree, a random subset of the attributes is selected to determine the best split. This additional step introduces diversity helping to reduce the correlation between the individual trees in the forest.

Once all the trees are built, predictions are made for an unseen data point x. In regression tasks, the predictions from each tree are averaged to obtain the final prediction as follows:

$$\hat{y}_i = \frac{1}{M} \sum_{j=1}^{M} T_j(x_i)$$

where:

- M is the total number of trees.
- $T_j(x_i)$ is the prediction of the j^{th} tree for sample x_i.
- y_i is the predicted value for sample x_i.

In classification tasks, each tree's prediction is considered as a vote, and the class with the most votes across all trees is chosen as the final prediction:

$$\hat{y}_i = \text{vote}(C_j)$$

where:

- C_j represents the class predicted by the j^{th} tree for sample x_i.
- $\text{vote}(C_j)$ returns the most frequent class in C_j.
- y_i is the predicted value for sample x_i.

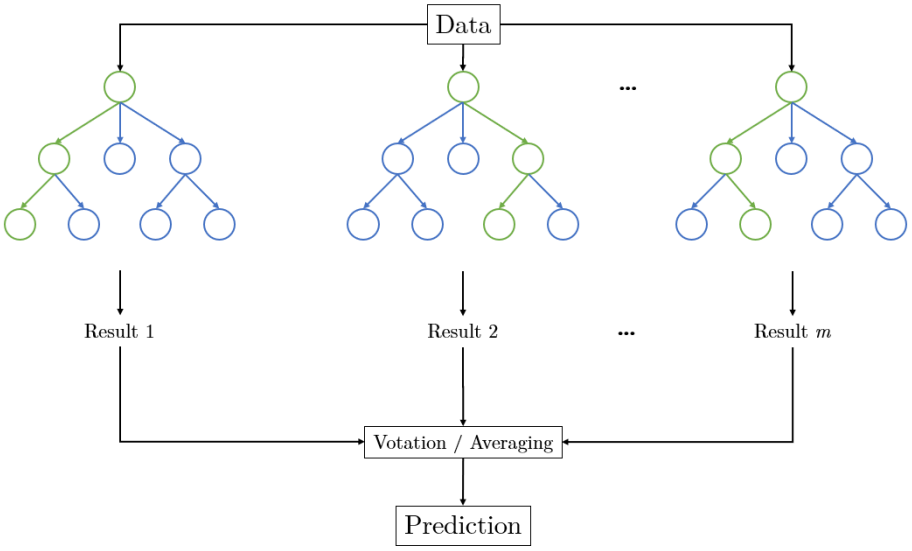


Figure 4: *Graphical representation of Random Forest algorithm.*

Neural Networks Neural Networks are a family of algorithms inspired by the biological structure of the human brain. They are formed by a collection of interconnected units called artificial neurons (analogous to biological neurons). These artificial neurons work together to process information and create a model capable of learning from data [22].

These neurons are grouped in different layers, in the first of the layers we find our inputs and in the last of all our output, the prediction. The intermediate layers are known as hidden layers.

As shown in Figure 5, each neuron receives as many inputs as there are neurons in the previous layer and outputs as many values as there are neurons in the next layer. For the input layer, we have that it is composed by the m features.

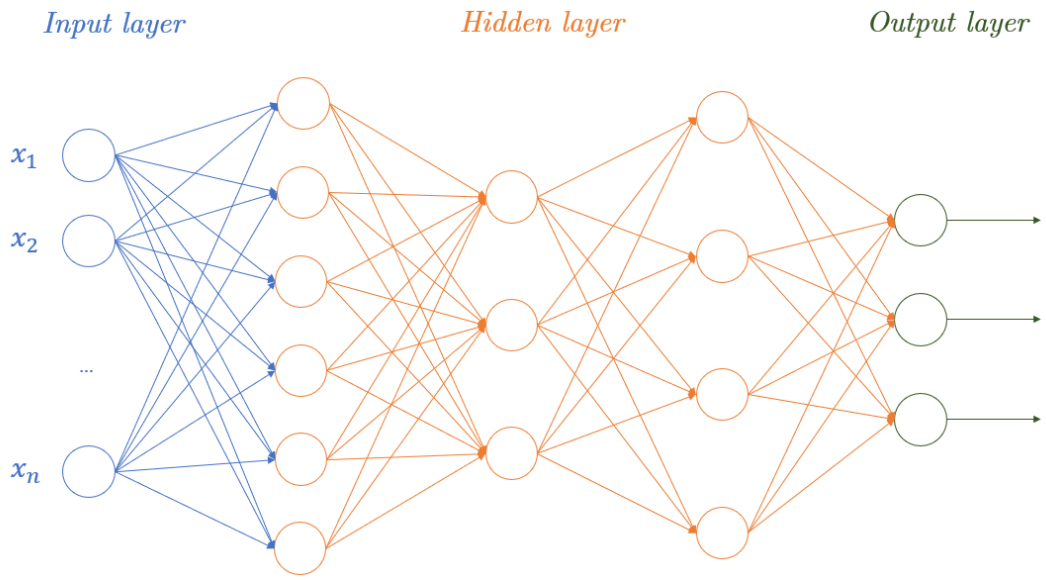


Figure 5: *Graphical representation of the behaviour of a Neural Network.*

In general, for the construction of a Neural Network, the following steps are followed:

1. Initially the network has no knowledge since it is assigned arbitrary values. Therefore it produces "random" responses.

2. The first step is the forward propagation through all the Neural Network.

   (a) Each neuron applies a transformation to its input values to produce the output values. Specifically, for each neuron j in layer l the output is computed as shown in Equation (3).

$$a_j^l = activation_l(\sum_i w_{ij}^l a_i^{l-1}) \tag{3}$$

   where:

   - $activation_l$ is a nonlinear function that introduces non-linearity.
   - w_{ij}^l is the weight connecting neuron i in layer $l-1$ to neuron j in layer l.
   - a_i^{l-1} is the output of neuron i in the previous layer.

   In the case of the activation function there are many of them that can be selected. Among the most common, we find the Sigmoid Function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

the Hyperbolic Tangent Function (tanh):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

or the Rectified Linear Unit Function (ReLU):

$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

It goes without saying that the choice of the activation function for each layer is fundamental to the performance of the Neural Network.

(b) Once the data has passed through the network, it produces an output. The output value is compared with the actual value by means of a loss function, and the weights of each neuron are adjusted accordingly through a process called backpropagation.

i. We define a loss function $L(\hat{y}, y)$ that measures the difference between the predicted and actual values.

ii. This previous difference is then propagated backward through the network layer by layer. This involves calculating the partial derivative of the loss function with respect to each weight, w_{ij}^l, in the network using the chain rule.

iii. The previously calculated values are used to update the weights in a direction that minimises the overall loss. For a general weight w:

$$w' \leftarrow w - \eta \frac{\partial L}{\partial w}$$

where:

- w' is the updated weight.
- w is the previous weight.
- η is the learning rate, which controls the step size of the updates.
- $\frac{\partial L}{\partial w}$ is the derivative of the loss function L with respect weight w.

3. Once the Neural Network is trained, new data can be passed in order to make

predictions.

XGBoost Extreme Gradient Boosting (XGBoost) is an optimised algorithm applied to the Gradient Boosting method [23]. Similar to Random Forests, this is an ensemble that employs decision trees but the Gradient Boosting algorithm boosts each decision tree by trying to minimise the error of its predecessor.

The algorithm of XGBoost begins with an initial prediction, often set as the average of the target variable in the training data. Let us denote this initial prediction as $\hat{y}^{(0)}$. After doing this, we build T different regression decision trees as follows:

For each iteration $i \in \{1, \ldots, T\}$:

1. The difference between the actual target values and the current prediction is calculated, resulting in the residuals:

$$r_i^{(t)} = y_i - \hat{y}_i^{(t-1)}$$

where:

- y_i is the actual target value.
- $\hat{y}_i^{(t-1)}$ is the current prediction.
- $r_i^{(t)}$ is the residual for the i^{th} value in the t^{th} iteration.

2. A new regression decision tree $f_t(x)$ is constructed using these residuals ($r_i^{(t)}$) as the target variable. This tree aims to minimise the errors made by the previous trees.

3. After this tree is constructed, its predictions are incorporated into the final prediction by the weighted sum:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta \cdot f_t(x_i)$$

where:

- $\hat{y}_i^{(t-1)}$ is the previous prediction.
- η is a weight known as the learning rate. It is an hyperparameter to be determined.
- $f_t(x_i)$ is a decision tree.

- $\hat{y}_i^{(t)}$ is the current updated prediction.

This process continues iteratively until a stopping criterion is met, such as reaching a maximum number of trees (T) or achieving a desired level of prediction accuracy.

Once trained, the XGBoost model can predict the target variable for new data points by passing them through the ensemble of decision trees and combining the individual predictions using the learned weights.

2.2.3 Hyperparameter Tuning

All the Machine Learning techniques we are introducing in this work require from some hyperparameters to be tuned. These hyperparameters directly affect the performance of each model. Among all the possible values an hyperparameter can take we have to choose one. For this purpose, cross-validation can be used as presented below.

Cross-Validation A crucial step in Machine Learning is evaluating how good a model is when it comes to *generalising*, meaning how well it performs on unseen data. A common approach involves splitting the data into training and testing sets. However, the model's performance can be sensitive to this specific split.

Cross-validation is presented as a solution to this by repeatedly splitting the data into multiple (k) folds and evaluating the model on each fold. First of all, we need to split all our data into k groups (folds). After doing this, we iterate through each fold as follows:

For each $i \in \{1, \ldots, k\}$:

1. Fold i is designated as the test set $X_{\text{test},i}$.
2. The remaining $k-1$ folds are combined into the training set $X_{\text{train},i}$ (notice that each pair of training and testing sets is disjoint, meaning they share no common cases ($X_{\text{test},i} \cap X_{\text{train},i} = \emptyset$) [24]).
3. The model is trained on the training set and then evaluated on the testing set (the $k-1$ folds that are not used in training set).

After running the models with the k folds (training and testing different partitions), we take the metrics that define the performance of each model and we average them to estimate how well our model generalises to new data. Looking at the process by which

this method is defined, we have that k is a parameter we need to define and that strongly affects the performance.

Tuning Hyperparameters with Cross-Validation We first define a set of values for each hyperparameter we want to tune. We want to determine which one is the best value for our model among the values we have defined for a particular hyperparameter. Within the cross-validation loop (described above):

1. For each combination of hyperparameters:
   (a) Train the model on the training set using those specific hyperparameters.
   (b) Evaluate the model's performance on the test set.
2. The set of hyperparameters that gives the best average performance across all folds is chosen as the optimal configuration.

Applying cross-validation offers a way of determining the best values for some hyperparameters in a way so we maximise our model's performance.

2.2.4 Error measurement metrics

The following metrics are a measure of error in the predictions of our target variable, which provides us information about how far the predictions are with respect to the actual values.

