Machine Learning Engineer Nanodegree

Capstone Project - Predicting Functional Threshold Power (FTP)

James Kinley, October, 2018

I. Definition

Project Overview

Professional and amature cyclists alike use power output (watts) as a primary measure of performance. The key power metric that is used to assess fitness, calibrate training levels, and pace races is Functional Threshold Power (FTP), which is a measure of a cyclists sustainable power over a period of time[1].

More specifically, from a physiological perspective, a cyclists FTP sits at the point where the amount of lactate produced by their muscles is fractionally below their bodies ability to remove it[2]. This is also known as Lactate Threshold (LT). Just below the LT, the body is able to balance energy supply versus demand. Whereas just above it, excess lactate will build up and performance will deteriorate. Accurately measuring FTP is important because it is used to optimise training levels and specific sessions can be targeted at a cyclists FTP with the goal of increasing it.

This project evaluates the application of machine learning for predicting an athlete's FTP using data collected from previous training sessions.

Problem Statement

There are multiple ways to determine FTP, but the easiest way is to take a formal FTP test. This can be done in a sports science lab, or there are a number of fitness applications that include specific FTP tests, and paying customers can take the tests whenever they like[3]. However, in practice the test is typically only taken at the start of every training block and incremental changes in FTP are not measured or used for recalibration until a test is retaken. In addition to this, an athlete is not able to predict what effect their training will have on their FTP in the future. At its peak, cycling is all about marginal gains and being able to predict changes in FTP based on current and planned training without having to take the test is a competitive advantage.

Modern athletes are generally couch data analysts and they record all of their training data. Fitness applications (such as Strava) capture an array of ride related metrics including:

- Cadence (pedal revolutions per minute)
- Heart rate
- Elevation (accent and decent)
- Power output (measured in watts)
- Speed

[1] https://blog.trainerroad.com/what-ftp-really-means-to-cyclists/
[2] https://www.trainingpeaks.com/blog/what-is-lactate-and-lactate-threshold/
[3] https://support.trainerroad.com/hc/en-us/articles/201993760-FTP-Testing-The-Cornerstone-of-Training

Strava stores this data forever and provides an export function so that users can download their entire history. The download includes a zipped folder containing all of a users previous activities stored in FIT files. The Flexible and Interoperable Data Transfer (FIT) format is designed for the storing of data that originates from sport, fitness, and health devices[4].

Multiple Strava exports have been acquired from amature athletes to use as input for the project. The exports include hundreds of rides, ranging from short high-intensity rides performed on indoor bikes (turbo trainers), to long multi-hour rides over mixed terrain. All athletes are experienced cyclists and ride multiple times a week.

The solution is to extract and process the cycling-related FIT files from the Strava exports and prepare the metrics for training a machine learning model. The hypothesis is that a combination of the metrics, trended over time can be used to accurately predict a cyclists FTP, without them having to take a formal FTP test. This in turn will enable continuous calibration of training levels to optimise performance.

Metrics

FTP (the dependent variable) is a natural number from a small range (100-500 watts). Predicting this value is a regression problem and three common regression metrics will be used to quantify the performance of the solution:

- **Mean squared error (MSE)**: the average squared difference between the estimated and actual FTP values

$$MSE(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2.$$

- **R^2 score (coefficient of determination)**: the proportion of the variance in the observed FTP values (dependent variable) that is predictable from the given performance metrics (independent variables). I.e. how well future samples are likely to be predicted by the model

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{samples}-1} (y_i - \bar{y})^2}$$

- **Adjusted R^2 score**: the R^2 score (above) suffers from a spuriously increasing score as more independent variables are added to the model. Consequently, a model with more features may appear to have a better fit simply because it has more features. The adjusted R^2 scores takes the number of features into account, and penalises the score when an independent variable has little effect

$$\bar{R}^2 = 1 - (1 - R^2) \frac{n-1}{n-p-1}$$

[4] https://www.thisisant.com/developer/ant/ant-fs-and-fit1/

II. Analysis

Data Exploration

A Python script was written (**strava_export.py**) to extract the relevant files from the Strava exports. Each export includes:

- **profile.csv**: athlete related data (e.g. sex and weight)
- **activities.csv**: list of the athlete's activities, including names of associated FIT files
- **activities**: directory containing the athlete's FIT files

The script extracts the cycling-related FIT files and processes them using the **python-fitparse** library[5]. The athlete data is combined into one Pandas dataframe, and the ride data into a second dataframe. To aid data exploration, the 25 ride-related variables, were split into 4 groups.

Cadence and heart rate

Cadence is the number of revolutions per minute (rpm) an athlete can pedal in a given gear.

- 88% of rides have cadence data
- The interquartile range (IQR) is narrow, but within expected values
- A maximum cadence of 246 is physically difficult to reach, so this is possibly a sensor error or other data anomaly. Addressed by removing the 134 outliers
- Cadence variables have minimal skew

Heart rate is measured in beats per minute (bpm). Not all athletes wear a heart rate monitor during every ride, if at all, so the data is expected to be sparse.

- 54% of rides have heart rate data
- Statistics are sensible and within expected ranges. A maximum heart rate of 214 is rare, but not physically impossible
- Marginally higher skew than the cadence variables

	avg_cadence	max_cadence	avg_heart_rate	max_heart_rate
count	550.00	550.00	337.00	337.00
mean	80.05	114.61	140.80	165.27
std	11.54	28.17	16.92	18.80
min	5.00	9.00	0.00	0.00
25%	77.00	100.00	133.00	157.00
50%	82.00	110.00	142.00	168.00
75%	86.00	121.00	149.00	177.00
max	101.00	**246.00**	178.00	**214.00**

	avg_cadence	max_cadence	avg_heart_rate	max_heart_rate
nan	78.00	78.00	291.00	291.00
nan_perc	12.42	12.42	46.34	46.34
skew	-2.96	1.45	**-3.30**	**-4.20**
outliers	98.00	**134.00**	0.00	0.00

[5] https://github.com/dtcooper/python-fitparse

The histograms below show how the cadence and heart rate data is distributed. The horizontal axes represents the rpm and bpm bins respectively.

