Monitoring Athletes Performance Through Wearable Devices

Group 13 Members

Lin Mi

Tingli Qiu

Spoorthi Suresh Avvanna

Chitra Naga Durga Sindhusha Veluguleti

Yuhan Zhao

Supervisor

Milad Chenaghlou

Department of Computing and Information Systems,

The University of Melbourne

Client

Ollie Allan

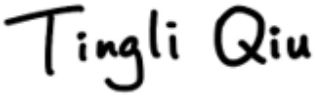
Director and Head Coach, TRI-ALLIANCE

Declaration of Authorship

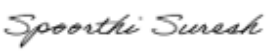
We certify that this report does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of our knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

This report is 11620 words in length (excluding text in images, table, bibliographies and appendices).

Lin Mi

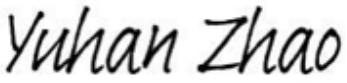


Tingli Qiu



Spoorthi Suresh Avvanna

Chitra Naga Durga Sindhusha Veluguleti



Yuhan Zhao

Abstract

Monitoring athletes is of vital importance to ascertain that the individual training program works well for them. With the advancements in technology, the idea of inculcating scientific techniques to load monitoring in athletes has increased. Data from wearable devices are being used to track their daily practice sessions. The goal of this project is to capture Fatigue, Fitness and Form as metrics in order to determine if the athletes are over-training or under-training. An analytical module was built with an end to end workflow to process the data from Garmin watches and generate Performance Management Charts (PMC) to achieve our desired goal. Machine Learning techniques such as Regularised linear regression, Gradient Boosting and Long Short-Term Memory Networks were built to estimate the Training Stress Scores (TSS) which accounts for the physiological stress and training load of the athletes' workout. Various mathematical formulas by researches such as Joe Friel and Andy Coogan were used to arrive at this.

Acknowledgements

We would like to express our sincere gratitude to our client Ollie Allan, for his constant feedback and support throughout the project. We would also like to extend our thanks to Dr Eduardo Araujo Oliveira for helping us with additional data and providing timely suggestions.

We want to thank our subject co-ordinator Michael Kirley and project supervisor Milad Chenaghlou for their guidance throughout the project.

Finally, we thank our family and friends for their unfathomable encouragement throughout our course of study.

Contents

List of Tables

List of Figures

1 Introduction

With the advancements in technology, the urge to quantify aspects of training to improve the athletes' training cycle has been the holy grail of sports scientists', coaches, and athletes. The main objective of monitoring the data is to improve the athlete's performance by avoiding overtraining and overreaching. Wearable sensors are employed to procure real-time data about the person and the environment. The objective of inculcating scientific approaches in designing the training program is not only to identify the progress of the athlete but also to make frequent modifications to the program for maximising the athletes' performance. Based on the feedback from the monitored data, loads are adjusted frequently to balance the fatigue. The optimization of the training program entirely depends on managing what the athletes do and how they respond to the designed training program. TRIMP is one of the methods used to evaluate the training load, and this method uses training volume or heart rate as a metric. To measure the impact of exercise, Training Stress Score (TSS) is used to evaluate the external load and Performance Management Chart (PMC) is used to track and plan the fitness and fatigue of the athlete. The coaches must safeguard the health of athletes to prevent over-training and under-training, which leads to failures and injuries. While monitoring the performances of athletes, the external and internal loads are considered as well. Internal load involves the psychological state of the athlete. While tracking the training loads, the units can be thought of as either external or internal but traditionally external load is the main base for many tracking devices, and this is independent of internal characteristics.

The aim of this project is to analyse the data from Garmin watches worn by athletes during their training sessions for different activities such as running, swimming, and cycling to determine if they are over-training or under-training.

2 Related work

There are many technologies for tracking the performances of athletes, and these are available commercially in different forms. Some of them are Concussion-Based Wearable sensors, Wearable Device for Anterior Cruciate, ligament injury Mitigation. Continuous modifications in the training program of the athletes ensures that they are achieving their goals. Hence, an adjustment in training loads are mandatory, and this may result in the increase or decrease of the fatigue. Fatigue is perceived as physical or mental tiredness, resulting in significant reductions in sports performance. In general, wearable are installed with IMU for kinematics, GPS for movement records, and heart rate detectors for athletes to record their training activities, by which fatigue prediction is made via either machine learning techniques or expert review on statistics. Here are some exemplary fatigue prediction investigations.

2.1 Over-training or Under-training

2.1.1 Features

Training Load [2][5][6][7][8][9][10] (Our prediction model is based on this) is a general description of how much exercise an athlete has performed during a training session. In detail, training load can be expressed as a collective representation of chronic training load (CTL), intensity factor (IF), training stress balance (TSB), acute training load (ATL), and training stress score (TSS).

Training Stress Score (TSS) is an overall estimate of training load and physiological stress after a training session, by taking into consideration the training duration and intensity. Different exercises have different kinetic patterns that require TSS metrics specifically designed given exercise natures and given different provided measurement equipment (some measure heart rate), TSS can be computed in different ways. Some typical TSSes are

- ***Running Training Stress Score (rTSS)*** [1][8] Step pace is the most significant stress indicator, computed by taking GPS (for longitudinal distance) and IMU (for motion acceleration as well as other kinetic metrics). Running intensity is considered, graded to different physiological cost levels.
- ***Swim Training Stress Score (sTSS)*** sTSS considers classified swim pace, covered swim distance, and exercise duration. Swim pace refers to a functional swimming threshold speed, calculated in accordance with laboratory-measured power output.
- ***Cycling Training Stress Score (cTSS)*** [10] The power meter can be installed on a bike to measure power output, by which cTSS is determined by reading and computing cycling energy output. It can also be evaluated by kinetics, GPS, and heart rate.
- ***Heart Rate Training Stress Score (hrTSS)*** Calculation of hrTSS is based on recorded heart rate

zones, which indicate training stress levels. A great heart rate indicates intense training activity.

- ***TRIMP Training Stress Score (tTSS)*** [8][9][10] TRIMPS stands for Training Impulse. TRIMPS is a comprehensive exercise descriptor that considers training duration and intensity and coined the term "dose" as a single estimate of a session of training, computed by heart rate related metrics and semantic statistics.
- ***Session RPE*** [9] TSS can be directly scored by RPE.

Chronic Training Load (CLT) is a set of exercise records over a long period, such as six weeks, while Acute Training Load (ATL, treated as fatigue) is for a short training period, typical of one week. Training Stress Balance (TSB) is the result of comparing CLT with ATL, e.g., comparison via CLT/ATL ratio or load gap via CLT - ATL.

Heart Rate Variability (HRV) [9] is the most frequently used fatigue indicator and is referenced in all research experiments for fatigue prediction. Several HRV metrics are studied to explore possible correlations between fatigue and HRV: mean, Interval Variance, Absolute value of low- and the high-frequency component of RR variability power and center frequency of high-frequency component.

Kinetic statistics [2] [4] refers to a general concept of an athlete's motion, usually detected by IMU for 1) body and limb acceleration, 2) changes of acceleration, 3) movement speed and by GPS for longitudinal distance recording. Some exercises require repeated motions, e.g., swimming requires consistent strikes.

2.1.2 Predictive Models

Fitness-Fatigue Bannister model [5][6]is a classic descriptor that illustrates the relationship between training and fatigue. It states that immediately after exercise, there is a reduction in sports performance as a positive training response (main fitness after-effect) is late. By reviewing training metrics (TSS) such as training intensity, rest interval, and frequency, an athlete's performance can be predicted, that sports performance responds negatively (interpreted as fatigue) after immediate finished training, and positively after a period of recovery time. By computing CTL and ATL corresponding to fitness and fatigue, respectively, performance enhancement can be expected.

Machine learning techniques are used in supervised training to fit input features (fatigue indicators) to labels (quantified physiological or psychological fatigue).Some examples are *Artificial Neural Network (ANN)* [1], *Elastic Net and LASSO* [1], and *Support Vector Machine (SVM)*, which defines normal and fatigue time zones for different athletes during training [9].

2.1.3 Existing Research Experiments

- **Fatigue prediction for outdoor runners by machine learning** [1]
  Upon collecting kinetic data and derivatives of statistical significance, four machine learning algorithms are employed for prediction: GBRT, ANN, ELASTIC NET and LASSO, where GBRT displays the lowest MAE (Mean Average Error).

- **Cycling fatigue evaluation** [4]
  Heart Rate Variability (HRV) and electrocardiographic (ECG) are significant fatigue indicators and this research attempts to identify feature control parameters to evaluate fatigue by SVM.

- **Training load quantification for fatigue prognosis** [5][6]
  The two research experiments summarise a list of internal and external training load quantifiable indicators, and review Fitness-Fatigue Bannister Model arguing that physiological responses show a strong correlation relationship with training stresses.

- **Cyclist performance evaluation via training load monitoring** [10]
  This study examines relationships between different training load measures and performance variations for well-trained cyclists. It reveals strong correlations between performance changes and training load measures when considering individual athletes' physiological characteristics.

2.2 Performance prediction

Implementation of computerized approaches to athletic performance evaluations and improvements sees increased popularity in recent years. Upon examinations on several academic research papers, students provide an overview of research relative to swimming, running, and cycling.

2.2.1 Swimming:

This report has reviewed the paper by [11], which has used the regression and neural models to predict the competitive swimmer's performance. The model has considered the following body features of the swimmers:

• Body fitness such as lung capacity

• Body strength such as standing long jump

• Anthropometric variables such as hand and foot sizes, body weight and body height, hand length, arm span, Rohrer Coefficient, BMI (body mass index)

• Swimming technique including turn, glide, distance per stroke cycle

• Swimming strategy such as speed in particular distances

The total sample number is 189 swimmers, who has been training for four years. Training including the fundamental body workout and swimming skills training. The athletes' swimming performance will be evaluated for the testing group with 50m and 800m distance, and the completion time will be recorded. The model used for model training is the multidimensional statistical analysis. For generalization and prediction of swimming results, Multilayer Perceptron (MLP) neural models were used to model the 50 m and 800 m performances with the following structures: 4-3-1 and 8-4-1 [four and 8 A. MASZCZYK, et al. eight input neurons (variables), respectively, one hidden layer (with three and four neurons, respectively, and one outcome].

The training and testing data results show non-linearity, and the feature variables were transformed to a linear function. Eventually, the time for the racing in 50m is predicted to be:

$$Y1(50m) = 147.64 - 25.54 * G + 64.41 * FL - 53.24 * BH - 32.43 * 25mAA10.11$$

Here, Y1(50m) is the time required to finish the 50m racing, Where G is the glide, FL is the foot length, BH is the body height, mAA is the 25mAA parameter accounts for the arms n cycles for 25m racing.

For 800m racing, the predicted model is based on 8 parameters:

$$Y2(800m) = 90.43 - 0.27 * 50mLL + 11.71 * VLC + 11.21 * 800mALAS + 10.72 * HL - 18.61 * 25mAA + 5.19 * RC\check{}7.23SLJ\check{}6.45 * AAc8.72$$

Here Y2(800m) is the time required to finish the 800m racing, where 50mLL is the flutter kick with a kickboard, VLC is the vital lung capacity, HL is the hand length, RC is the Rohrer coefficient, SLJ is the standing long jump, AAc is the AA cycle.

The neural model with multiple layers has been approved to be an effective and accurate tool to predict the racing's swimming performance.