Mean Squared Error (MSE) Measures the average of the squares of the errors. It quantifies the average of the square difference between the predicted and actual values. It is given by Equation (4).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

where:

- n represents the number of observations.
- y_i represents the actual value of the target variable for the i^{th} sample.
- $\hat{y}_i$ represents the predicted value of the target variable for the i^{th} sample.

Root Mean Squared Error (RMSE) It is the square root of the mean of the squared differences between actual and predicted values. This is basically the squared root of the MSE (Equation (4)) and it is given by Equation (5).

Root Mean Squared Error:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5}$$

where:

- n represents the number of samples in the dataset.
- y_i represents the actual value of the target variable for the i^{th} sample.
- $\hat{y}_i$ represents the predicted value of the target variable for the i^{th} sample.

2.2.5 Ranking metrics

In addition to error measurement metrics, which, as explained in that section, measure how far quantitatively the predictions are from the actual values, we introduce ranking metrics. These metrics measure how well the models estimate the order of the classification it is trying to predict. We could have a case where our model is overestimating or underestimating the target variable (for instance it is giving very large finishing times to all cyclists) but is properly ordering the cyclists in order of finish (it is properly estimating the final classification). In this particular case we would have high values in our error measurement metrics but good ones in ranking metrics.

Some of these parameters include the idea of relevance. Relevance refers to how important a cyclist is for the prediction. In our case, a rider who has actually placed high in the classifications is more relevant for prediction than a rider in lower positions. That is, for some of the metrics, predicting accurately riders from the top of the classification has a higher weight. In some cases, only the accuracy of the first k elements is considered.

For the sake of clarity, let us define two sets $C = c_1, c_2, \ldots, c_m$ and $\hat{C} = \hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m$ where C represents the actual classification order and $\hat{C}$ represents the order assigned by the predictions.

Kendall Tau Distance Introduced by Kendall in [25], this metric only calculates the number of inversions needed to convert C to $\hat{C}$. The higher the number of inversions the higher the value of the metric indicating a lower prediction accuracy. The normalised Kendall Tau Distance taking values between -1 and 1 is given by:

$$\tau = \frac{2 \cdot (n_c - n_d)}{n(n-1)}$$

where:

- τ represents the Kendall Tau Distance.
- n_c is the number of concordant pairs.
- n_d is the number of discordant pairs.
- n is the total number of pairs.

Suppose we have two predictions. In the first one we have predicted correctly a higher number of positions, but we have failed to predict the podium (first three positions). With this metric, this first ranking would get a better value than one in which we have predicted correctly the podium but failed in lower positions of the table.

This is commonly presented as a limitation, however, it offers relevant information since we can consider that it is just as difficult to predict that a mid-ranked cyclist is ranked in that exact position as it is to predict the winner. Therefore, it depends more on the interpretation that is made, if we are interested in predicting high positions the metrics below are more useful. In any case better prediction of high positions does not necessarily indicate that the model is better, it can be better for this specific purpose but not in general [26].

Note that we can set a k value that limits the metric to the first k positions of the ranking without considering the rest.

Discounted Cumulative Gain (DCG) The first step in calculating this metric is to define a *gain* function. Usually, for an element i with a relevance R_i the gain is defined as:

$$gain_i = 2^{R_i} - 1$$

To define the relevance R_i for the i^{th} cyclist we can take the position p_i in which the cyclist arrives. However, as this value is lower for better (more relevant) cyclists, we redefine it

as:

$$R_i = \frac{1}{p_i}$$

Additionally, to compensate for the gain, a discount function is defined that penalises each element proportionally, discounting more the highest elements in ranking. For element in position i:

$$discount_i = log_2(i + 1)$$

Therefore, our metric considering the first k elements of the ranking is finally defined as:

$$DCG@k = \sum_{i=1}^{k} \frac{gain_i}{discount_i}$$

Additionally, this value can be normalised by dividing it by the ideal DCG (value for a perfect ranking) to obtain the Normalised Discounted Cumulative Gain (nDCG) which takes values between 0 and 1:

$$nDCG = \frac{DCG@k}{DCG_{ideal}@k}$$

Note that k can take different values, for each of them, the meaning of the metric changes. For example, if it is three, we are focusing on evaluating the accuracy of the podium, but if we define it as the total number of riders, we are considering the entire classification, although the highest positions are still rewarded in the metric [26].

Both Kendall Tau Distance and nDCG can take a maximum value of one which is the ideal prediction. Therefore, unlike the error metrics where a lower value represents a better prediction, for these ranking metrics a higher value represents a better prediction.

3 Dataset generation

To implement and perform even the smallest Machine Learning model, it is essential to have a properly designed and structured dataset [27]. In the case of real Machine Learning applications, this component takes a fundamental role, since the results of our work strongly depends on the quality of the dataset for training and subsequent evaluation.

This section walks through the process of dataset generation. The dataset is built from nothing by collecting unstructured data from various sources. In addition to detailing the whole process, the challenges encountered and the solutions proposed to solve them are also indicated.

3.1 Sources

Cycling, like any successful sport, enjoys wide media coverage. Consequently, it is not difficult to have access to the winners of most of the races. On the other hand, although it is becoming more and more common, the most specific data about races is not always exposed in media. In this sense, the UCI, as the managing body, records more regularly all kinds of statistics. However, it does not offer an open, convenient and accessible way to consult this information.

3.1.1 *ProCyclingStats*

Fortunately, there is a website [28] that has done the work of compiling these data and offers it openly. *ProCyclingStats* offers through its website an intuitive and pleasant way to access the record of classifications and other information such as teams, cyclists, race withdrawals or cancelled races on a large number of events.

The site has a search engine and different main tabs to navigate mainly through cyclists, teams and races. By accessing a race you can consult the different editions available (they vary depending on the event), as well as more general information about the event. Once you click on the year, all the available information appears, which also varies depending on the edition. The main problem is that there is no way to export this data, so it is

necessary to implement web scraping techniques to develop this task. This means that acquiring the dataset is a significant hurdle in achieving the objectives of this project. In the context of a Machine Learning application for cycling, data collection proves to be a substantial challenge.

At user level, the web offers an experience mainly based on navigation through links, in this sense, if it is wanted to access a race, the most intuitive way would be to use the search engine, select the race and then search for the desired edition. This is not a very optimal way to work with web scraping. Fortunately, the page follows a very well organised link structure so that the race edition, the type of classification or the stage appear explicitly in the link. This structure is respected in the majority of occasions and makes the whole process easier.

One of the main drawbacks of this website is the lack of data representing the route of the race. Although there is some concrete data such as the length of the race, this is not enough. While it is true that there is an image with the profile of the race, this does not offer a way to extract the information directly through web scraping techniques.

To solve this, we look for sources where this information is available, finding the solution on another website: *LaFlammeRouge*.

3.1.2 *La Flamme Rouge*

This website [29] provides downloadable GPX files representing the route of cycling races.

GPX is an open source map file format that is defined in its documentation [30] as *"a light-weight XML data format for the interchange of GPS data (way-points, routes, and tracks) between applications and Web services on the Internet"*. It is a widely used format and it is relatively easy to extract information from these files since it is an XML-based format. This makes it easily understandable by both software and humans making it easy to process.

The volume of available route data is much smaller than other types of data such as classifications from the previous web. For this reason, the bottleneck arises in this aspect, limiting the volume of usable data in the application of the designed models.

This website functions primarily as a forum where cycling routes can be shared and edited. The website is not as intuitive as *ProCyclingStats* and is more confusing to navigate at user level. For the extraction of routes it works in a similar way, having to first access the desired race and then choose the edition. In the case of multiple-day races, the stages appear after the edition. The option to download the GPX file and its visualisation on the map appear in the last place.

3.2 Web scraping

As indicated above in section 3.1.1, the first page does not contain a direct way to export the data it contains. The second page, on the other hand, has a direct download button for the GPX file, but reaching the page containing this button is somewhat laborious. Collecting data from either of the two sources is not trivial, so it is decided to implement web scraping techniques for data extraction (represented in Figure 6). In this sense, it is decided to use Python as a programming language together with some libraries that allow this task to be carried out. These libraries are BeautifulSoap and Selenium. Complementary libraries are used, but they do not respond to the process of accessing the content of the web pages *per se*, so their mention is omitted.

In this process we try to extract as many variables as possible, even if some of them are not used, it is a preliminary process and we try to ensure that the available data is not a limitation.

As explained above in section 2.1.3, the extraction is limited to top level races. Within UWT, there are 1.UWT races which refer to one-day races and 2.UWT referring to multiple-day races. However, this type of classification was established in 2014 with the previous system being more confusing. Therefore, it is decided to take races of the highest category between 2014 and 2023 (10 years). It is also rare to find GPX routes for earlier races. It is worth mentioning that the category of races may vary along years and some may not be held anymore, so not all races have all these editions. However, we can ensure that for all those years we have all celebrated races belonging to the highest category.

Finally, as some races may have guest teams, the data is extracted for all the teams of the two highest categories and for all their riders. In the case of GPX routes, they are

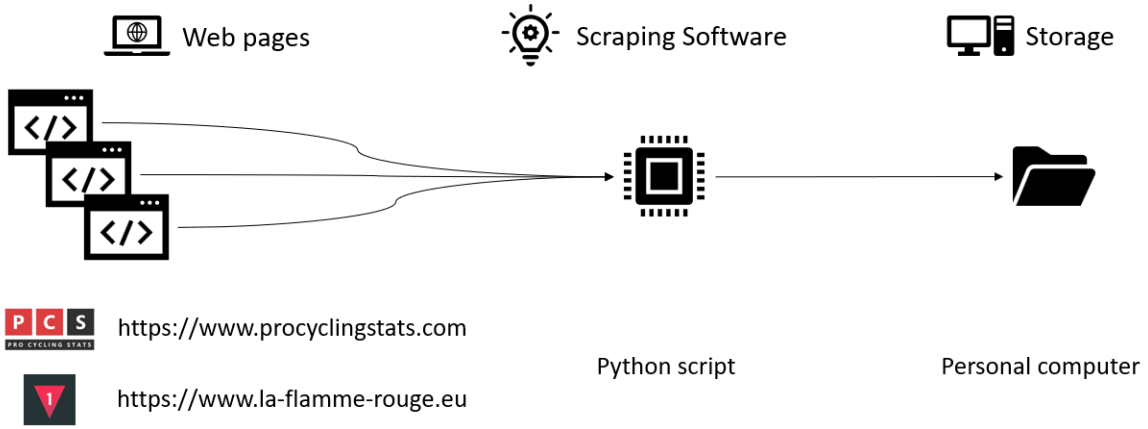


Figure 6: *Process of web scraping for the generation of the dataset.*

extracted for those races with available data. Remember that this is the bottleneck. The challenges encountered in this process are detailed below.

