FACULTY OF ENGINEERING AND ARCHITECTURE

Predicting cycling results using machine learning

Emiel De Spiegeleer
Student number: 01405160

Supervisors: Prof. dr. ir. Luc Martens, Dr. ir. Toon De Pessemier
Counsellors: Dr. ir. Toon De Pessemier, Kris Vanhecke

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Academic year 2018-2019

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Predicting cycling results using machine learning

Emiel De Spiegeleer
Student number: 01405160

Supervisors: Prof. dr. ir. Luc Martens, Dr. ir. Toon De Pessemier
Counsellors: Dr. ir. Toon De Pessemier, Kris Vanhecke

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Computer Science Engineering

Academic year 2018-2019

GHENT UNIVERSITY

Preface

First of all I would like to thank prof. dr. ir. Luc Martens and dr. ir. Toon De Pessemier for making this thesis possible and providing me with valuable feedback whenever needed.

I would also like to express my gratitude to my parents, brothers, friends and girlfriend for their support, encouragement, constructive criticism and pretence to completely understand my poorly explained problems. A special thank you to my parents for providing me with everything I needed to obtain a higher education.

Finally, I should thank my cat for tirelessly voicing her opinions about my thesis and sometimes even going as far as trying to write the thesis herself in my absence.

Emiel De Spiegeleer - June 2019

Permission for use

The author gives permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use.

In all cases of other use, the copyright terms have to be respected, in particular with regard to the obligation to state explicitly the source when quoting results from this master dissertation.

Emiel De Spiegeleer - May 31, 2019

Abstract

Although much research exists about forecasting sport results using machine learning, road cycling has been neglected in this area. This thesis research uses machine learning and publicly available data to forecast three road cycling results: the average velocity of a stage, the difference between the average stage velocity and the velocity of a rider and, finally, the head-to-head wins between two riders in a stage. The stages under study were the Tour de France, Vuelta a España and Giro d'Italia from 2016-2018.

Different regression algorithms as well as recency weighting were tried for the forecasting of the first two results. The gradient boosting algorithm combined with recency weighting of up to two days in the past obtained the best results for the mean stage velocity prediction. A mean absolute error (MAE) of 0.4660 m/s was realized. For the forecasting of the difference between the average stage velocity and the velocity of a rider, a MAE of 0.1576 m/s was obtained using ridge regression and recency weighting of up to 90 days in the past. The two results were both a significant improvement over the baselines.

Gradient boosting also obtained the best head-to-head forecasting result. More important, including features based on the previously predicted difference between mean stage and rider velocity improved the prediction accuracy and resulted in an accuracy of 73.16%. This is again a significant improvement over the baseline.

These results demonstrate that machine learning in combination with public data can be used to predict cycling results with a reasonable accuracy.

Keywords: cycling, machine learning, Tour de France, Vuelta a España, Giro d'Italia

Predicting cycling results using machine learning

Emiel De Spiegeleer

Supervisor(s): Prof. dr. ir. Luc Martens, Dr. ir. Toon De Pessemier

Abstract—**Although much research exists about forecasting sport results using machine learning, road cycling has been neglected in this area. This thesis research uses machine learning and publicly available data to forecast three road cycling results: the average velocity of a stage, the difference between the average stage velocity and the velocity of a rider and, finally, the head-to-head wins between two riders in a stage. The stages under study were the Tour de France, Vuelta a España and Giro d'Italia from 2016-2018. Different regression algorithms as well as recency weighting were tried for the forecasting of the first two results. The gradient boosting algorithm combined with recency weighting of up to two days in the past obtained the best results for the mean stage velocity prediction. A mean absolute error (MAE) of 0.4660 m/s was realized. For the forecasting of the difference between the average stage velocity and the velocity of a rider, a MAE of 0.1576 m/s was obtained using ridge regression and recency weighting of up to 90 days in the past. The two results were both a significant improvement over the baselines. Gradient boosting also obtained the best head-to-head forecasting result. More important, including features based on the previously predicted difference between mean stage and rider velocity improved the prediction accuracy and resulted in an accuracy of 73.16%. This is again a significant improvement over the baseline. These results demonstrate that machine learning in combination with public data can be used to predict cycling results with a reasonable accuracy.**

Keywords—**cycling, machine learning, Tour de France, Vuelta a España, Giro d'Italia**

I. Introduction

ROAD cycling has properties that make it somewhat different to predict compared to two-contestant games (e.g. football and tennis) and even to other multi-contestant races (e.g. swimming and marathon). The most important one is that the influence of the route is huge. A mainly flat stage is expected to have a totally different top ten compared to a stage with a lot of steep hills. Every stage has a unique combination of length, slopes and surfaces (e.g. asphalt with or without pebbles, cobblestones). A second property that makes cycling special is the combination of teams and individuals. Teamwork is important, but only one person can win (except in e.g. team time trials). This can give rise to an interesting dynamic between team members.

A. Related work

Before discussing literature about the prediction of cycling results an important distinction has to be made between the words *cycling race* and *cycling stage*. Both are used, but mean something different. A cycling race is an abstract entity that combines different stages under some common name and theme. Examples of a cycling race are the Tour of Flanders and Vuelta a España. A cycling stage is a concrete contest with a concrete route on a certain date. For example the Tour of Flanders 2017 or the 6^{th} stage of the Vuelta a España of 2015.

Olds et al. [1] and Martin et al. [2] presented and validated mathematical models for road-cycling performance. They took into account aerodynamic drag, rolling resistance, friction in the bearings and chain drive system, and changes in kinetic and potential energy. Martin et al. calculated predictions of the cyclists power outputs that were close to the measured power outputs. The power output of a rider is a measure of how hard the rider has to work and is measured in watts (W). However, they only validated their mathematical models on a short and rather flat route with a concrete surface, which are not real stage situations. Kataoka and Gray [3] recently used a data-driven approach in order to predict the power output of professional riders in real-time. Using live GPS data, hand-crafted feature engineering using physics knowledge, automatic feature generation using autoencoder and a random forest algorithm, they were able to reduce the MAE by 56.79% (from 139.17W to 60.13W) compared to a model using only physics. According to Pezzoli et al. [4] the most influential meteorological variables for cycling performance are air temperature, wind, rain and air humidity. However, this conclusion was based upon qualitative research with focus groups, so should not be considered proven.

The research of Goff et al. [5]–[11] used inclined planes, physics and mathematical models to predict the stage-winning times of the Tour de France, from 2003 up until 2016. Included in the models were required power, drag and friction. Factors such as weather, team strategies, crashes and sharp turns were not taken into account. For the 2015 Tour de France, they predicted 12 stages below 5% mean absolute relative error, seven of which were below 1%. Only one of the predictions had an error worse than 10%. For the 2016 Tour de France they predicted 14 stages below 5% error, four of which were below 1% error.

II. Dataset

The dataset contains the stages from 2013 to 2018 included in the races of the UCI World Tour 2018 and the results of the top 200 cyclists on the Individual UCI World ranking for men at the end of 2018 in these stages. This information was scraped from the website procyclingstats.com [12]. Only the team time-trial (TTT) stages were not scraped for their rider results because those stages are rather rare and very different from the normal mass-start and individual time-trial (ITT) stages. Fig. 1 gives a schematic overview of this data collection process.

Only the historical results of a rider are not enough to forecast cycling results. One of the reasons predicting cycling results is so different from e.g. tennis or soccer is that many stages have a completely different route and this has a major impact on the results. Therefore, a list of coordinates was collected for each of the stages of the Tour de France, Vuelta a España and Giro d'Italia from 2013 to 2018. These coordinates were retrieved manually from different websites [13]–[16]. Thereafter, for every collected coordinate the elevation was retrieved using the Google Elevation API [17].

Another possible influence on the cycling results is the weather. The API used to gather this information is the one from World Weather Online [18]. This API provides historical

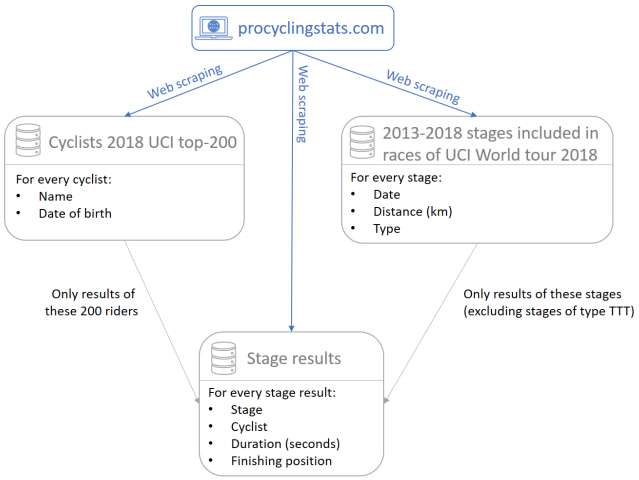


Fig. 1. Data collection process for cyclists, stages & stage results

weather forecasts based on latitude and longitude coordinates. However, because most stages contain thousands of coordinates, retrieving the weather forecast for every coordinate would be infeasible. It would also be quite pointless, as the weather will not really change every few kilometers. Therefore, the weather information was collected every 50km of a stage. Because the exact time riders passed a certain location is not publicly available, the collected weather information is the average during the daytime on that location. Examples of data gathered this way are: precipitation in millimeters, temperature in degrees Celsius, wind speed in kilometers per hour and wind direction in degrees.

III. Feature engineering

Many features were created using the collected raw data. First, features based on the total route and elevations such as the total distance, percent flat, percent up, percent up steep, percent down, average angle and average elevation were created. Some of these features were also multiplied with the distance to obtain features such as the meters up and meters down. These features describe important aspects of a route. However, one can image that a cyclist will have a different mean velocity if the finish of a stage is on a steep slope compared to a finish on a flat surface, even if the features for the whole stage route are the same. In order to take that into account those features were not only created for the whole stage, but also for the first, second, third and fourth quarter and for the final three kilometers of a stage. Finally, a boolean value that is true if a stage is an ITT was also added to the feature set. This resulted in a total of 133 features that characterize a route.

Secondly, features based on the weather were created. This includes straightforward features such as the average temperature, average pressure and average precipitation but also two more complex features: the average air density and the average wind influence.The air density is calculated using the temperature, humidity and pressure and expressed in kg/m^3. The wind influence is calculated by multiplying the wind velocity with the cosinus of the difference of the bearing (direction) of the cyclists and of the wind. Each one of the created features is multiplied by the distance causing a doubling of the amount of features. All these features are calculated for the whole stage route. The two wind influence features are also calculated for the first, second, third and fourth quarter and for the final three kilometres of a stage. This resulted in a total of 30 weather related features.

Thirdly, four features specific for each rider were engineered: the age of the rider and the time in minutes the rider has cycled one day, three days and seven days before the currently forecasted stage. These features will be tested in the forecasting of the difference between the average stage velocity and the velocity of a rider

Finally, extra features to use in the head-to-head forecasts were created. These include the total number of historical head-to-head wins between two riders as well as the head-to-head wins up until one year and up until ten days before the forecasted stage. All the stages from 2013 to 2018 that are included in the UCI World tour races of 2018 were taken into account. These are the stages from where the results were scraped. Also included were the head-to-head wins up until one year ago on flat stages and on hilly stages for all the collected stages of the Tour de France, Giro d'Italia and Vuelta d'España. A stage is considered to be flat if the percentage flat of the total stage is larger than or equal to 0.8 and hilly otherwise. From these features some derivative features such as the difference between the head-to-head wins of two riders were constructed. This resulted in a total of 19 features.

IV. Methodology

All the reported forecasting results make use of so-called rolling-origin cross validation. This means that each fold contains one observation to forecast and all the observations prior to that one are used as training data. Fig. 2 illustrates the concept. Every line represents a fold where the circles are the data samples of a certain stage. The samples are ordered chronologically from left to right, so each line can be considered a timeline. The blue dots represent the training set and the red dots the validation set. The white-gray circles are data samples that are not taken into account for that fold. The dataset is also split in a training set and a test set. Another something in common for all the forecasts is that features are first scaled before being passed to the machine learning algorithm. This is done using a standard scaler that standardizes features by removing the mean and scaling them to unit variance.

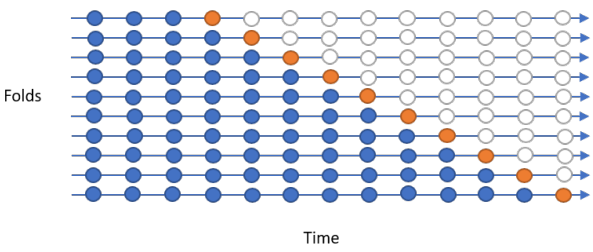


Fig. 2. Diagram illustrating concept of rolling-origin evaluations

The methodology followed for the prediction of the mean stage velocity and the difference between the average stage velocity and the velocity of a rider is identical. First, two baselines

are established: one using the mean of all the previous data samples as forecast and another one using the value of the nearest neighbor. The second baseline uses four features to obtain the nearest neighbor: the percentage up, flat and down of the whole stage route and a boolean indicating whether the stage is of type ITT. Thereafter, a forward selection and backward elimination procedure is performed with different machine learning regression algorithms. This results in different combinations of algorithms and features; each obtains a different mean absolute error (MAE). The influence of recency weighting is then explored on the algorithm(s) that obtained the lowest MAE. This means that more recent data samples get more weight in the forecast by duplicating them a certain amount of times in the training set. Finally, the hyperparameters of the machine learning algorithm are tuned. This is done using a grid search. For both types of forecasting the stages between 2016 and 2018 are predicted.

The methodology for the head-to-head forecasts also starts with establishing a baseline; the rider with the most head-to-head wins up until a year ago in the flat or hilly stages of the three Grand Tours is forecasted as the winner. Depending on whether the forecasted stage is flat or hilly, the head-to-head wins on the flat or hilly stages will be used. Thereafter, different classification algorithms are tried with the features rider age, percentage up, flat and down and stage type is ITT and the features based on the historical-head-to-head wins. The same algorithms are tried again with the same features, but this time also features based on the forecasted difference between the average stage velocity and the velocity of a rider are included. This is done to test if adding those features can improve the head-to-head predictions. The stages of 2017 and 2018 are forecasted. This ensures that the first stage of 2017 already has the 2016 forecasted differences as training data.

V. Results

In this section the results of the three types of forecasting are summarized.

A. Mean stage velocity

The gradient boosting algorithm combined with recency weighting of up to 2 days ago obtains the best result. The MAE for the training set is 0.3758 m/s (with 95% CI: 0.3198 m/s - 0.4362 m/s). The test set however only gets an MAE of 0.4660 m/s (95% CI: 0.3400-0.6014). Table I shows the MAE and CI for the two baselines and gradient boosting for both the training set and the test set. Compared to the nearest neighbor baseline, an improvement for the test set of 0.4167 m/s is obtained (95% CI: 0.1810 m/s - 0.6628 m/s). If we look at the predictions for the total duration in seconds for the stages, we get a mean absolute percentage error (MAPE) of 3.49% (95% CI: 2.97% - 4.06%) for the training set and 4.06% (95% CI: 2.97% - 5.26%) for the test set.

B. Individual rider velocity

The ridge regression algorithm combined with recency weighting of up to 90 days ago obtains the best results. The resulting MAE for the difference between mean stage velocity and rider velocity for all 200 riders is 0.1560 m/s (95% CI: 0.1517 m/s - 0.1603 m/s) for the training set and 0.1576 m/s (95% CI: 0.1481-0.1673) for the test set. This is an improvement of 0.0394 m/s (95% CI: 0.0262 m/s - 0.0526 m/s) compared to the best of the mean baseline. Table II shows the results for the two baselines and ridge regression.