2.2.2 Running:

This report reviews the paper by [12], who used XGBoost to predict runners' completion time in the marathon race while extending the model to precisely predict whether the competitors will slow down in the remaining laps.

In this experiment, the subjects included 7931 well-trained and healthy runners who completed one of the New York, London, or Dublin marathons. For each competitor, important features are required to be recorded in order to predict a marathon finish time, which shows following:

- Pacing: Measured in minutes per kilometer
- Heart rate: Measured in beats per minute
- Cadence: Represent the number of steps taken per minute

Note that in the prediction process, In order to build the predictive model, each race needs to be sampled while the average value of each (pace, heart rate, and cadence) within the interval should be obtained. Specifically, multiple window sizes, containing 1km, 5km, and the full race to that point, are adopted in this process which corresponding to the short, medium, and long-term efforts. This detailed analysis is worthwhile because there are pacing changes, physical effort, and running form in different running stages. This monitor can track these changes as well as classify and evaluate their performance.

Finally, evaluate this model's performance compared to the baseline model, which is used frequently in the marathon race, that is, to record the finish time. One of the baseline model problems is that it is based on the assumption that marathon runners maintain uniform speed during the whole process, which is impossible in the real race. The baseline prediction method is calculated as

$$Predicted\ Finish\ Time(s) = MRP * 42.2 * 60$$

where MRP is the mean pace of the runner to that point.

Comparing it to this baseline is worthy to see the scale of errors in the baseline model caused by the even-running strategy. It turns out the improved model can not only improve the precision of finish time prediction by over 4 minutes between 7.5 and 23.5 km but also can reflect the slower pace of participants in the second half of the race.

2.2.3 Cycling:

This report reviews the paper by [13], which has used the Fuzzy Inference System (FIS) and Adaptive Neural Network-Fuzzy Inference System (ANFIS) system to classify the cyclist current performance status and predict the cycling athletes' performance.

The experimental testing samples are 12 cyclists from 24 to 29 years old without any health condition. The body features are measured in real-time using sensors attached to the body. The test is made all indoors with a trainer bike, the speed of which is measured with sensors. The body features of the cyclists required for this experiment are:

- Heart pulse is measured to be used to calculate the heart rate variability (HRV)
- Maximum heart rate (mHR): Normally denote as 220-age.
- Resting heart rate (rHR): rHr of each participant should be record before each training session

• Reserved heart rate (HRr):

$$mHR - rHR$$

• Target heart rate (HRt): HRr x intensity + rHR

• Body Temperature: Body temperature of the participant should maintain around 37°C

• Core temperature

• Skin temperature

• Cycling speed

lifestyle is one of the key factors that can affect cyclists' performance; cyclist should follow some regulatory constraints to perform the accuracy of the experiment such as the participant should have ample sleep, the participant should not take any drug medicine, the participant should maintain a constant cadence at all time, etc. Overall diagram for FIS- and ANFIS-based system as shown in the figure below

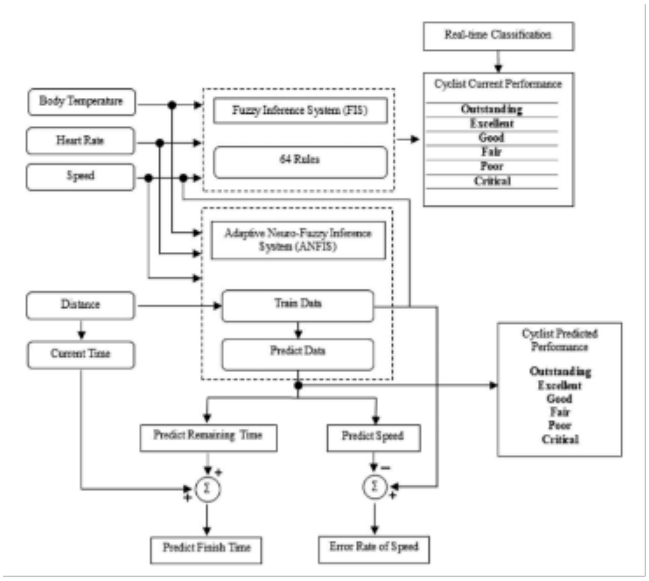


Figure 1: Overall diagram for FIS- and ANFIS-based system

The inputs are body temperatures, heart rate, and speed, as shown above. The FIS-based predictor first receives real-time data from the sensors attached to each participant and then predicts the current state of participant performance in which classified into six different groups:

• Outstanding

• Excellent

• Good

• Fair

- Poor
- Critical

While the ANFIS system uses real-time data to train the ANN algorithm and predict the cycling speed produced by the participant, it can also estimate the remaining cycling loop's time taken. The predicted time taken to finish the whole lap is using the following equation

$$Tf = T + (l * \frac{n}{S})$$

Where, T represents the time used for the previous course, l is the remaining distance to finish to the whole lap. and S(m/s) is the predicted speed produced by ANFIS for the remaining laps, n is the remaining laps.

The prediction precision could be improved by increasing the sampling numbers since we have limited test samples, leading to errors for prediction.

3 Methodology

3.1 Performance Management Chart

PMC, which is short for Performance Management Chart, is one of the effective tools to assess if an athlete is training as suggested[14]. Both over-training and under-training might lead to injuries. What's more, even when an athlete is not over or under training, he could also not take the optimal training load. It is always the goal to let the athletes have training loads as appropriate as possible. PMC is able to identify whether an athlete is in the optimal training zone or not [15].

Therefore, PMC is taken to be the key to achieving the project main goal. PMC contains four major elements, which are Training Stress Scores (TSS), Acute Training Load (ATL), Chronic Training Load (CTL), and Training Stress Balance (TSB) [14]. All of these elements are calculated from TSS. By taking the average of the workouts that have done from the past 7 days, we can estimate ATL, which is also known as Fatigue. An exponentially weighted average of the last 42 days of training is used to estimate CTL, which is also called Fitness. Fitness reflects the training that an athlete has done over the last 3 months. By subtracting Fatigue from Fitness, we obtain TSB or Form [16].

With Fitness alone, we can't tell if an athlete is ready to race at his/her best. There are zones in the PMC indicating stages in a training process, which are divided based on TSB. A large magnitude negative TSB indicates that the athlete is carrying too much fatigue and not on form, or in other words, the athlete is over-training and he/she is in "High Risk Zone". A large positive Form, however, indicates the athlete is not training enough [22]. Keeping the athletes in the optimal training zone is the way to let them keep in good shape while avoiding injuries. The figure below gives an idea of how zones are actually distributed in a PMC, and we will illustrate the plot and the what the making process is in the later parts of this report. There are also special cases, athletes may have lower TSB than the floor of the optimal zone, but they shed fatigue at a higher rate than losing fitness by tapering and come into form on the race day [14]. This can also be monitored by PMC.

3.2 Estimating TSS and Generating PMC

The most important element for making PMC is Training Stress Scores (TSS), but the variable values and also the related variable values are widely missing in the datasets. The team implemented a system written in Python for processing the data, estimating missing TSS values, and generating the Performance Management Chart (PMC).

The estimations of TSS are different for different activities, which is mainly reflected in the difference of features involved. For example, when estimating TSS for running, we consider features like 'Elevation Gain' and 'Avg Ground Contact Time', but for swimming, we consider 'Avg Stroke Rate' and 'Avg SWOLF'. The main idea of estimating TSS is we use the records where TSS values are not missing as training

sets, extract features from both spreadsheet data and additional data, build activity-wise models, and then fill in the missing values with predictions. The details of how we process the data, what features we use, and what models we have will be shown in the later sections.

Once we have all the TSS values ready, we can take the rolling averages to calculate the Fatigue and Fitness, and further calculate the Form. We can monitor the athletes' body conditions by identifying which zones the athletes' Forms are in.

3.3 Our System

We have developed a non-hardcoded and reusable data modeling system. It is both user and developer friendly.

For users, all the things that need to be prepared are getting data ready and having a computer with Python 3.6 installed. Then a user can just follow the instructions in README.md of the project, and the rest is automated.

The system control is based on argument parsers, and the argument options are open to the developers, so they can easily test each module of the system when necessary. It is a strong foundation for future students who continue to work on athlete performance predictions.

The system contains several modules, and the modules are in two types: major modules that participate directly in data processing and analyzing, and supportive modules that could help users and future developers understand the data. Major modules include

- ***Main Module:*** Controls the execution of the program and I/O of the system.
- ***System Initialize Module:*** Initializes necessary configurations of the system.
- ***Fit Files Conversion Module:*** Converts the .fit files to .csv files.
- ***Data Cleaner Module:*** Cleans the data. Contains the spreadsheet data cleaner class and the additional data cleaner class.
- ***Feature Engineering Module:*** Extracts features from the raw data. Contains the spreadsheet feature engineering class and the additional data feature engineering class.
- ***Data Merge Module:*** Merge the additional data and spreadsheet data for each athlete.
- ***Modeling Module:*** Builds and save models for TSS and select the best model to record in JSON.
- ***PMC Generation Module:*** Generate Performance Management Chart.

Refer to the diagram below for a more visualized understanding.

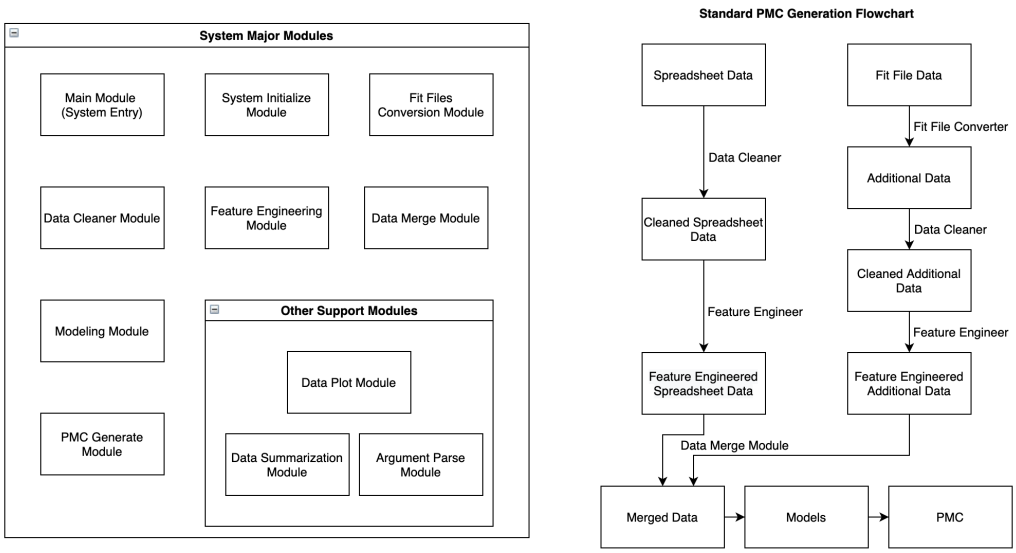


Figure 2: System Structure and How the System Works

We have two different types of datasets, and we call them spreadsheet data and additional data. The one spreadsheet dataset contains activity records within a period of time, however one additional dataset contains only one activity on a certain day. The details of the datasets will be explained in the 'Data' section. The features in the two types of datasets are quite different, while they both have their characteristics, so we process the datasets separately and merge them before modeling.