3.2.1 Challenges

Different procedure for one-day and multiple-day races One-day races have a different page format than multiple-day races. In one-day races, the rider information is in one table, while in multiple-day races, the information is in multiple tables. This makes the web scraping procedure different for each type of race.

Additional complexity due to lack of fixed structure As there is no fixed structure in the pages, web scraping becomes more complicated. Each year presents different structures, and it is necessary to consider multiple cases, such as the different types of classifications. The type of classifications available is analysed for each year, since, for example, depending on the race or edition there may or may not be classification by teams.

Team Stages The team stages follow a radically different structure to the other classifications. There is no single winner, as a team wins. Although the times of each team are applied to their individual riders, the classification on the website is shown as a team. Such stages are usually held on the first day of the race (so that the teams have all the riders) and are unusual. It is decided to ignore the stage classification, as times are represented by the general classification for that day. It is important to note that it could

be the case that, due to age, the general classification data for that day is not available; however, this case is ignored as it is very particular and has not been detected.

Race participants This information does not appear explicitly on the website, but can be inferred from the classification of the first stage. It is understood that riders who do not appear in the classification of the first stage have not started the race, so they have not participated in it. For one-day races, it is understood that riders who do not appear in the classification of the race have not participated. These assumptions can be made since the classifications also show riders who did not finish (DNF).

Doping During the web scraping process we realise that there are disqualifications that have been made *a posteriori* for doping positives. These data distort the rest as they do not correspond to a real performance of the athlete. After looking for sources in which to find these cases, we see that in *ProCyclingStats* the position number of the cyclist who has been disqualified for doping appears crossed out in the classification, for instance ~~5~~ instead of 5. The script is modified to take this case into account.

Different URL format depending on race For the final classifications, it is needed to navigate to the last stage and view the classifications for this stage. Sometimes these classifications come in the following format:

`https://www.procyclingstats.com/race/renewi-tour/2021/stage-7-points`

and sometimes they follow the format:

`https://www.procyclingstats.com/race/renewi-tour/2019/points`

For the general classification it always follows the second format, but for the rest of the classifications it varies. This is solved by checking if the final part of the URL starts with the name of the stage or not.

Incorrect URL for some classifications Related to the previous problem, races following the second format show a problem with the team classification. In this case, the team classification is displayed in the following URL:

`https://www.procyclingstats.com/race/renewi-tour/2019/teams`

If we access this link directly, it shows a list of the participating teams, but does not show the team classification. However, it is the link to which the website redirects if we click on

the teams tab in the last stage. This web error is ignored as it only happens occasionally so we sacrifice data volume as some races do not have this information.

Downloading GPX files The desired download folder to keep them organised is not correctly set by the program. It is decided to download them all to the browser's default download folder. Once downloaded, they are moved to their corresponding destination folder together with the rest of the data for that race.

Changes in web's usage policy Despite not doing so at first, the website from which GPX files are extracted has changed its usage policy. Now it is more restrictive due to the use of bots for web scraping. This means that the scraping process is stopped and modified, as the data cannot be accessed without an account. This is easily solved by creating an account, but it is also needed to change the Python library to work more comfortably with login process.

IP address blocking The website from which GPX files are extracted blocks IP addresses that make requests that appear to be automated. This means that the scraping process is stopped on several occasions. To avoid this, random timeouts are set between requests, simulating more human behaviour, but slowing down the process.

3.3 Structure

The dataset follows a structured organisation inspired by *ProCyclingStats*. It is divided into two main folders: one for one-day races and another for multiple-day races. Within each category, there are subfolders for each race, named after the race itself. Inside these race folders, there are further subfolders named with the year of the event. For one-day races, the data for that particular race and year is directly accessible. For multiple-day races, additional subfolders represent each stage, containing the data specific to that stage. This hierarchical structure is depicted in Figure 7.

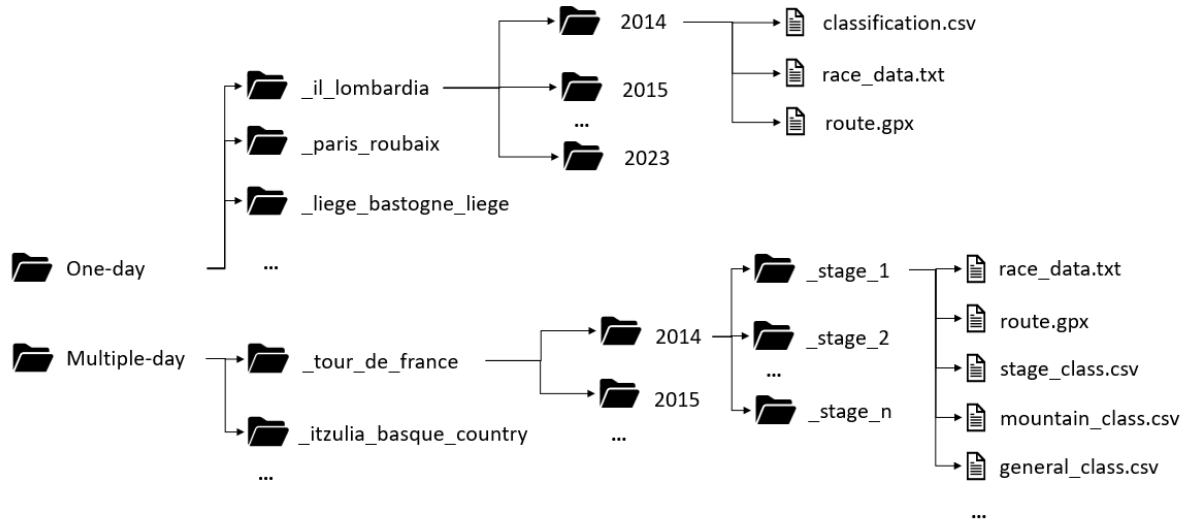


Figure 7: *Folder tree representing the dataset structure.*

In addition to this, there are two folders with information on teams and riders. These folders have a simpler structure. The files containing the information are directly accessible. In both the teams folder and the riders folder, there are as many files as years available. In these files appear the teams/riders that participated in that year and their corresponding information (remember that they are the ones that were in the first and second categories established by the UCI).

For teams and riders, data is stored in CSV files, with each line representing either a team or a rider. Teams' files include information such as team name, status, and URL, while riders' files contain details like name, birth date, nationality, and various performance metrics such as one-day races, GC, time trial (TT), sprint, and climber points. These metrics reflect the rider's career performance and should be interpreted with caution. Additionally, the files include UCI points indicating the rider's current standing, awarded by the UCI in each race. It's worth noting that these points represent the current position, not that of the previous year. In any case, as explained earlier, in this initial step, all available data is obtained regardless of whether it is useful later on.

Returning to the races, three types of data are established: classifications, route information, and general race data. Classifications, regardless of their type, are stored in CSV files where each line represents a cyclist, ordered from best to worst. Typically, cyclists are paired with their times. In the case of points classifications, total points are stored instead of time, but the concept remains the same. For route information, a GPX file is processed to extract specific usable data. Lastly, there is a TXT file containing details

such as start and end dates, the average speed of the winner, and the start and finish locations. It's important to note that the more recent the race, the more information is available, but not all races have the same data.

Finally, Table 1 and 2 represent the volume of data contained within the created dataset.

Event type	**Number of Races**	**Number of Classifications**	**Number of Routes**
One-day	169	169	165
Multiple-day	1459	9208	1370
Total	**1628**	**9377**	**1535**

Table 1: *Summary of race classifications.*

Type of Information	**Value**
Number of teams	546
Number of riders	3176

Table 2: *Summary of additional data.*

It is worth noting that teams in the cycling world can be quite diffuse as a concept. Although there is continuity in the sports management, changes of sponsors, administrative management and riders are very common from year to year. It is true that there are teams that remain stable over a some years. Additionally, it is important to know that each classification gives information about the performance of several cyclists.

4 Literature review for feature selection

This section provides a detailed work on the feature selection from the dataset by both reviewing the existing literature and providing general culture-based thoughts.

In the analysis of performance in any sporting discipline, age is always a determinant factor. It is obvious that the passage of time deteriorates people's physical abilities, but at the same time one can think that experience is a fundamental factor. In this sense, there are numerous studies on the analysis of sporting performance and age. For example, at the 2012 London Olympics, 72% of the participating athletes were aged between 20 and 30. 99% were under the age of 40 [31]. If we take this to the world of cycling, we find that the age at which peak performance is reached is between 26 and 28 depending on the category [32].

Another component that can affect a cyclist's performance in addition to age is the time of year in which they were born. Numerous studies [33]–[35] suggest that people born earlier in the year have a greater chance of success as elite athletes compared to those born later in the year. One of the sources consulted [33], in its title, suggests: *"When "where" is more important than "when": Birthplace and birthdate effects on the achievement of sporting expertise"*. Although this characteristic has been suggested to condition access to elite sport, there is no evidence that, once at the highest level, it conditions the athlete's performance. In any case, we decide to create a variable to measure this characteristic. To do so, we construct a numerical attribute that represents the days passed from the beginning of the year until the date of birth. In this way, we establish in which period of the year an athlete was born.

Inspired by the previous idea, it is intuitive to think that the time of the season in which a race takes place can have an influence on certain athletes. In this sense, studies such as [36] by Hopker, Coleman, and Passfield suggest the existence of changes in cyclists' performance during the season. In order to be able to quantify this moment and see if it influences performance, it is decided to calculate a numerical variable that represents the days elapsed from the beginning of the year until the date of the race.

Additionally, the moment of the season can also be important because of the weather. Weather can be considered to affect the performance in a race since both the weather and variations in the weather during a cycling race can have significant consequences on the

outcome of the event. It is difficult to determine whether this affects positively or negatively as it depends on both the weather variation and the cyclist. For the development of our project, it is difficult to find a considerable amount of climate-related data, as the sources consulted contain almost no information on this factor. Therefore, and taking into account that most of the top level events (the *UCI WorldTour* ones, which are the ones we are considering) take place in Europe ($82,86\%$, see Figure 8), the best we have to represent the climate is the numerical variable presented above indicating the moment of the year.