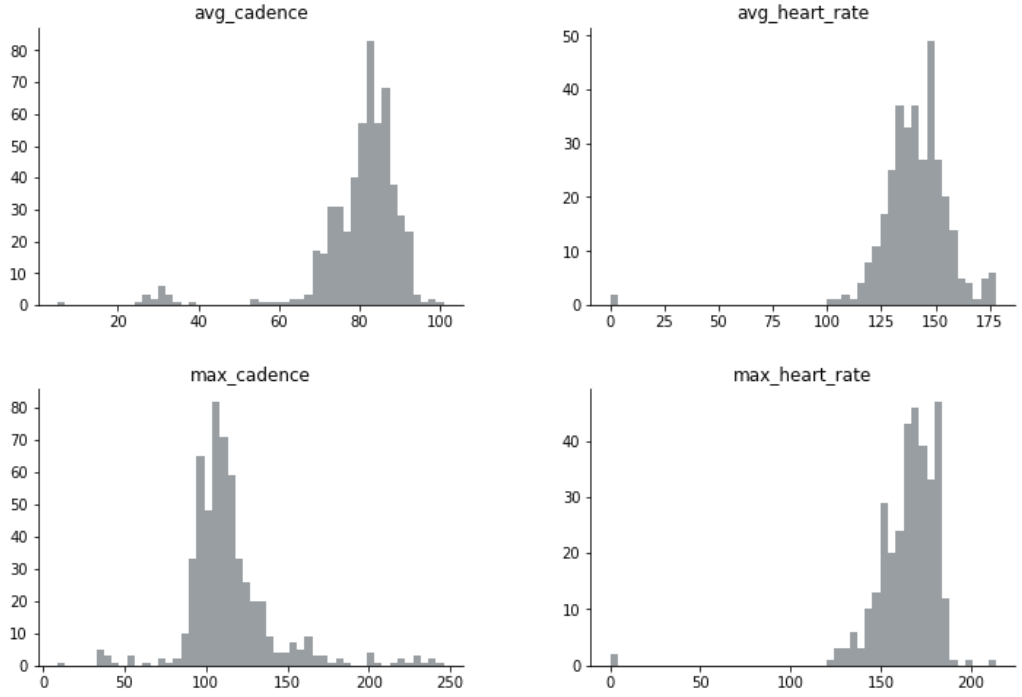

Figure 1. Cadence & Heart Rate Distribution

Power and stress

Power data is collected from a power meter (strain gauge) attached to a bike, or indoor "turbo" trainer. Not all athletes have a power meter so poor coverage was expected across the Strava data. More specific cycling applications, such as TrainerRoad, rely on power numbers and will have better coverage.

- 47% of rides have power data
- Maximum power is highly skewed due to the 9 outliers (a maximum power of 22,276 watts is physically impossible). Address by removing the 9 outliers
- Normalised power is an important variable. Also known as weighted average power, it takes ride variation into account (e.g. terrain, intervals, rest, etc) to provide a better estimate of average power, as if the power output was constant for the entire ride. Normalised power provides the ability to compare effort between rides
- Considering that the dataset includes accurate FTP values (i.e. the dependent variable) Strava's estimated threshold power variable can be omitted

	avg_power	max_power	normalized_power	threshold_power
count	295.00	295.00	295.00	288.00
mean	175.06	615.34	192.79	224.95
std	33.26	1781.05	36.48	48.83
min	1.00	64.00	6.00	200.00
25%	162.00	293.50	176.00	200.00
50%	175.00	365.00	193.00	200.00
75%	189.00	584.50	208.00	245.25
max	297.00	**22276.00**	473.00	400.00

	avg_power	max_power	normalized_power	threshold_power
nan	333.00	333.00	333.00	340.00
nan_perc	53.03	53.03	53.03	54.14
skew	-0.73	**11.07**	0.86	2.46
outliers	0.00	**9.00**	0.00	0.00

The histograms below show how the power variables are distributed. The horizontal axes represents the power in watts.

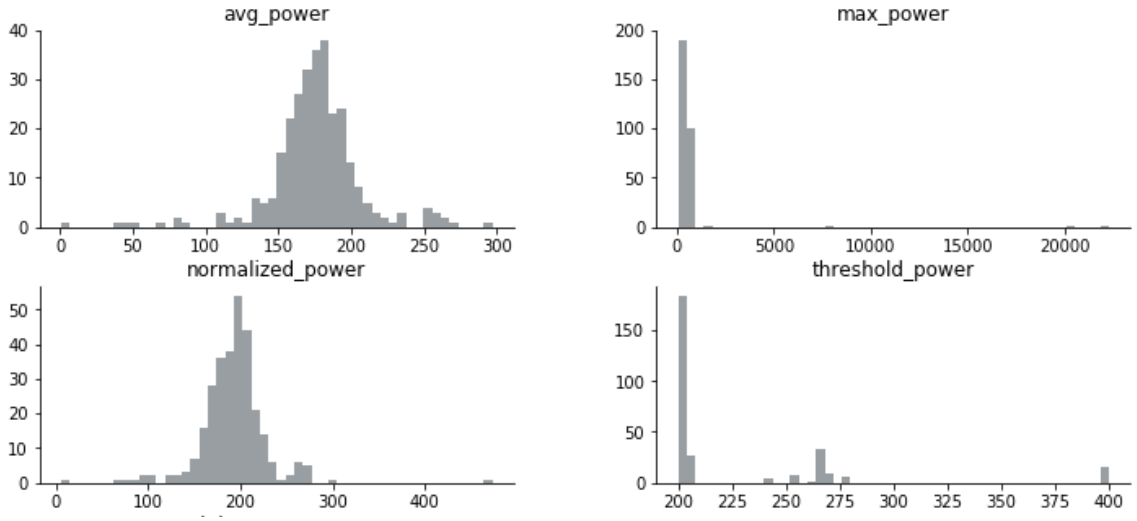

Figure 2. Power Distribution

The training stress score (TSS) is a measure of how hard a ride was. The higher the TSS, the more potential fitness gained from the ride, and the greater need for recovery afterwards[6].

- TSS is highly skewed due to the 27 outliers. Address by removing the 27 outliers

	training_stress_score
count	293.00
mean	111.14
std	152.81
min	0.00
25%	52.20
50%	87.80
75%	141.50
max	2388.70
nan	335.00
nan_perc	53.34
skew	**11.57**
outliers	**27.00**

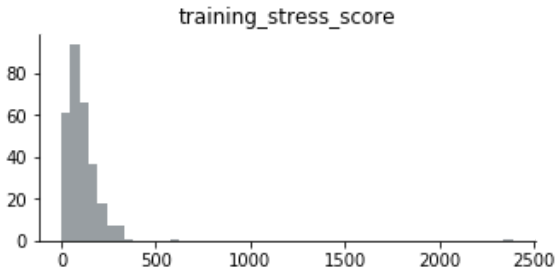

Figure 3. TSS Distribution

Speed

- Speed is reported in miles per hour (mph)
- Good coverage and within expected ranges
- Minimal skew
- Average speed has 124 outliers that were removed
- Maximum speed has 113 outliers that were removed
- Enhanced speed variables are the same as their unenhanced counterparts and have been omitted

	avg_speed	enhanced_avg_speed	enhanced_max_speed	max_speed
count	628.00	628.00	529.00	529.00
mean	6.49	6.49	14.06	14.06
std	3.39	3.39	3.53	3.53
min	0.00	0.00	0.00	0.00
25%	6.11	6.11	12.32	12.32
50%	7.53	7.53	14.63	14.63
75%	8.49	8.49	16.19	16.19
max	18.06	18.06	38.15	38.15

	avg_speed	enhance_avg_speed	enhance_max_speed	max_speed
nan	0.00	0.00	99.00	99.00
nan_perc	0.00	0.00	15.76	15.76
skew	-0.98	-0.98	-0.64	-0.64
outliers	124.00	124.00	113.00	113.00

[6] https://www.trainingpeaks.com/blog/normalized-power-intensity-factor-training-stress/

The histograms below show how the speed variables are distributed. The horizontal axes represents the speed in mph.