After the system has been initialized, the additional data will be first converted to .csv files from .fit files, and then both types of datasets will be clean throughout Data Cleaner Module. Features will be extracted from both types of datasets and after that, the datasets will be merged for modeling.

```
monitoring-athletes-performance
   ├── main
   |   ├── main.py
   |   ├── system_initialize.py
   |   ├── data_loader.py
   |   ├── data_cleaning.py
   |   ├── data_feature_engineering.py
   |   ├── data_merge.py
   |   ├── data_modeling.py
   |   ├── requirements.txt
   |   |── config
   |   |   ├── data_file_names.cfg
   |   |   ├── activity_types.cfg
   |   |   ├── column_data_types.cfg
   |   |   └── athlete_personal_info.json
   |   └── fit_file_convert
   |       ├── convert_fit_to_csv.py
   |       └── process_all.py
   ├── data
   |   ├── Eduardo Oliveira.csv
   |   ├── Xu Chen.csv
   |   ├── Carly Hart.csv
   |   ├── fit_eduardo_oliveira
   |   ├── fit_xu_chen
   |   └── fit_carly_hart
   |
   ├── log
   ├── models
   ├── plots
   |
   ├── LICENSE
   └── README.md
```

Figure 3: System Directory Layout

How to Run Our System

User's Option

Open the system built-in command-line tools or Terminal in a Python IDE, you can go to the main folder of the project, and run the following command.

```
python main.py --athletes-names xu_chen eduardo_oliveira --initialize-system=False --generate-pmc=Tru
```

The example above means you don't want to initialize the system (say it's your second time running this system), but you do want Performance Management Charts for Xu Chen and Eduardo Oliveira (say you have updated the fit files). Please refer to Additional Help for more options.

Developer's Option

For the developers who work further based on this system, we recommend changing options in `main.py` for tests, and here's an overview.

```
if __name__ == '__main__':
    athletes_names = ['eduardo_oliveira', 'xu_chen', 'carly_hart']
    internal_args = ['--athletes-names={}'.format(' '.join(athletes_names)),
                     '--initialize-system=False', '--clean-data=False', '--process-feature-engineeri
                     '--build-model=False', '--generate-pmc=True']
    main(internal_args)
```

Figure 4: How to Run the System

4 Data

The main goal of this project is to assess if the athletes are training as they should, to avoid injuries while keeping fit. This is done by analysing the Performance management Charts (PMC) for over-training, under-training and accurate training for each individual athlete.In order to do so we use data from the Garmin watches.Since Heart rate is an important attribute for estimating TSS values and many data points had this feature missing, additional data was requested.

4.1 Data Description

4.1.1 Spreadsheet Datasets

The data-set comprises of athlete information from 12 participants, 6 males and 6 females. Each data-set contains mainly two types of features where one is the activity-related, and the other is performance-related. For instance, the type and time duration of activity are associated with the former whereas the athlete's average speed and heart rate are categorized as the latter.

These athlete activities have been recorded for ∼4 years except for one athlete for whom we have ∼10 years of data. The age-bracket of these athletes is quite large, ranging between 27-50 years. Most of them have had a history of sports injury except for two athletes who have no known history of injuries. The number of data entries varies for each athlete. For instance, some of them have ∼1000 entries in the data set but some have fewer entries, as low as 100.

Activity for each athlete can be broadly classified into four major types: Running, Swimming, Cycling and Strength Training. We only focus on the first three for our analysis. To predict for TSS scores, we need the information of each athlete and their tendency for each activity type. Figure 5 provides an overview of this.

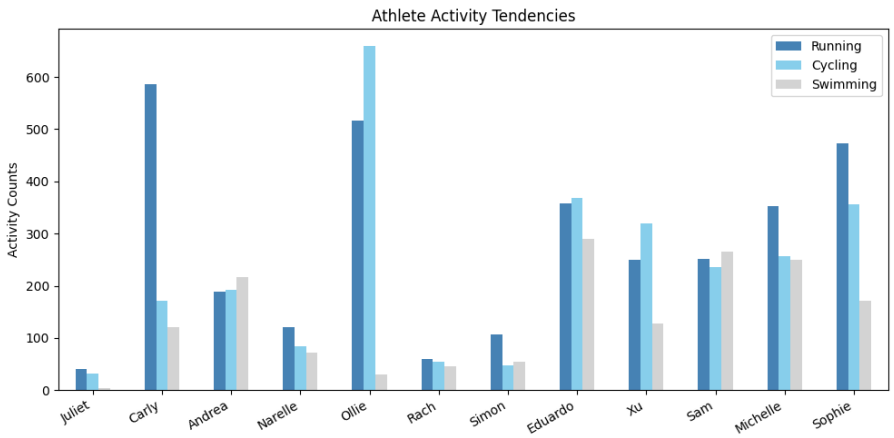


Figure 5: Activity Tendencies

Each of these athletes can be broadly classified into Novice, Intermediate, and Advanced type based

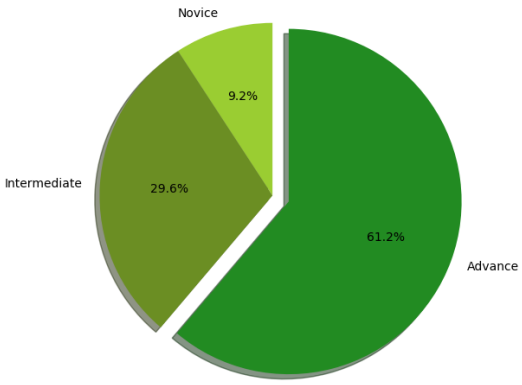


Figure 6: Athletes Levels - Data Contributions

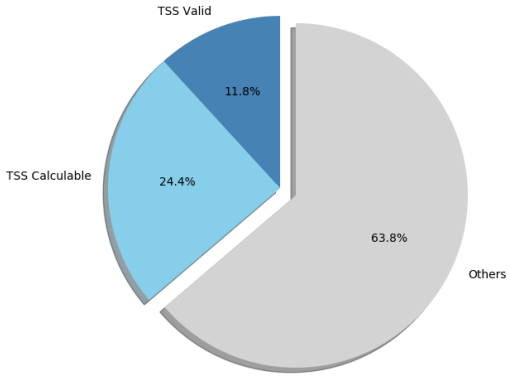


Figure 7: TSS Overviews

on their level of activities. This information adds immense value in predicting their performance and fine-tuning the model for each activity type. The distribution in terms of these can be observed in Figure 6.

Each athlete dataset has ∼44 feature columns. EDA was done to identify important columns and to gain an understanding of how the variables are distributed. More on this has been discussed in section 4.2. Using the information gathered during EDA, data sets were cleaned and prepared for analysis. One key takeaway from EDA was the column Training Stress Score. This column proved to be significantly important to achieve the goal of this project. However, it was observed that only ∼11.8% of the TSS values were valid. This can be observed in figure 7. The section of ∼24.4% in the distribution of TSS highlights the values that can be calculated or estimated using other columns in the data set. A request was made to the client for additional data to reduce the unknown TSS ratio of ∼63.8%. The additional data zooms in on the athlete activity for each day and captures the data at progressive time intervals in terms of seconds to capture Heart rate and other metrics.

4.1.2 Additional Datasets

Besides the spreadsheet datasets, three athletes provided us some additional data. The additional data is exported from athletes' sports watches, and in the format '.fit'. A '.fit' dataset contains location and time information, environment information (like temperature), and athletic information (like heart rate, running cadence) during an activity.

Each additional dataset is the log for one activity during a certain period of time, each athlete, therefore, could have multiple additional datasets. For example, one athlete swims in the morning and runs in the afternoon someday, he will have two additional datasets for that day.

As the '.fit' format cannot be analysed directly, a number of problems had to be fixed. More on this has been discussed in section 4.2.3.

Additional data include four activity types (running, swimming, cycling, and training), one can tell an athlete's activity tendency based on the number of data files for each activity. Some athletes have a balanced number of data files for all activities but some athletes tend to work more on one or two activities.

Additional datasets have 14 features, including features for the time, location, environment, and athletic status, as mentioned before. The length of a dataset depends on the duration of the activity, ideally, there's a 1 to 3 seconds difference between records in a dataset.

4.2 Data Exploration

This section briefly describes the approaches used by the team to get familiar with the data set by exploring the obvious and hidden patterns. Doing this not only revealed the various characteristics of the athletes but also provided a sense of familiarity with their behaviors. A combination of manual processing and automated checks on the data set was done for this purpose using a series of univariate and bivariate analysis. This type of descriptive statistics not only helped in gaining a better picture of all the columns but also eased the process of identifying outliers.

4.2.1 Univariate Analysis

The main purpose of this analysis is to summarize a single data column using central tendencies such as mean, mode, median, min, and max for continuous variables and frequencies for categorical values. In addition to line charts such as the one below, interquartile ranges were also used for certain columns to check for the spread of the variable. An example has been illustrated in Figure 8. for an athlete whose 'Date' column alone revealed a plethora of information about their routine.

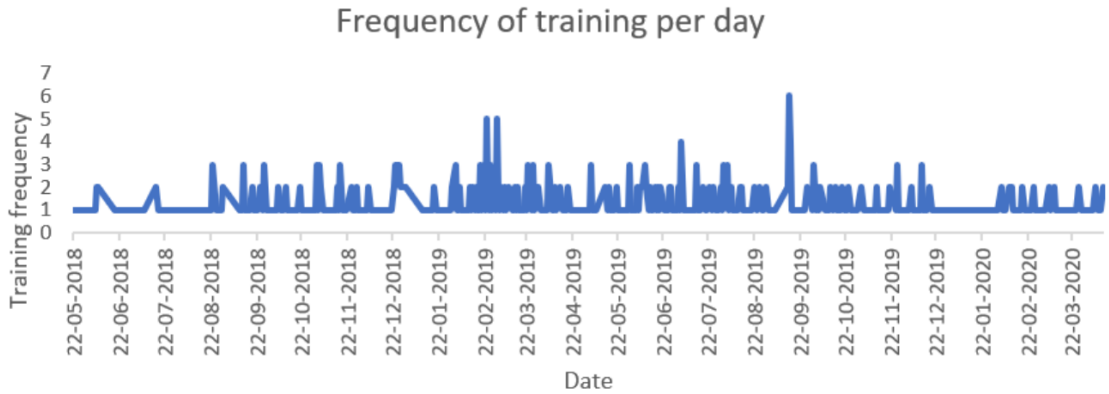


Figure 8: Training Frequency Distribution

It can be observed from the graph that in a span of two years the athlete on average, trains for a maximum of two-three times a day. However, a peak can be observed on '14-09-2019'. A basic outlier analysis on this column alone flags this scenario but does not reveal too much information. This

generates opportunities for further exploration using Bivariate analysis.

4.2.2 Bivariate Analysis

This type of analysis is typically used to identify the relationship between two variables. Correlations between variables were checked to gather a fair idea around how the values of one change with the increase or decrease in the other. For instance, Distance and Training Stress Score are highly correlated with a coefficient value of 0.8. This gives an idea about which variables could introduce multicollinearity in the model and can thus be taken care of. Another interesting aspect of this type of analysis is that it could add more value to the univariate analysis by enabling a better understanding of the existing feature sets.

This can be extended to the example discussed in subsection 4.2.1. Combining the training frequencies with the competition column in Figure 9 reveals that compared to all others, the athlete has trained extensively for the Queenscliff cycling and running competition. This confirms that the outlier is an important data point and not to be neglected as further analysis could be done to check how the sudden peak in training frequency has affected the athlete in terms of fatigue and fitness.