	Training set		
	CI lower	**MAE**	**CI upper**
Mean baseline	1.0156	1.1314	1.2540
Nearest neighbor baseline	0.5616	0.6360	0.7135
Gradient boosting	0.3198	0.3758	0.4362
	Test set		
	CI lower	**MAE**	**CI upper**
Mean baseline	0.6659	0.9194	1.1975
Nearest neighbor baseline	0.6904	0.8827	1.0944
Gradient boosting	0.3400	0.4660	0.6014

TABLE I
MEAN STAGE VELOCITY FORECASTING RESULTS

	Training set		
	CI lower	**MAE**	**CI upper**
Mean baseline	0.2016	0.2059	0.2103
Nearest neighbor baseline	0.2207	0.2265	0.2324
Ridge regression	0.1517	0.1560	0.1603
	Test set		
	CI lower	**MAE**	**CI upper**
Mean baseline	0.1880	0.1970	0.2063
Nearest neighbor baseline	0.1890	0.2005	0.2123
Ridge regression	0.1481	0.1576	0.1673

TABLE II
MEAN STAGE VELOCITY MINUS RIDER VELOCITY FORECASTING RESULTS

C. Head-to-head wins

Gradient boosting obtains the best results. When the individual rider predictions are not included the mean stage accuracy is 71.35% (95% CI: 69.84% - 72.86%) which is an improvement over the baseline. Adding the predicted difference features an accuracy of 73.16% (95% CI: 71.58% - 74.72%) was obtained. This means an accuracy improvement of 1.81%. This improvement varies, at the 95% confidence level, from -0.36%, a small decrease, to 3.97%, a substantial increase.

VI. Conclusion & future work

Machine learning was used in order to predict road cycling results. The focus was put on the stages of the Giro d'Italia, Vuelta a España and Tour de France from 2016 to 2018. Different combinations of machine learning algorithms, features and recency weighting were tested in order to predict the mean velocity of a stage, the mean stage velocity minus the velocity of a rider and the head-to-head wins between two riders in a stage. The baselines for all three forecast types were improved significantly. Therefore, it has been demonstrated that machine learning in combination with public data can be used to predict cycling results with a reasonable accuracy. However, the beauty of cycling, as with many other sports, is that it remains

very unpredictable. Future research could explore other stages, machine learning techniques and features. It is expected that advanced physics and expert domain knowledge will lead to better features.

REFERENCES

[1] T. S. Olds, K. I. Norton, E. L. A. Lowe, S. Olive, F. Reay, and S. Ly, "MODELING ROAD-CYCLING PERFORMANCE", *Journal of Applied Physiology*, vol. 78, no. 4, pp. 1596–1611, 1995.

[2] J. C. Martin, D. L. Milliken, J. E. Cobb, K. L. McFadden, and A. R. Coggan, "Validation of a mathematical model for road cycling power", *Journal of Applied Biomechanics*, vol. 14, no. 3, pp. 276–291, 1998.

[3] Y. Kataoka and P. Gray, "Real-time power performance prediction in tour de France", in *CEUR Workshop Proceedings*, vol. 2284, 2018, pp. 127–136.

[4] A. Pezzoli, E. Cristofori, M. Moncalero, F. Giacometto, and A. Boscolo, *Effect of the Environment on the Sport Performance*, P. P. Correia, J. Barreiros, and J. Cabri, Eds. 2014, pp. 167–170.

[5] B. L. Hannas and J. E. Goff, "Model of the 2003 Tour de France", 2004.

[6] J. E. Goff, "Predicting winning times for stages of the 2011 tour de france using an inclined-plane model", in *Procedia Engineering*, ser. Procedia Engineering, P. Drane and J. Sherwood, Eds., vol. 34, 2012, pp. 670–675.

[7] B. Lee Hannas and J. E. Goff, "Inclined-plane model of the 2004 Tour de France", *European Journal of Physics*, vol. 26, no. 2, pp. 251–259, 2005.

[8] B. A. Ramsey and J. E. Goff, "Predicting Tour de France stage-winning times with continuous power and drag area models and high speeds in 2013", *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 228, no. 2, pp. 125–135, 2014.

[9] C. M. Hobson and J. E. Goff, "Improving Tour de France modeling with allometric scaling", *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 229, no. 3, pp. 183–191, 2015.

[10] C. M. Hobson and J. E. Goff, "Tour de France Modeling: 2015 Results and Comparisons with Elite Cyclist Power Data", in *Procedia Engineering*, vol. 147, 2016, pp. 607–612.

[11] C. M. Hobson and J. E. Goff, "Using the 2011-16 Tours de France to refine prediction model and elicit racing strategies", *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 231, no. 3, pp. 232–240, 2017.

[12] Procyclingstats, [Online]. Available: `https://www.procyclingstats.com/` (visited on 01/21/2019).

[13] Velowire, [Online]. Available: `https://www.velowire.com/blogcat/35/en/open-street-maps-google-maps-google-earth.html` (visited on 02/10/2019).

[14] Radtoto, [Online]. Available: `http://www.radtoto.com/index.php?na=3100` (visited on 02/13/2019).

[15] Gpsies, [Online]. Available: `https://www.gpsies.com/` (visited on 02/04/2019).

[16] Ridewiththegps.com, [Online]. Available: `https://ridewithgps.com/` (visited on 02/04/2019).

[17] Google elevation api, [Online]. Available: `https://developers.google.com/maps/documentation/elevation/start`.

[18] World weather online, [Online]. Available: `https://www.worldweatheronline.com/developer/api/historical-weather-api.aspx`.

Contents

1. Used abbreviations

TTT	Team Time-Trial
ITT	Individual Time-Trial
MAPE	Mean Absolute Percentage Error
MAE	Mean Absolute Error
CI	Confidence Interval
OLS	Ordinary Least Squares
BN	Bayesian Network
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
ANN	Artificial Neural Network
MLP	Multilayer Perceptron

2. Introduction

Humans have always been fascinated by the future. From the reading of bird intestines to empirically proven physics models, the methods to predict this future have greatly improved over time. With the rise of computers, machine learning techniques and the world wide web we have created powerful new prediction tools. In this thesis, I make use of these tools to predict the results of professional road cycling competitions.

2.1 Literature review

In this literature review, I will first identify and synthesize the relevant literature within the field of forecasting sport results using machine learning. Thereafter, I will show that predicting cycling results using public data and machine learning is still an unexplored area.

2.1.1 Predicting sport results

A lot of research has already been done on the prediction of sport results for two-contestant sport games such as football, association football (soccer) and tennis. Already in 1980 Harville [1] predicted the differences in scores of National Football League games using linear models based on measures of team strength and the home-field advantage. These team strengths were found using a maximum likelihood procedure. This method resulted in an average absolute error that was slightly larger than the bookmaker predictions. In 1982, Maher [2] used a Poisson model and measures for the attack and defence of a team, taking into account home and away matches, to predict association football scores. Again, maximum likelihood was used to estimate these attack and defence measures. Further improvements to this method were tried by Dixon and Coles [3] and, together with a good betting strategy, they were able to yield a positive return on investment. The home advantage and some measure of team strength were also used in 1993 to predict major league baseball in the United States [4]. However, this time, strengths were allowed to change after every game. The outcomes of future games were predicted using Markov chain sampling. With the two most used methods to predict association football being modelling win–draw–lose match results directly and modelling the goals scored and conceded by each team, Goddard [5] compared the forecasting performance of these two methods. For the

first one he used ordered probit regression and for the second one bivariate Poisson regression. He concluded that both methods gave similar results, but that "the best forecasting performance is achieved using a hybrid specification, combining a results-based dependent variable with goals-based lagged performance covariates". Joseph, Fenton and Neil [6] compared a Bayesian Network (BN) for association football created by a domain expert with four machine learning algorithms: MC4 (a decision tree learner), a Naive Bayesian learner, Data Driven Bayesian and a K-nearest neighbour learner. They concluded, using a small dataset of 76 matches, that the expert BN had the best predictive accuracy. However, when comparing 48 experts and 31 statistical systems forecasting American football winners, Song, Boulier and Stekler [7] did not find a significant difference in prediction accuracy between expert and statistical systems. The bookmaker odds on the other hand substantially outperformed both systems. This seems to indicate that a combination of statistical models and expert domain knowledge can generate better forecasts than either one of those methods alone.

Throughout the years, many other related and novel techniques were tried to create the best forecasting models for two-contestant sport games. Constantinou, Fenton and Neil [8] used a BN model with both objective and subjective data to generate a profit from betting on the English Premier League. Boshnakov, Kharrat and McHale [9] also generated a profit using a bivariate Weibull count model combined with a Kelly-like betting strategy. Berrar, Lopes and Dubitzky [10] won the 2017 Soccer Prediction Challenge using k-nearest neighbour and two new methods for constructing predictive features from soccer match data: recency feature extraction and rating feature learning. A hierarchical Markov model [11] and Bradley-Terry type model [12] have been used to generate a positive return on investment for tennis betting. Cricket games have been predicted with an accuracy of 75% and positive betting profit using Twitter posts and a Support Vector Machine (SVM) [13]. Kaur and Jain [14] combine a fuzzy approach and SVM technique and Neudorfer and Rosset [15] combine the least squares pairwise comparison model and isotonic regression to predict basketball results with a reasonable accuracy.

Another type of sports, besides two-contestant games, are multi-contestant races. This includes sports such as swimming, horse racing, running and cycling. Less research exists in this area compared to two-contestant games. Edelmann-Nusser, Hohmann and Henneberg [16] used a neural net to predict the performance of a swimmer in the finals of the 200 meters backstroke at the Olympic Games of 2000 in Sydney. Specific, private, information about the training schedule of the swimmer was used. The error of the prediction was only 0.05 seconds. This

demonstrates that neural nets could be useful to forecast races. Davoodi and Khanteymoori [17] also used neural nets to forecast the finishing times of horses in horse races. These predictions were used to forecast the finishing positions of the horses with an average accuracy of 77%. This time, only publicly accessible information was used. Chapters 5 and 6 will use a similar method; the head-to-head results will be forecasted using the predicted individual rider finishing times. Neural nets will be one of the algorithms tried to create those forecasts. Linear methods have also been proven effective in the prediction of race performances. Przednowek, Iskra and Cieszkowski [18] tested ordinary least squares (OLS), ridge and lasso linear regression to predict the results of 500-meter sprint races. Features were created from private training data of the athletes. Ridge and lasso regression outperformed OLS regression, with lasso achieving slightly better results than ridge. Linear models combined with training data of athletes were also used in order to forecast race walking performances [19]. The algorithms tried were OLS, ridge, lasso and elastic net regression. Again, ridge and lasso outperformed OLS regression with lasso achieving the best results. The main purpose of lasso regression is to eliminate the least important features and it is therefore suitable as a feature selection algorithm. However, this thesis will use forward selection and backward elimination to select the best features. Therefore, lasso regression will not be used. OLS and ridge regression on the other hand will be tried as forecasting algorithms in chapter 5. Ruiz-Mayo, Pulido and Martínez-Muñoz [20] compared linear regression and non-linear bagging with regression trees for the forecasting of marathon performances. Features are created from training data of the athletes. The non-linear bagging method provided the best result with a mean absolute error (MAE) of seven minutes. Chapter 5 will test the use of regression trees and bagging in order to predict the cycling performance.

2.1.2 Predicting cycling results

Before discussing literature about the prediction of cycling results an important distinction has to be made between the words *cycling race* and *cycling stage*. Both are used in this thesis, but mean something different. A cycling race is an abstract entity that combines different stages under some common name and theme. Examples of a cycling race are the Tour of Flanders and Vuelta a España. A cycling stage is a concrete contest with a concrete route on a certain date. For example the Tour of Flanders 2017 or the 6^{th} stage of the Vuelta a España of 2015.

Road cycling has properties that make it somewhat different to predict compared to two-contestant games and even to other multi-contestant races. The most important one is that the influence of

the route is huge. A mainly flat stage is expected to have a totally different top ten compared to a stage with a lot of steep hills. Every stage has a unique combination of length, slopes and surfaces (e.g. asphalt with or without pebbles, cobblestones). A second property that makes cycling special is the combination of teams and individuals. Teamwork is important, but only one person can win (except in team time trials). This can give rise to an interesting dynamic between team members. In this subsection I look at literature that is specifically about cycling.

Olds et al. [21] and Martin et al. [22] presented and validated mathematical models for road-cycling performance. They took into account aerodynamic drag, rolling resistance, friction in the bearings and chain drive system, and changes in kinetic and potential energy. Martin et al. calculated predictions of the cyclists power outputs that were close to the measured power outputs. The power output of a rider is a measure of how hard the rider has to work and is measured in watts (W). However, they only validated their mathematical models on a short and rather flat route with a concrete surface, which are not real stage situations. Kataoka and Gray [23] recently used a data-driven approach in order to predict the power output of professional riders in real-time. Using live GPS data, hand-crafted feature engineering using physics knowledge, automatic feature generation using autoencoder and a random forest algorithm, they were able to reduce the MAE by 56.79% (from 139.17W to 60.13W) compared to a model using only physics. According to Pezzoli et al. [24] the most influential meteorological variables for cycling performance are air temperature, wind, rain and air humidity. However, this conclusion was based upon qualitative research with focus groups, so should not be considered proven.

The research of Goff et al. [25]–[31] used inclined planes, physics and mathematical models to predict the stage-winning times of the Tour de France, from 2003 up until 2016. Included in the models were required power, drag and friction. Factors such as weather, team strategies, crashes and sharp turns were not taken into account. For the 2015 Tour de France, they predicted 12 stages below 5% mean absolute relative error, seven of which were below 1%. Only one of the predictions had an error worse than 10%. For the 2016 Tour de France they predicted 14 stages below 5% error, four of which were below 1% error.

2.2 Goal

Two types of road cycling results are very interesting to predict: the mean velocity, both of individual riders and the average for a stage, and the head-to-head wins. Head-to-head wins mean

predicting which one of two riders in a certain stage will be the first one to finish. Predicting these road cycling results with a reasonable accuracy is important for the following groups:

- **betting agencies:** accurate forecasts allow them to maximize their profit
- **competition organizers:** having an estimate how long the stage will take allows better planning
- **coaches:** they can adjust their strategy or training schedule in view of the forecast for the next game

As seen in the previous section, a lot of research exists about using public data to predict sports such as association football, tennis and basketball. However, road cycling has properties that make predicting its results a harder challenge. To the best of my knowledge, no studies are yet available in the literature about the forecasting of road cycling results using public data and machine learning.

The goal is thus to predict the results of professional road cycling competitions using public data and machine learning in order to help competition organizers, betting agencies and coaches and to have a baseline for potential future research. More specifically, I will look at the results of the top 200 cyclists on the 2018 Individual UCI World ranking for men in the three so-called Grand Tours: the Vuelta d'España, Giro d'Italia and Tour de France. These stages were chosen because the Grand Tours are the most important (according to the UCI World ranking) cycling competitions and, as such, the most interesting to predict. They are also somewhat similar with every Grand Tour consisting of 21 stages with at least a few mountainous ones. I will predict the average velocity of a stage, use this to predict the velocity of every cyclist individually and use those results in order to predict the head-to-head wins between two riders in a stage.

2.3 Thesis outline

Chapter 3 describes the data collection phase. It explains what data were gathered from where. It is divided in three parts with every part exploring a different data source: cyclists and stages, stage routes and weather.

Chapter 4 gives an overview of the different techniques related to machine learning used in this thesis. It starts with the choice of programming language and libraries. Thereafter, a short

summary of the used machine learning algorithms is provided. Then follows an explanation of how cross-validation, training-test set splitting and feature selection are handled in this thesis. Finally, the method to calculate confidence intervals (CIs) for the mean error and accuracy of machine learning algorithms is explained.

Chapter 5 handles everything about predicting the mean stage and individual rider velocity. It starts with the features that are engineered from the raw data collected in chapter 3. Thereafter, the process of the prediction of the mean stage velocity is described followed by the prediction of the individual rider velocity. Both sections use the same process; first, baselines are created, then a combination of machine learning algorithm and features is chosen, thereafter, some possible optimizations are examined and finally, the results are discussed.

Chapter 6 describes how the prediction of head-to-head results was handled. The main focus of this chapter is to see if the individual rider velocity predicted in chapter 5 could help in predicting the head-to-head wins.

Chapter 7 gives a brief summary of the main results and some suggestions for future work.

3. Data gathering

Without data, there is no machine learning. In this chapter, I explain what data I gathered, what the sources of the data are and how I gathered those data.

3.1 Cyclists & stages

To start predicting the results of cycling competitions at least some basic information about stages, cyclists and end results of cyclists in stages is needed. This information can be found on the website procyclingstats.com [32]. This website does not offer a public API, so I used *Python 3.6* and the *BeautifulSoup* library to scrape the data from the website. All the scraped data were then stored in a local MySQL database.