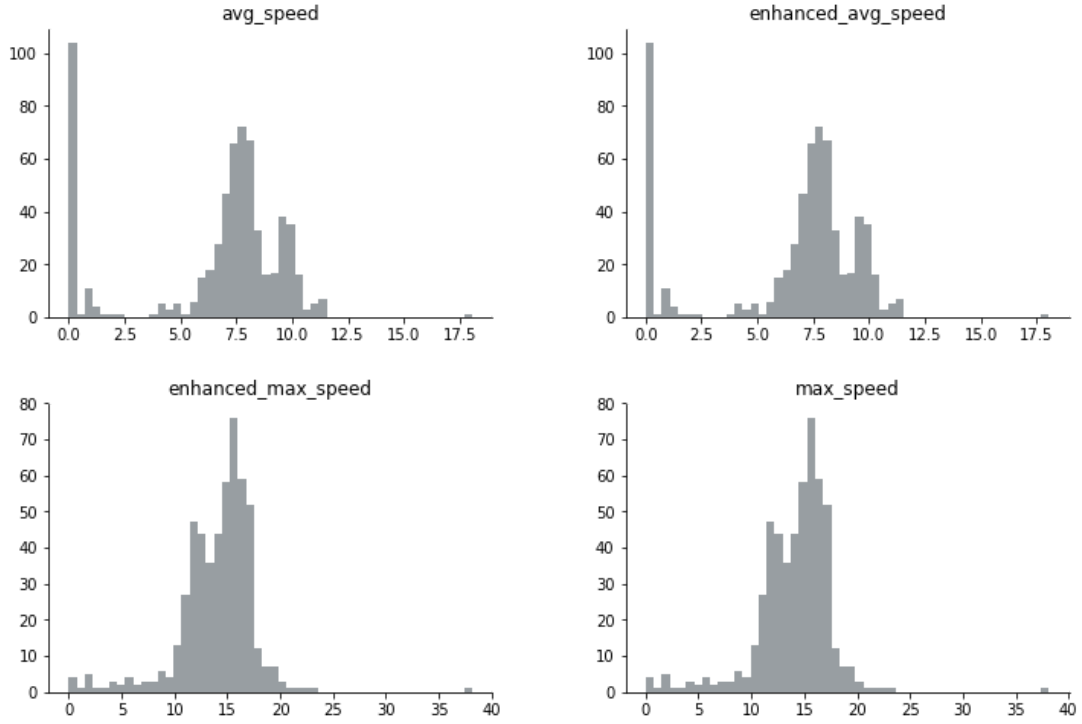

Figure 4. Speed Distribution

Totals

- The total variables have good coverage, are within expected ranges, and have minimal skew
- There are a small number of outliers that were removed
- The ascent and descent variables have a large population of zero examples due to indoor rides
- Total fat calories contains no valuable data and was omitted

	total_ascent	total_descent	total_cycles	total_calories
count	553.00	551.00	506.00	605.00
mean	338.47	339.25	6802.61	1162.14
std	376.38	377.48	4703.19	943.41
min	0.00	0.00	1.00	0.00
25%	2.00	2.00	3671.25	555.00
50%	213.00	208.00	5414.50	852.00
75%	590.00	596.00	8541.25	1500.00
max	2222.00	2233.00	27438.00	5713.00

	total_ascent	total_descent	total_cycles	total_calories
nan	75.00	77.00	122.00	23.00
nan_perc	11.94	12.26	19.43	3.66
skew	1.14	1.19	1.35	1.73
outliers	7.00	7.00	23.00	44.00

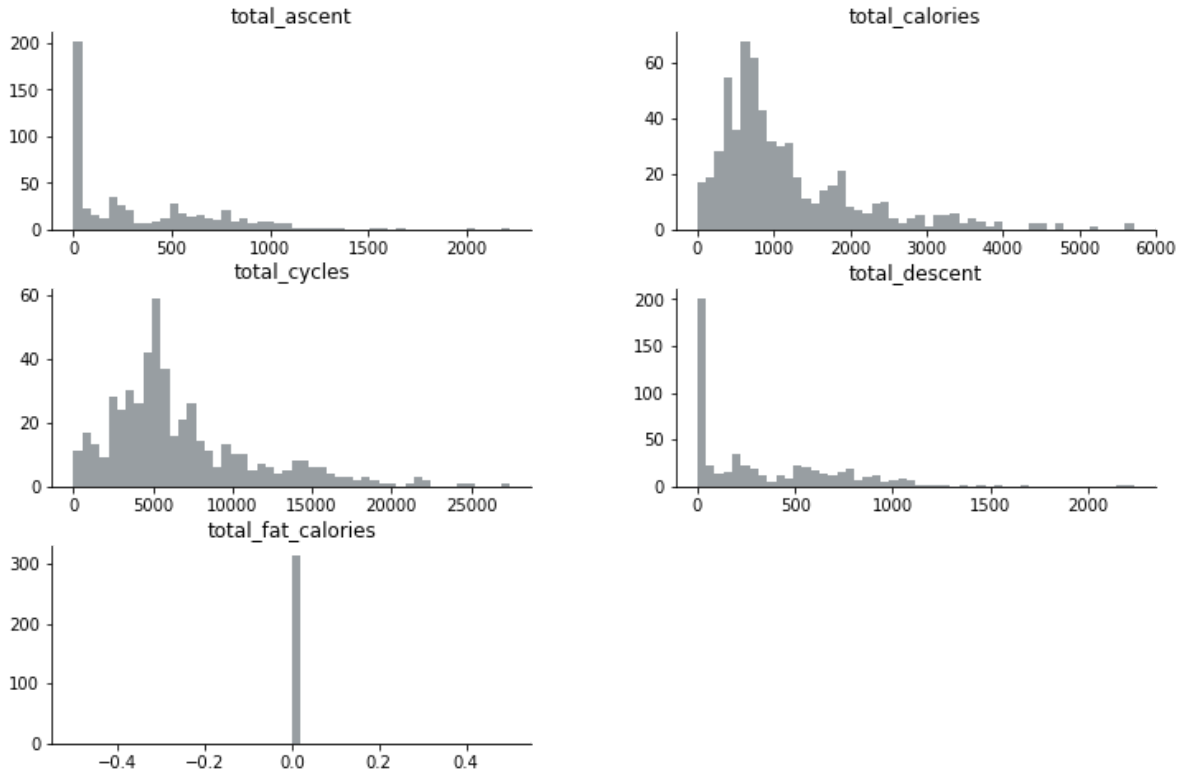

Figure 5. Totals Distribution

- Total elapsed time is highly skewed due to the 27 outlying values, which were removed
- Total distance and total timer time also have outlying values that were removed, but these variables have minimal skew

	total_distance	total_elapsed_time	total_timer_time
count	628.00	628.00	628.00
mean	38136.23	16923.31	5571.79
std	33924.77	98511.40	4080.81
min	0.00	0.67	0.67
25%	9041.35	2987.90	2734.45
50%	31701.35	4200.18	4091.18
75%	53002.00	8345.99	7190.24
max	164129.56	1721086.11	27256.93

	total_distance	total_elapsed_time	total_timer_time
nan	0.00	0.00	0.00
nan_perc	0.00	0.00	0.00
skew	1.01	11.62	1.45
outliers	17.00	27.00	25.00