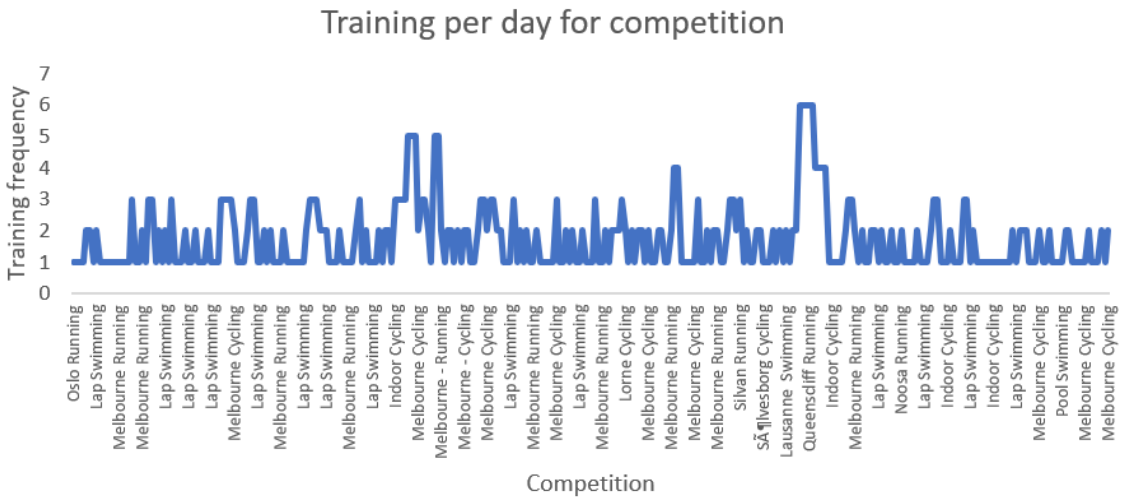


Figure 9: Training Frequency w.r.t Competition per day

A bivariate analysis on the additional data helped us understand how the heart rate for an activity varies with respect to time. The example Figure 10 reflects the data capture for an athlete for Cycling on '2018-08-05'. It can be observed how the heart rate increases as the activity progresses. Sudden troughs can be observed which can be attributed to the problems with syncing of the Garmin watch.

Exploratory Data Analysis thus not only exposes the hidden patterns but also aids in understanding which data point is a true anomaly and which is not. In addition to this, it also supports better feature engineering as a lot of columns can be eliminated based on Univariate analysis. The distribution of the variable also gives an idea about the kind of imputations that can be done in case of missing values.

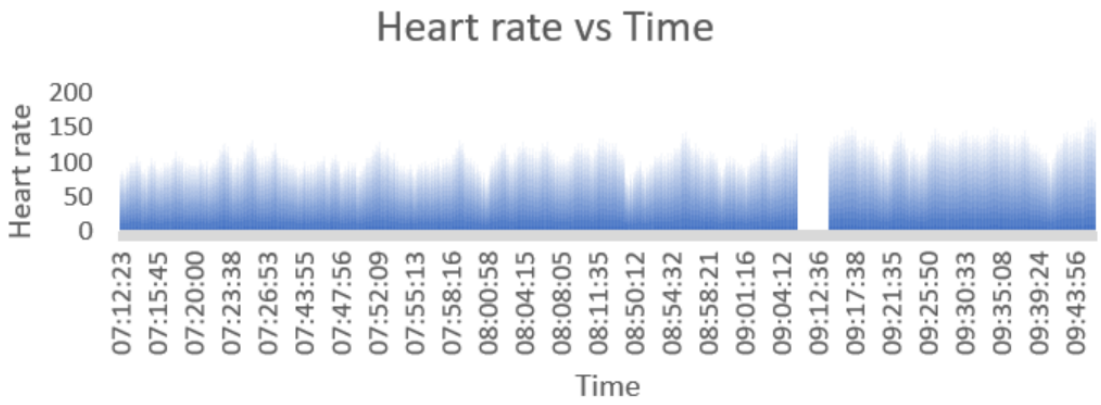


Figure 10: Heart Rate Distribution

Performing this type of analysis on different data sets helps in comprehending the data and fine-tuning the model better.

4.2.3 Problems in Data sets:

Since each data set is unique to a specific athlete, there arises an issue of conflict when trying to merge different data sets while generating summaries. Although, not all columns differ, in many scenarios the Garmin watch captures additional data. 'Max. Resp' for instance is one such feature captured as part of one athlete owing to their configuration setting in the watch. This column, however, has been absent in all other data sets. Such issues have been handled carefully by either filtering them out as unnecessary or by collecting additional data wherever necessary.

The distribution of data plays a major role in data prediction. Often, with a better representative sample, the model performance improves drastically.The athlete data analyzed in this report faces challenges due to many sparse columns, including missing values in the most important column Training Stress Score (TSS) which is of utmost importance to achieve our main goal. Apart from the individual athlete data being sporadic, the data points captured for some of the novice athletes are as few as 200. This difference in data size results in some of the athletes being underrepresented. This poses more difficulty in building a generalized model involving all athletes.

While comparing different athlete types, in addition to the issue of having uncommon columns, the data format varies substantially. For instances where a pattern could be observed in the format difference, a quick fix was made by correcting the pattern type. However, in many columns, data format changes abruptly and no pattern can be observed. These types of problems have been carefully handled to ensure that a combined normalized data set for all athletes can be created. More on the techniques used to handle these cases have been discussed in the Data cleaning section.

The additional data received as fit files to cater for the missing values have a plethora of problems. Many

files are empty as the Garmin device could not sync properly when the athlete was training.

As mentioned before, we cannot handle '.fit' files directly, so we use python scripts given by the clients to convert these '.fit' files to '.csv' files. Due to some reasons in scripts and the data, the scripts could convert about only 40% of the files, and after making some changes to the scripts, the team ended up converting 82% - 90% of the files.

There are some other problems with the additional data that are caused by the actual training habits. There are very few swimming datasets since athletes usually don't wear watches during swimming. The data is not always accurate, because some of the records are manually added by the athletes after their training. Some athletes don't like wearing watches, or sometimes they forget to wear during training.

Some problems are from the data itself or the database implementation. Sometimes, the activities or the dates don't match with those in spreadsheets for a certain athlete.

Even after successful conversion, we observed that important columns such as Heart rate is missing completely for numerous files. Thus, only those activities were considered for analysis wherein the heart rate could be mapped from additional data.

4.3 Data Cleaning

Data Cleaning played a significant role in the project. Identifying the missing values and outliers with visual and quantitative methods was necessary to understand underlying patterns and relationships between features. The detailed discussion about the steps involved in data cleaning for building the predictive models are as follows.:

Data Sourcing

To start off, we read in our data set and split it based on the activity types to achieve our goal.

Preliminary Data Processing:

We do this to generate a simple preview and statistics for our dataset. To account for inconsistency in the data types we treat the data as following:

- For categorical columns, i.e., columns that take on a limited, usually fixed, many possible values, we set their type as "category." E.g., Activity Type, Title are all categorical data.
- For numeric columns, we set their type as "float" (floating-point), E.g., Distance, Calories, Avg HR, Max HR, Avg Speed, Max Speed, Elev Gain, Elev Loss, Avg Bike Cadence, Max Bike, Cadence,

Normalized Power, Training Stress Score, Max Avg Power, Avg Power, Max Power, Total Strokes, Avg. Swolf, Avg Stroke Rate, Number of Laps, Time_sec are all numeric data.

- Date is set as "datetime64" data type.

Selecting relevant columns

In this step, we choose the informative features. If the feature is useful and still had a high missing value percentage, we selected the feature and used more sophisticated methods such as missing data modeling. After this step, we zeroed in on 23 features.

Inconsistent data

It is imperative to have the data set follow specific standards to fit a model. We have to make sure the values in the same column are on the same scale. Inconsistent data type was handled as below:

- Capitalization: To avoid inconsistent capitalization, we transformed all the categorical values to lower cases, removed white spaces and commas.
- Formats: We observed the varying formats for Date-Time, so we changed the time to %H: %M: %S format, converted the feature from string to Date Time format. We replaced the column values having "...","–" with NaN to handle the missing values.

4.3.1 Missing values

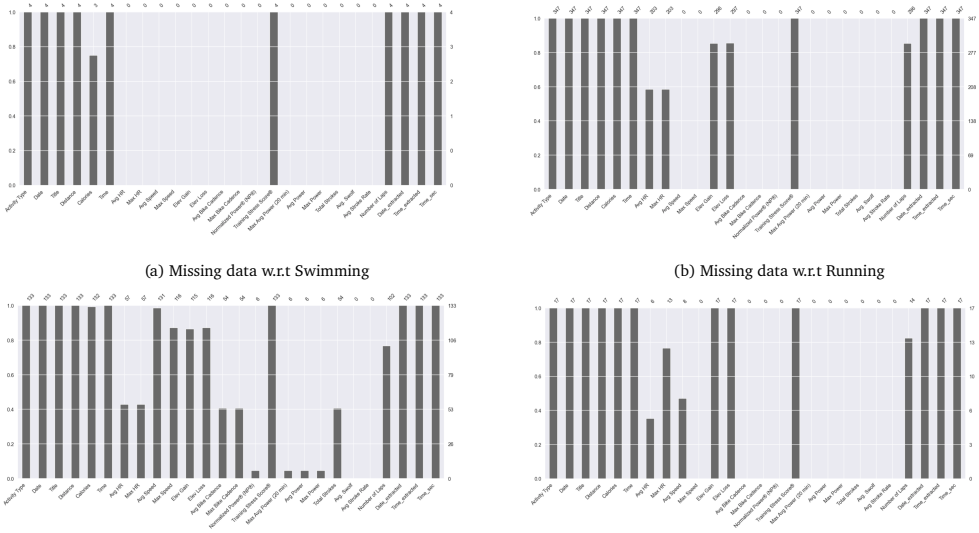
The next step after standardization is handling missing data. From the results, we observed that the data set had 18 columns with missing values. We used different techniques to know more about missing data in our dataset.

Detecting missing values numerically

To get the idea about the distribution of missing values numerically, we generated the percentage of missing data in each feature of the data set and dropped the columns containing more than 60% of data missing.

Detecting missing data visually using the Missingno library

To graphically analyze the missingness of the data, we used the Missingno library. Missingno library generates a bar chart that provides an overview of the data set's completeness.



(a) Missing data w.r.t Swimming

(b) Missing data w.r.t Running

(c) Missing data w.r.t Cycling

(d) Missing data w.r.t Others

Figure 11: Missing data w.r.t Activity-Type

Visualizing the locations of the missing data

To visualize the missing data locations, we used msno.matrix. The example in Fig. 12 is a dataset is distributed having 1140 data points. The plot generated by this matrix appears white wherever there are missing values. The Avg power, Max power, and Max Avg power columns seem to have the highest number of missing values at an all-activity level. When the data set was further split based on activity types, we observed that the columns Avg power, Max power, and Max Avg power were missing for activity types swimming and running but were completely filled for cycling. This makes sense, considering the fact that power is recorded only during cycling. To get a better idea about how features are correlated with one another, we used heatmaps.