First, the top 200 cyclists on the Individual UCI World ranking for men at the end of 2018 were scraped. This ranking is a 52-week ranking that is updated every week where riders get points for their results in stages and tours [33]. The Tour de France awards the most points followed by the Vuelta d'España and Giro d'Italia. The reason why I chose to only scrape these 200 cyclists is twofold. First, 200 cyclists are enough to gather reliable statistics while remaining computationally feasible. Secondly, the best riders are the most interesting to predict. The Individual UCI World ranking seemed like a valid way to select those top riders.

Thereafter, all the stages of the Vuelta a España, Giro d'Italia and Tour de France from 2013 to 2018 were scraped and stored. In chapter 6, riders will be compared in these Grand Tours stages. In order to compare two riders it would be interesting to collect their results not only in the Grand Tours but in all the important stages. A stage is considered to be important if it is part of one of the races included in the UCI World Tour 2018. The Vuelta, Giro and Tour are included in these races. Therefore, all the stages from the 2013-2018 editions of those races were scraped and stored. For every stage the type (e.g. individual time-trial (ITT)), distance in kilometers and the date were retrieved. Finally, for every one of the 200 saved riders the time duration and finish position of these stages were saved. Only the team time-trial (TTT) stages were not scraped for their rider results because those stages are rather rare and very different from the normal mass-start and ITT stages. Fig. 3.1 gives a schematic overview of this data

collection process.

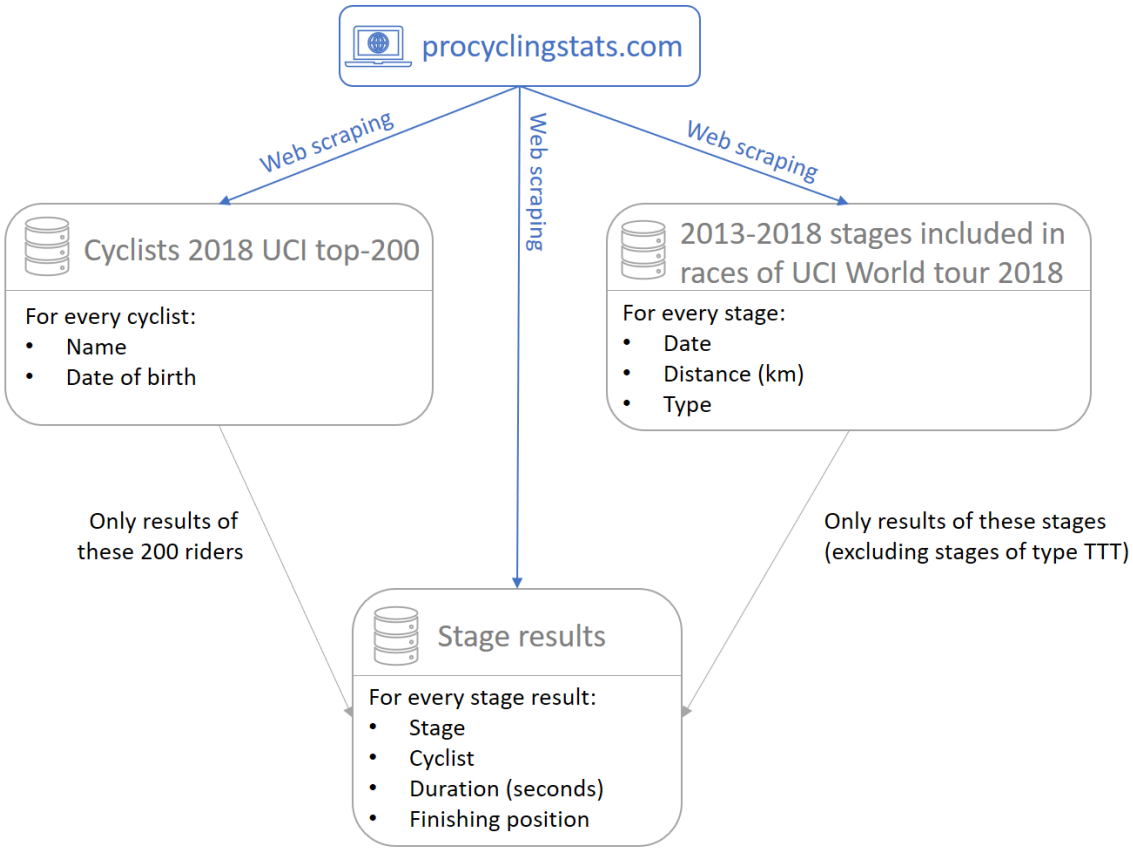


Figure 3.1: Data collection process for cyclists & stages

3.2 Routes

Of course, only the distance of a route and the historical results of a rider are not enough to predict future cycling results. One of the reasons predicting cycling results is so different from e.g. tennis or soccer is that many stages have a completely different route and this has a major impact on the results. More detailed info about the routes is thus needed.

Ideally, for every stage there would be a list of latitude and longitude coordinates of the route together with the elevation for each of these coordinates. However, these coordinates were not available in a structured manner that allowed automatic collection. Therefore, that information had to be retrieved manually. Fortunately, cycling is a popular sport and fans all around the world like to know the exact route of the stages of some of the major cycling races. That way they can view the route beforehand using e.g. Google Earth or they can ride the route themselves in their free time. This has resulted in websites that allow cyclists to upload route coordinates or have uploaded some routes themselves. These routes can then be freely downloaded. The

routes of the stages from 2013 up until 2018 of the three Grand Tours were collected.

A lot of the routes come from velowire.com [34]. This is a blog dedicated to the cycling sport that uploads stage routes for some of the major cycling races. The routes collected from there are:

- Tour de France 2013-14-15-16-17-18
- Giro d'Italia 2013-14-15-16-17
- Vuelta a España 2013-14

For the missing routes, I used the following websites:

- radtoto.com: Giro d'Italia 2018, Vuelta a España 2018 [35]
- gpsies.com: Vuelta a España 2015-16-18 [36]
- ridewithgps.com: Vuelta a España 2017 [37]

Every route was visually compared with the map of the stage on procyclingstats.com to make sure that it was more or less the same. Small and isolated deviations from the exact stage route could have gone undetected. However, the features created with the help of those coordinates will be based on the elevations and weather. Neither changes considerably over a few meters. Also, those features will always be an average over some part of the stage. Therefore, small and isolated deviations should not have a significant influence on the forecasts.

The next step was to gather the elevation for every collected coordinate in the routes. This could be done automatically thanks to the Google Elevation API [38]. This API allows to upload coordinates and retrieve the elevation in meters. Using Python, all of the coordinates from the routes were run through this API and saved in the database.

3.3 Weather

Another possible influence on the cycling results is the weather. One can imagine that a rider that rides the same route twice, but the first time with tailwinds, a temperature of 15°C and not a single drop of rain will be faster than a second time when there are headwinds, a temperature of 25°C and heavy rain. Therefore it is important to consider the weather in the predictions.

One caveat when collecting weather data is that not the real weather conditions should be retrieved, but the weather forecasts. When making a prediction, the real weather conditions are not known beforehand; only the forecasts are known. The API used to gather this information is the one from World Weather Online [39]. This API provides historical weather forecasts based on latitude and longitude coordinates. However, because most stages contain thousands of coordinates, retrieving the weather forecast for every coordinate would be infeasible. It would also be quite pointless, as the weather will not really change every few kilometers. Therefore, I decided to collect the weather information every 50km of a stage. That seemed like a good trade-off between weather accuracy and the duration of retrieving these data. The exact time riders passed a certain location is not publicly available. Therefore, the collected weather information is the average during the daytime on that location. The data that was gathered this way was:

- cloud cover in percentage (%)
- pressure in millibar (mb)
- visibility in kilometer (km)
- humidity in percentage (%)
- precipitation in millimeter (mm)
- wind speed in kilometers per hour
- wind direction in degrees (°)
- temperature in degrees Celsius (°C)
- feeling temperature in degrees Celsius (°C)
- UV index

4. Machine learning techniques

This chapter gives a short overview of the libraries and techniques related to machine learning that will be used in the remainder of this thesis.

4.1 Programming language & libraries

All the code written for this thesis uses Python 3 as programming language. There are two reasons for this choice. The first one is that Python is a high-level, interpreted and dynamically-typed language. This allows easy and quick programming without the need to indicate every type beforehand or wait for the program to compile. The second, main, reason is that Python is one of the most used programming languages for machine learning [40]. This has resulted in a large amount of libraries and online information that can be used freely. The four most important libraries used in this thesis are:

- **NumPy** allows efficient handling of and mathematical operations on large multidimensional arrays [41]. It is, among other things, used to calculate the CIs with bootstrapping. Data passed to the machine learning algorithms is always first converted to a NumPy array.
- **Pandas** provides data structures and operations for highly optimized manipulation and analysis of data [42]. It uses NumPy under the hood. It is used in this thesis to store the data samples as an in-memory table, a so-called DataFrame, and calculate new feature columns derived from other columns. A DataFrame can easily be converted to a NumPy array in order to pass it on to the machine learning algorithms.
- **SciPy** is a Python-based ecosystem of open-source software for scientific and technical computing [43]. It includes Pandas and NumPy as its core packages. In addition to Pandas and NumPy, SciPy is also used to create plots and to power many of the machine learning algorithms.
- **Scikit-learn** is a machine learning library build on top of SciPy [44]. It features algorithms for classification, regression and clustering as well as many other machine learning related functions. All the machine learning algorithms used in this thesis are implemented by

Scikit-learn.

4.2 Algorithms

This thesis uses six algorithms. Each of these algorithms is briefly discussed below.

4.2.1 Linear & ridge regression

Linear regression models the relationship between one or more explanatory variables and a scalar response variable in a linear way. More concretely, assume a data set of n vectors where every vector is of the form $\{x_1, x_2, ..., x_j, y\}$ where $\{x_1, x_2, ..., x_j\}$ are the j explanatory variables and y is the response variable. Linear regression assumes the relationship to be of the form $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_j x_j + \epsilon$, where ϵ is called the residual or error term and β_0 the intercept. The part of the right-hand side without ϵ is denoted as $\hat{y}$ and is the value predicted by the linear function. The values for $\beta_0, \beta_1, ..., \beta_j$ can be estimated using many different methods. In this thesis, OLS is used. This method chooses the betas in such a way that the sum of the squares of all the error terms ($\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$) is minimized. The main advantage of linear regression is that it is a very simple and intuitive method with a limited amount of hyperparameters to tune. It creates easy interpretable models. One only has to look at the beta coefficient of a feature to understand if this feature has a positive or negative effect on the response variable and the magnitude of that effect. Linear regression has the main disadvantage of assuming that all features are linearly correlated with the response variable. It is also sensitive to multicollinearity.

Ridge regression is a so-called regularization method for the linear regression model. It tries to keep the betas small and thus obtain a simpler model. In order to keep those betas small it penalises the loss function (OLS). When using ridge regression, the loss function to minimize becomes
$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^{n}(\beta_{i1}^2 + \beta_{i2}^2 + ... + \beta_{ij}^2)$
This is nothing more than the OLS loss function plus a penalty depending on the size of the betas. The most important hyperparameter to tune is α. Depending on the size of α the size of the betas will be larger or smaller. The main advantage of ridge regression compared to OLS is that ridge regression can handle multicollinearity better [45], [46].

4.2.2 K-nearest neighbors

K-nearest neighbors (KNN) is one of the simplest machine learning techniques. Assume we want to predict a response variable y using a vector of explanatory variables $\{x_1, x_2, ..., x_j\}$. KNN looks at the K closest neighbors of explanatory variables in de dataset. If we are dealing with a classification task, the K neighbors will vote for their own response variable. For a regression task some kind of averaging is used between the values of the K neighbors. It is possible to take the distance between the explanatory variables of a neighbor and $\{x_1, x_2, ..., x_j\}$ into account for the voting or averaging. There are many possible distance metrics, but the most used one is the Euclidean distance. Assume a neighbor with explanatory variables $\{z_1, z_2, ..., z_j\}$, then the Euclidean distance is $\sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + ... + (x_j - z_j)^2}$. KNN has the advantage of being a simple and intuitive algorithm with no assumptions on the underlying data distribution. There are only two important parameters to choose: the number of neighbors to take into account and the distance metric. On the other hand, KNN is sensitive to noise and overfitting. The performance scales with the number of data samples so, when the dataset is huge, performance can be an issue [47], [48].

4.2.3 Random forest

Random forest is an ensemble method that can be used for both classification and regression. It consists of many base predictors, almost always decision trees, that together vote for the final prediction. This is called bagging: many independent predictors are combined using some averaging technique. Random forests using decision trees can model complex non-linear relationships. In a normal decision tree, rules will be created based on all the observations and features. However, the decision trees in a random forest are build from randomly selected observations and features. By combining many of these randomly build decision trees the prediction error as well as the chances of overfitting can be reduced. This on the other hand increases the complexity of the model and decreases the performance. The most important hyperparameters are the number of decision trees used and hyperparameters for the individual trees such as the maximum depth of a tree [47], [49].

4.2.4 Gradient boosting

Gradient boosting is also an ensemble method, often based on decision trees, that can be used for both classification and regression. Instead of bagging it uses boosting. Boosting is an

ensemble technique where the base predictors are not combined independently but sequentially. Subsequent predictors learn from the mistakes of previous ones and try to reduce the error. A simplified high-level summary of the gradient boosting algorithm would be: fit a (simple) predictor on the data, then look at the error residuals and thereafter fit a new predictor on the error residuals. Keep fitting new predictors on the new error residuals until a condition (e.g. overfitting) is met. In this thesis, decision trees will be used as the base predictors. Gradient boosting has shown considerable success in various applications. However, this comes at the cost of a complex model that can take some time to train and to evaluate. The model can also suffer from overfitting if the number of base predictors is too high. Besides lowering the amount of base predictors, it is also possible to shrink the contribution of each base predictor. This method can result in better generalization. The shrinking factor is sometimes called the learning rate. The most important hyperparameters for gradient boosting are: the number of decision trees used, the learning rate and hyperparameters for the individual trees [49], [50].

4.2.5 Multilayer perceptron

A multilayer perceptron (MLP) is one of the simplest and most used types of artificial neural networks (ANNs). An ANN is a type of model loosely based on the human brain. It consists of a collection of connected nodes, also called artificial neurons. These nodes can have multiple inputs and multiple outputs that are connected with other nodes. Every node contains a threshold and a function that maps the inputs and a discrete time parameter to the outputs. Each connection is also assigned a weight. Based on these weights and the threshold, a node can decide to ignore the inputs. In order to let a neural network 'learn', typically the weight and threshold values are adjusted in order to fit the data. Often such an ANN is structured in layers. The nodes of a layer can only connect to the nodes of the next layer. The first layer, where the data is inserted, is called the input layer; the middle layers are called hidden layers and the final layer is called the output layer. An MLP is such a layered ANN that consists of an input layer, one ore more hidden layers and an output layer. MLPs are capable of modelling very complex functions and, if they have enough training data, they can handle irrelevant features and noise. The disadvantages are that they take long to train, are sensitive for the choice of hyperparameters, need a lot of training data and are hard to interpret. The main hyperparameters for MLPs are the number of hidden layers and artificial neurons in every hidden layer as well as the algorithm that determines the weights of the connections [47], [51].

4.3 Cross-validation

When testing different features, machine learning algorithms and hyperparameters it is important to have a reliable method of comparing the errors with these different choices. If the ultimate goal is to make predictions for new, unseen data then the different choices should be compared using the error on data the machine learning algorithm has never seen before. One way to achieve this is to split the dataset in three non-overlapping parts: the training, validation and test set. The training set is used to train the algorithm, the validation set to calculate an intermediate error in order to compare different choices and the test set to calculate the final error and make sure at the very end of the machine learning process that the chosen combination of features, machine learning algorithm and hyperparameters obtains decent results. However, for small datasets it is often not feasible to split it in three large enough representative parts. A way to overcome this problem is cross-validation. Instead of using three parts, only a training and test dataset are used. When calculating the error to compare different choices, the training set is split into K parts. For every part, the error is calculated with the part as validation set and the rest as training set. One such a validation-training set combination is called a fold. In the end, the average error of the K folds is considered to be the overall error. If K equals the amount of data in the training set, then it is called leave-one-out cross-validation.