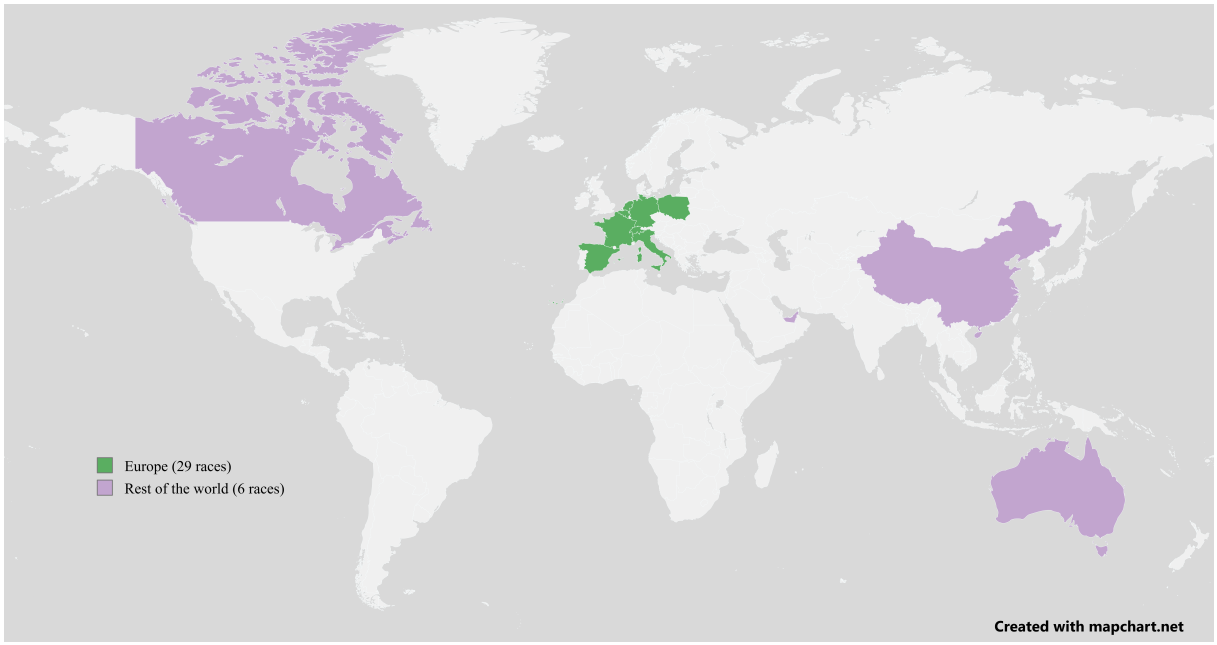


Figure 8: *Map of the countries hosting at least one UCI WorldTour race.*

Studies such as [37] suggest that some physical characteristics such as height or weight have an influence on a better performance depending on the type of the cycling event (in the case of the cited study, it refers to track cycling). Looking at the professional cycling scene, one can easily understand that the best *"climbers"*, the ones that better perform in the toughest mountains, have a common physical characteristic such as light weight or even short height. Similarly, for races where the finish line sprint is the deciding factor, cyclists with more power are more likely to win, normally having a higher weight because of their strong legs. Following this concept that correlates the physical characteristics of the cyclist with the profile of the race (or sometimes just a part of the race), for the project we consider these ideas by means of some variables. These variables may include weight, height and age together with the elevation gain and loss of the entire race but also of some phases of the race.

Finally, although the cyclist's performance may seem to be only conditioned by his/her own abilities and the characteristics of the course, it is also influenced by the competitors he or she has. A race can have a higher level of demand depending on the cyclists who participate in it; the effort (and time) required to win a low level race is not the same as one where the big stars meet. In this sense, one of the sources consulted has a numerical parameter that establishes the quality of the start list of each race determined by its participants.

In the Experimental setup, we detail all the chosen variables for the development of the project, specifically in Variables.

5 Experimental setup

In this section we present the different scenarios designed for the application of the Machine Learning techniques already proposed in Section 2.2.2. These scenarios experiment with different variables, different race types for training and testing, and finally applying all these settings to different Machine Learning models.

All configurations tested are detailed below. In addition, for clarity, they are depicted in Figure 9.

5.1 Splitting the dataset

In the following, we describe the different data we use for the training and evaluation of the models as well as the partitions we make for these two sets.

A quick reminder of how the dataset is structured. On the one hand we have the one-day races and on the other hand the multiple-day races. The latter are the same as the former, except that each stage is an individual race with its corresponding route and classifications. Looking at this, one can think of multiple ways of dividing the dataset, meaning by dataset in this case the totality of the available data. Therefore, the four different ways that we chose to run our models are described below. In each group, the used data and a brief explanation of the motivation behind is explained.

To concisely illustrate the volume of the different configurations, Table 3 shows the number of entries for each configuration. These are the different configurations we make for the distribution of the dataset.

One-day races for both training and testing For this first configuration we take all available data within the one-day races between the years 2014 and 2023. We generate a *DataFrame* object (given by pandas Python library) with the data in order to include it in our model building and evaluation process.

Multiple-day races for both training and testing Similarly to the previous case we take all races between the years 2014-2023 but this time for multiple-day races. It is worth remembering that we take each stage of a multiple-day race and treat them as

individual races. Again, we build a *DataFrame* with this data.

One-day for training and multiple-day for testing As mentioned in the previous group, we treat the stages of a multiple-day race as one-day races. Technically nothing differentiates a stage of a multiple-day race from a one-day race. But the real world tells us that you don't compete the same way in a one-day race where only winning matters as you do in a stage race where there may be other types of objectives. We can see if this is true by finding out if our model can generalise adequately to individual stages learning from one-day races. Finally, it is worth noting that although not usual, for this group the training set is smaller than the testing set due to the nature of this approach.

All available races for both training and testing Finally, following the logical evolution of the groups presented above, we take all available data, since, although not always, a larger volume of data can help a model to perform better.

Dataset	**Configuration**	**Training Entries**	**Testing Entries**	**Total Entries**
Dataset1	One-day races for both training and testing	14504	4058	18562
Dataset2	Multiple-day races for both training and testing	160872	37618	198490
Dataset3	One-day for training and multiple-day for testing	18562	198490	217052
Dataset4	All available races for both training and testing	175376	41676	217052

Table 3: *Number of entries for each configuration.*

Finally, it is important to note that for all configurations a ratio of 0.8 and 0.2 is used for the training and evaluation partitions respectively. Except for the third case where the partitioning is defined by the nature of the approach.

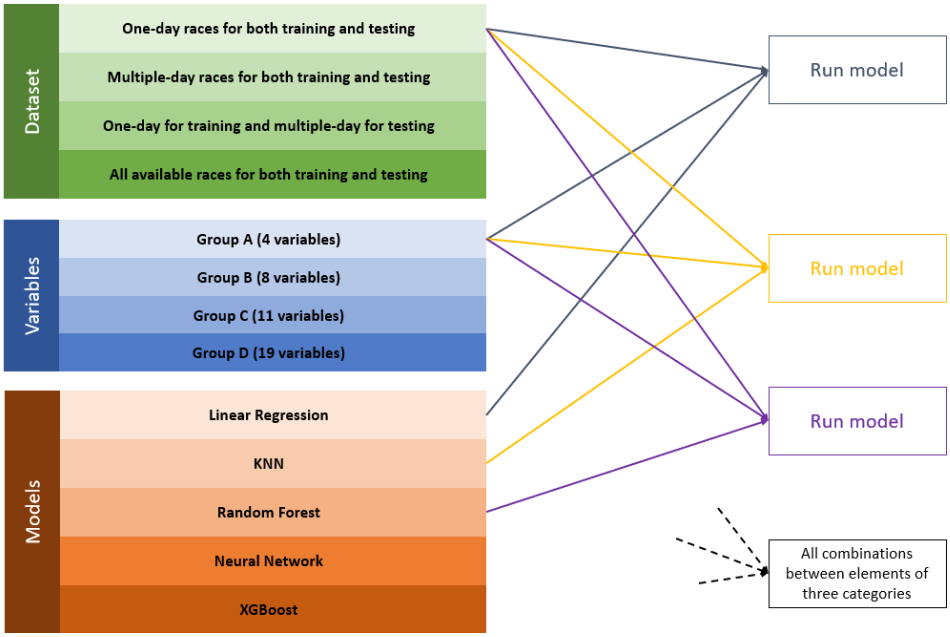


Figure 9: *Representation of the configurations that constitute the experimental setup.*

5.2 Variables

We group the variables into four different groups, starting with the simplest elements and adding more complexity. Some groups may include the variables from the previous one together with some new ones.

Our target variable is *time*. It tells us how long it takes for a cyclist to complete a race in seconds.

Group A For the first group of variables we only use the most basic elements that describe a race and the characteristics of a cyclist. The four variables we have are the following:

- *distance* represents the total distance for a particular race in metres.
- *height* is the cyclist's measured height in centimetres.
- *weight* is the cyclist's measured weight in kilograms.
- *age* is the age represented in days that the cyclist has when a race takes place.

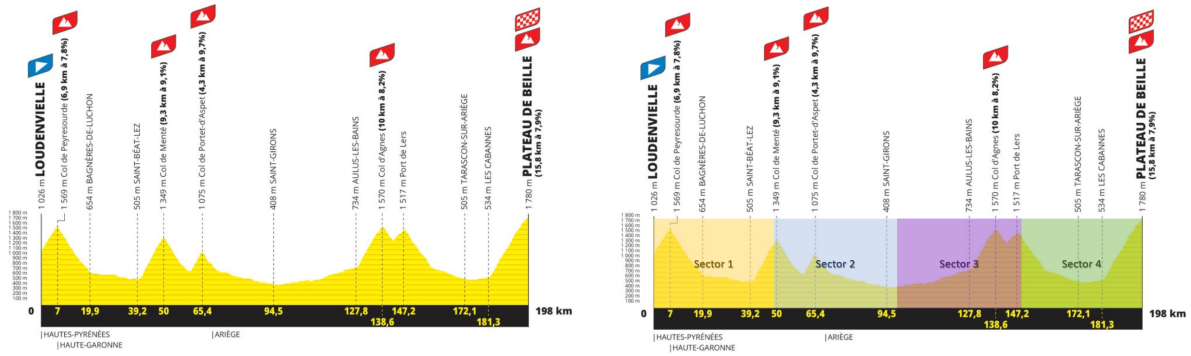
So, the variables composing this group are: *distance*, *height*, *weight* and *age*.

Group B For this second group of variables, in addition to all of the above, we add some other variables that provide more information about the cyclists. We add the following four variables:

- *nationality* is the nationality with which the cyclists are registered with UCI.
- *startlist_quality* is the score *ProCyclingStats* gives to each race. It is a parameter to measure the level of the participants of a given race.
- *race_year_days* are the days that passed since the beginning of the year until the day of the race. It indicates the period of the season in which we are.
- *rider_year_days* are the days that passed since the beginning of the year until the birth date of the cyclists. It indicates the period of the year in which a cyclist was born. The reason of including this attribute is explained in Section 4.

In total we have the eight variables, the previous four and the four new ones. The eight variables composing this group are: *distance*, *height*, *weight*, *age*, *nationality*, *startlist_quality*, *race_year_days* and *rider_year_days*.

Group C In this third group we have more specific information about the race route. For this purpose, we take the GPX files. From these files we can get the total distance we are already using and also the elevation gain and loss. In addition, we decide to divide the race into four sectors and get the elevations and distances corresponding to these four sectors (Figure 10 illustrates this idea). This can be useful to better determine the moments of the race.