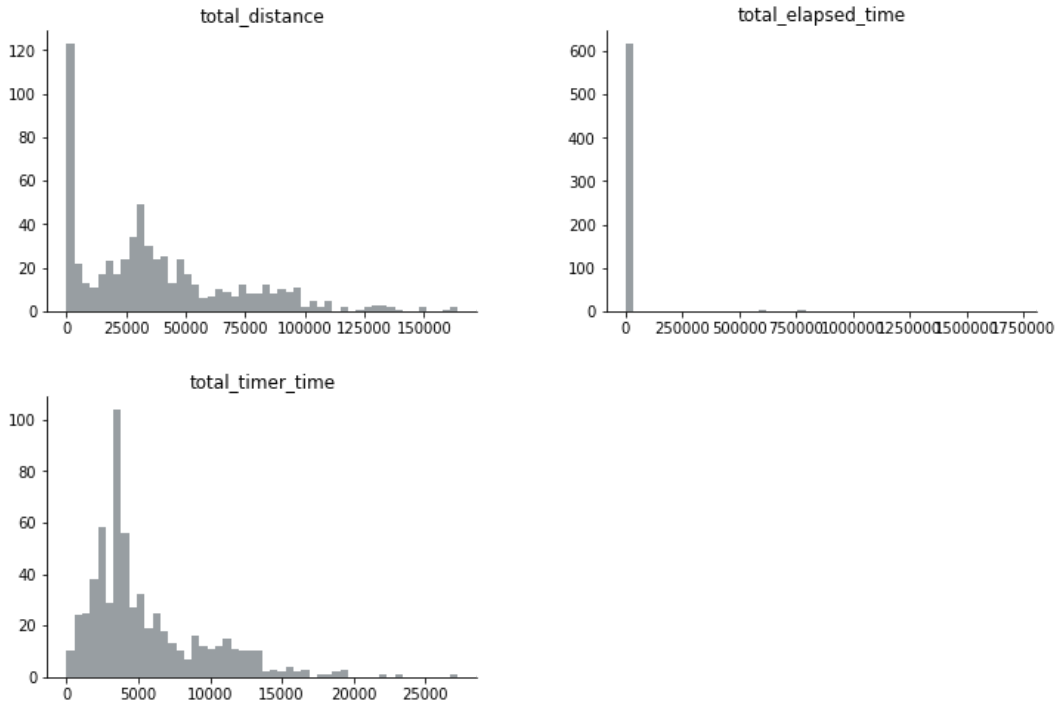

Figure 6. Totals Distribution

Exploratory Visualization

As described above, TSS is a measure of ride difficulty, and a priori is that a consistently high TSS over many weeks, combined with the right amount of recovery, will result in optimal fitness gains. A high TSS can be achieved on short high-intensity rides, but in general it is the long hilly rides that result in the highest TSS.

The scatter matrix below shows the correlation between ride duration (total timer time), ascent, distance, and TSS. There is a strong correlation between ride duration, ascent, and distance. This is expected, the longer the ride, the further you travel, and the more hills you have to climb. There is also a good correlation between TSS and the other variables, which gives support to the prior and validates these are good features to include in the model.

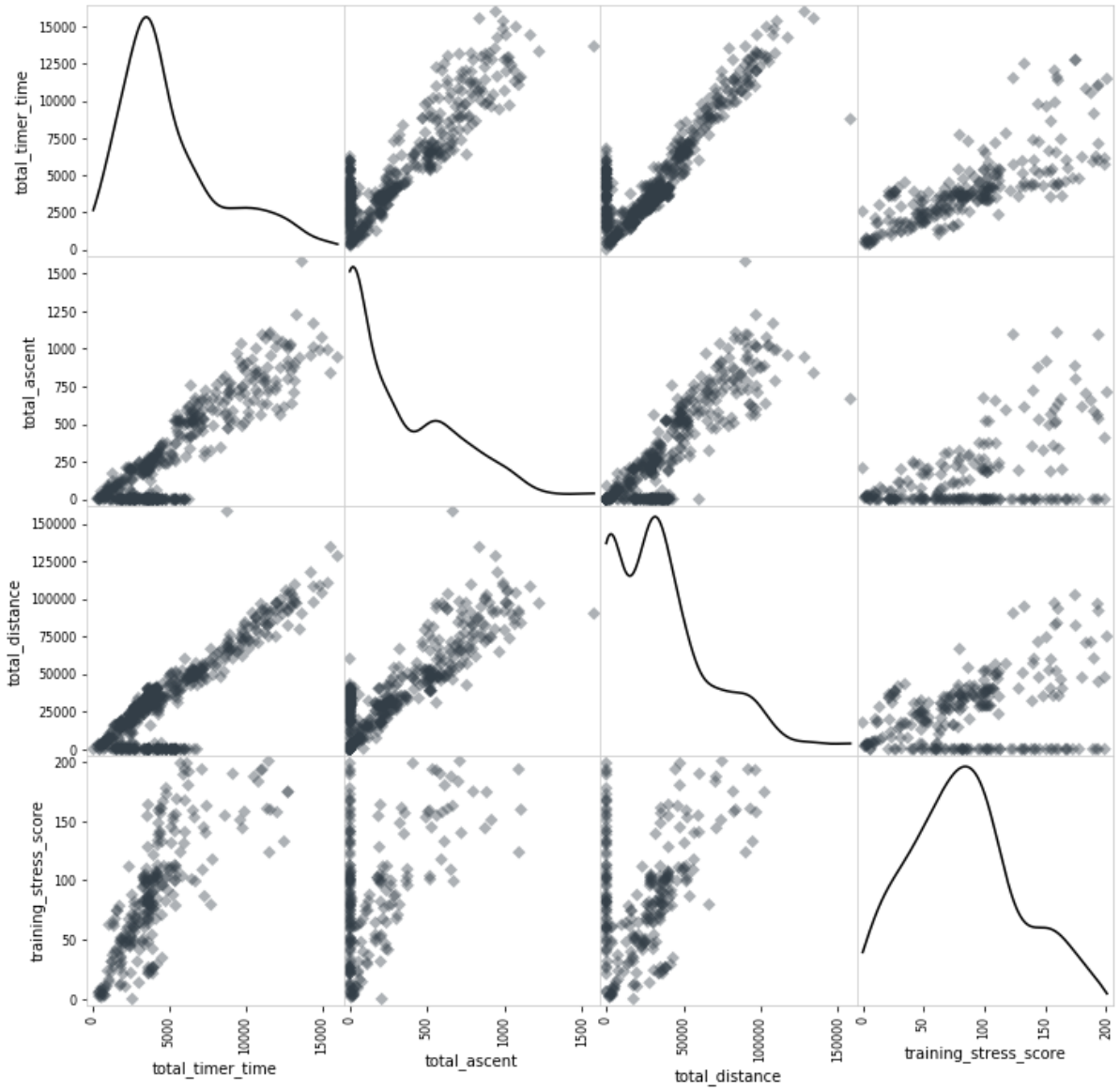

Figure 7. Duration Scatter Matrix

Algorithms and Techniques

Predicting FTP is a linear regression problem. FTP is the target (dependent) variable which is expected to be a linear combination of the input (independent) variables. In this case, the ride metrics. Put simply, linear regression models fit a line through the observed data. The fitting process estimates a set of parameters (coefficients) that represent the best fitting line, which can then be used to predict the value of the target variable for new observations. Four generalised linear regression models were evaluated for this project:

Ordinary Least Squares (OLS)

OLS estimates the coefficients by minimising the residual sum of squares between the observations and the predicted line.