In this thesis, however, normal cross-validation is not a good way to measure the prediction error. Data samples are time dependent. It would not be correct to predict the results of the Tour de France of 2014 using the results of 2015. Therefore this thesis uses so-called rolling-origin evaluations [52]. This means that each fold contains observations from the same point in time as validation set and all the observations prior to that one are considered to be training set. Fig. 4.1 illustrates the concept. Every line represents a fold where the circles are the data samples of a certain stage. The samples are ordered chronologically from left to right, so each line can be considered a timeline. The blue dots represent the training set and the red dots the validation set. The white-gray circles are data samples that are not taken into account for that fold. In this thesis, unless explicitly mentioned otherwise, whenever cross-validation is used it will be rolling-origin evaluations.

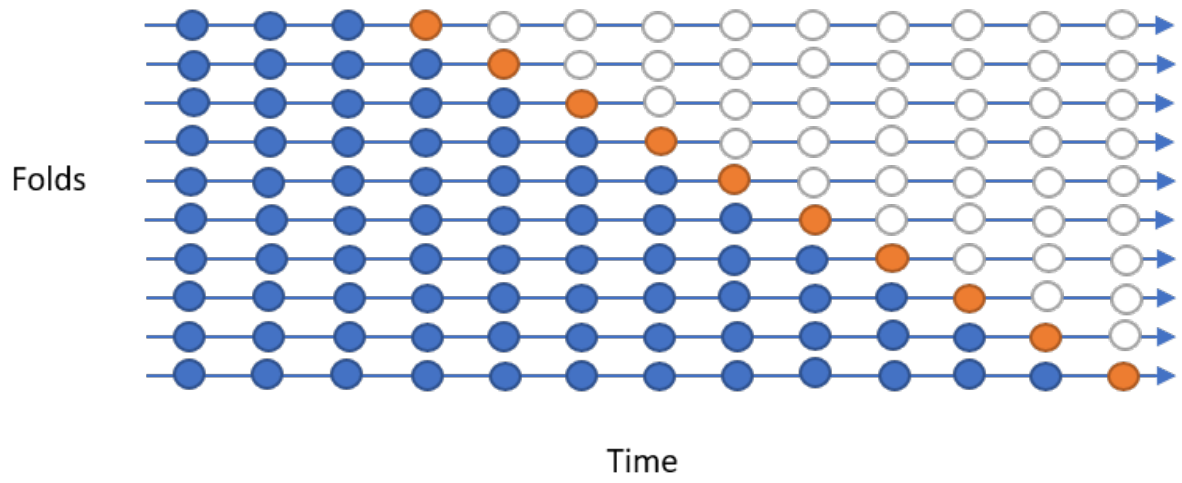


Figure 4.1: Diagram illustrating concept of rolling-origin evaluations

4.4 Training & test set

As mentioned in section 4.3, it is important to have a training and test set. At the very end of the machine learning process, the test set makes sure that the selected combination of features, machine learning models and hyperparameters that performed well on the training set were not a coincidence. The test set in this thesis consists of three randomly selected stages for each Grand Tour each year, from 2016 up until 2018. This means that the test set contains 27 stages. Table 4.1 shows for every year and Grand Tour the stages in the test set.

	2016	**2017**	**2018**
Giro d'Italia	1, 2, 11	4, 15, 20	3, 6, 10
Tour de France	2, 6, 14	5, 19, 20	4, 15, 18
Vuelta a España	5, 9, 11	2, 13, 19	12, 14, 19

Table 4.1: The numbers of the stages in the test set

4.5 Forward selection & backward elimination

In chapter 5, there will be many features for a rather small set of data. This means that some sort of feature selection procedure will have to be applied. The two methods used are forward selection and backward elimination. Forward selection starts with zero features, tests the addition of one feature from all the available features and adds the feature that obtains the best result to the model. This process is repeated with more and more features added until a certain stop criterion. The test criterion used to compare the addition of different features will, in this thesis,

be the mean absolute prediction error obtained using cross validation. The forward regression will stop when none of the remaining features lowers that error. Backward elimination is a very similar process, but instead starts with all the features included in the model and tries to remove features one-by-one. In each iteration, the feature that lowers the mean absolute prediction error the most will be removed. There will be no stop criterion. Features will keep on being removed until only one feature is left. For every number of features, the corresponding feature set and mean absolute prediction error will be retained. The feature set that in the whole process resulted in the lowest error will be the final set.

4.6 Bootstrapping

Bootstrapping is a technique to estimate statistics on a population that relies on random sampling with replacement [53]. The greatest advantages of the method are its simplicity and generality. It will be used in this thesis to calculate the 95% confidence intervals (CI) for the mean of the results of a machine learning model. The technique works by taking samples from a large dataset. Every sample contains the same amount of observations as the large dataset. However, the observations in the sample are randomly chosen from the large dataset and a chosen observation is not removed from the dataset so can be chosen again and again. The mean of every sample is calculated and added to a list of means. To calculate the 95% CI, the 2.5 and 97.5 percentiles of this list are taken. In this thesis 100 000 bootstrap samples are used to calculate a CI.

5. Predicting velocity

In this chapter I will explain how I created features, selected a relevant subset of those features and used that subset to forecast the mean duration for a stage and the duration of the individual riders. All the reported error values are calculated using cross-validation.

5.1 Feature engineering

Using the raw data gathered in chapter 3 it is possible to create many features. This section describes the types of features that have been engineered based on the route information, weather predictions and individual rider information.

5.1.1 Route

With coordinates and associated elevations, the distance and angle between two coordinates can be calculated. The distance between two points on the earth is curved and can be calculated using WGS-84 [54]. However, because the distances are very small, the error when using the Cartesian coordinate system and straight lines would be negligible. This means that the latitude/longitude coordinates have to be converted to Cartesian earth-centered, earth-fixed (ECEF) coordinates. This can be done using the Pymap3D Python library [55]. The result is two points (x_1, y_1, z_1) and (x_2, y_2, z_2) on the surface of the earth. The straight distance between these two points can be calculated using $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$. However, these two points are on the theoretical surface of the ellipsoid earth, where the elevation due to mountains and hills is not yet taken into account. The elevation e is included using $d_{with\_elevation} = \sqrt{d^2 + (e_2 - e_1)^2}$. The angle α in radians can then be calculated using the distance d and elevation e by means of $\alpha = \arctan \frac{e_2 - e_1}{d}$. Thereafter the angle is converted from radians to degrees. Figure 5.1 gives a schematic 2D view of the situation. The green curve represents the theoretical surface of an ellipsoid earth.

Using the collected elevation and calculated distance and angle between two points, the following features for a route can be created:

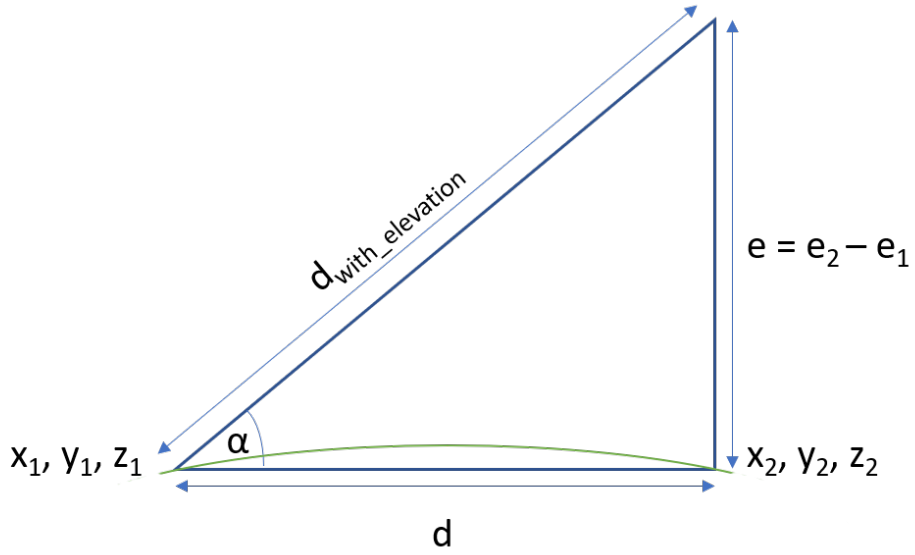


Figure 5.1: Schematic 2D view of distance and angle between two ESEC points

- **stage_is_ITT:** typically, time trials have a higher mean velocity than normal mass-start stages, so this feature is 1 if the stage is an ITT stage and 0 otherwise.
- **total_length_m:** the total length of the route in meters.
- **percent_up_steep:** percentage of the route that the degree of the slope is $> 20°$.
- **percent_up:** percentage of the route that the degree of the slope is $> 3.5°$ but $\leq 20°$.
- **percent_flat:** percentage of the route that the degree of the slope is $\leq 3.5°$ and $\geq$ -3.5°.
- **percent_down:** percentage of the route that the degree of the slope is $<$ -3.5° but $\geq$ -20°.
- **percent_down_steep:** percentage of the route that the degree of the slope is $<$ -20°.
- **percent_up_total:** sum of *percent_up_steep* and *percent_up*.
- **percent_up_flat_total:** sum of *percent_up_total* and *percent_flat*.
- **percent_down_total:** sum of *percent_down_steep* and *percent_down*.
- **percent_down_flat_total:** sum of *percent_down_total* and *percent_flat*.
- **avg_angle:** weighted mean angle of the route; calculated by taking the sum of $distance \times angle$ between every two consecutive coordinates and dividing this by the total distance.
- **avg_elevation:** weighted mean elevation; calculated by taking the sum of $distance \times \frac{e_1+e_2}{2}$ between every two consecutive coordinates and dividing this by the total distance.

- **..._times_distance:** every one of the features (except for *stage_is_ITT* and *total_length_m*) mentioned above are multiplied by the *total_length_m*. This results in features such as e.g. avg_angle_times_distance and percent_down_times_distance.

These features describe important aspects of a route. However, one can image that a cyclist will have a different mean velocity if the finish of a stage is on a steep slope compared to a finish on a flat surface, even if the features mentioned above are the same for the whole stage. In order to take these differences into account I did not only create those features for the whole stage, but also for the first, second, third and fourth quarter and for the final three kilometers of a stage. This results in a total of 133 features that characterize a route.

5.1.2 Weather

With the raw weather data collected every 50 kilometers, the following features can be engineered:

- **avg_temperature_c:** weighted mean temperature in °C.
- **avg_precip_mm:** weighted mean precipitation in millimeters.
- **avg_visibility:** weighted mean visibility in kilometers.
- **avg_cloud_cover:** weighted mean cloud cover in percentage.
- **avg_humidity:** weighted mean humidity in percentage.
- **avg_pressure_pa:** weighted mean atmospheric pressure in Pascal.
- **avg_uv_index:** weighted mean UV index.
- **avg_temp_feeling_c:** weighted mean feeling temperature in °C.
- **avg_air_density:** weighted mean air density in kg/m^3; calculated using temperature, humidity and pressure [56].
- **avg_wind_force:** weighted mean wind force measure; calculated by multiplying the wind velocity (m/s) with the cosinus of the difference of the bearing (direction) of the cyclists and of the wind. This is not physically correct, because the influence of the wind also depends on the velocity of the riders. However, because that velocity is unknown, this measure was created to give an indication of the influence of the wind on the rider velocity.

- **..._times_distance:** every one of the features mentioned above are multiplied by the total distance (in meters). This results in features such as e.g. avg_precip_mm_times_distance and avg_wind_force_times_distance.

The weather related features are only calculated for the full route, except for the *avg_wind_force* and *avg_wind_force_times_distance*. They are calculated for a whole stage, but also for the first, second, third and fourth quarter and for the final three kilometres of a stage. This results in a total of 30 weather related features.

5.1.3 Individual rider

The following features that give some more information about an individual rider are created:

- **rider_age:** the age of the rider.
- **rider_duration_one_day:** in order to have some measure of fatigue, this feature is created. It is the time in minutes the rider has cycled the day before the forecasted stage.
- **rider_duration_three_days:** the sum of the time in minutes the rider has cycled up until three days before the forecasted stage.
- **rider_duration_seven_days:** the sum of the time in minutes the rider has cycled up until seven days before the forecasted stage.

5.2 Mean stage velocity

This section describes the forecasting of the mean velocity of the riders of a stage. Instead of forecasting the duration in seconds, I choose to forecast the mean velocity in m/s. In the preliminary tests this obtained better predictions. It also allows easier comparison of the results because they are in the same order of magnitude. Before use as input for a machine learning algorithm, features are transformed using the scikit-learn standard scaler. This scaler standardizes features by removing the mean and scaling to unit variance. This is important for machine learning algorithms that use a distance metric (e.g. K-nearest neighbors).

The stages from 2013 to 2015 were used as the initial training samples. This resulted in 180 training samples for the first prediction in 2016. There are 365 retrieved stages in total between 2013 and the end of 2018. This means that, starting from 2016, 185 samples were forecasted

using rolling-origin cross-validation. Of those 185 samples 27 were in the test set and 158 in the training/validation set.

5.2.1 Baselines

When using machine learning it is important to compare your models with other simple and intuitive models. That way you know if using more advanced models and features is really worth the added complexity and computational cost. The predictions of these simple and intuitive models are called baselines.

A first and very naive baseline to predict the mean velocity of a stage would be to use the mean of all the previous stages. This baseline gives a mean absolute error (MAE) of 1.1314 m/s for the training set (95% confidence interval (CI): 1.0156-1.2540). For the test set the MAE is 0.9194 m/s (95% CI: 0.6659-1.1975).

A second, more advanced, baseline comes from the intuitive notion that in order to predict something, we look at a similar event in the past and give as prediction the outcome of that past event. Concretely, this is the same as using a K-nearest neighbors algorithm, but with only one neighbour. The used features were the four most basic: *percent_up_total_full*, *percent_flat_full*, *percent_down_total_full* and *stage_is_ITT*. This baseline improves upon the previous one and gives a MAE of 0.6360 m/s for the training set (95% CI: 0.5616-0.7135). The MAE for the test set is 0.8827 m/s (95% CI: 0.6904-1.0944).

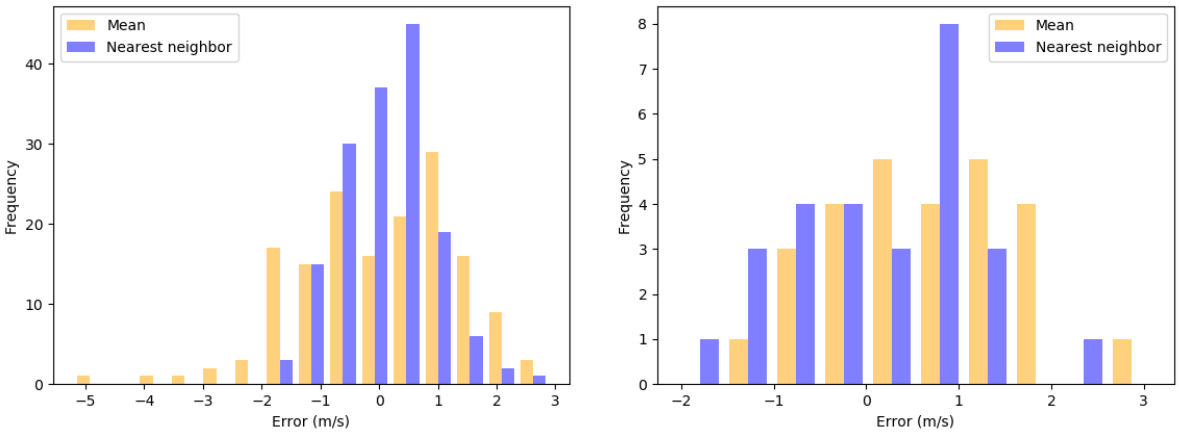
Figure 5.2 shows a histogram with the errors of the two baselines divided over 15 bins for the train set and 10 bins for the test set. In the graph we can see that the distribution of these errors is approximately normal.

The purpose of the following subsections will be to find the best combination of machine learning algorithm and features that at least improves upon the baselines.

5.2.2 Feature and model selection

To find a good combination of a machine learning model and features, forward selection and backward elimination was applied with the route- and weather-based features and six different algorithms using the default scikit-learn hyperparameters:

- **Linear regression:** OLS linear regression with intercept.