(a) **(b)**

Figure 10: *Route of a race (a) and the division we make in the route (b) represented in the same race profile. Stage profile belonging to A.S.O. and taken from the Tour de France official website.*

Unlike the previous group, this one does not include all the variables of the previous groups but is composed of the variables that represent the characteristics of a race ignoring the characteristics of the cyclists. We only keep the total distance of the race, adding the following new ones:

- *course_type* is the type of the race. A race can be categorised as *"Individual Time Trial"*, *"Plain"*, *"Medium Mountain"* or *"High Mountain"*.
- *elevation_gain* are the total metres ascended in a race.
- *elevation_loss* are the total metres descended in a race.
- *elevation_gain_n* are the metres ascended in sector n of the race, for $n \in \{1, \ldots, 4\}$.
- *elevation_loss_n* are the metres descended in sector n of the race, for $n \in \{1, \ldots, 4\}$.

In total there are eleven new variables together with the distance. The twelve variables composing this group are the following: *distance*, *course_type*, *elevation_gain*, *elevation_loss*, *elevation_gain_n* (x4) and *elevation_loss_n* (x4).

Group D Lastly, we define a group that contains all the variables that are defined in the three previous groups. A total of nineteen variables are present here. These nineteen variables are the following: *distance*, *height*, *weight*, *age*, *nationality*, *startlist_quality*, *race_year_days*, *rider_year_days*, *course_type*, *elevation_gain*, *elevation_loss*, *elevation_gain_n* (x4) and *elevation_loss_n* (x4).

5.3 Models

This section presents the different techniques that are used for the construction of the models. Each technique has certain hyperparameters that must be determined. This can be done manually or taking a range of values and apply cross-validation to choose the best combination of hyperparameters. This concept is explained in the section Tuning Hyperparameters with Cross-Validation.

In the following, each technique is presented together with the hyperparameters to be considered. In addition, the ranges or groups of values for which cross-validation is used are defined. For ease of understanding, Table 4 shows in an organised and summarised way all the information presented below.

For all the models, when predicting the target variable *time*, in order to determine the best hyperparameters, we maximise the negative Mean Squared Error (MSE) with cross-validation.

Linear Regression We describe two types of linear models: Ridge (Equation (1)) and Lasso (Equation (2)). These two differ from the OLS solution because they introduce regularisation to avoid overfitting. These two methods are accompanied by an additional parameter known as regularisation strength commonly represented by the Greek letter α.

This technique is explained earlier in Section 2.2.2, particularly in Linear Regression.

K-Nearest Neighbours In the case of KNN, the first hyperparameter to consider is the number of nearest neighbours k to be considered for prediction. Then, to determine which neighbours are the nearest it is essential to define a distance function that measures this distance. Finally, we also decide to test whether to assign weights to each of the k nearest neighbours depending on the distance or whether they all participate equally in the final prediction.

This technique is explained earlier in Section 2.2.2, particularly in K-Nearest Neighbours (KNN).

Random Forests For the case of Random Forests, the key hyperparameters to be considered are, firstly, the number of trees. This is fundamental since it controls the number of decision trees that are present in the forest. For each decision tree we also define some values for the maximum depth. Finally, we have the minimum samples split that controls the minimum number of samples required to split an internal node.

This technique is explained earlier in Section 2.2.2, particularly in Random Forest.

Neural Networks For the case of Neural Networks, we define the number of hidden layers (excluding input and output). In addition, the number of neurons in each of these hidden layers is defined. Finally, the activation function is always set to $ReLu$ and the regularisation strength is introduced again.

This technique is explained earlier in Section 2.2.2, particularly in Neural Networks.

XGBoost For the case of XGBoost we define up to seven hyperparameters. Firstly, we have the number of boosting iterations or trees to build together with the loss function

to be optimised during training. Additionally, we have the maximum depth of each tree in the boosting process, the minimum child weight and the learning rate (also known as shrinkage) which determines how much the algorithm updates the weights in each iteration. Finally, we define the subsample ratio and column subsample ratio for each boosting iteration which determines how is our data used in the training process.

This technique is explained earlier in Section 2.2.2, particularly in XGBoost.

Performance measurement metrics For the Kendall Tau Distance, since it is a ranking metric that does not reward better prediction of top ranking positions, we consider k as the entire ranking. This allows us to see, in general terms, how well it predicts positions in the whole ranking and not just in the top.

For the case of the nDCG we do reward high ranking positions, knowing this we could consider the whole ranking again but we decide to limit k to 10 to see how well we predict the top 10. After all, this metric is interesting for us to see how well we can predict the victory or the big favourites of a race, so we can directly not consider the rest of the positions for this metric.

	Hyperparameter	Range
Linear Reg.	Method	[Ridge, Lasso]
	Regularisation strength α	[0.05, 0.5, 2.0]
KNN	Number of neighbours	[1, 3, 5, 10]
	Distance function	[Euclidean, Manhattan]
	Weights	[Uniform, Distance]
Random Forest	Number of trees	[25, 50, 100, 200]
	Maximum depth of trees	[5, 10, 20, None]
	Minimum samples split	[2, 5, 10, 25, 50]
Neural Networks	Number of hidden layers	[1, 2, 3, 6]
	Number of neurons for each layer	[32, 64, 100]
	Regularisation strength α	[0.05, 0.5, 2.0]
XGBoost	Number of boosting rounds	[25, 50, 100]
	Loss function	[MSE, MAE]
	Maximum depth	[5, 10, 20, None]
	Minimum child weight	[1, 3, 5, 10]
	Learning rate η	[0.01, 0.05, 0.1, 0.5]
	Subsample ratio	[0.7, 1.0]
	Column subsample ratio	[0.7, 1.0]

Table 4: *Hyperparameters and their tested ranges for different techniques.*

6 Results and discussion

This section presents a review of the results obtained after executing all the configurations described above in the experimental setup.

6.1 Dataset 1

Recall that in dataset 1 we trained with one-day races and tested also with one-day races in a ratio of 0.8 and 0.2 respectively (see Table 3).

As far as the RMSE metric is concerned (Table 5), the best group is A in the linear regression model. However, it is worth noting that the best result of all is obtained with group D, which is the one with the largest number of variables since it contains the variables of the rest of the group.

Furthermore, for the ranking metrics (Table 6 and 7), we see that in the case of Kendall no value is too good and none of them stands out especially from the rest. However, for the case of nDCG we see that group C offers very good results and also Random Forest.

RMSE	**Linear R.**	**KNN**	**Random F.**	**Neural N.**	**XGBoost**	**Group avg.**
groupA	1087.87	1236.85	1234.47	1127.18	1248.04	**1186.88**
groupB	1086.64	1634.08	1356.12	3371.10	1170.84	1723.76
groupC	811.08	2058.42	1272.61	856.60	1188.11	1237.36
groupD	805.82	1960.01	1250.27	923.67	1094.93	1206.94
Model avg.	**947.85**	1722.34	1278.37	1569.64	1175.48	-

Table 5: *Values of the RMSE metric for each group of variables for dataset 1 across models.*

Kendall τ	**Linear R.**	**KNN**	**Random F.**	**Neural N.**	**XGBoost**	**Group avg.**
groupA	0.009435	0.028255	-0.039282	0.009692	0.002376	**0.002095**
groupB	0.004597	0.011274	-0.265993	-0.164961	0.009190	-0.081179
groupC	-0.170825	-0.164961	-0.245791	-0.167845	-0.164961	-0.182877
groupD	-0.164961	0.015565	-0.011223	0.007323	0.026289	-0.025401
Model avg.	-0.080438	**-0.027467**	-0.140572	-0.078948	-0.031777	-

Table 6: *Values of the Kendall τ metric for each group of variables for dataset 1 across models.*

nDCG	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	0.105124	0.110446	0.513236	0.107714	0.104235	0.188151
groupB	0.099237	0.093934	0.630003	0.632783	0.154551	0.322102
groupC	0.566338	0.632783	0.632783	0.540486	0.632783	**0.601035**
groupD	0.632783	0.110041	0.167263	0.116265	0.129268	0.231124
Model avg.	0.350871	0.236801	**0.485821**	0.349312	0.255210	-

Table 7: *Values of the nDCG metric for each group of variables for dataset 1 across models.*

6.2 Dataset 2

This dataset is in concept the most similar to the first one as we train and test with races of the same type (single stages of multiple-day races in this case). Again we introduce a training and test ratio of 0.8 and 0.2 respectively (see Table 3).

For the RMSE (Table 8), again, the Linear Regression model is the most accurate and this time group D is the best in terms of mean values. This time the ranking metrics (Table 9 and 10) coincide in terms of best group of variables and best model, which are group C with Random Forest. This combination was already present in dataset 1 with the nDCG metric and is repeated from here on.

RMSE	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	1822.96	1982.54	2248.03	1830.91	1905.78	1958.05
groupB	1860.28	2063.96	1884.12	1831.88	1830.57	1894.16
groupC	1384.63	3458.55	1474.04	1747.49	1438.63	1900.67
groupD	1391.53	2845.02	1472.79	1594.45	1457.03	**1752.16**
Model avg.	**1614.85**	2587.52	1769.74	1751.18	1658.00	-

Table 8: *Values of the RMSE metric for each group of variables for dataset 2 across models.*

Kendall τ	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	-0.018693	0.001940	0.246690	0.014559	0.093453	0.067590
groupB	0.091338	-0.002988	0.234778	-0.030067	-0.010581	0.056496
groupC	0.090299	0.091338	0.234778	0.088804	0.091338	**0.119312**
groupD	-0.023130	0.002163	0.160248	-0.030472	-0.017443	0.018273
Model avg.	0.034954	0.023113	**0.219124**	0.010706	0.039192	-

Table 9: *Values of the Kendall* τ *metric for each group of variables for dataset 2 across models.*

nDCG	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	0.062703	0.076752	0.675417	0.104608	0.645329	0.312962
groupB	0.683583	0.079028	0.683583	0.079192	0.351637	0.375405
groupC	0.584838	0.683583	0.683583	0.528884	0.683583	**0.632894**
groupD	0.066962	0.075782	0.517598	0.067016	0.100520	0.165576
Model avg.	0.349521	0.228786	**0.640045**	0.194925	0.445267	-

Table 10: *Values of the nDCG metric for each group of variables for dataset 2 across models.*

6.3 Dataset 3

This dataset contains one-day races as training for later testing with multiple-day races (see Table 3).

Following the previous tendencies, we have that the best value for RMSE (Table 11) is obtained with the Linear Regression model although this time it is not the group of variables D, but B that is the best. Similarly, group C and Random Forest offer results that stand out from the rest of the configurations as far as ranking metrics are concerned (Table 12 and 13).