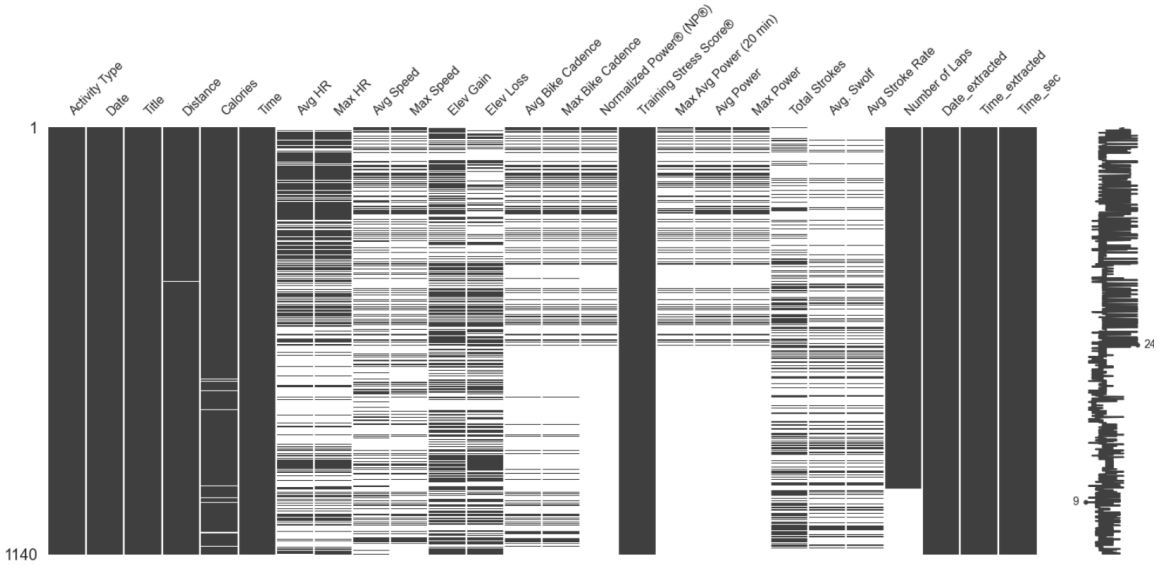


Figure 12: Locations of Missing Data

Reasons for missingness in the data

There are three possible reasons for missing data, Missing Completely at Random (MCAR), Missing at Random (MAR), Missing Not at Random (MNAR). To find the correlations between missing values of different features, we used a heat map and dendogram.

Heatmap

It was used to find the correlations among features, as shown in Figure 13. A positive correlation is in proportion to the level of darkness in blue, as indicated by the bar on the right side. There are positive correlations at different levels between" Avg HR", Avg Speed", "Max Speed"," Elev Gain", "Elev Loss"," Avg Bike Cadence", "Max Bike Cadence", "Normalized Power"," Training Stress Score"," Max Avg Power"," Avg Power"," Max Power". The highest correlation is between "Max Bike Cadence" and "Avg Bike Cadence," which is 1. The collinearity among the predictor variables was a factor considered into account while modeling.

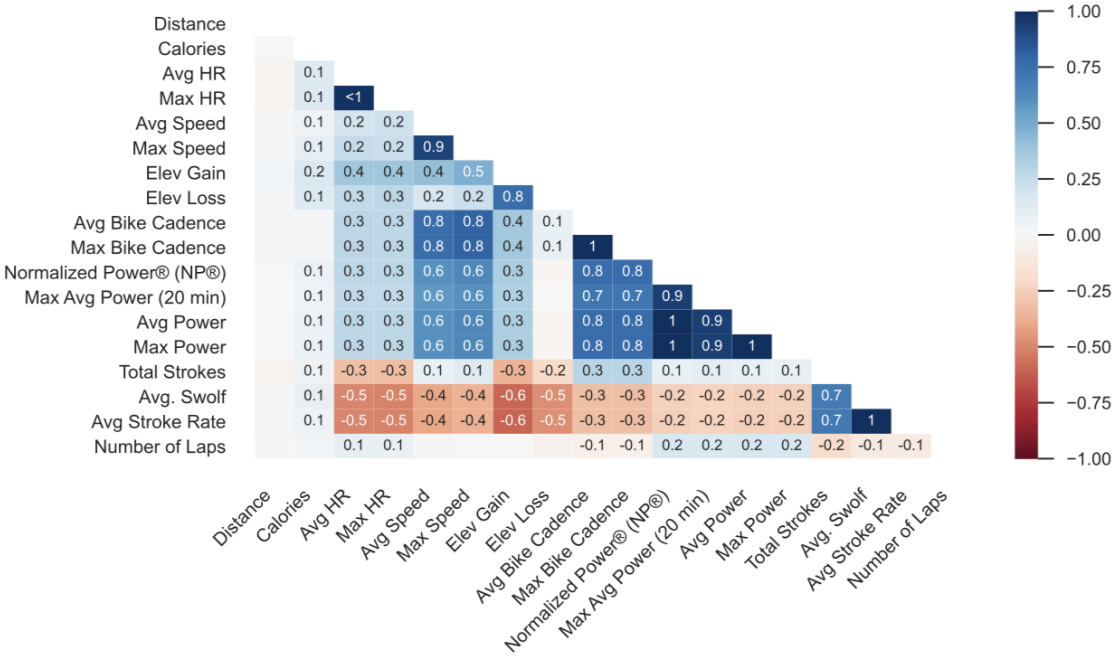


Figure 13: Heatmap

Dendogram

The dendogram generates a tree diagram of missingness. It groups the positively correlated variables. Cluster leaves which linked together at a distance of zero fully predict one another's presence. One variable might always be empty when another is filled, or they might always both be filled or both empty. The missingness of Avg Swolf tends to be more similar to Avg Stroke Rate than to Total Strokes and so on. However, in this particular case, the correlation is high since Avg Swolf has very few missing

values.

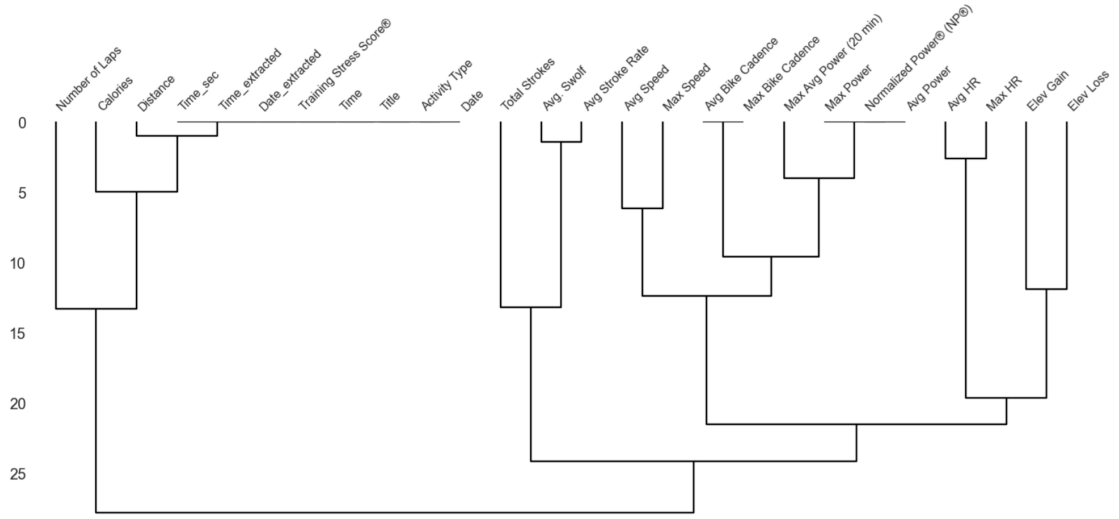


Figure 14: Dendogram

Imputing the missing values

After identifying the patterns, we imputed the missing values depending on the nature of the problem and data. Imputation techniques were classified based on the type of features and the reason for missingness. If the feature is a numerical variable, we imputed the missing values with the mean, mice, and knn imputations. If the feature is categorical, we imputed the missing values with mode imputations. Luckily, our dataset has no missing value for categorical features.

4.3.2 Outlier detection

The final step in our EDA is outlier detection. Whether the feature is numeric or categorical, we used different techniques to study its distribution to detect outliers, such as univariate or multivariate. We used different outlier Methods like Interquartile Range, Standard deviation, Z score(Since the given features in the additional data set are numerical, we apply a z score test to handle the outlier), and local outlier factor to identify univariate and multivariate outliers. The main idea behind IQR is to build boxplots. We generated boxplots to measure the statistical dispersion and data variability by dividing the dataset into quartiles. Standard deviation techniques understand the variance or how much data points are spread out from the mean. Finally, we used the local outlier factor method for the detection of multivariate outliers. In the technique, a point is labeled as an outlier if the density around that point is significantly different from its neighbour.

4.3.3 Missing values in Additional Data

There are both advantages and disadvantages when handling missing values in additional datasets. The good thing is that the data points within a dataset are from one continuous time series so interpolation makes a lot of sense on some of the columns, and the bad side is there are usually hundreds and thousands of files for one athlete so complicated regressions is not a good choice considering the overall performance. Here we use several types of imputations.

We first format all types of missing values as 'np.nan', and then check the missing value percentage for each column. We will not process imputation if the missing percentage is high to avoid bringing in too much noise.

Then we decide what kind of imputation techniques should be applied to each column. For example, Interpolation can be applied on latitude, longitude, and altitude. Some of the columns can be imputed together, so we group those columns, like (speed, heart rate, cadence) and (timestamp, distance). While some columns can only apply univariate imputation, like 'temperature' and 'timezone'. We save the imputation technique information in a configuration file and load the configuration file as a dictionary when the module is used and the class is instantiated.

We find out the columns that need imputations and find the relevant techniques in the dictionary. Then we apply imputations and check the missing value percentages again after imputation. Finally, a logger will be saved which contains the missing value percentage before and after.

5 Feature Engineering

To better represent our problem of estimating over-training, under-training, and accurate training of athletes, it was necessary to accurately approximate the Training Stress Score (TSS) values. As discussed earlier, 63.8 percent of these values are missing. A lot of research was done to reduce this percentage of unknown values. As the data we have does not favour a uniformed TSS value across all activity type using the formula below, activity based TSS were computed.

$$\frac{sec * NP * IF}{FTP * 3600} * 100$$

where, $sec =$ Duration of workout in seconds, $NP =$ Normalized Pace, $IF =$ Intensity Factor and $FTP =$ Functional Threshold Pace.

We have rTSS for running, sTSS for swimming and cyclingTSS for cycling. The formulas we use to compute them are as follows:

5.1 rTSS

$$\frac{sec * NGP * IF}{FTP * 3600} * 100$$

is generally used to estimate rTSS. where, $sec =$ Duration of workout in seconds, $NGP =$ Normalized graded Pace, $IF =$ Intensity Factor and $FTP =$ Functional Threshold Pace.

Owing to data constraints as discussed earlier, alternate techniques were explored. Various researchers such as Joe Friel and Andy Coogan have introduced formulas for running which have long proven to be accurate. We thus use Joe Friel's running formula to estimate rTSS.

Various Heart rate Zones [25] are estimated using the distribution of lactate threshold. These zones can be thought of as a means to specify how hard the athlete trains and is unique to an athlete. Lower the zone, easier is the workout. Lactate threshold is basically the average heart rate of the last 20 minutes of a 30-minute trial, usually estimated at the beginning of a training period. An approximation of the lactate threshold was estimated by the team using the continuous heart rate captured by the additional data. Once the training zones have been established for an athlete, Each activity is mapped based on average heart rate and duration of the activity. TSS is obtained by multiplying TSS per hour for that zone with the duration of the workout. The mapping of heart rate zones with respect to TSS per hour is as follows in Table 1.