Ridge Regression (L2 regularisation)

Ridge Regression also minimises the residual sum of squares, but includes a regularisation parameter $\boldsymbol{\alpha}$ that imposes a penalty on the size of the coefficients. Increasing the value of $\boldsymbol{\alpha}$ shrinks the value of the coefficients, which results in the model being less prone to overfitting.

Lasso Regression (L1 regularisation)

Lasso Regression takes a similar approach to Ridge in that it also uses a regularisation parameter $\boldsymbol{\alpha}$ to penalise large coefficients. The key difference is Lasso has built-in variable selection. Lasso's coefficients are laplace distributed (as opposed to normally distributed) which means that it shrinks the less important variable coefficients to zero, effectively eliminating them from the model.

Bayesian Regression

This model takes a Bayesian approach to the linear regression problem[7]. Bayesian Regression includes two regularisation parameters $\boldsymbol{\alpha}$ (precision of the noise) and $\boldsymbol{\lambda}$ (precision of the coefficients). Unlike the other models, the parameters are not fixed and instead they are given as priors in the form of probability distributions. A third parameter $\boldsymbol{w}$ is given for the prior probability distribution of the coefficients. $\boldsymbol{w}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\lambda}$ are optimised during the model fitting process using the observed data.

Benchmark

Formal FTP test results provided by the TrainerRoad cycling application will be used to benchmark model performance. TrainerRoad provides three formal FTP tests:

- **Ramp test**: uses gradual increases in target power until the rider is no longer able to match, typically within 15-20 minutes. FTP is calculated as 0.75 of the highest 1-minute power[8]
- **20-minute test**: 20-minute time trial. FTP is calculated as 0.95 of the average power[9]
- **8-minute test**: Two 8-minute intervals. FTP is calculated as 0.90 of the average power over the two intervals

Data shared by the athletes include formal FTP results, recorded at different times, using one of the three formal methods above.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.BayesianRidge.html
[8] https://support.trainerroad.com/hc/en-us/articles/360003910912-FTP-How-to-Test
[9] https://support.trainerroad.com/hc/en-us/articles/115003401846--FTP-How-it-s-Calculated

III. Methodology

Data Preprocessing

The data exploration step identified a number of preprocessing steps that were required to prepare the data ready for model training:

- Redundant variables were removed: **enhanced_avg_speed**, **enhanced_max_speed**, **threshold_power**, **total_fat_calories**
- Tukey's method was used to identify and remove outliers
    - Tukey's method takes the lower (Q_1) and upper (Q_3) quartiles of a variables range and defines an outlier to be an observation outside of the range:

      $\boldsymbol{k} = 1.5$
      $[Q_1 - \boldsymbol{k}(Q_3 - Q_1), Q_3 + \boldsymbol{k}(Q_3 - Q_1)]$

- Three variables were discovered to have significant positive skew which was being caused by outliers. Removing the outliers significantly reduced the skew, and avoided the need to apply a log-transformation

	skew_before	skew_after
max_power	11.07	0.80
total_elapsed_time	11.57	1.20
training_stress_score	11.62	0.37

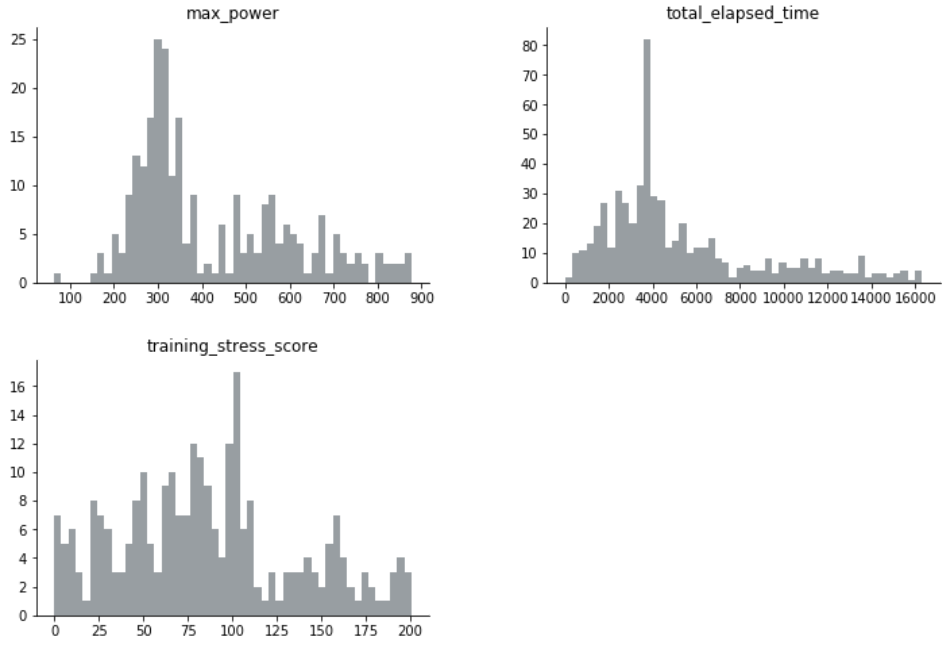

Figure 8. Unskewed Variables

- Finally the variables are standardised by removing the mean and scaling to unit variance

Implementation

During the implementation of the model it quickly became apparent that further data preparation was needed. The formal tests described in the benchmark section estimate FTP using the data captured from a single ride. However, these rides are specifically designed to push athletes to their limit in order to capture the required data at that point in time.

This is not representative of the observations in the prepared dataset, and this is by design. The aim of this project is to predict FTP at any point in time, based on the data captured from a single ride, and the cumulation of previous rides over a given period.

To address this problem an algorithm named **Deep Feature Synthesis (DFS)**[10] was used. DFS automates the feature engineering process by generating new statistical features from a given dataset. In this case, DFS was applied to the preprocessed dataset to generate new features that capture the desired cumulation of rides for each athlete. The DFS algorithm is implemented by the Python library **Featuretools**[11].