RMSE	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	2084.82	4869.11	4674.36	3168.05	5067.14	3972.70
groupB	2072.73	5219.40	4661.84	2112.77	4685.06	**3750.36**
groupC	1720.23	6317.25	4876.03	1860.81	4964.45	3947.75
groupD	1732.03	6244.91	4849.21	2073.98	4780.16	3936.06
Model avg.	**1902.45**	5662.67	4765.36	2303.90	4874.20	-

Table 11: *Values of the RMSE metric for each group of variables for dataset 3 across models.*

Kendall τ	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	0.025937	0.007921	0.332225	0.029943	0.027287	0.084663
groupB	0.023947	0.005446	-0.006588	0.022485	0.000042	0.009066
groupC	0.129180	0.129770	0.356080	0.127523	0.129770	**0.174465**
groupD	0.129770	-0.003514	0.295255	-0.019844	-0.010671	0.078199
Model avg.	0.077209	0.034906	**0.244243**	0.040027	0.036607	-

Table 12: *Values of the Kendall τ metric for each group of variables for dataset 3 across models.*

nDCG	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	0.098957	0.081183	0.663828	0.103001	0.099001	0.209194
groupB	0.090007	0.079599	0.077652	0.106902	0.095336	0.089899
groupC	0.660559	0.684892	0.684892	0.592299	0.684892	**0.661507**
groupD	0.684892	0.076969	0.630725	0.075747	0.084704	0.310607
Model avg.	0.383604	0.230661	**0.514274**	0.219487	0.240983	-

Table 13: *Values of the nDCG metric for each group of variables for dataset 3 across models.*

6.4 Dataset 4

Finally, this dataset contains all available one-day and multiple-day races and again uses the ratio 0.8 and 0.2 for training and test respectively.

To confirm the tendency, again the model that most accurately determines the times according to RMSE (Table 14) is the Linear Regression model. Once again we have the group of variables D as the most remarkable. For the ranking metrics (Table 15 and 16) we see that we continue to have group C with Random Forest as the best configuration.

RMSE	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	1782.16	1922.89	2189.39	1776.70	1851.94	1904.62
groupB	1782.55	2009.39	1882.32	5605.61	1766.77	2609.33
groupC	1328.19	3451.53	1534.90	1305.52	1424.30	1808.89
groupD	1331.44	2817.74	1506.04	1393.71	1360.72	**1681.93**
Model avg.	**1556.09**	2550.39	1778.16	2520.38	1600.93	-

Table 14: *Values of the RMSE metric for each group of variables for dataset 4 across models.*

Kendall τ	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	0.015953	0.002296	0.173653	0.005985	-0.000109	0.039555
groupB	0.059189	0.003752	0.208848	0.054078	0.003426	0.065859
groupC	0.057693	0.059189	0.174498	0.057023	0.059189	**0.081518**
groupD	-0.020564	0.003051	0.119030	-0.029517	-0.018297	0.010741
Model avg.	0.028068	0.017072	**0.169007**	0.021892	0.011052	-

Table 15: *Values of the Kendall* τ *metric for each group of variables for dataset 4 across models.*

nDCG	Linear R.	KNN	Random F.	Neural N.	XGBoost	Group avg.
groupA	0.082617	0.081990	0.665221	0.084038	0.085885	0.199950
groupB	0.677211	0.084927	0.604962	0.218621	0.180484	0.353241
groupC	0.592323	0.677211	0.677211	0.457820	0.677211	**0.616355**
groupD	0.065904	0.081589	0.482261	0.065820	0.108432	0.160801
Model avg.	0.354514	0.231430	**0.607414**	0.206575	0.263003	-

Table 16: *Values of the nDCG metric for each group of variables for dataset 4 across models.*

We see that, in terms of RMSE, the model that best determines the times is the Linear Regression model. Although it is true that in some datasets the difference with respect to other models is not so great, in some others it is, and in any case it is always the best. As far as the ranking metrics are concerned, it is very remarkable that the regression models based on random forests offer practically all the time, and for both metrics, the best results. These results are considerably better than for the other models, with the differences being considerably larger than in the case of RMSE.

6.5 Comparison between datasets

Finally, in tables 17, 18 and 19 we show the mean values obtained for each of the evaluation metrics across the different models. This mean is represented by comparing each dataset with each group of variables in order to determine which combination of data gives the best results after completing the experiment.

We can see that analysing the results of RMSE (Table 17) that indicate how well the times have been predicted individually without considering the ordination for each race, the best results appear in dataset 1 with the group of variables D. It must be remembered that group D contains all the possible variables, so it may be logical that with a greater type of data the models are able to establish a better correlation between these variables and the target variable *time*. Dataset 1 is the one that offers the best results together with dataset 2, they coincide in that they do not mix different types of races (one-day races and individual stages of multiple-day races). Dataset 4 is, together with dataset 3, the one with the largest number of entries (see Table 3). In this dataset individual stages are mixed with one-day races. In contrast to dataset 3, where training with one-day races is used to predict individual stages, in dataset 4 both are mixed for training and testing.

This makes sense because if it is decided that individual stages are to be considered as one-day races and not only as stages within a race, they need to be treated equally in both training and test. This mitigates any possible differences between these two types of races, since, if there exist differences, with dataset 3 the models may overfit to the peculiarities of the one-day races and and not be able to generalise to individual stages.

The ranking metrics provide the best results for group C in which no rider variables are considered. This group does not contain data on individual cyclists so it would not be surprising if it performed good overall approximations in terms of time precision (RMSE), but it is strange that it is able to rank them better. It should be noted that group C does not consider the cyclists‘ variables, so it is predicting the same time for all of them. From this point on, the sorting is done arbitrarily by means of identifiers assigned within the logic of the dataset. In short, the results of this group should not be taken into account too much. However, seeing the good results of this arbitrariness, we can think that the prediction of rankings based on the estimated time is not a good way to approximate the problem of predicting individual positions. For the rest of the groups we see that the results do not differ much from each other, being all of them quite bad. In any case, these are difficult metrics to interpret, since, although the value is higher for better predictions, it is difficult to establish what difference in values can be considered as being a considerably better constructed ranking.

RMSE	**dataset1**	**dataset2**	**dataset3**	**dataset4**	**Group average**
groupA	1186.88	1958.05	3972.70	1904.62	2255.56
groupB	1723.76	1894.2	3750.36	2609.33	2494.40
groupC	1237.36	1900.67	3947.75	1808.89	2223.67
groupD	1206.94	1752.16	3936.06	1681.93	**2144.27**
Dataset average	**1338.74**	1876.26	3901.72	2001.19	-

Table 17: *Mean value of the RMSE metric for each combination of variables and dataset across all models.*

Kendall	**dataset1**	**dataset2**	**dataset3**	**dataset4**	**Group average**
groupA	-0.008803	0.037055	0.042428	0.016939	0.021905
groupB	0.063935	0.027808	0.008797	0.036658	0.002332
groupC	-0.182877	0.119312	0.174465	0.081518	**0.048104**
groupD	-0.025401	0.018273	0.078199	0.010741	0.020453
Dataset average	-0.070254	0.050612	**0.075972**	0.036464	-

Table 18: *Mean value of the Kendall τ metric for each combination of variables and dataset across all models.*

nDCG	dataset1	dataset2	dataset3	dataset4	Group average
groupA	0.188151	0.312962	0.209194	0.199950	0.227564
groupB	0.322102	0.375405	0.089899	0.353241	0.285162
groupC	0.601035	0.632894	0.661507	0.616355	**0.627948**
groupD	0.231124	0.165576	0.310607	0.160801	0.217027
Dataset average	0.335603	**0.371709**	0.317802	0.332587	-

Table 19: *Mean value of the nDCG metric for each combination of variables and dataset across all models.*

6.6 Hyperparameters

This section gives a brief review of the hyperparameters selected with cross-validation for our models. Tables 20, 21, 22, 23 and 24 of the appendix show more detailed information on this topic.

Linear Regression We find that for all configurations, the best regularisation method has been Lasso except for one occasion. For the regularisation strength α, there is a greater variation of values, with 0.05 being the most common, being chosen in almost half of the occasions. However, for the models that give the best values in the evaluation metrics, 1.0 is chosen as regularisation strength.

KNN We see that the most common k, being chosen in more than half of the occasions, is 10. In the case of the calculation of the distances, half of the configurations choose the Manhattan distance and the other half Euclidean. For the groups of variables A and B, which are those with the least number of variables, Euclidean is chosen and for C and D, which have the greatest number of variables, Manhattan is chosen. For the case of the weights for each nearest neighbour, the most repeated value is uniform. However, in the configuration for which we obtain the best ranking metrics we have a k value of 1 where the hyperparameter of the weights becomes meaningless.

Ranodm Forest Some of the configurations (marked in Table 22 with a footnote) have fewer hyperparameters due to the high computational cost. Therefore, we cannot draw as many conclusions about the hyperparameters. In any case, in most cases a number of trees of 25 or 50 is chosen, never shooting up to values as high as 100 or 200 even though they are tested. Similarly, the maximum tree depth is always limited to between 5 and 10 and

never left without a maximum depth (None), so this is an important hyperparameter and it is worth to specify it. For the hyperparameter that determines the minimum number of samples required to split a node the full range of possible values appears in the results.

Neural Network It is noteworthy that for absolutely all cases 2 hidden layers are chosen. In fact, for the three best performing models in our three performance measurement metrics, the number of neurons and the regularisation strength α is different and does not repeat.

XGBoost In the majority of cases the boosting rounds hyperparameter (n_estimators in Table 24) is set to 100 which is the upper bound of the set of values that have been defined for this hyperparameter. This may mean that even higher values could be used. In any case, sometimes a very high number of rounds can lead to overfitting and techniques such as early stopping should be introduced to avoid this. This remains as a possible improvement to be applied. In the case of the loss function we have that half of the time MAE is chosen and half of the time MSE is chosen. We also have that for the maximum depth of each tree the only values that appear are more extreme values such as 5 and None, never being set to 10 or 20. The learning rate remains stable at the two intermediate values of 0.05 and 0.1, never dropping to 0.01 and rarely being set to 0.5. For "subsample", which determines the ratio of the data used when bootstrapping, we have that sometimes it is set to 0.7 but in many others it is omitted (left at 1.0, all data is used). For "column subsample by tree" that determines the ratio of features that are being used to train each tree, we have something similar being sometimes set to 0.7 and other times not applying it (set to 1.0).

7 Conclusions and future lines

Looking at the results, we have that the models have very different configurations when they predict finishing times more accurately compared to when they predict the classification more precisely. We also see that the results given by the ranking metrics are not very representative. Furthermore, as explained above, one could predict times very badly but if this deviation was respected for all riders by overestimating or underestimating their times, good rankings could be built. This can make us understand that the approach to the problem of estimating times and estimating positions may have a different nature. Although by estimating times one can construct a ranking, it does not seem to be the best way to do so, since, in the obtained results, the best predicted ranking rarely has the best estimated times. In fact, our *"regression models approach"* responds directly to the need to estimate times properly, and not rankings, since our target variable is time and we always aim to minimise its error. Therefore, we believe that analysing RMSE values is more appropriate to draw conclusions regarding future lines and possible improvements.