Individual heart rate zones are set by calculating $Lactate\,Threshold * percent\,in\,decimal$ as seen in Table 2.

Avg HR Zone	TSS per Hour
1(low)	20
1	30
1(high)	40
2(low)	50
2(high)	60
3	70
4	80
5a	100
5b	120
5c	140

Table 1: HR Zone to TSS mapping (Running)

Zone	Percentage
1	65%-81%
2	82%-88%
3	89%-93%
4	94%-100%
5A	101%-102%
5B	103%-105%
5C	106%+

Table 2: Setting individual HR Zone (Running)

5.2 sTSS

The section below will introduce two methods to estimate the sTss score. Several definitions associated with sTSS should be elaborated before further calculation

5.2.1 Related Conceptions:

Critical Swim Speed (CSS): CSS approximates lactate threshold speed. This indicator is significant in the improvement of sporters because it determines a sporter's ability to sprint and whether he/she can work anaerobically. So, in estimating sTSS, CSS needs to be estimated first, which serves as a threshold in estimating sTSS. CSS can be calculated easily, and it is recommended to repeat the CSS test every six weeks or every four weeks if athletes feel the fitness is developing fast. However, if athletes' training is not working well, then take the CSS test every 6 weeks. The calculation procedure is explained in detail in the following:

1. Athletes need to do a thorough warm up and some built-up exercises to prepare for the CSS test in order to get as precise as a possible estimation for CSS.

2. Finish a 400 m time-trial swim. The standard test should be started with a wall push rather than diving into the water while try to be as even as possible during the whole process and do not change the pace drastically. Write down the time denoted as T2.

3. Athletes need to get complete recovery after the 400 m time-trial swim and come back to the best state.

4. Finish another 200 m time-trial swim. The specific requirements are the same as the 400 meters.

Write down the time denoted as T1

5. Calculate CSS with the formula:

$$\frac{D2 - D1}{T2 - T1}$$

where, D1 represents 200 meters, D2 represents 400 meters, T1 represents the time required for the athlete to swim 200 meters in seconds, and T2 represents the time required for the athlete to swim 400 meters in seconds.

6. CSS needs to be converted from m/sec into time per 100m for further calculations.

Duration: Record total time for swimming in second not including rest.

Distance: Record total distance in meters for swimming covered by duration.

Pace:

$$Duration * \frac{100}{Distance}$$

where, pace is in-unit s/100 meters.

Estimated Duration:

$$Pace * \frac{Distance}{100}$$

where, estimated duration is in-unit seconds.

5.2.2 Calculation Methods:

There are mainly two ways to estimate sTSS: using pace and distance or using swim time and distance. And there is one critical point in the process of calculation that rest time need to be excluded for the precision requirement of estimation.

In the first estimation method, data of swim time and distance are necessary. The definitions of swim time and distance are simple, referring to the estimated duration (rest time excluded) and the distance covered. The sTSS can be derived as:

$$\frac{CSS^3}{Pace} * \frac{Estimated\ Duration}{3600} * 100$$

In the second estimation method, records of pace and distance are required. The definitions of swim pace and distance, referring to the velocity of swimming and the distance covered. Then sTSS can be derived directly from formula:

$$\frac{CSS^3}{Pace} * \frac{Duration}{3600} * 100$$

Avg HR Zone	Percentage of FTP
1	<55%
2	56%-75%
3	76%-90%
4	91%-105%
5	106%-120%
6	121%-150%

Table 3: FTP to HR Zone mapping (Cycling)

5.3 Cycling TSS

The average power produced by a rider over one hour of training is FTP. It is expressed in terms of watt per kilo. It is basically a fraction of the power by the athletes' weight. The fitness of an athlete is generally measured in terms of whether his/her FTP value has increased over a four-week duration, provided the weight has not changed. Similar to the training zones in running, based on the FTP values, zones are established. Following is mapping of FTP to respective zones defined by Allen and Coggan. The FTP to Avg. Heart Zone mapping can be observed in Table 3 [24].

These zones are then used to train the athletes based on their requirements. For instance, zone one and two are used for warmups. Road racers target zone three, whereas zone four is generally useful for those athletes aiming to increase their FTP. The latter zones are suited for hard sprint training. As the TSS mapping for these zones were not explicitly available and the data already catered to TSS values for cycling, this area was not explored further.

6 Data Merging

There are two types of datasets for this project, spreadsheet data, and additional data. A spreadsheet dataset contains all the activity summaries for one athlete with a certain time range, so each athlete has one spreadsheet dataset. Additional datasets are transformed from fit files, which are the activity records from athletes' sports watches. Each athlete could have multiple additional datasets, and each dataset is for one activity at a certain time only. For example, one athlete swims in the morning and runs in the afternoon someday, he will have two additional datasets for that day.

Given the fact that the Training Stress Score values are widely missing in the spreadsheet datasets, which will have a huge impact on future modeling steps, we decided to try to as many TSS from additional datasets as possible and merge the two types of datasets.

Ideally, each row in a spreadsheet dataset should match one additional dataset, and all the things need to be done are calculate TSS values using the TSS definition formula based on additional datasets and fill the values into the spreadsheet dataset, and then we can build models based on existing TSS values and predict the missing TSS values.

However, there are many problems merging the two types of datasets. For example, for one activity on someday, the time ranges in spreadsheet data and additional data don't match each other. Another problem is activity types don't match each other or it is supposed to have several activities on someday but the spreadsheet data has only one record. We assume those problems are caused by the process of generating spreadsheet datasets.

The plot below shows the merging Logic.

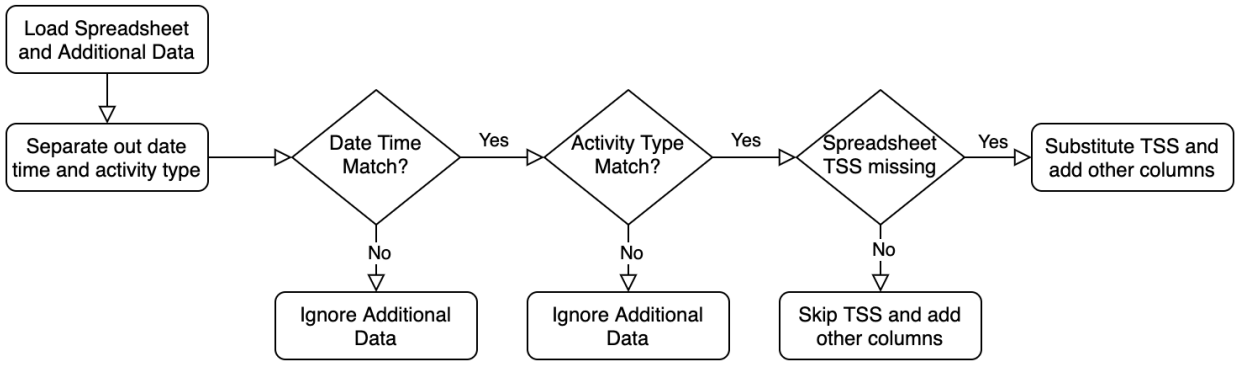


Figure 15: Merge Logic

7 Data Modelling

As the data spans across three activity types, and the PMC chart is at an all-activity level, two approaches were considered for modelling.

1) **Classification**: Classifiers were built for each athlete at an all-activity level to predict for over-training, under-training, and accurate training. The results were not as good since the features vary across different activities and only a handful of them are suitable to model at an overall level. For instance, Max. Power is an important attribute for cycling but the data is limited to only this activity type. To overcome problems such as this regression was preferred.

2) **Regression**: The models are built at activity level to predict for TSS Scores. These TSS values are later used to calculate Fatigue, Fitness and Form to generate PMC charts used to analyse the trends to check for the athletes' over-training and under-training.

7.1 Neural Network

We used two types of neural networks, regular deeply connected neural network, and Long Short-Term Memory (LSTM).

A deeply connected neural network is also known as a deep feedforward neural network, and the difference from the second type is its node connections do not form cycles or loops. In this type of neural network, the data or information goes only one direction, that is forward towards the output. This network contains three types of layers: one input layer, zero to many hidden layers, and one output layer. The main principle behind this network is that each node in the hidden layers takes a linear combination of inputs from previous layer nodes and uses an activation function to bring non-linearity in, and outputs some values to the next layer.

LSTM is one type of recurrent neural network, and unlike the former one, LSTM has feedback connections. The reason why we tried LSTM is that some of the features of our data are sequences or time series, for example, the Fitness, Fatigue, and Performance of an athlete are accumulations, and LSTM can process sequences of data too besides single data points. We chose LSTM instead of regular RNN is due to the fact that LSTM usually performs better than RNN in most cases because of its architecture.

A unit of LSTM includes an input gate, an output gate, a forget gate, and a cell. The cell remembers information from the past and the gates regulate the flow of information into and out of the cell. To be more specific, the input gate takes the input and the last hidden state, through a sigmoid function, and pass the values to the cell and other gates. The forget gate controls what information remains in the cell and what to forget by computation and using activation functions sigmoid and tanh. The output

gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

LSTM Implementation was done based on the frame *tensorflow.keras*.DNN implementation is similar and much simpler so here we are going to show the LSTM only. Since LSTM usually takes a 3d-array as an input, we transform the 2d data frame to 3d by using *numpy.reshape*. The model has three layers, which are one LSTM layer, one Dense hidden layer, and one Dense output layer. The model includes a 0.2 dropout to avoid overfitting and a batch normalization for better performance and stability. The optimizer is 'adam' with a learning rate of 0.01, and the loss is 'mean absolute error'. The model is trained on 30 epochs with batch size equals 4. Below is the code for implementation.

```
verbose, epochs, batch_size = 0, 30, 4
neural_network = Sequential()
neural_network.add(layers.LSTM(units = 32,
        input_shape = (X_train.shape[1], 1), return_sequences=True))
neural_network.add(layers.Dense(256, activation='relu'))
neural_network.add(layers.Dropout(0.2))
neural_network.add(layers.BatchNormalization())
neural_network.add(layers.Dense(1, kernel_initializer='normal',
    activation='linear'))
opt = optimizers.Adam(learning_rate=0.01)
neural_network.compile(loss='mean_absolute_error', optimizer=opt,
    metrics=['mean_absolute_error'])
```

7.2 Random Forest

Random Forest Regression is a supervised machine learning algorithm, using several decision trees to train and predict the estimate of a specific indicator. The core of the algorithm lies in its idea of "Ensemble learning", which is a combination of "bootstrap aggregating" from Breimans and "random subspace method" from Tin Kam Ho. It is widely used in a marketing promotion, income analysis, risk prediction, personal recommendation, and other areas.

The two core concepts of Random Forest are "Decision Tree" and "Ensemble Learning". For our project, in the prediction process of missing TSS, the ensemble learning mechanism will calculate the average of n TSS estimates from n independent decision trees. Finally, this average TSS values will be the output of the model. The "Ensemble Learning" idea, which is represented in its crowd decision process, makes this model be one of the most accurate among other machine learning algorithms.