(a) Training set

(b) Test set

Figure 5.2: Binned errors of baselines for predicting mean stage duration

	Forward (m/s)	**# features**	**Backward (m/s)**	**# features**
Linear regression	0.3932	21	0.4071	31
Ridge regression	0.3939	22	0.3912	40
K-nearest neighbors	0.4857	4	0.4184	55
Random forest	0.4545	4	0.3845	21
Gradient boosting	0.3875	11	0.3843	15
Multilayer perceptron	0.4680	2	0.4810	3

Table 5.1: Mean absolute errors of different models and feature selection algorithms

- **Ridge regression:** linear least squares with l2 regularization; alpha parameter is 1.
- **K-nearest neighbors:** five neighbours are used for a prediction.
- **Random forest:** uses 10 estimators.
- **Gradient boosting:** uses 100 estimators.
- **Multilayer perceptron:** uses 1 hidden layer with 100 neurons. Because the default weight optimization algorithm (adam) is not useful for a relatively small dataset (< 100k) I opted for one deviation from the default scikit-learn hyperparameters: the use of the L-BFGS optimization algorithm.

The mean absolute errors obtained with these selection algorithms can be seen in table 5.1. Ridge regression obtains the best result of the linear algorithms and gradient boosting of the non-linear ones, so these two algorithms will be examined further.

5.2.3 Recency weighting

The results of stages from three years ago will obviously be less relevant than the results of recent stages. This can be because the bikes are more optimized, the training schema's of cyclists have improved, the current generation of riders is simply better, etc. To take this into account recency weighting is used. This ensures that more recent data points are considered to be more important. There are multiple ways to do recency weighting but I opted to use the very simple method of duplicating more recent data. For example, a stage of one day ago could be duplicated 10 times and a stage of 5 days ago only 5 times. The two functions I used to get the number of duplications based on the days ago are exponential and logarithmic decay. The formula's are:

- **Exponential decay:** $W(days\_ago) = A \times e^{k \times days\_ago} + C$, with $A > 0$ and $k < 0$
- **Logarithmic decay:** $W(days\_ago) = A \times \ln(days\_ago) + C$, with $A < 0$

The resulting float values are casted to an int and that int value is the amount of times the data sample will be duplicated. A data sample should never be deleted (duplicated 0 times), no matter how old it is, because it still contains some information. This is ensured by taking the maximum value between 1 and $W(days\_ago)$. For exponential decay this allows us to fix $C = 1$.

Both ridge regression and gradient boosting are tested with recency weighting. To find the optimal parameters for the exponential and logarithmic decay functions a grid search was used with the following values:

- Exponential decay
  - **A:** 0.1, 0.5, 0.9, 1, 1.5, 2, 2.5, 3, 3.5, 4, 6, 8, 10, ..., 98, 100
  - **k:** -0.1, -0.5, -0.9, -1, -1.5, -2, -2.5, -3, -3.5, -4
  - **C:** 1
- Logarithmic decay
  - **A:** -0.1, -0.5, -0.9, -1, -1.5, -2, -2.5, -3, -3.5, -4
  - **C:** 2, 4, ..., 98, 100

This resulted in 580 combinations for exponential decay and 500 combinations for logarithmic decay.

Ridge regression did not improve with any of the parameters stated above. Gradient boosting on the other hand did improve from an MAE of 0.3843 m/s to an MAE of 0.3815 m/s. This MAE was obtained when using exponential decay with $A = 8$, $k = -0.9$, $C = 1$ and logarithmic decay with $A = -2$, $C = 4$. Both have the exact same effect: a sample of a stage of one day ago is duplicated four times and a sample of two days ago is duplicated two times. All the other stages are only included one time. The effect is therefore the same as using the linear function $W(days\_ago) = \frac{4}{days\_ago}$.

5.2.4 Hyperparameter tuning

The next step is to tune the hyperparameters of the two selected algorithms.

Ridge regression

The method used to tune the hyperparameters for ridge regression is a grid search. First, a search space is constructed with possible values for every hyperparameter. Then, all the combinations of parameters are tried and the mean absolute error is calculated for each of these combinations using cross validation and the 40 features previously obtained with backward elimination. The hyperparameters that resulted in the lowest MAE are chosen.

For ridge regression the tried values for the hyperparameters are:

- **alpha:** 0, 0.1, 0.2, ..., 1.9, 2.0
- **fit_intercept:** True or False
- **normalize:** True or False
- **solver:** 'svd', 'cholesky', 'sparse_cg', 'lsqr', 'sag', 'saga'

This results in a search space of 480 possible combinations. After trying all those combinations the best result was: alpha=0.7, fit_intercept=True, normalize=False and solver='svd'. With these hyperparameters the MAE improved slightly over using the default parameters: from 0.3912 m/s to 0.3899 m/s. Fig. 5.3 shows the MAE using the training set for different solvers with varying alpha values. The fit_intercept parameter is True and normalize is False. The svd

and cholesky solvers are difficult to distinguish because they obtain exactly the same results. These graphs show that the optimum alpha value of 0.7 is not the result of random effects because it shows clear patterns with a clear minimum.

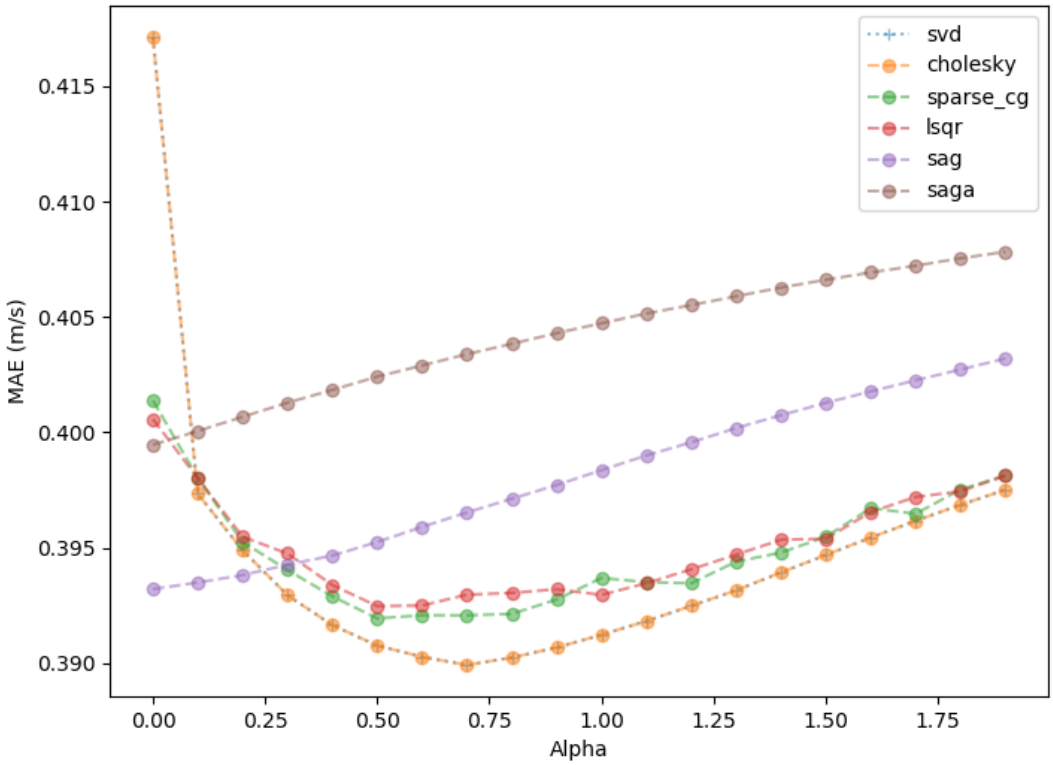


Figure 5.3: MAE using training set with varying alphas for different solvers and fit_intercept=True and normalize=False

Gradient boosting regression

Because gradient boosting has more hyperparameters than ridge regression and is computationally a lot more expensive to train, an extensive grid search would be infeasible. Therefore I will tune only the most important parameters and not in one grid search, but split into three grid searches. After every grid search, the parameters considered in that search are fixed. The first three parameters that will be searched are loss, criterion and alpha. The tried values for these are:

- **loss:** 'ls', 'lad', 'huber' or 'quantile'
- **criterion:** 'friedman_mse', 'mse' or 'mae'

- **alpha:** 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

The alpha parameter is only used when the loss is quantile. This results in a search space of 36 possible combinations. The best result was obtained using the default values: loss = 'ls' and criterion = 'friedman_mse'.

Next, the tree-specific parameters are tuned. These are max_depth, min_samples_split, min_samples_leaf and max_features. The following values are tried:

- **max_depth:** 1, 2, 3, ..., 14, 15
- **min_samples_split:** 2, 3, ..., 19, 20
- **min_samples_leaf:** 1, 2, ..., 14, 15
- **max_features:** 1, 2, ..., 14, 15

This results in a search space of 64 125 possible combinations. The best result obtained was with max_depth = 3, min_samples_split = 19, min_samples_leaf = 1 and max_features = 10. These parameters resulted in an MAE of 0.3784 m/s. Fig. 5.4 shows for each of these hyperparameters the influence on the MAE while keeping the other hyperparameters constant at the previously obtained best values.

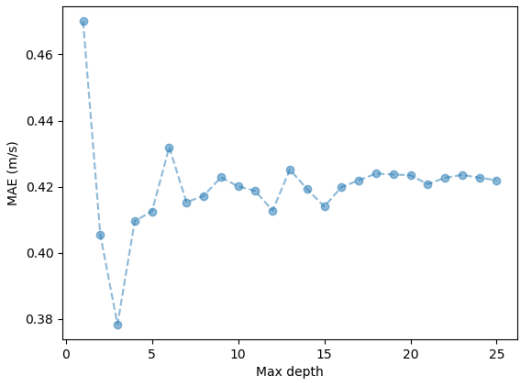
Finally, subsample and n_estimators are tuned. The following values are tried:

- **subsample:** 0.5, 0.55, ..., 0.95, 1
- **n_estimators:** 50, 60, ..., 190, 200

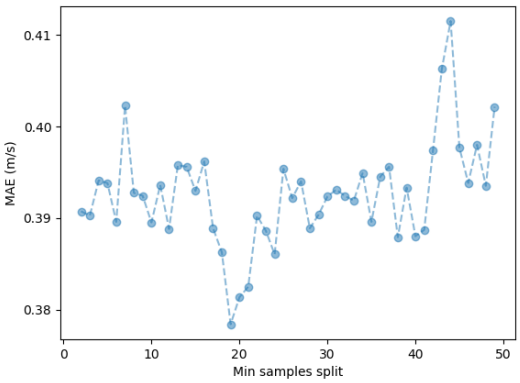
This results in a search space of 176 possible combinations. The best result obtained was with subsample = 1.0 and n_estimators = 90. These parameters resulted in an MAE of 0.3758 m/s. Fig. 5.5 shows for these two hyperparameters the influence on the MAE while keeping the other hyperparameters constant at the previously obtained best values.

5.2.5 Results

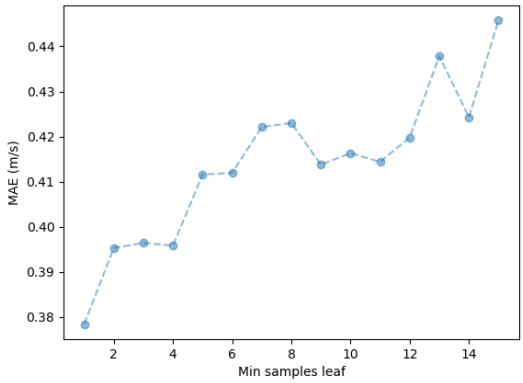
After the hyperparameter tuning, gradient boosting obtains the best result. The MAE for the training set is 0.3758 m/s (with 95% CI: 0.3198 m/s - 0.4362 m/s). The test set however only gets an MAE of 0.4660 m/s (95% CI: 0.3400-0.6014). Figure 5.6 shows a histogram with the



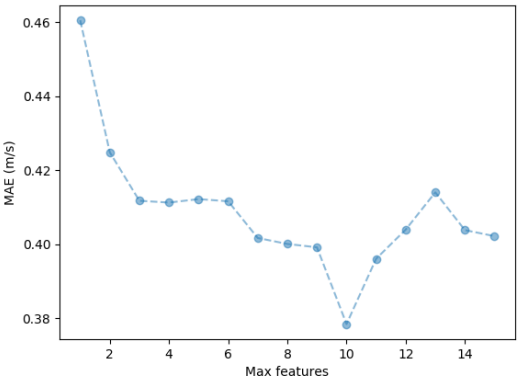
(a) Max depth parameter with optimum at 3



(b) Min samples split parameter with optimum at 19

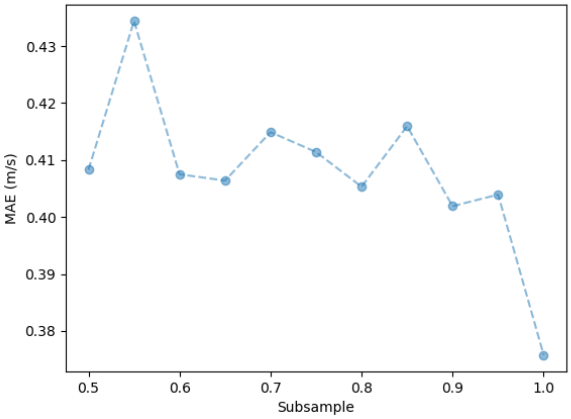


(c) Min samples leaf parameter with optimum at 1

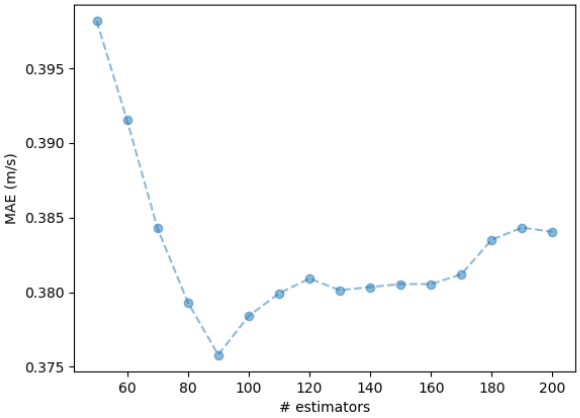


(d) Max features parameter with optimum at 10

Figure 5.4: Influence of tree-specific hyperparameters on the MAE



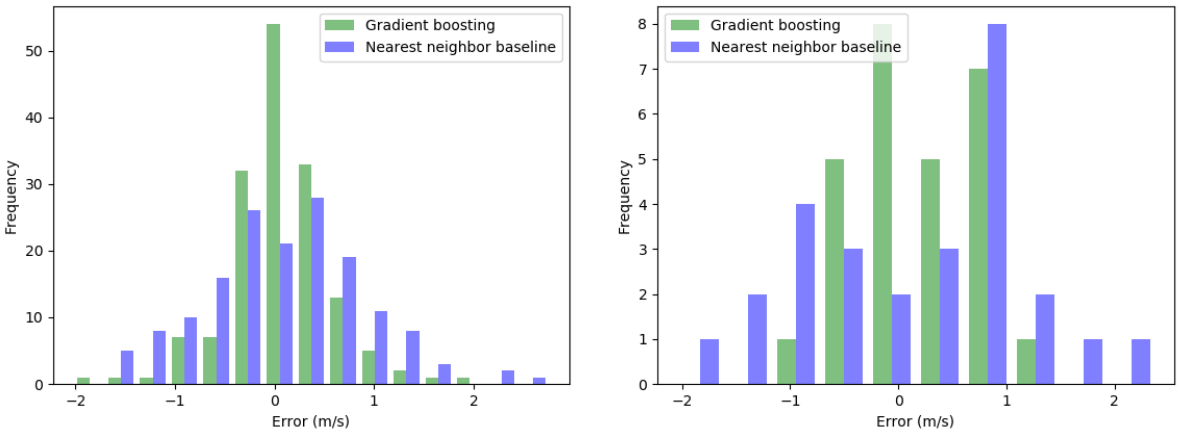
(a) Subsample parameter with optimum at 1



(b) Number of estimators parameter with optimum at 90

Figure 5.5: Influence of subsample and number of estimators on the MAE

errors of the gradient boosting algorithm compared with the nearest neighbor baseline. The errors are divided over 15 bins for the training set and 10 bins for the test set. In the graph we



(a) Training set

(b) Test set

Figure 5.6: Binned errors of gradient boosting for predicting mean stage duration compared with linear regression baseline

	Training set			*Test set*		
	CI lower	**MAE**	**CI upper**	**CI lower**	**MAE**	**CI upper**
Mean	1.0156	1.1314	1.2540	0.6659	0.9194	1.1975
Nearest neighbor	0.5616	0.6360	0.7135	0.6904	0.8827	1.0944
Gradient boosting	0.3198	0.3758	0.4362	0.3400	0.4660	0.6014

Table 5.2: MAE and CI of baselines and gradient boosting

can see that the distribution of these errors is approximately normal. Table 5.2 shows the MAE and CI for the two baselines and gradient boosting for both the training set and the test set. Using this table, it is clear that gradient boosting gives better results than the mean and nearest neighbor baselines. Compared to the nearest neighbor baseline, an improvement of 0.4167 m/s is obtained (95% CI: 0.1810 m/s - 0.6628 m/s). The 95% CI for the test set MAE with only stages from the Tour de France is 0.2825 m/s - 0.6954 m/s, for the Giro d'Italia this is 0.2779 m/s - 0.8313 m/s and for the Vuelta a España 0.2246 m/s - 0.5369 m/s. This means that at the 95% confidence level there is no significant difference between the MAE of the three Grand Tours.