In this sense, we have that the group of variables D offers the best average results, and that analysing the datasets individually in two of them it offers the best result, and when it is not the best, it is not far from being it. Similarly, within the datasets we notice that the best one is the first, where we try to predict one-day races based on training with one-day races. It is true that dataset 2 (multiple-day races) does not mix races like datasets 3 and 4 do, but it does not give as good results, which may indicate that more context is needed for the individual stages within a multiple-day race. Additionally, the poor RMSE results of dataset 3 indicate that we cannot treat individual stages from multiple-day races as single day races.

In terms of data, following the previous ideas, there is still a need to make more use of existing data such as complementary classifications. Predicting individual stages as one-day races may not be a fully polished approximation and this can be seen in the results. When it comes to predicting a one-day race we can say that the best rider wins. On the other hand, in a stage, in the middle of a grand tour, it may be that more ambitious goals such as the general classification and not just a stage win make riders race differently. In addition, since the group with greatest number of variables has the best results, one can intuitively think of collecting more data, so maybe more accurate data on weather conditions or cyclists (medical information, for example) could improve the results.

Once the whole dataset is processed and ready to be applied, one new approach could be to design classification models. This could be done by sorting by winner and non-winner for each of the races and trying to predict a single champion. Additionally, some hyperparameters could be adjusted by offering a different range of values if it is considered that some combinations are missing (this could be the case in the boosting rounds of XGBoost).

Although the challenge of predicting results, not in cycling but in any sport, is enormous, accurate approximations can be achieved. In this sense, the basis of this project, together with certain improvements such as those described, can be applied in the real world. It should be remembered that road cycling races are held on the streets of the world practically during the whole year, on many occasions paralysing large cities. In order to mitigate the consequences of this, passing times are estimated at certain points in order to be able to manage traffic among other things. It is well known that these predictions are not always accurate and work could be done on this aspect to see if the passing times predicted by the race organisations can be improved with Machine Learning techniques, being this just an example of possible applications in real life.

References

[1] W. Andreff, "The correlation between economic underdevelopment and sport," *European Sport Management Quarterly*, vol. 1, no. 4, pp. 251–279, 2001. DOI: 10.1080/16184740108721902. eprint: https://doi.org/10.1080/16184740108721902. [Online]. Available: https://doi.org/10.1080/16184740108721902.

[2] D. Van Reeth, "Globalization in professional road cycling," in *The Economics of Professional Road Cycling*, D. Van Reeth, Ed. Springer International Publishing, 2022, pp. 337–367, ISBN: 978-3-031-11258-4. DOI: 10.1007/978-3-031-11258-4_14. [Online]. Available: https://doi.org/10.1007/978-3-031-11258-4_14.

[3] "UCI Cycling Regulations." (Mar. 4, 2024), [Online]. Available: https://assets.ctfassets.net/761l7gh5x5an/6FEzFHeA2oKMBGb5sdIvQ7/6ecdf625b5175759daed42c522204ef1/2-ROA-20240205-E.pdf (visited on Mar. 11, 2024).

[4] A. Baker. "The most watched sporting events in the world - roadtrips." (Nov. 3, 2023), [Online]. Available: https://www.roadtrips.com/blog/the-most-watched-sporting-events-in-the-world/ (visited on Mar. 11, 2024).

[5] M. Bopp, D. Sims, and D. Piatkowski, "The bicycle: A technological and social history," in Jan. 2018, pp. 1–19, ISBN: 9780128126424. DOI: 10.1016/B978-0-12-812642-4.00001-5.

[6] J.-F. Mignot, "The history of professional road cycling," in *The Economics of Professional Road Cycling*, D. Van Reeth and D. J. Larson, Eds. Cham: Springer International Publishing, 2016, pp. 7–31, ISBN: 978-3-319-22312-4. DOI: 10.1007/978-3-319-22312-4_2. [Online]. Available: https://doi.org/10.1007/978-3-319-22312-4_2.

[7] "A Short History of PBP — Randonneurs USA." (Mar. 11, 2024), [Online]. Available: https://rusa.org/pages/PBP-short-history (visited on Mar. 11, 2024).

[8] "A True Classic: The History of Paris-Brest-Paris." (Oct. 30, 2017), [Online]. Available: https://stories.strava.com/articles/a-true-classic-the-history-of-paris-brest-paris (visited on Mar. 11, 2024).

[9] D. Van Reeth, "Tv broadcasting of road cycling races," in *The Economics of Professional Road Cycling*, D. Van Reeth, Ed. Cham: Springer International Publishing, 2022, pp. 123–161, ISBN: 978-3-031-11258-4. DOI: 10.1007/978-3-031-11258-4_6. [Online]. Available: https://doi.org/10.1007/978-3-031-11258-4_6.

[10] "UCI - History — UCI." (Mar. 11, 2024), [Online]. Available: https://www.uci.org/uci-history/4E4a552SyXOUBCtsZkEN8v (visited on Mar. 11, 2024).

[11] "Improving the Bicycle – USC Viterbi School of Engineering." (Dec. 29, 2002), [Online]. Available: https://illumin.usc.edu/improving-the-bicycle/ (visited on Mar. 11, 2024).

[12] "Sporting stakes." (Mar. 11, 2024), [Online]. Available: https://www.letour.fr/en/the-race/sporting-stakes (visited on Mar. 11, 2024).

[13] "Numpy -." (Mar. 12, 2024), [Online]. Available: https://numpy.org/ (visited on Mar. 18, 2024).

[14] "Pandas - python data analysis library." (Mar. 17, 2024), [Online]. Available: https://pandas.pydata.org/ (visited on Mar. 18, 2024).

[15] "Scikit-learn: Machine learning in python — scikit-learn 1.4.1 documentation." (Mar. 18, 2024), [Online]. Available: https://scikit-learn.org/stable/ (visited on Mar. 18, 2024).

[16] S. Weisberg, *Applied linear regression*. John Wiley & Sons, 2005, vol. 528.

[17] J. Groß, *Linear regression*. Springer Science & Business Media, 2003, vol. 175.

[18] M. Tranmer and M. Elliot, "Multiple linear regression," *The Cathie Marsh Centre for Census and Survey Research (CCSR)*, vol. 5, no. 5, pp. 1–5, 2008.

[19] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, R. Meersman, Z. Tari, and D. C. Schmidt, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 986–996, ISBN: 978-3-540-39964-3.

[20] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[21] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324.

[22] S. O. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ: Pearson, Nov. 2008.

[23] "Xgboost documentation — xgboost 2.0.3 documentation." (Dec. 19, 2023), [Online]. Available: https://xgboost.readthedocs.io/en/stable/ (visited on May 10, 2024).

[24] D. Berrar, "Cross-validation," in Jan. 2018, ISBN: 9780128096338. DOI: 10.1016/B978-0-12-809633-8.20349-X.

[25] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938, ISSN: 00063444. [Online]. Available: http://www.jstor.org/stable/2332226 (visited on May 26, 2024).

[26] V. Efimov. "Comprehensive guide to ranking evaluation metrics — by vyacheslav efimov — towards data science." (Oct. 19, 2023), [Online]. Available: https://towardsdatascience.com/comprehensive-guide-to-ranking-evaluation-metrics-7d10382c1025 (visited on May 21, 2024).

[27] A. Jain, H. Patel, L. Nagalapatti, *et al.*, "Overview and importance of data quality for machine learning tasks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 3561–3562, ISBN: 9781450379984. DOI: 10.1145/3394486.3406477. [Online]. Available: https://doi.org/10.1145/3394486.3406477.

[28] "Procyclingstats." (), [Online]. Available: https://www.procyclingstats.com.

[29] "La flamme rouge." (), [Online]. Available: https://www.la-flamme-rouge.eu.

[30] "GPX: the GPS Exchange Format." (Mar. 11, 2024), [Online]. Available: https://www.topografix.com/gpx.asp (visited on Mar. 11, 2024).

[31] A. Longo, C. Siffredi, M. Cardey, G. Aquilino, and N. Lentini, "Age of peak performance in olympic sports: A comparative research among disciplines," *Journal of Human Sport and Exercise*, vol. 11, Dec. 2016. DOI: 10.14198/jhse.2016.111.03.

[32] T. V. Leonid Kholkine Steven Latré and A.-W. de Leeuw, "Age of peak performance in professional road cycling," *Journal of Sports Sciences*, vol. 41, no. 3, pp. 298–306, 2023, PMID: 37139786. DOI: 10.1080/02640414.2023.2208998. eprint: https://doi.org/10.1080/02640414.2023.2208998. [Online]. Available: https://doi.org/10.1080/02640414.2023.2208998.

[33] J. B. Jean Côté Dany J. Macdonald and B. Abernethy, "When "where" is more important than "when": Birthplace and birthdate effects on the achievement of sporting expertise," *Journal of Sports Sciences*, vol. 24, no. 10, pp. 1065–1073, 2006, PMID: 17115521. DOI: 10.1080/02640410500432490. eprint: https://doi.org/10.1080/02640410500432490. [Online]. Available: https://doi.org/10.1080/02640410500432490.

[34] J. Musch and S. Grondin, "Unequal competition as an impediment to personal development: A review of the relative age effect in sport," *Developmental Review*, vol. 21, no. 2, pp. 147–167, 2001, ISSN: 0273-2297. DOI: https://doi.org/10.100

6/drev.2000.0516. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273229700905161.

[35] J. Musch and R. Hay, "The relative age effect in soccer: Cross-cultural evidence for a systematic discrimination against children born late in the competition year," *Sociology of Sport Journal*, vol. 16, no. 1, pp. 54–64, 1999. DOI: 10.1123/ssj.16.1.54. [Online]. Available: https://journals.humankinetics.com/view/journals/ssj/16/1/article-p54.xml.

[36] J. Hopker, D. Coleman, and L. Passfield, "Changes in cycling efficiency during a competitive season," *Medicine & Science in Sports & Exercise*, vol. 41, no. 4, 2009, ISSN: 0195-9131. DOI: 10.1249/MSS.0b013e31818f2ab2. [Online]. Available: https://journals.lww.com/acsm-msse/fulltext/2009/04000/changes_in_cycling_efficiency_during_a_competitive.22.aspx.

[37] N. P. Craig and K. I. Norton, "Characteristics of track cycling," *Sports Medicine*, vol. 31, no. 7, pp. 457–468, Jun. 2001, ISSN: 1179-2035. DOI: 10.2165/00007256-200131070-00001. [Online]. Available: https://doi.org/10.2165/00007256-200131070-00001.