There are many advantages of random forest and its application in our project. We know that it is difficult for a model to predict the outcome with high-dimension data for the calculation ability limitation. For our project, high-dimensions of data(features) are adopted in the prediction of TSS; the

random forest is able to process these high-dimensions features simultaneously without extra work in the variable deletion. Moreover, The random forest can address the missing data efficiently, even if the missing part is quite large that does not affect the performance of the algorithm. Another good point is that Random Forest is not sensitive to outliers.

However, the disadvantages should also be mentioned as well. The decision process of this model is unobservable. We can not investigate its specific mechanisms but only tune the parameters based on experience and compare among their outputs, limiting the improvement of TSS precision. Another potential problem of our project is that the ability of a random forest to handle huge data sets. The bootstrap mechanism requires to make a prediction for n times, according to the parameter of the model. Currently, it works well considering the relatively small scale of our data set, but its slow ineffective disadvantages may show up when dealing with big data.

7.3 XG Boost

Extreme Gradient Boosting is a scalable Tree boosting system. It is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient descent framework designed for speed and performance. XGBoost is a highly flexible and versatile Model that can work through most ranking, classification, and regression predictive modeling problems. The main objective of XGBoost is to make the best use of available resources to train the model. Its key implementation features are Sparse Aware implementation, Block Structure, Continued Training. The main advantage of XGBoost is its Execution speed and Model performance, Compared to the gradient boosting's other implementations. The reason for this is XGBoost improves the Gradient Boosting Machines through systems optimization and algorithmic enhancements. Its regularization parameter reduces variance in the data. Aside from the regularization parameter, this algorithm leverages a learning rate and subsamples from the features, which increases its ability to generalize.

As we have a high dimensional and sparse data set, XGBoost regressor with l1 regularization was used to reduce the regression tree functions' complexity. XGBoost combines estimates of a set of simpler weaker models to predict for TSS accurately. Here, the weak learners are regression trees, and each regression treemaps an input data point to one of its leaves containing a continuous score. The Sparsity patterns in the data will be handled by automatically 'learning' the best missing value depending on training loss, so it predicts for TSS more efficiently. As we have distributed weighted quantile sketch algorithm to find the optimal split points among the weighted datasets effectively. The training proceeds iteratively. New trees will be created to predict the preceding tree's errors at each iteration and use a gradient descent algorithm to minimize the loss when adding new models. The final prediction for TSS is the sum of predictions from each tree.

7.4 Adaboost

Adaboost is an ensemble Learning algorithm proposed in 1997. The basic principle of adaboost algorithm is to combine several weak classifiers, often using single-layer decision tree, reasonably and make them into a strong classifier. It is a great graphical approach to solve probability problems. And the graph below graphically explains how adaboost works.

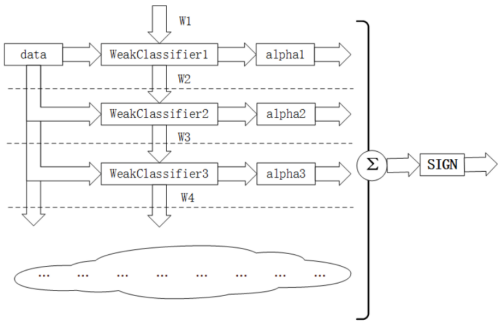


Figure 16: How WeekClassifiers work in Adaboost

Adaboost uses iteration as its core method. Each iteration only trains one weak classifier, and the trained weak classifier will participate in the use of the next iteration. In this way, in the Nth iteration, there are N weak classifiers, N-1 of which are previously trained and their various parameters are not changed in the Nth iteration. In the end, the final classification output depends on the comprehensive effect of these N classifiers. The procedures of adaboost can be summarized in the following steps:

1) Define data used for training is

$$T = (x_1, y_1), ((x_2, y_2), \ldots, (x_m, y_m)$$

Give the kth classifier a weight to each sample in the data used for train

$$D(k) = (w_{k1}, w_{k2}, \ldots, w_{km}), w_{1i} = \frac{1}{m}, i = 1, 2, \ldots, m$$

2) Train a new classifier by using all the training data.

3) Calculate the error rate of the new classifier and calculate a weight of the new classifier based on its error rate.

$$e = \frac{numberofuncorrectpredict}{numberofalldata}, \alpha_k = \frac{1}{2} ln(\frac{1 - e_k}{e_k})$$

4) Rebuild the weight. The weight of kth classifier and the weight for the k+1th classifier are

$$D(k) = (w_{k1}, w_{k2}, \ldots, w_{km})$$

$$w_{k+1,i} = \frac{w_{ki}}{Z_K} exp(-\alpha_k y_i G_k(x_i)), Z_K = \sum_{i=1}^{m} w_{ki} exp(-\alpha_k y_i G_k(x_i))$$

From the equation above, if the classification of number i data is wrong, $y_i G_k(x_i)$is negative, leading the number i data's weight to be higher in the next new classifier. Otherwise the $y_i G_k(x_i)$should be positive.

5) Use the new weight to train and get a new classifier like step 3. Repeat until a classifier getting error rate 0 appears or number of iterations is reached.

6) Do summary to all the classifiers

$$G(x) = sign[f(x)] = sign[\alpha_1 G_1(x) + \alpha_2 G_2(x) + \cdots + \alpha_2 G_2(x)]$$

$G_i(x)$ is classifier and α_1 is the weight of classifier.

In conclusion, adaboost uses weak classifiers in case of overfitting, especially useful when not sure how to split data into training data and testing data. Adaboost is also flexible because it can apply different classification algorithms as its weak classifiers. In this way, adaboost can be always highly accurate for it can be regarded as a combination of all classification algorithms with careful and concise weights.

All algorithms have disadvantages, even adaboost. The number of iterations, the number of weak classifiers, is not easy to set. However, some exhausting methods such as grid search can solve this problem well, ignoring time-consuming. And due to adaboost make full learn of all training data, outliers may decrease the accuracy a lot, which is a classification algorithm common problem.

7.5 Linear Regression

Being the simplest and most basic statistical technique for predicting continuous variables, linear regression was implemented. It performed well in most scenarios except for few. When the model was built using all the features, it had a low R^2 value. Forward Selection was thus implemented to automatically select the best features in the model. This improved the R^2 and subsequently reduced the MAE. When the linear regression was subjected to cross validation, the scores varied drastically, it can be observed from table 4.

Although an attempt was made to eliminate the outliers using local outlier factor method, it did not improve the model. To account for this uncertainty regularization was introduced.

Regularization: A penalty term is added to the cost function [26]. Two approaches were experimented with:

1) Lasso This is also known as L1 regularization. The model in this case is penalized by absolute weight co-efficients. given by the formula

$$\beta^{lasso} = arg\, min_\beta \sum_{n=1}^{N} \frac{1}{2}(y_n - \beta x_n)^2 + \lambda \sum_{i=1}^{p} |\beta_i|$$

Cross Validation	CV-1	CV-2	CV-3	CV-4	CV-5
Linear Regression	0.9337086	0.93416888	0.96406222	-5.0978903	0.85348659
Lasso	0.92481707	0.93561479	0.94354603	0.94357303	0.86545017

Table 4: Cross Validation scores for cycling before and after Regularization

2) Ridge A linear function of the squared coefficients is added as a constraint to penalise the model. This is also called as L2 regularization given by the formula

$$\sum_{i=n}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

The key difference amongst the two is that in case of Ridge all the coefficients are retained in the model only their values are shrunk, whereas Lasso can be used as a feature selection as the insignificant coefficients are removed.

Since our data has multicollinearity, regularization helps to a great extent. Lasso was preferred over Ridge due to its property of zeroing out the coefficients. This in a way enables us to identify important variables for each activity type. As a next step, TSS can be computed using only this information instead of collecting all the 44 columns easing the analysis. The important variable type for each activity are as follows:

- Distance, min Temp, No. of unique weekly activities, Avg. HR, Stride length and Vertical ratio are important for running
- Calories, No. of laps, Max. Speed, Elev. Gain, Max. power, Max HR and Max. Bike Cadence are important for Cycling

It can be observed from table 4 that the cross-validation scores post Regularization is more or less consistent across all the folds as compared to linear regression without regularization. This model performs the best for Cycling as compared to other activity types.

Several techniques such as Regularization using elastic net ,variable transformations and factor analysis could be implemented as a next step to improve the current model.

8 Evaluation

Evaluation allows us to check whether the goals that were set out were met or not. After training the model, we tested it against an unused part of the data set for evaluation to see the model's proficiency. Model evaluation helps us find the best model that fits our data and how well the chosen model will work in the future. We used Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-squared values for model performance evaluation and comparison. MAE gives us the absolute difference between the target value and the value predicted by the model. MAE is more robust to outliers. RMSE is the square root mean squared error. Compared to the MAE, RMSE penalizes more on large errors. R-Squared value indicates the proportion of the variance in the data set explained by the model.

8.1 Error Analysis

We performed Grid search and Cross-validation on our models. The intention of performing cross-validation is to test the model's ability for predicting TSS (target) based on some "new data "and for hyperparameter tuning. By "new data," we refer to new settings of experiments such as athletes with different and unseen feature values. Using cv = 3 in our analysis for instance, we split our data set into 3 folds where it uses each fold as testing set at some point. In the first iteration, we test the model using the first fold and the remaining data for training the model and make predictions. We repeat until each fold of the 3 folds has been used as a testing set. So, we performed Cross-Validation to evaluate the fit of the model to data and see how well it predicts the TSS. Simultaneously, this method allows us to check for overfitting.

8.2 Hyperparameter tuning

Unlike model parameters learned through a machine learning model, the Hyperparameter directly controls the training model. It has a remarkable impact on the model's performance. Choosing appropriate hyperparameters will increase the model's efficiency. To find the optimal parameter, we used a grid search. Grid Search evaluates all the combinations from a given range of hyper-parameters for each model and returns the best accuracy parameters. We used the same performance metric for evaluating the model and the Hyperparameter tunning

8.3 Results

After obtaining the optimal parameters from Grid search, we Finally used our models on the test data to predict for TSS

From Table 5 and Table 6, we can see that the XGBoost is doing a reasonably good job in predicting.The results displayed in the table are averaged across the activity type for all activities. The prediction is

Running			
Model	MAE	RMSE	R^2
XGBoost	29.91	46.28	0.64
RandomForest	27.95	46.31	0.63
LinearRegression	24.28	50.38	0.57
NeuralNetwork	24.81	41.87	0.71
AdaBoost	31.74	59.76	0.20

Table 5: Results w.r.t Running

Cycling			
Model	MAE	RMSE	R^2
XGBoost	13.76	21.65	0.94
RandomForest	16.66	25.03	0.93
LinearRegression	16.94	22.14	0.94
NeuralNetwork	45.67	60.82	0.75
AdaBoost	38.81	51.18	0.70

Table 6: Results w.r.t Cycling

not far from the actual values, although in some cases, the prediction falls well inside the range of true values. Overall the model is doing a decent job predicting unseen samples, and no significant overfitting is detected. Random Forest comes close followed by Linear Regression, Neural Network and ADA Boost.

The better results for cycling can be attributed to the data capture. Although, the source code for swimming was implemented, we did not include the results as very few valid data points were available for modelling.

9 PMC Generation

9.1 Preparatory Work

The $generate\_pmc$ module in the system is responsible for generating the PMC. After building and selecting the best models for each activity type for each athlete, the module will take the best modules to predict TSS values for the records which values are missing, and full the values in. Ideally, after this step, all the TSS values should be either already there before or be predicted, although some of the activities have very few records and are not suitable as a training set to predict missing values.