Sckit-learn provides a method to get the feature importances from a fitted gradient boosting model. The 15 features used in the gradient boosting regression together with their importance relative to the most important feature can be seen in fig. 5.7. This importance is calculated based on a fitting of all the 365 data samples of the stages from 2013 to the end of 2018. The percentage of the total stage route that goes down or is flat is clearly the most important feature.

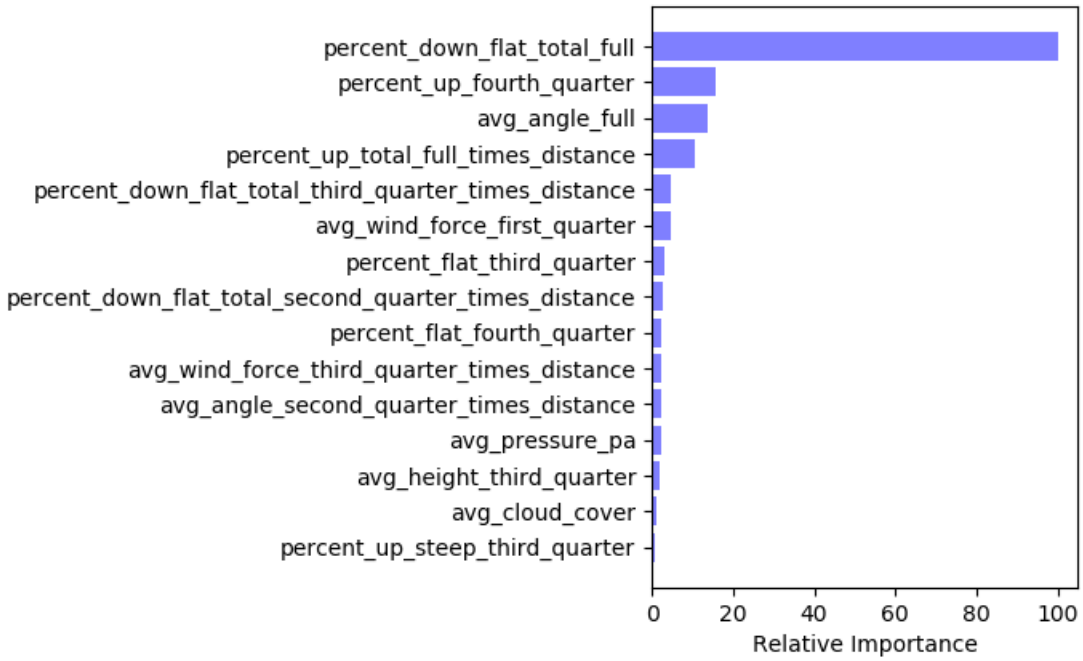
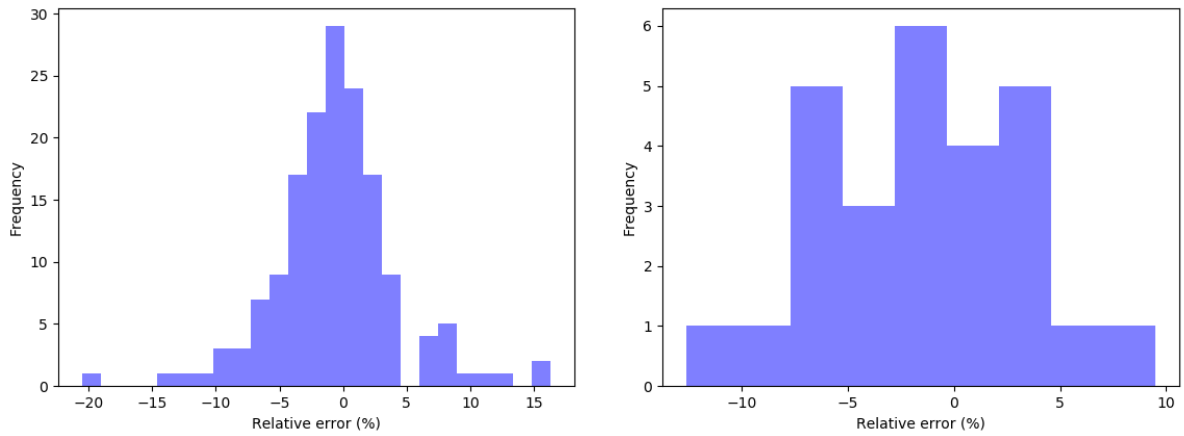


Figure 5.7: Relative feature importance for gradient boosting

At about 15% of the importance of percent_down_flat_total_full are the percentage of the last quarter of the stage route that goes up and the weighted mean slope of the total stage route. The total metres the cyclists have to climb in the whole stage is the fourth most important feature, at about 10% of the importance of percent_down_flat_total_full. All the other features have a relative importance of less than 5%.

If we look at the predictions for the total duration in seconds for the stages, we get a mean absolute percentage error (MAPE) of 3.49% (95% CI: 2.97% - 4.06%) for the training set and 4.06% (95% CI: 2.97% - 5.26%) for the test set. The distributions for the relative errors can be seen in fig. 5.8. The histogram uses 25 bins for the training set and 9 bins for the test set.



(a) Training set

(b) Test set

Figure 5.8: Binned relative errors of gradient boosting for predicting mean stage duration

5.3 Individual rider velocity

This section describes the forecasting of the velocity of a stage for each individual rider. The main purpose of these forecasts is to compare different riders, so we are less interested in the exact mean velocities of the riders. Therefore, I chose to forecast the difference between the mean velocity of a stage and the mean velocity of each of the riders in that stage. The prediction for the mean velocity of a rider then equals the sum of that predicted difference and the predicted mean velocity of the stage. Instead of trying to find features that give information about the intrinsic qualities of a cyclist I opted to create a different model for every rider. The main benefit is that the intrinsic qualities are automatically taken into account. If, for example, a rider is very good at climbing, the model will give more weight to the *percent_up_full* feature. The disadvantage is that the amount of data for every rider is rather limited. Before using features, they are transformed using a standard scaler. The predictions start from 2016 up until 2018. A minimum of 50 data samples for one rider is chosen in order to make a prediction.

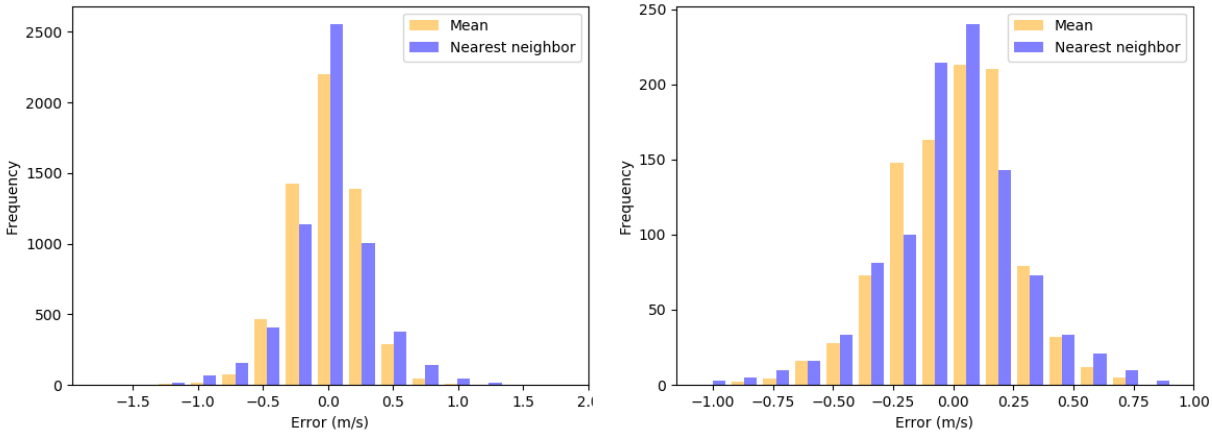
5.3.1 Baselines

The exact same baselines as in the prediction of the mean stage velocity are used.

The first baseline to predict the difference between the mean velocity of a stage and the mean velocity of each of the riders in that stage would be to use the mean predictor. This baseline gives an MAE of 0.2059 m/s for the training set (95% CI: 0.2016-0.2103). For the test set the MAE is 0.1970 m/s (95% CI: 0.1880-0.2063).

The second baseline uses a K-nearest neighbors algorithm, but with only one neighbour. The used features are: *percent_up_total_full*, *percent_flat_full*, *stage_is_ITT* and *percent_down_total_full*. However, this baseline does not improve upon the previous one. The MAE for the training set is 0.2265 m/s (95% CI: 0.2207-0.2324) and for the test set 0.2005 m/s (95% CI: 0.1890-0.2123).

Figure 5.9 shows a histogram with the errors of the two baselines divided over 15 bins for both the training and test set. In the graph we can see that the distribution of these errors is approximately normal.



(a) Training set

(b) Test set

Figure 5.9: Binned errors of baselines for predicting difference between mean stage velocity and rider velocity

The purpose of the following section will be to find the best combination of machine learning algorithm and features that at least improves upon the two baselines.

5.3.2 Feature and model selection

To find a good combination of a machine learning model and features, forward selection and backward elimination were applied. The route-, weather- and individual rider-based features in combination with the two best performing algorithms from the mean stage forecasting (ridge and gradient boosting) using the default scikit-learn hyperparameters were tried. Because there are 200 riders saved in the database, the computational cost of these selection algorithms would be about 200 times higher compared with the one from the mean stage velocity forecasting. Therefore, the feature selection was done with only the UCI World ranking top 50 cyclists. However, even then the backward elimination with gradient boosting proved to be too computationally heavy to run in a few days. Only in this case the backward elimination was done using the top 10 riders and the n_estimators parameter was reduced from the default 100 to 20.

	Forward (m/s)	# features	Backward (m/s)	# features
Ridge regression	0.1646	6	0.1644	4
Gradient boosting	0.1777	7	0.1749	9

Table 5.3: Mean absolute errors for all 200 riders of different models and feature selection algorithms

The MAEs of the training set obtained with these selection algorithms can be seen in table 5.3. These MAEs are calculated using all 200 of the riders. Ridge regression obtains the best results, so this algorithm will be further examined.

5.3.3 Recency weighting

Recency weighting was tested for ridge regression using the features obtained from the backward elimination. Both logarithmic and exponential decay were tested with the same possible parameters as for the mean stage prediction. The UCI world ranking 2018 top-50 cyclists were used to find the best decay method and parameters. The best result using the exponential decay function is obtained with $A = 26$, $k = -0.1$, $C = 1$ and results in an MAE for all 200 riders of 0.1577 m/s. The effect is that a sample of a stage of one day ago is duplicated 24 times and that the last stage that is duplicated is a stage of 32 days ago. Using the logarithmic decay function the best result is obtained with $A = -4$ and $C = 20$ and results in an MAE of 0.1573 m/s. The effect is that a sample of a stage of one day ago is duplicated 20 times and that the last stage that is duplicated is a stage of 90 days ago. Figure 5.10 shows both of the decay functions, both in their continuous version and in their discrete, casted to int, version.

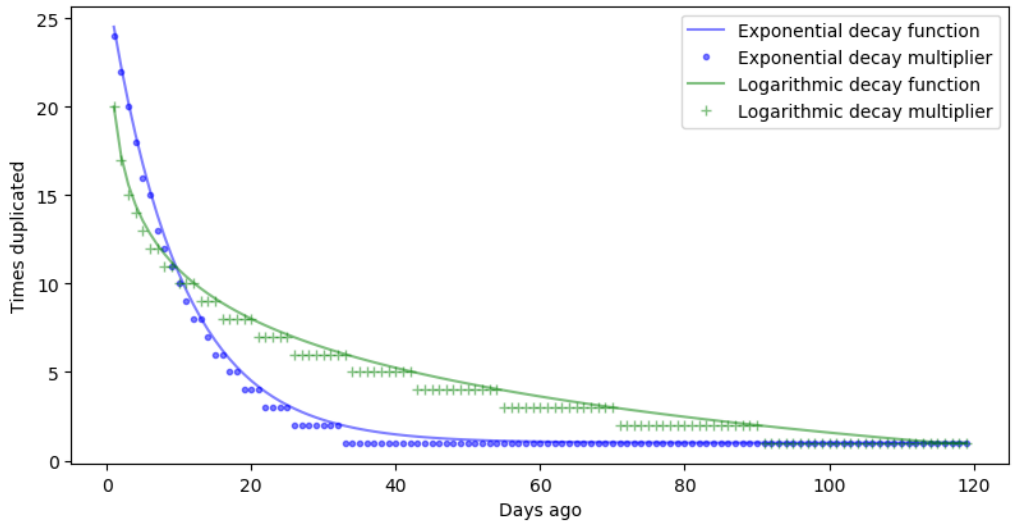


Figure 5.10: Exponential and logarithmic decay functions

5.3.4 Hyperparameter tuning

A grid search is used to tune the hyperparameters of the ridge regression. The exact same values as for the mean stage duration are tried, except for the alpha parameter. The alpha parameter is tried with the following values: 0, 0.01, 0.02, ..., 0.09, 0.1. This results in a search space of 264 possibilities. The best result with the UCI top-50 riders was obtained with the following hyperparameters: alpha=0.05, fit_intercept=True, normalize=True and solver='lsqr'. With these hyperparameters the MAE for all the 200 riders improved slightly over using the default parameters: from 0.1573 m/s to 0.1560 m/s. Fig. 5.11 shows the MAE using the training set for different solvers with varying alpha values. The fit_intercept and normalize parameters are True. The svd and cholesky solvers are difficult to distinguish because they obtain exactly the same results. These graphs show clear patterns with the lsqr solver and alpha = 0.05 as a clear minimum.