One important aspect of DFS is its ability to handle time[12]. The preprocessed data includes a timestamp for every ride and DFS uses it as a cutoff point when generating new features. For example, if a particular ride was recorded on 2018-11-22 then DFS will only count the athletes other rides prior to that timestamp when generating new features. DFS also provides the ability to set a training window to limit the amount of historical data that will be included prior to a given cutoff time. In this case, a training window of six months was specified, meaning that the models predictive capacity is based on the data collected from a single ride and an aggregation of the athletes efforts over the previous six months.

Below is a sample of the generated features chosen from the final set of 59:

```
1. <Feature: intensity_factor>
2. <Feature: normalized_power>
3. <Feature: total_ascent>
4. <Feature: total_distance>
5. <Feature: total_timer_time>
6. <Feature: training_stress_score>
7. <Feature: training_stress_score > 141.5>
8. <Feature: athletes.AVG_TIME_BETWEEN(rides.timestamp)>
9. <Feature: athletes.COUNT(rides)>
10.<Feature: athletes.MEAN(rides.intensity_factor)>
11.<Feature: athletes.MEAN(rides.normalized_power)>
12.<Feature: athletes.MEAN(rides.total_ascent)>
13.<Feature: athletes.MEAN(rides.total_distance)>
14.<Feature: athletes.MEAN(rides.total_timer_time)>
15.<Feature: athletes.MEAN(rides.training_stress_score)>
16.<Feature: athletes.TREND(rides.intensity_factor, timestamp)>
17.<Feature: athletes.TREND(rides.total_ascent, timestamp)>
18.<Feature: athletes.TREND(rides.training_stress_score, timestamp)>
19.<Feature: athletes.TREND(rides.normalized_power, timestamp)>
20.<Feature: athletes.TREND(rides.total_distance, timestamp)>
21.<Feature: athletes.TREND(rides.total_timer_time, timestamp)>
22.<Feature: athletes.PERCENT_TRUE(rides.training_stress_score > 141.5)>
```

[10] http://www.jmaxkanter.com/static/papers/DSAA_DSM_2015.pdf
[11] https://docs.featuretools.com/index.html
[12] https://docs.featuretools.com/automated_feature_engineering/handling_time.html

The feature matrix is split into training (75%) and testing sets (25%). The training set is used to fit models and the testing set is used to quantify model performance. OLS was selected as the first model to evaluate because it does not have any parameters to optimise.

- Number of training examples: 429
- Number of test examples: 143
- Number of features: 59

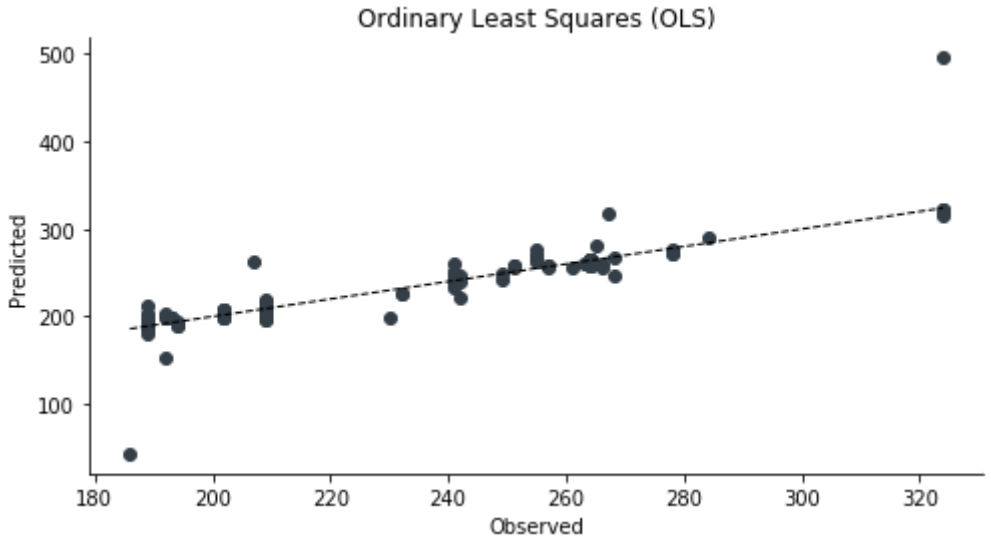

Figure 9. OLS Performance

Metric	OLS
Mean squared error	450.30
R^2 score	0.65
R^2 (adjusted) score	0.40

Refinement

The next step was to compare the performance of the initial OLS solution versus the other linear regression models. As discussed in **Algorithms and Techniques**, both the Ridge and Lasso algorithms include a regularisation parameter α that applies a penalty to the fitted coefficients:

- If the regularisation is too weak then the model can be prone to overfitting the training data and it will not perform well on new observations
- Likewise, if the regularisation is too strong then the coefficients will be shrunk excessively, causing the model to underfit the training data, also reducing its predictive capacity
- Lasso has the added risk that if the regularisation is too strong then important variables may be left out of the model altogether

There are multiple ways to find the optimal value for α. Scikit Learn's **GridSearchCV** method could have been used, but the easiest solution was to use **RidgeCV** and **LassoCV**

respectively. These implementations find the best value for α from a given array of values using built-in cross-validation over multiple folds of the training data. For example:

```
ridge = linear_model.RidgeCV(alphas=np.linspace(0.5, 10., num=20), cv=5)
```

As the table below shows, Ridge and Lasso Regression with optimised values for α perform significantly better than the unregularised OLS algorithm. In this case, Ridge marginally outperforms Lasso.

Metric	Ridge	Lasso
Mean squared error	168.12	186.17
R^2 score	0.87	0.86
R^2 (adjusted) score	0.78	0.75
Best α	1.00	0.50

The final model to be evaluated is Bayesian Regression. This probabilistic approach to linear regression uses default probability distributions as prior values for it parameters, and the optimal values are estimated during the model fitting process. Bayesian Regression gives the best performance out of the four models tested and is used as the final model.

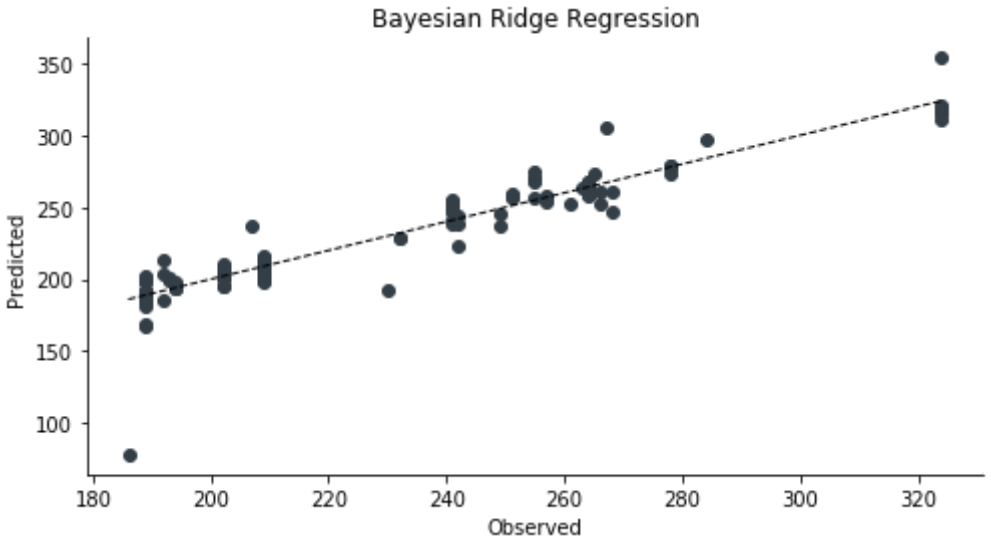

Figure 10. Bayesian Regression Performance

Metric	Bayesian
Mean squared error	164.11
R^2 score	0.87
R^2 (adjusted) score	0.78
Best α	0.0145
Best λ	0.0037

IV. Results

Model Evaluation and Validation

The final model uses the same regularisation algorithm as applied in Ridge Regression, so it is not a surprise that the performance of the Bayesian and Ridge models is similar.