Appendix A Hyperparameters and complete results

Table 20: *Results for Linear Regression models.*

Variables	Dataset	Hyperparameters	RMSE	Kendall's τ	nDCG@k
groupA	dataset1	regularisation type=ridge, α=0.05	1087.865700	0.009435	0.105124
groupA	dataset2	regularisation type=lasso, α=0.05	1822.957321	-0.018693	0.062703
groupA	dataset3	regularisation type=lasso, α=0.05	2084.823081	0.025937	0.098957
groupA	dataset4	regularisation type=lasso, α=0.05	1782.164103	0.015953	0.082617
groupB	dataset1	regularisation type=lasso, α=1.0	1086.643305	0.004597	0.099237
groupB	dataset2	regularisation type=lasso, α=0.5	1860.277739	0.091338	0.683583
groupB	dataset3	regularisation type=lasso, α=1.0	2072.728431	0.023947	0.090007
groupB	dataset4	regularisation type=lasso, α=0.1	1782.549940	0.059189	0.677211
groupC	dataset1	regularisation type=lasso, α=0.5	811.075056	-0.170825	0.566338
groupC	dataset2	regularisation type=lasso, α=0.05	1384.634231	0.090299	0.584838
groupC	dataset3	regularisation type=lasso, α=0.5	1720.233374	0.129180	0.660559
groupC	dataset4	regularisation type=lasso, α=0.05	1328.187807	0.057693	0.592323
groupD	**dataset1**	**regularisation type=lasso, α=1.0**	**805.818443**	-0.164961	0.632783
groupD	dataset2	regularisation type=lasso, α=0.05	1391.529587	-0.023130	0.066962
groupD	**dataset3**	**regularisation type=lasso, α=1.0**	1732.025383	**0.129770**	**0.684892**
groupD	dataset4	regularisation type=lasso, α=0.1	1331.438694	-0.020564	0.065904

Table 21: *Results for KNN regression models.*

Variables	Dataset	Hyperparameters	RMSE	Kendall's τ	nDCG@k
groupA	**dataset1**	**k=10, distance=euclidean, weights=distance**	**1236.854463**	0.028255	0.110446
groupA	dataset2	k=10, distance=euclidean, weights=uniform	1982.544800	0.001940	0.076752
groupA	dataset3	k=10, distance=euclidean, weights=distance	4869.114916	0.007921	0.081183
groupA	dataset4	k=10, distance=euclidean, weights=uniform	1922.894145	0.002296	0.081990
groupB	dataset1	k=5, distance=euclidean, weights=distance	1634.078391	0.011274	0.093934
groupB	dataset2	k=10, distance=euclidean, weights=distance	2063.958723	-0.002988	0.079028
groupB	dataset3	k=5, distance=euclidean, weights=distance	5219.397162	0.005446	0.079599
groupB	dataset4	k=10, distance=euclidean, weights=distance	2009.387105	0.003752	0.084927
groupC	dataset1	k=10, distance=manhattan, weights=distance	2058.421459	-0.164961	0.632783
groupC	dataset2	k=3, distance=manhattan, weights=uniform	3458.548750	0.091338	0.683583
groupC	**dataset3**	**k=1, distance=manhattan, ~~weights=uniform~~**	6317.248067	**0.129770**	**0.684892**
groupC	dataset4	k=1, distance=manhattan, ~~weights=uniform~~	3451.532158	0.059189	0.677211
groupD	dataset1	k=10, distance=manhattan, weights=distance	1960.014788	0.015565	0.110041
groupD	dataset2	k=10, distance=manhattan, weights=uniform	2845.017421	0.002163	0.075782
groupD	dataset3	k=10, distance=manhattan, weights=distance	6244.905351	-0.003514	0.076969
groupD	dataset4	k=10, distance=manhattan, weights=uniform	2817.735248	0.003051	0.081589

Table 22: *Results for Random Forest regression models.*

Variables	Dataset	Hyperparameters	RMSE	Kendall's τ	nDCG@k
groupA	**dataset1**	**n estimators=50, max depth=5, min samples split=2**	**1234.474220**	-0.093775	0.513236

groupA	dataset2	n estimators=25, max depth=5, min samples split=10	2248.032951	0.094018	0.675417
groupA	dataset3	n estimators=25, max depth=5, min samples split=2	4674.364025	0.121052	0.663828
groupA	dataset4	n estimators=50, max depth=5, min samples split=5	2189.388787	0.060568	0.665221
groupB	dataset1	n estimators=25, max depth=5, min samples split=50	1356.121460	-0.179773	0.630003
groupB	dataset2	n estimators=50, max depth=5, min samples split=5	1884.118211	0.091338	0.683583
groupB	dataset3	n estimators=50, max depth=20, min samples split=25	4661.842956	-0.007933	0.077652
[1]groupB	dataset4	n estimators=50, max depth=10, min samples split=10	1882.322858	0.062842	0.604962
groupC	dataset1	n estimators=50, max depth=None, min samples split=5	1272.605001	-0.245791	0.632783
groupC	dataset2	n estimators=25, max depth=10, min samples split=10	1474.035554	0.234778	0.683583
groupC	**dataset3**	**n estimators=25, max depth=None, min samples split=50**	4876.029764	**0.356080**	**0.684892**
[1]groupC	dataset4	n estimators=50, max depth=10, min samples split=5	1534.902196	0.174498	0.677211
groupD	dataset1	n estimators=50, max depth=10, min samples split=25	1250.273683	-0.011223	0.167263
[1]groupD	dataset2	n estimators=25, max depth=10, min samples split=25	1472.792302	0.160248	0.517598
[1]groupD	dataset3	n estimators=50, max depth=5, min samples split=25	4849.213541	0.295255	0.630725

[1]Due to the long execution times, the hyperparameters are limited for the cases marked with this note. New hyperparameters: number of tress: [25, 50, 100], maximum depth of trees: [None, 10] and minimum samples split: [5, 10, 25].

[1]groupD	dataset4	n estimators=50, max depth=10, min samples split=5	1506.041281	0.119030	0.482261

Table 23: *Results for Neural Network regression models.*

Variables	Dataset	Hyperparameters	RMSE	Kendall's τ	nDCG@k
groupA	dataset1	hidden layers=2, neurons=32, α=2.0	1127.180051	0.009692	0.107714
groupA	dataset2	hidden layers=2, neurons=100, α=0.5	1830.912800	0.014559	0.104608
groupA	dataset3	hidden layers=2, neurons=100, α=2.0	3168.054116	0.029943	0.103001
groupA	dataset4	hidden layers=2, neurons=100, α=2.0	1776.697228	0.005985	0.084038
groupB	**dataset1**	**hidden layers=2, neurons=64, α=0.5**	3371.104301	-0.164961	**0.632783**
groupB	dataset2	hidden layers=2, neurons=64, α=2.0	1831.877152	-0.030067	0.079192
groupB	dataset3	hidden layers=2, neurons=64, α=2.0	2112.773873	0.022485	0.106902
groupB	dataset4	hidden layers=2, neurons=32, α=0.5	5605.614011	0.054078	0.218621
groupC	**dataset1**	**hidden layers=2, neurons=100, α=2.0**	**856.596052**	-0.167845	0.540486
groupC	dataset2	hidden layers=2, neurons=64, α=0.5	1747.491485	0.088804	0.528884
groupC	**dataset3**	**hidden layers=2, neurons=32, α=0.05**	1860.811103	**0.127523**	0.592299
groupC	dataset4	hidden layers=2, neurons=64, α=2.0	1305.518037	0.057023	0.457820
groupD	dataset1	hidden layers=2, neurons=64, α=0.05	923.669144	0.007323	0.116265
groupD	dataset2	hidden layers=2, neurons=100, α=0.05	1594.452870	-0.030472	0.067016
groupD	dataset3	hidden layers=2, neurons=32, α=0.5	2073.980225	-0.019844	0.075747
groupD	dataset4	hidden layers=2, neurons=32, α=0.5	1393.709269	-0.029517	0.065820

Table 24: *Results for XGBoost regression models.*

Variables	Dataset	Hyperparameters	RMSE	Kendall's τ	nDCG@k
groupA	dataset1	loss=MSE, n estimators=100, max depth=5, min child weight=5, learning rate=0.05, subsample=0.7, colsample by tree=0.7	1248.043324	0.002376	0.104235
groupA	dataset2	loss=MAE, n estimators=25, max depth=5, min child weight=5, learning rate=0.1, subsample=0.7, colsample by tree=1.0	1905.778485	0.093453	0.645329
groupA	dataset3	loss=MAE, n estimators=50, max depth=5, min child weight=3, learning rate=0.1, subsample=1.0, colsample by tree=0.7	5067.139045	0.027287	0.099000
groupA	dataset4	loss=MSE, n estimators=100, max depth=5, min child weight=3, learning rate=0.1, subsample=1.0, colsample by tree=0.7	1851.939195	-0.000109	0.085885
groupB	dataset1	loss=MAE, n estimators=50, max depth=5, min child weight=10, learning rate=0.5, subsample=0.7, colsample by tree=1.0	1170.835842	0.009190	0.154551
groupB	dataset2	loss=MSE, n estimators=100, max depth=5, min child weight=3, learning rate=0.05, subsample=0.7, colsample by tree=1.0	1830.573962	-0.010581	0.351637
groupB	dataset3	loss=MAE, n estimators=100, max depth=None, min child weight=1, learning rate=0.5, subsample=0.7, colsample by tree=1.0	4685.062201	0.000042	0.095334

groupB	dataset4	loss=MSE, n estimators=100, max depth=5, min child weight=5, learning rate=0.1, subsample=1.0, colsample by tree=1.0	1766.770781	0.003426	0.180484
groupC	dataset1	loss=MAE, n estimators=100, max depth=None, min child weight=10, learning rate=0.05, subsample=1.0, colsample by tree=1.0	1188.109003	-0.164961	0.632783
groupC	dataset2	loss=MAE, n estimators=100, max depth=5, min child weight=1, learning rate=0.1, subsample=0.7, colsample by tree=1.0	1438.632354	0.091338	0.683583
groupC	**dataset3**	**loss=MSE, n estimators=100, max depth=5, min child weight=10, learning rate=0.05, subsample=1.0, colsample by tree=1.0**	4964.445286	**0.129770**	**0.685374**
groupC	dataset4	loss=MSE, n estimators=50, max depth=None, min child weight=1, learning rate=0.1, subsample=0.7, colsample by tree=1.0	1424.298572	0.059189	0.677211
groupD	**dataset1**	**loss=MAE, n estimators=100, max depth=5, min child weight=1, learning rate=0.1, subsample=0.7, colsample by tree=1.0**	**1094.927521**	0.026289	0.129268
groupD	dataset2	loss=MSE, n estimators=100, max depth=5, min child weight=10, learning rate=0.1, subsample=0.7, colsample by tree=0.7	1457.031750	-0.017443	0.100520

groupD	dataset3	loss=MAE, n estimators=100, max depth=5, min child weight=10, learning rate=0.1, subsample=1.0, colsample by tree=1.0	4780.164120	-0.010671	0.084934
groupD	dataset4	loss=MSE, n estimators=100, max depth=5, min child weight=3, learning rate=0.1, subsample=1.0, colsample by tree=0.7	1360.718633	-0.018297	0.108432