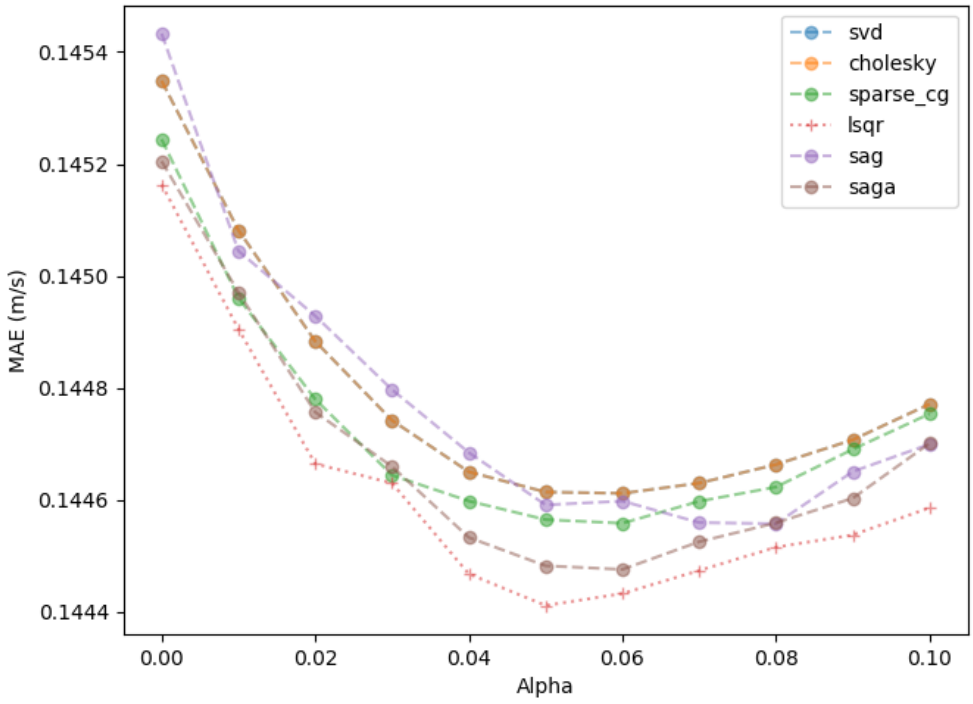
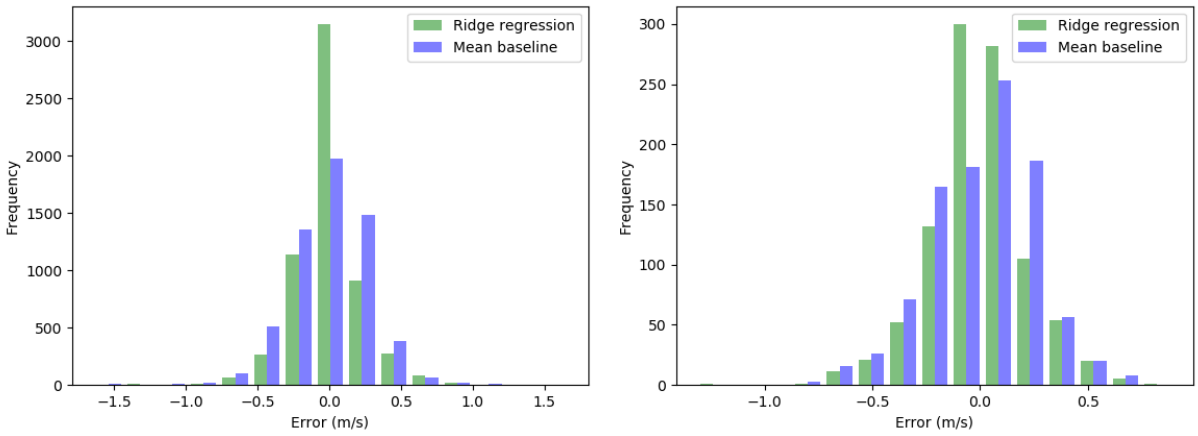


Figure 5.11: MAE individual rider velocity using training set with varying alphas for different solvers and fit_intercept=True and normalize=True

5.3.5 Results

The resulting MAE for the difference between mean stage velocity and rider velocity for all 200 riders is 0.1560 m/s (95% CI: 0.1517 m/s - 0.1603 m/s) for the training set and 0.1576 m/s (95% CI: 0.1481-0.1673) for the test set. This is an improvement of 0.0394 m/s (95% CI: 0.0262 m/s - 0.0526 m/s) compared to the best of the two baselines, the mean baseline. Fig. 5.12 shows the distribution of the final results compared with the mean baseline. The four features that are included in the final ridge regression are: stage_is_ITT, percent_down_full, percent_up_fourth_quarter, percent_down_total_first_quarter.



(a) Training set

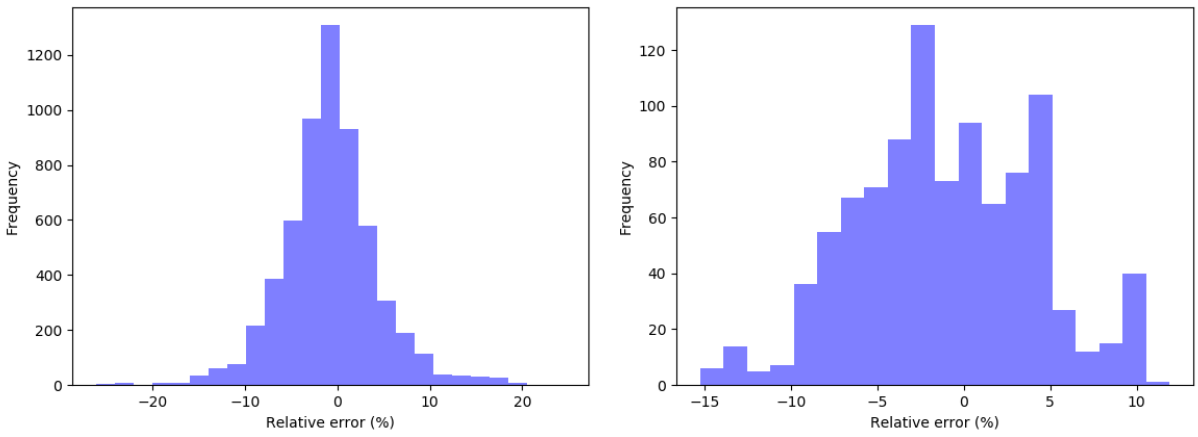
(b) Test set

Figure 5.12: Binned errors of ridge regression for predicting difference between mean stage velocity and rider velocity

The 95% CI for the test set MAE with only stages from the Tour de France is 0.1107 m/s - 0.1380 m/s, for the Giro d'Italia this is 0.1632 m/s - 0.1971 m/s and for the Vuelta a España 0.1715 m/s - 0.2101 m/s. This means that at the 95% confidence level there is no significant difference between the MAE of the Giro d'Italia and the Vuelta a España, but there is with the Tour de France. The Tour de France has a lower MAE for the difference between mean stage velocity and rider velocity. A possible explanation could be that for many riders the Tour de France is the most important cycling event and therefore they will do their utmost to win or help their teammates. This in turn will reduce the unpredictability of the individual rider performances.

If we look at the predictions for the duration in seconds for the individual riders, we get a mean absolute percentage error of 3.82% (95% CI: 3.73% - 3.91%) for the training set and 4.30%

(95% CI: 4.11% - 4.49%) for the test set. The distributions for the relative errors can be seen in fig. 5.13. The histogram uses 25 bins for the training set and 20 bins for the test set.



(a) Training set

(b) Test set

Figure 5.13: Binned relative errors of ridge regression for predicting rider duration

6. Predicting head-to-head

This section describes the forecasting of the head-to-head wins between riders. The head-to-head winner between two riders in a certain stage is the rider that finishes before the other rider. The main question this section will try to answer is whether the previously predicted individual rider duration can help in predicting these head-to-head wins. Therefore, the focus will not be on gathering a lot of features, but only a limited number of intuitive features will be used. For the same reason there will be no extensive tuning of the hyperparameters nor any form of recency weighting.

6.1 Feature engineering

The following new features, comparing two riders, are created:

- **rider_hth_wins_total:** the total amount of times the rider has finished a stage earlier than an other rider. All the stages from 2013 up until 2018 that are included in the UCI World tour races of 2018 are taken into account. These are the stages from where the results were scraped (see chapter Data gathering, section Cyclists & stages).
- **rider_hth_wins_recent:** using the same stages as for *rider_hth_wins_total*, but only up until 10 days before the forecasted stage.
- **rider_hth_wins_one_year:** using the same stages as for *rider_hth_wins_total*, but only up until one year before the forecasted stage.
- **rider_hth_wins_flat_stage_one_year:** the amount of times the rider has finished a flat stage earlier than the other, compared, rider, up until one year before the forecasted stage. A stage is considered flat if the feature *percent_flat_full* is larger than or equal to 0.8. Because information about the route topology is used, only stages for which these data were collected can be taken into account. This means only stages of the Tour de France, Giro d'Italia and Vuelta d'España from 2013 up until 2018 are used.
- **rider_hth_wins_hilly_stage_one_year:** the same as *rider_hth_wins_flat_stage_one_year*, but for stages that have *percent_flat_full* < 0.8.

These features are always created for two riders in a certain stage, comparing them. From these features some derivative features are constructed:

- **hth_wins_..._difference:** the difference between the head-to-head wins of two riders. This feature is created for the five new features mentioned above.
- **predicted_velocity_diff_difference:** the predicted difference in velocity between the mean stage velocity and the velocity of a rider (see chapter Predicting velocity, section Individual rider velocity) for two riders is subtracted from each other to create this feature.
- **rider1_wins_based_on_..._difference:** for each of the 'difference' features mentioned above a simple boolean value is also calculated. This value is True if according to that difference the so-called rider1 of the two riders that are being compared should finish before the other rider. In other words if rider1 has had more head-to-head wins in the past compared to rider2, according to one of the possible head-to-head wins measures mentioned above, this value will be True. This value is also calculated for the *predicted_velocity_diff_difference* feature. It will be True if the feature is < 0, meaning that the predicted difference in velocity between the mean stage velocity and the velocity of a rider for rider1 is smaller than for rider2.
- **rider_1_wins_based_on_flat_or_hilly_stage:** True if *percent_flat_full* >= 0.8 and rider1 should win according to the *rider1_wins_based_on_flat_stage_difference* feature or when *percent_flat_full* < 0.8 and rider1 should win according to the *rider1_wins_based_on_hilly_stage_difference* feature.

Together with the features created in chapter 5, these are the possible features for predicting head-to-head wins.

6.2 Baseline

Predictions are made for the three Grand Tours of 2017 and 2018. The results of 2016 are used as initial training data. The accuracy for every stage is predicted using cross-validation and the final metric is the mean of all the accuracies of these stages. Because the amount of stages predicted is rather small (123 stages) and because no feature selection nor hyperparameter tuning nor recency weighting will be used, cross-validation should suffice. Therefore, a test set will not be used. In order to make sure that the data is balanced, every data sample comparing two

riders (rider1 and rider2) is duplicated and flipped, switching the role and features of rider1 and rider2. This ensures that the amount of samples where rider1 wins is the exact same as the amount of samples where rider2 wins. However, this also means that the amount of training samples is doubled.

The baseline uses only the *rider_1_wins_based_on_flat_or_hilly_stage* feature to make a prediction. This more or less corresponds to the basic intuitive prediction of many supporters that a rider that has in the past year in most cases finished earlier than another rider, while making a distinction between flat stages and more hilly stages, will have the highest chance of again finishing earlier than that other rider. This baseline obtains a mean stage accuracy of 68.36% (95% CI: 66.63% - 70.05%). Fig. 6.1 shows a binned histogram with the accuracy of the 123 predicted stages. Five stages have an accuracy below 50%.

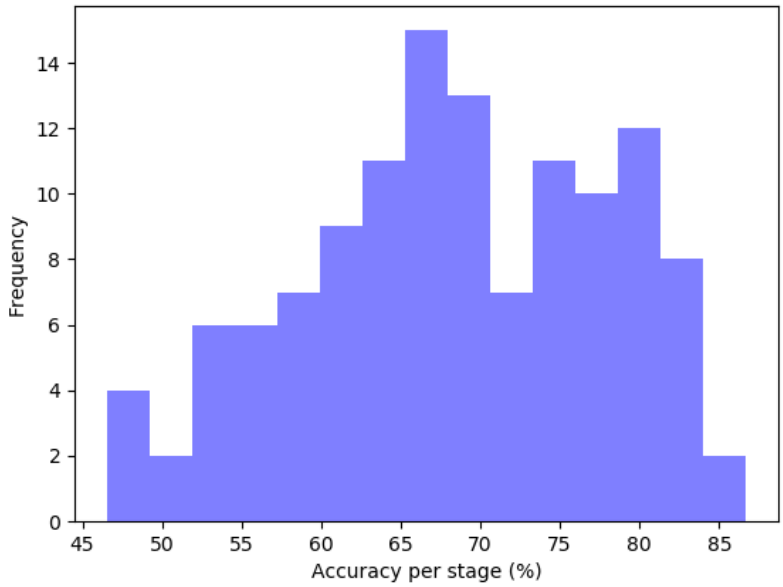


Figure 6.1: Binned baseline accuracies for head-to-head predictions of the three Grand Tours from 2017 and 2018

6.3 Model selection

Four classification algorithms are tested, with mostly default hyperparameters: K-nearest neighbors, random forest, gradient boosting and multi-layer perceptron. The only deviations from the default hyperparameters are that the multi-layer perceptron uses max_iter=500 and alpha=0.01 instead of the default max_iter=200 and alpha=0.0001. This was done because otherwise the

	without individual rider prediction (%)	including individual rider prediction (%)
K-nearest neighbors	68.15	69.83
Random forest	68.29	70.30
Gradient boosting	71.35	73.16
Multilayer perceptron	71.25	72.32

Table 6.1: Mean stage head-to-head accuracy using different models and two sets of features

algorithm that fixes the weights of the links would not converge. Two different sets of features were used. The first one includes all the features newly created in section 6.1 except for the features related to the predicted difference between the mean stage velocity and the individual rider velocity, predicted in chapter 5. These are combined with the features *rider_age*, for both riders, *stage_is_ITT*, *percent_up_total_full*, *percent_down_total_full* and *percent_flat_full* (see section 5.1 for information about these features). The second set includes all the features from the first set, but this time also the features related to the individual rider prediction. Table 6.1 shows the results of these algorithms.

6.4 Results

The most important thing we can see in the previous section is that for all four of the algorithms the accuracy improves when using the individual rider prediction. This suggests that the predictions from section 5.3 can be useful in order to predict the head-to-head wins between riders.

Gradient boosting obtains the best results of those four algorithms. When the individual rider predictions are not included the mean stage accuracy is 71.35% (95% CI: 69.84% - 72.86%). Adding the predicted difference features an accuracy of 73.16% (95% CI: 71.58% - 74.72%) was obtained. This means an accuracy improvement of 1.81%. This improvement varies, at the 95% confidence level, from -0.36%, a small decrease, to 3.97%, a substantial increase. Fig. 6.2 shows a binned histogram with the mean stage accuracies of the baseline and the ones using the gradient boosting and individual rider predictions. When using gradient boosting and all the features, there are no stages anymore with a mean accuracy lower than 50%.

The 95% CI for the mean prediction accuracy for only the Tour de France is 70.09%-74.87%, for the Giro d'Italia this is 71.18%-77.17% and for the Vuelta a España 70.11%-75.37%. This means that at the 95% confidence level there is no statistically significant difference between

the mean prediction accuracy of the three Grand Tours.

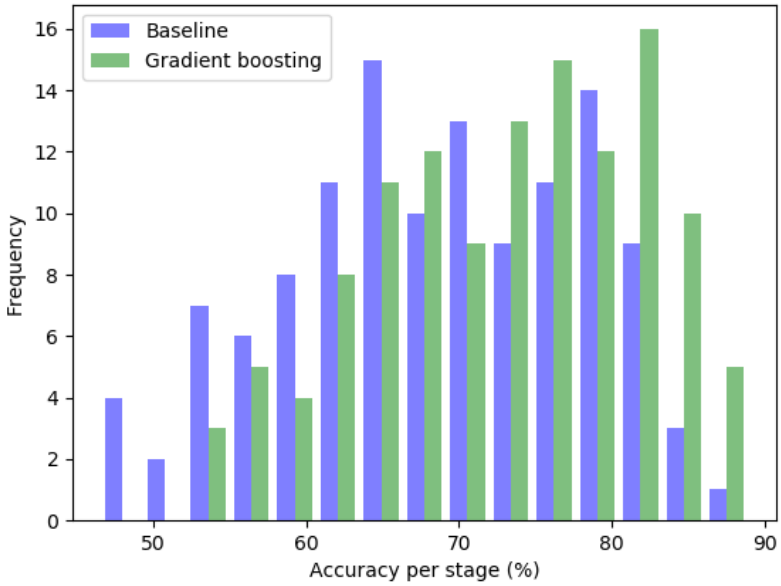


Figure 6.2: Binned accuracies for head-to-head predictions
of the three Grand Tours from 2017 and 2018

Fig. 6.3 show the relative importance of the features compared to the most important feature. The gradient boosting algorithm was fitted on all the available head-to-head data for the three Grand Tours from 2016 up until 2018. Only the 12 features with a relative importance larger than 0.5% are included in the plot. This plot clearly shows that the difference in the predicted individual rider performance between the two riders is the most important feature. The feature used as baseline is the second most important one. The difference in head-to-head wins of two riders in stages up until 10 days before the predicted stage is the third most important feature.