Bayesian was chosen as the final model because of its ability to self-optimise its parameters. The model accepts prior knowledge about the problem and adapts its knowledge to the given observations. It can also continue to fine-tune its parameters given more observations over time. In practice, model performance met expectations using the default priors and relatively few observations. A test set was used to evaluate performance and it generalises well to unseen data.

Another advantage of the Bayesian model is its output is a distribution of probable values. Its **predict** method returns the mean of the distribution (most likely value) and the standard deviation. The standard deviation can be used to quantify the model's confidence in the prediction. A large standard deviation indicates a wider distribution, which in turn represents a greater uncertainty in the prediction (the probable values are spread out).

The robustness of the model was evaluated by training and testing using different splits of the dataset:

- The alpha and lambda parameters fluctuate by small amounts
- These perturbations have a small effect on the results
- The final model appears to be relatively sensitive to changes in the training data
- Further testing of the model is required before the results can be trusted

Justification

Figure 11 below plots one athlete's observed versus predicted FTP values. The lighter green boundary around the predicted line shows the model's confidence (standard deviation in the posterior distribution).

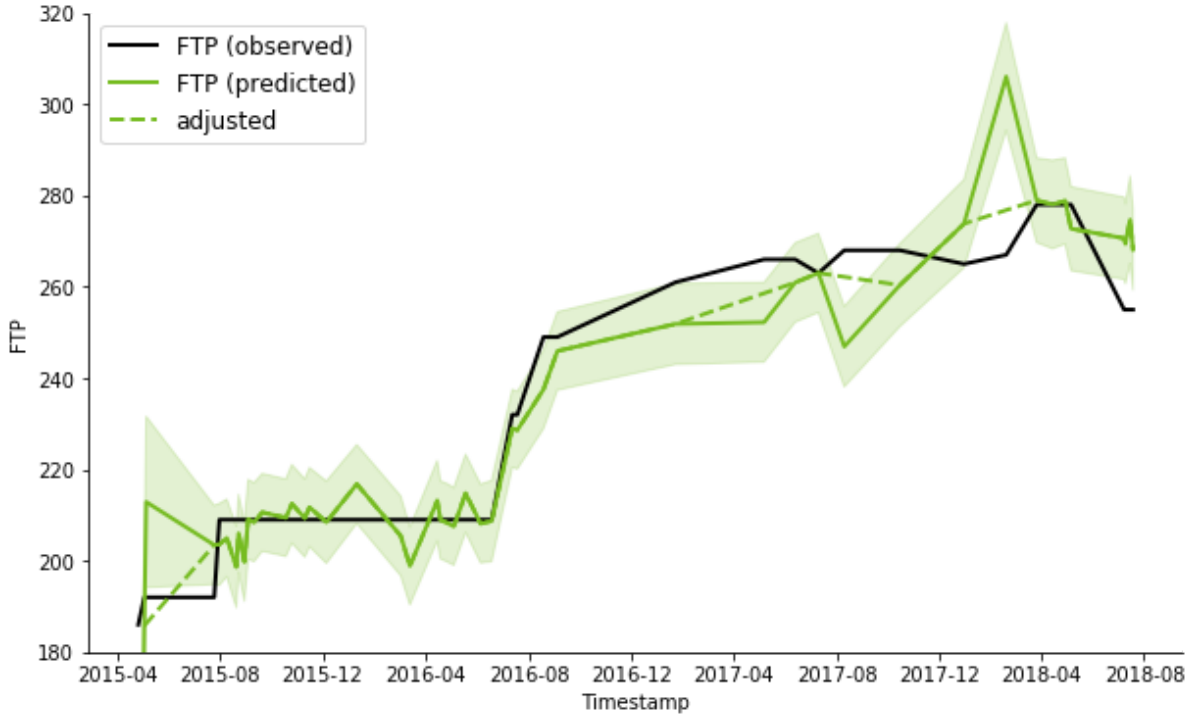

Figure 11. Athlete FTP Predictions

- The model is less confident to begin with, but it gains in confidence as more observations are seen
- There are four anomalous predictions that distort the prediction line. The adjusted prediction line is show by the green dotted line
- Out of the 143 rides from the test set, the 90th percentile of predictions are within 5% (11 watts) of the observations

For comparison, **figure 12** below plots the predictions for a second athlete. In this example, the prediction line adjusts well to the athletes efforts. Early in 2018 the predicted FTP begins to rise before and inline with the observed FTP recorded at the end of March. Likewise, the predicted FTP decreases towards the end of the year before the formal test is taken at the end of September. Given that there is less data for this athlete, the lower confidence (larger standard deviation) is to be expected.