An interesting question is which stages were predicted well and which poorly and why. Fig. 6.4 plots the total percentage of a stage that is flat against the mean stage head-to-head prediction accuracy. In this plot there seems to be some kind of trend: the lower the percentage that is flat, the higher the mean stage accuracy. This means that hilly stages were predicted better than flat stages. This could be expected because hilly stages typically result in larger mean stage velocity differences between riders. However, for stages that are more flat, the variability is high. When looking at the different route and weather features collected for a stage, no explanation for that variability was found. Neither did the type of stage (ITT or normal mass-start stage) nor the specific Grand Tour provide an explanation. Therefore this question remains open.

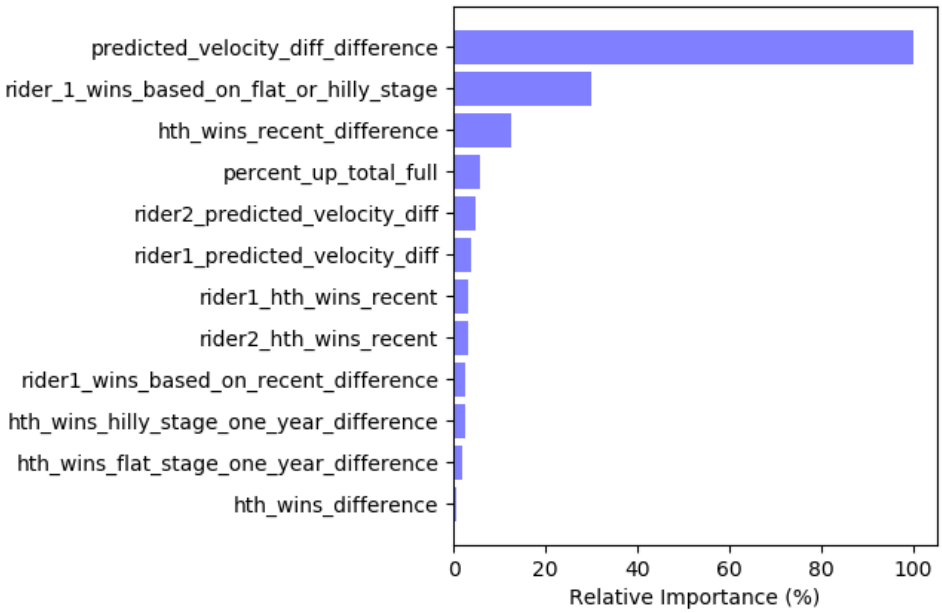


Figure 6.3: Feature importance relative to the most important feature for predicting mean stage head-to-head accuracy using gradient boosting

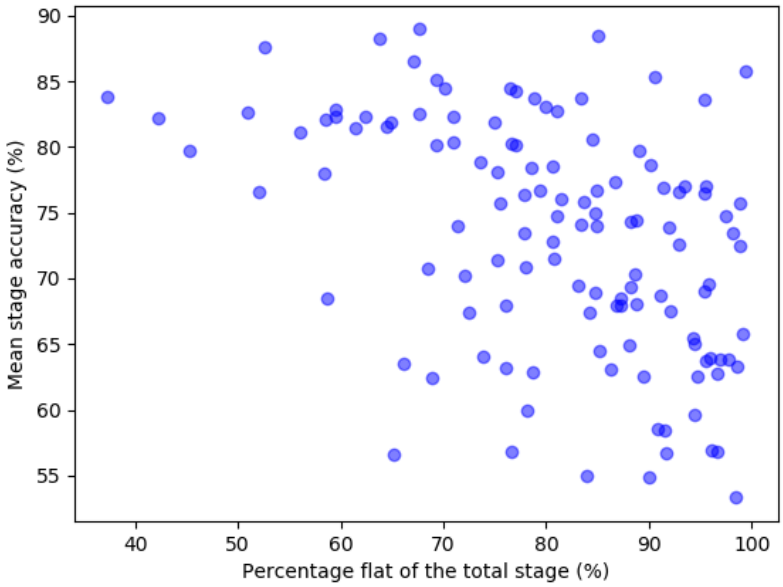


Figure 6.4: Mean stage accuracy for head-to-head predictions compared with the total percentage of a stage that is flat

7. Conclusion

This final chapter will give an overview of the main results and make suggestions on how to improve and extend this work.

7.1 Main results

Machine learning was used in order to predict road cycling results. The focus was put on the stages of the three Grand Tours (Giro d'Italia, Vuelta a España and Tour de France) from 2016 to 2018.

First of all, the mean velocity was predicted for these stages. With a gradient boosting algorithm, backwards feature elimination and recency weighting, a MAE of 0.4660 m/s was obtained for the test set. This is an improvement of 0.4167 m/s compared to the MAE of the best of two baselines. If we look at the predictions for the total duration in seconds for the stages in the test set, a mean absolute percentage error (MAPE) of 4.06% was obtained.

Secondly, the difference between the mean stage and individual rider velocity was predicted. This difference is a measure for the performance of an individual rider. Ridge regression together with backwards elimination and recency weighting of up to 90 days before the predicted stage proved to be the best choice. The MAE obtained for the test set was 0.1576 m/s. This was an improvement of 0.0429 m/s compared to the MAE of the nearest neighbor baseline. When looking at the predictions for the duration in seconds for the individual riders a MAPE of 4.30% was obtained for the test set.

Finally, the question is whether this predicted difference can help in forecasting the head-to-head wins between riders. Using a gradient boosting algorithm, cross validation and, among others, features based on the predicted difference a mean stage accuracy of 73.16% was obtained. When the same algorithm is used without the features based on the predicted difference an accuracy of 71.35% was obtained. This means that adding the predicted difference features resulted in an accuracy improvement of 1.81%. At the 95% CI this varies from a small accuracy decrease of 0.36% to a substantial accuracy increase of 3.97%.

This thesis has demonstrated that machine learning in combination with public data can be used to predict cycling results with a reasonable accuracy. It provides insights into how to collect and process public cycling data and what information is important. It also explores the influence of recency weighting on the results and shows that predicting the difference between the individual rider and mean stage velocity is effective in forecasting head-to-head results. Finally, non-trivial baselines are provided for future research.

7.2 Future work

This thesis is the first one that predicts results of road cycling using machine learning. Hence, there is certainly room for improvements in future research. Three suggestions are given in the remainder of this section.

First, other stages could be researched. There are many other important stages than those from the three Grand Tours. With the data of more stages the results could potentially be improved. Another question could be whether the models used for the Grand Tours would also work for one-day races such as the Tour of Flanders.

Secondly, more features could be researched. It is expected that advanced physics and expert domain knowledge will lead to better features for the stage and individual rider duration predictions. One of those features could potentially even explain for the head-to-head predictions why some of the stages were predicted better than others. This thesis focused on the possibility of using the predicted difference in order to forecast head-to-head wins and as such uses very few additional features. Therefore, the head-to-head predictions could certainly be improved with more and better features.

Finally, other machine learning techniques and hyperparameters could be researched. Many more machine learning algorithms exist than the ones used in this thesis. Also, only a few algorithms have had their hyperparameters tuned. Maybe some of the algorithms that were only tested with their default hyperparameters would perform much better after tuning.

Bibliography

[1] D. Harville, "Predictions for National Football League games via linear-model methodology", *Journal of the American Statistical Association*, vol. 75, no. 371, pp. 516–524, 1980.

[2] M. J. Maher, "Modelling association football scores", *Statistica Neerlandica*, vol. 36, no. 3, pp. 109–118, 1982.

[3] M. J. Dixon and S. G. Coles, "Modelling Association Football Scores and Inefficiencies in the Football Betting Market", *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 46, no. 2, pp. 265–280, 1997.

[4] D. Barry and J. A. Hartigan, "Choice models for predicting divisional winners in major league baseball", *Journal of the American Statistical Association*, vol. 88, no. 423, pp. 766–774, 1993.

[5] J. Goddard, "Regression models for forecasting goals and match results in association football", *International Journal of forecasting*, vol. 21, no. 2, pp. 331–340, 2005.

[6] A. Joseph, N. E. Fenton, and M. Neil, "Predicting football results using Bayesian nets and other machine learning techniques", *Knowledge-Based Systems*, vol. 19, no. 7, pp. 544–553, 2006.

[7] C. U. Song, B. L. Boulier, and H. O. Stekler, "The comparative accuracy of judgmental and model forecasts of American football games", *International Journal of Forecasting*, vol. 23, no. 3, pp. 405–413, 2007.

[8] A. C. Constantinou, N. E. Fenton, and M. Neil, "Pi-football: A Bayesian network model for forecasting Association Football match outcomes", *Knowledge-Based Systems*, vol. 36, pp. 322–339, 2012.

[9] G. Boshnakov, T. Kharrat, and I. G. McHale, "A bivariate Weibull count model for forecasting association football scores", *International Journal of Forecasting*, vol. 33, no. 2, pp. 458–466, 2017.

[10] D. Berrar, P. Lopes, and W. Dubitzky, "Incorporating domain knowledge in machine learning for soccer outcome prediction", *Machine Learning*, vol. 108, no. 1, pp. 97–126, 2019.

[11] W. J. Knottenbelt, D. Spanias, and A. M. Madurska, "A common-opponent stochastic model for predicting the outcome of professional tennis matches", *Computers and Mathematics with Applications*, vol. 64, no. 12, pp. 3820–3827, 2012.

[12] I. McHale and A. Morton, "A Bradley-Terry type model for forecasting tennis match results", *International Journal of Forecasting*, vol. 27, no. 2, pp. 619–630, 2011.

[13] R. Ul Mustafa, M. S. Nawaz, M. I. U. Lali, T. Zia, and W. Mehmood, "Predicting the Cricket match outcome using crowd opinions on social networks: A comparative study of machine learning methods", *Malaysian Journal of Computer Science*, vol. 30, no. 1, pp. 63–76, 2017.

[14] H. Kaur and S. Jain, "Machine learning approaches to predict basketball game outcome", in *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall)*, IEEE, 2017, pp. 1–7.

[15] A. Neudorfer and S. Rosset, "Predicting the NCAA basketball tournament using isotonic least squares pairwise comparison model", *Journal of Quantitative Analysis in Sports*, vol. 14, no. 4, pp. 173–183, 2018.

[16] J. Edelmann-Nusser, A. Hohmann, and B. Henneberg, "Modeling and prediction of competitive performance in swimming upon neural networks", *European Journal of Sport Science*, vol. 2, no. 2, pp. 1–10, 2002.

[17] E. Davoodi and A. R. Khanteymoori, "Horse racing prediction using artificial neural networks", *Recent Advances in Neural Networks, Fuzzy Systems & Evolutionary Computing*, vol. 2010, pp. 155–160, 2010.

[18] K. Przednowek, J. Iskra, and S. Cieszkowski, "The use of selected linear models in predicting the results of 400-metre hurdles races", *Current research in motor control*, vol. 4, 2012.

[19] K. Przednowek and K. Wiktorowicz, "Prediction of the result in race walking using regularized regression models", *Journal of Theoretical and Applied Computer Science*, vol. 7, no. 2, pp. 45–58, 2013.

[20] D. Ruiz-Mayo, E. Pulido, and G. Martínez-Muñoz, "Marathon performance prediction of amateur runners based on training session data", *Machine Learning and Data Mining for Sports Analytics*, 2016.

[21] T. S. Olds, K. I. Norton, E. L. A. Lowe, S. Olive, F. Reay, and S. Ly, "MODELING ROAD-CYCLING PERFORMANCE", *Journal of Applied Physiology*, vol. 78, no. 4, pp. 1596–1611, 1995.

[22] J. C. Martin, D. L. Milliken, J. E. Cobb, K. L. McFadden, and A. R. Coggan, "Validation of a mathematical model for road cycling power", *Journal of Applied Biomechanics*, vol. 14, no. 3, pp. 276–291, 1998.

[23] Y. Kataoka and P. Gray, "Real-time power performance prediction in tour de France", in *CEUR Workshop Proceedings*, vol. 2284, 2018, pp. 127–136.

[24] A. Pezzoli, E. Cristofori, M. Moncalero, F. Giacometto, A. Boscolo, R. Bellasio, and J. Padoan, "Effect of the environment on the sport performance: Computer supported training-a case study for cycling sports", in *International Congress on Sports Science Research and Technology Support*, Springer, 2013, pp. 1–16.

[25] B. L. Hannas and J. E. Goff, "Model of the 2003 Tour de France", *American Journal of Physics*, vol. 72, no. 5, pp. 575–579, 2004.

[26] J. E. Goff, "Predicting winning times for stages of the 2011 tour de france using an inclined-plane model", *Procedia Engineering*, vol. 34, pp. 670–675, 2012.

[27] B. Lee Hannas and J. E. Goff, "Inclined-plane model of the 2004 Tour de France", *European Journal of Physics*, vol. 26, no. 2, pp. 251–259, 2005.

[28] B. A. Ramsey and J. E. Goff, "Predicting Tour de France stage-winning times with continuous power and drag area models and high speeds in 2013", *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 228, no. 2, pp. 125–135, 2014.

[29] C. M. Hobson and J. E. Goff, "Improving Tour de France modeling with allometric scaling", *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 229, no. 3, pp. 183–191, 2015.

[30] C. M. Hobson and J. E. Goff, "Tour de france modeling: 2015 results and comparisons with elite cyclist power data", vol. 147, 2016, pp. 607–612.

[31] C. M. Hobson and J. E. Goff, "Using the 2011-16 Tours de France to refine prediction model and elicit racing strategies", *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 231, no. 3, pp. 232–240, 2017.

[32] Procyclingstats, [Online]. Available: `https://www.procyclingstats.com/` (visited on 01/21/2019).

[33] UCI. (Feb. 5, 2019). Uci cycling regulations, Part 2 road races. p.65, UCI Union Cycliste Internationale, [Online]. Available: `https://www.uci.org/docs/default-source/rules-and-regulations/part-ii---road-races.pdf` (visited on 04/06/2019).

[34] Velowire, [Online]. Available: https://www.velowire.com/blogcat/35/en/open-street-maps-google-maps-google-earth.html (visited on 02/10/2019).

[35] Radtoto, [Online]. Available: http://www.radtoto.com/index.php?na=3100 (visited on 02/13/2019).

[36] Gpsies, [Online]. Available: https://www.gpsies.com/ (visited on 02/04/2019).

[37] Ridewiththegps.com, [Online]. Available: https://ridewithgps.com/ (visited on 02/04/2019).

[38] Google elevation api, [Online]. Available: https://developers.google.com/maps/documentation/elevation/start.

[39] World weather online, [Online]. Available: https://www.worldweatheronline.com/developer/api/historical-weather-api.aspx.

[40] (2019). The state of the octoverse: Machine learning, [Online]. Available: https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/.

[41] Numpy, [Online]. Available: https://www.numpy.org.

[42] Pandas, [Online]. Available: https://pandas.pydata.org.

[43] Scipy, [Online]. Available: https://www.scipy.org.

[44] Scikit-learn, [Online]. Available: https://scikit-learn.org.

[45] H. Duzan and N. S. B. M. Shariff, "Ridge regression for solving the multicollinearity problem: Review of methods and models", *Journal of Applied Sciences*, vol. 15, no. 3, p. 392, 2015.

[46] L. Firinguetti, G. Kibria, and R. Araya, "Study of partial least squares and ridge regression methods", *Communications in Statistics-Simulation and Computation*, vol. 46, no. 8, pp. 6631–6644, 2017.

[47] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms", in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, 2016, pp. 1310–1315.

[48] V. Prasath, H. A. A. Alfeilat, O. Lasassmeh, and A. Hassanat, "Distance and similarity measures effect on the performance of k-nearest neighbor classifier - a review", *arXiv preprint arXiv:1708.04321*, 2017.

[49] Y. Zhang and A. Haghani, "A gradient boosting method to improve travel time prediction", *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 308–324, 2015.

[50] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial", *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[51] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques", *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[52] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: An analysis and review", *International journal of forecasting*, vol. 16, no. 4, pp. 437–450, 2000.

[53] T. C. Hesterberg, "What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum", *The American Statistician*, vol. 69, no. 4, pp. 371–386, 2015.

[54] World geodetic system, [Online]. Available: https://en.wikipedia.org/wiki/World_Geodetic_System (visited on 04/20/2019).

[55] Pymap3d, [Online]. Available: https://scivision.github.io/pymap3d/ecef.html#pymap3d.ecef.geodetic2ecef (visited on 04/21/2019).

[56] Wiki air density, [Online]. Available: https://en.wikipedia.org/wiki/Density_of_air#Humid_air (visited on 04/22/2019).