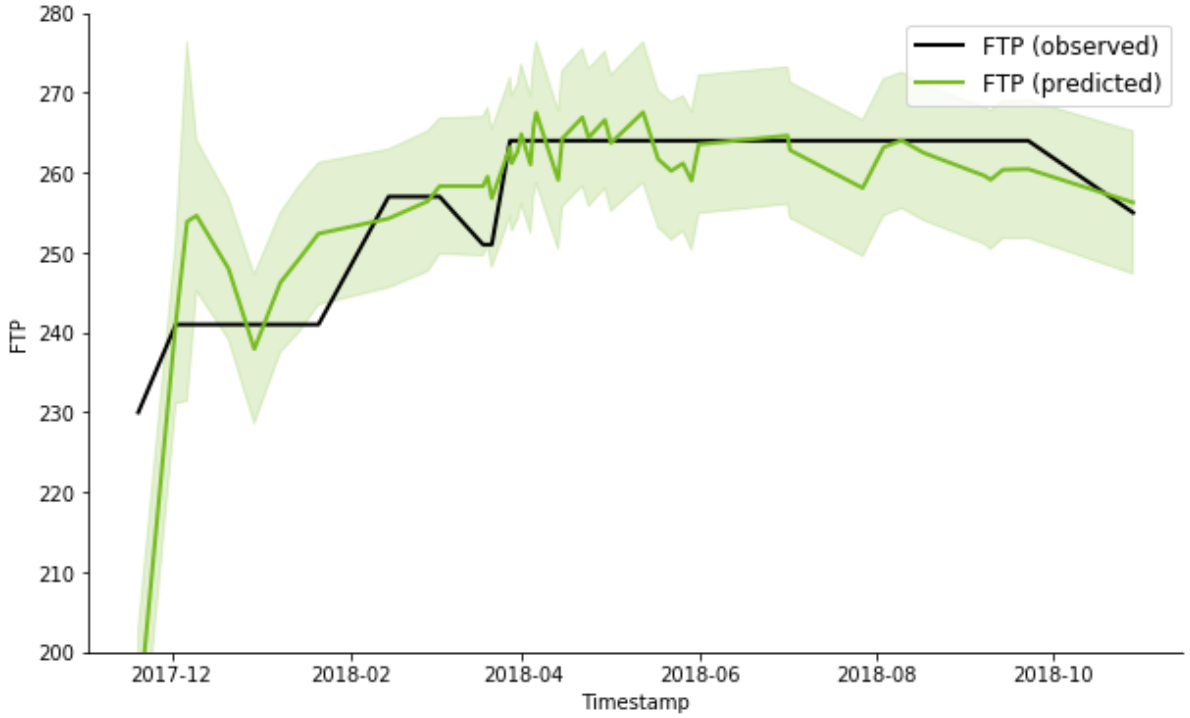

Figure 12. Athlete FTP Predictions

In summary, the initial results look promising. Micro-changes to FTP can be used to optimise training levels in near real-time, rather than having to wait for macro-changes to occur every couple of months as a result of taking formal FTP tests. Given more training and test data the model can be further optimised and should become more robust, increasing confidence in the predictions.

V. Conclusion

Free-Form Visualization

Figure 13 compares the feature coefficient values between the four models tested. It illustrates two important qualities about the dataset:

1. Regularization is needed to avoid overfitting. OLS appears to be trying to capture the noise in the training data hence why some of its coefficients have much larger values. This in turn increases the variance in the model and it does not perform as well on unseen data. The other models implement either Ridge or Lasso regularization which shrinks their coefficients closer or equal to zero. This reduces the variance and results in models that generalise well to unseen data
2. Features that represent the cumulative efforts of an athlete over time are better predictors of FTP. In the figure below, features 1-20 are data points from the current observation (e.g. normalized power generated during the given ride). Whereas features 21-59 capture the cumulate data points over time (e.g. average normalized power for the past 6 months). It is the cumulative features that have the larger coefficient values and thus more influence on the predicted FTP value

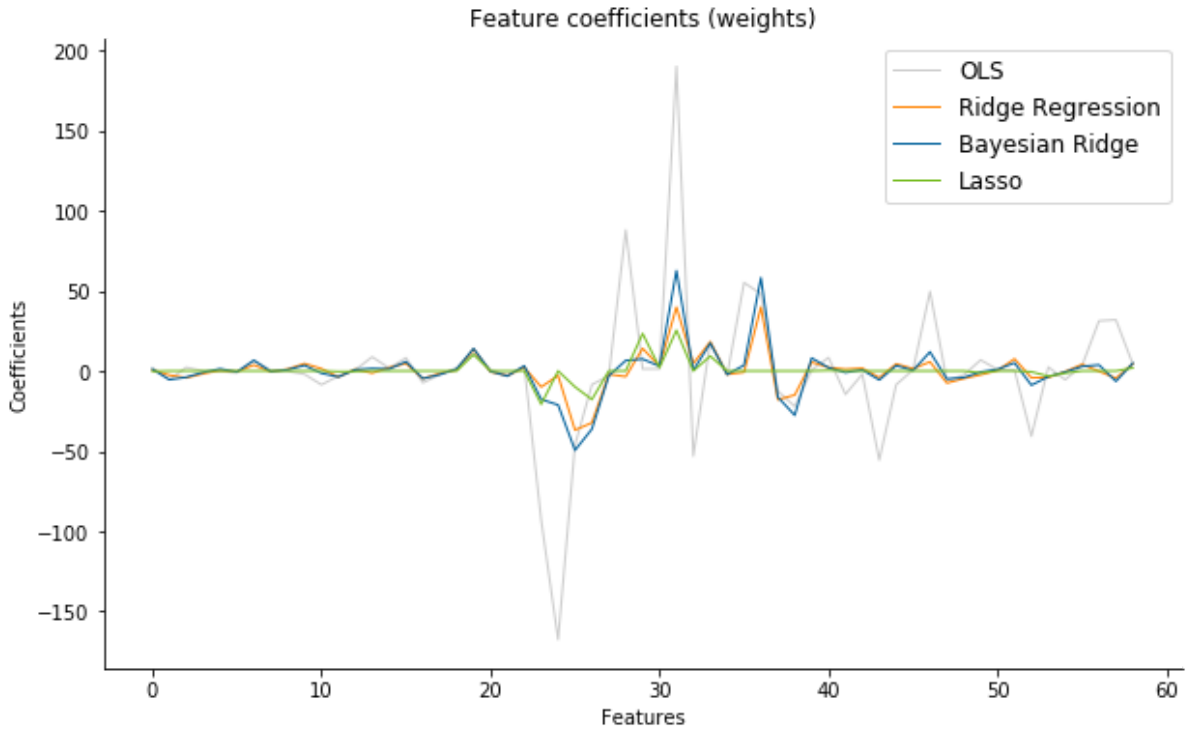

Figure 13. Model Coefficient Comparison

Reflection

The process used for this project can be summarised in the following steps:

1. The initial problem was clearly defined in a proposal
2. Multiple athletes agreed to share their Strava data
3. The cycling-related data was extracted from the Strava exports
4. The data was thoroughly explored and preprocessed
5. Automated feature engineering was applied to the dataset to generate new statistical features based on an athletes history
6. Four linear regression models were selected, evaluated, and optimised
7. The best performing model (Bayesian Regression) was chosen as the final solution
8. The final solution was analysed and documented

The most challenging aspect of the project was data acquisition. Data collected by Strava, and other related application, is private and it was difficult to persuade enough athletes to download and share their data for the purposes of this project. There were two interesting aspects of the project.

1. Learning about the Deep Feature Synthesis algorithm. Automating the feature engineering process using Featuretools has many advantages for machine learning projects
2. Learning how to apply Bayesian thinking to a linear regression problem, including the various types of probability distributions. Bayesian Inference is a powerful algorithm to have in a machine learning toolkit

Improvement

Given the final solution for this project as a benchmark there are a number of improvements that could be made:

- Evaluating the use of PyMC3[13] for implementing Bayesian Inference. PyMC3 has a comprehensive set of probability distribution to test, and its very well documented
- Circa 600 rides is all that could be acquired for this project. The model would benefit from considerably more data from a larger population of athletes. Including data from both sexes, and a wider range of physical characteristics and athletic abilities
- This project has focused on predicting an athlete's FTP using data from current and previous rides. Given an athlete's future training plan and estimates for variables such as duration, intensity, and TSS, the model could be used to predict future FTP values. It would be interesting to test how the future FTP changes, in terms of its value and confidence, as an athlete progresses through their training plan

[13] https://docs.pymc.io/