9.2 Adding ATL, CTL and TSB

When we get the data with TSS values ready, the $generate\_pmc$ module will add two columns called Fatigue (ATL) and Fitness (CTL) to the dataset. Fatigue is defined as the rolling average of the TSS values for the past 7 days, while Fitness is the rolling average of the TSS values for the past 42 days. This makes sense because the fatigue is highly related to the short-term Training Stress, and when it comes to the long-term, the training stress turns into the fitness of an athlete.

The challenging part is that, for the Fatigue/Fitness of the day we are working on, we cannot simply take the average of the TSS values from the several previous rows. For example, an athlete had a high load workout one week ago, the fatigue he got from that workout should not be considered equally as doing the workout yesterday. The solution that the team found is that using pandas rolling average based on dates. We first transform the data type of the 'Date' column to pandas DateTime, and use $pandas.Dataframe.rolling().mean()$ with setting the rolling period 42 and 7, and min_period 1 since we don't want to get 'NaN' for the first few days of the data.

After adding Fatigue and Fitness, the next step is adding Form (TSB), which is the variable that we take as the measurement of the training load for an athlete. The Form is simply Fitness minus Fatigue.

9.3 Plot PMC

We plot TSS, Fitness, Fatigue, and Form in a Performance Management Chart (PMC) refer to the website TrainingPeaks [14], we use subplots to distinguish the different scales for TSS and Fitness/Fatigue/-Form while they can share the same x-axis which is 'Date'.

We next plot zones (Transition Zone, Optimal Training Zone, High Risk Zone, etc.). Different athletes have different ranges of training period, and their ATL/CTL/TSB are quite different too, our non-hardcode plot function makes sure that the boundary of the zones and texts can be the plot in a nice way for all athletes.

Display/hide option for TSS is also added based on the client's suggestion because people who are not

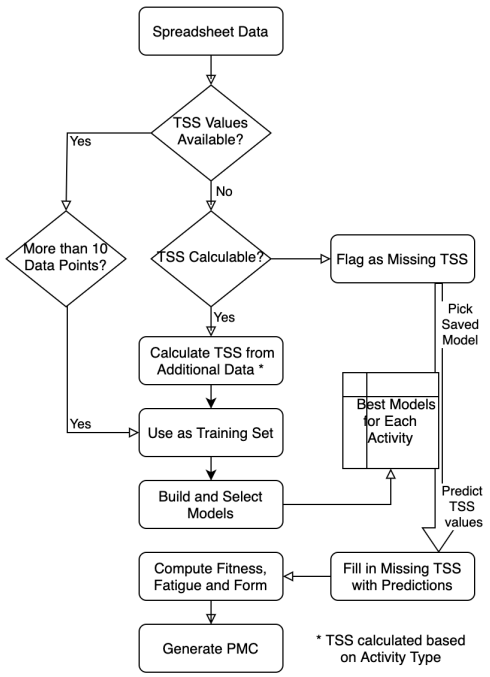


Figure 17: PMC Generation Logic

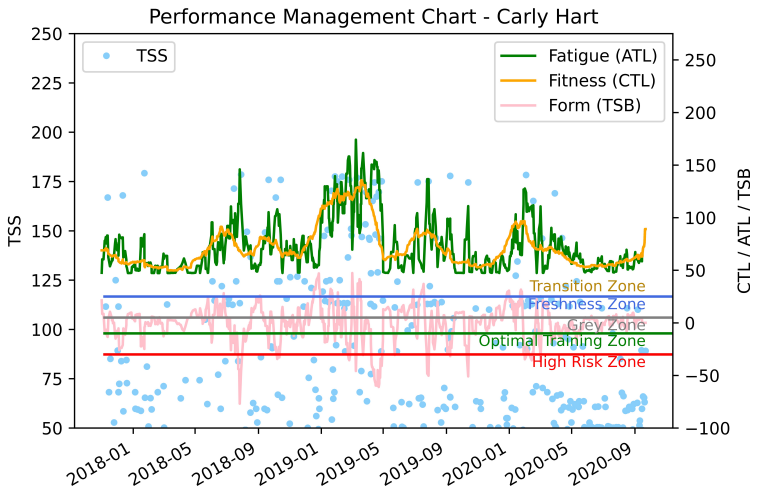


Figure 18: A preview of PMC Generated by System

familiar with training loads might get confused when they see the plot. Users can adjust the option by changing the input arguments at the entry of the system.

The figure above is the PMC for athlete Carly Hart generated by our system. We can see that Carly was generally in the optimal training zone and grey zone, which is good. However, Carly should also be careful of the days when he was in "High Risk Zone".

The plot also shows that as Fitness goes up, the Fatigue goes up as well and at a higher rate. Especially on days with high-TSS workouts, the Fatigue climbs even faster. It's quite intuitive because doing a hard workout makes people feel it in the coming days. Usually, consistent training will lead the Fatigue and Fitness to increase steadily. If there is a sharp drop in Fatigue and Fitness, we may consider that the athlete was suffering from sickness or injury [15].

10 Conclusion and future work

A number of enhancements could be made to our analytical module by better data capture. Values such as CSS (Critical Swim Speed) and Lactate Threshold are currently being approximated using all the '.fit' files for the athlete, once adequate data is available, a constraint could be imposed to approximate these values using data from the past 6 months or 3 months only. Alongside this, more research could be done with respect to estimating TSS scores for Strength Training activities. In our feature engineering, we mainly use techniques defined by Joe Friel and Andy Coogan, a lot of future work could centre around exploring other techniques.

The models built are currently being used to predict for missing TSS values. These can also be used to estimate TSS values for future training activities with slight modifications in the source code. Attributes such as Distance, No. of unique weekly activities, No. of laps, Speed, Power and Bike cadence proved to be important factors in keeping a check on athletes' activities to track their fitness, fatigue and form. These predictors are essential in determining if an athlete is over-training or under-training.

The current system automates the model selection process based on error rate. Further improvements could include definite models based on activity types or other valid constraints.

Extensive study was done to understand the behaviour of Performance prediction models (discussed in section 2.2). Future teams could use this to check for athlete performance and analyse if they can make a podium.

11 Appendix

Team Members and Roles

Name	Contributions
Lin Mi	Related work, Data cleaning, Feature engineering, Random Forest
Tingli Qiu	System Developer, Data Cleaning, Feature Engineering, PMC generation, LSTM
Spoorthi Suresh Avvanna	Data cleaning and Exploration, Feature engineering, Linear regression, Client communicator
Chitra Naga Durga Sindhusha Veluguleti	Introduction, Data cleaning, XG Boost and Model Evaluation
Yuhan Zhao	AdaBoost

System repository: `https://github.com/Data-Science-Project-G13/monitoring-athletes-performance`

Meeting Summaries: `https://github.com/Data-Science-Project-G13/monitoring-athletes-performance/tree/master/meeting_summaries`

References

[1] Op De Beéck, T., Meert, W., Schütte, K., Vanwanseele, B., & Davis, J. (2018, July). Fatigue prediction in outdoor runners via machine learning and sensor fusion. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 606-615).

[2] P. Leo. (2016). A contemporary approach to training load quantification in endurance athletes. Univerasity of Innsbruck.

[3] Wallace, L. K., Slattery, K. M., & Coutts, A. J. (2014). A comparison of methods for quantifying training load: relationships between modelled and actual training responses. European journal of applied physiology, 114(1), 11-20.

[4] Cardinale, M., & Varley, M. C. (2017). Wearable training-monitoring technology: applications, challenges, and opportunities. International journal of sports physiology and performance, 12(s2), S2-55.

[5] Cristina-Souza, G., Mariano, A., Souza-Rodrigues, C. C., Santos, P. S., Bertuzzi, R., Lima-Silva, A. E., & De-Oliveira, F. R. (2019). Monitoring training load in runners, throwers and sprinters/jumpers during a preparatory training camp. Journal of Physical Education & Sport, 19, 173-177.

[6] Pind, R., & Mäestu, J. (2017). Monitoring training load: necessity, methods and applications. Acta Kinesiologiae Universitatis Tartuensis, 23, 7-18.

[7] Sanders, D., Abt, G., Hesselink, M. K., Myers, T., & Akubat, I. (2017). Methods of monitoring training load and their relationships to changes in fitness and performance in competitive road cyclists. International journal of sports physiology and performance, 12(5), 668-675.

[8] Ramos, G., Vaz, J. R., Mendonça, G. V., Pezarat-Correia, P., Rodrigues, J., Alfaras, M., & Gamboa, H. (2020). Fatigue Evaluation through Machine Learning and a Global Fatigue Descriptor. Journal of Healthcare Engineering, 2020, 1-18.

[9] Halson, S. L. (2014). Monitoring training load to understand fatigue in athletes. Sports medicine, 44(2), 139-147.

[10] Pilarski, P. M., Qi, L., Ferguson-Pell, M., & Grange, S. (2013, June). Determining the time until muscle fatigue using temporally extended prediction learning. In Proceedings of the 18th International Functional Electrical Stimulation Society Conference (IFESS), Donostia-San Sebastian, Spain (pp. 7-8). ctional Electrical Stimulation Society Conference, pp. 1-4, 2013.

[11] Maszczyk, A. , Roczniok, R. , WasKiewicz, Z. , Czuba, M. , MikolAjec, K. , & Zajac, A. , et al. (2012). Application of regression and neural models to predict competitive swimming performance. Perceptual & Motor Skills, 114(2), 610-626.

[12] Berndsen, J. , Smyth, B. , & Lawlor, A. . (2019). Pace my race: recommendations for marathon running. the 13th ACM Conference. ACM.

[13] Sudin, S. , Md Shakaff, A. Y. , Zakaria, A. , Salleh, A. F. , Kamarudin, L. M. , & Azmi, N. , et al. (2018). Real-time track cycling performance prediction using anfis system. International Journal of Performance Analysis in Sport, 1-17.

[14] What is the Performance Management Chart?. (2020). Retrieved 31 May 2020, from *https://www.trainingpeaks.com/blog/what-is-the-performance-management-chart/*

[15] Friel, J. (2015). Managing Training Using TSB - Joe Friel. Retrieved 31 May 2020, from *https://joefrielsblog.com/managing-training-using-tsb/*

[16] ATL, CTL & TSB explained. (2007). Retrieved 31 May 2020, from *https://ianbarrington.com/2007/03/02/atl-ctl-tsb-explained/*

[17] TRIMP. (2020). Retrieved 31 May 2020, from *https://fellrnr.com/wiki/TRIMP#TRIMPavg-Average-Heart-Rate-Scaling*

[18] Sanders, D., Abt, G., Hesselink, M. K., Myers, T., & Akubat, I. (2017). Methods of monitoring training load and their relationships to changes in fitness and performance in competitive road cyclists. International journal of sports physiology and performance, 12(5), 668-675.

[19] Malone, S., & Collins, K. (2016). Relationship between individualized training impulse and aerobic fitness measures in hurling players across a training period. The Journal of Strength & Conditioning Research, 30(11), 3140-3145.

[20] Training Stress Scores (TSS) Explained. (2020). Retrieved 31 May 2020, from *https://help.trainingpeaks.com/hc/en-us/articles/204071944-Training-Stress-Scores-TSS-Explained*

[21] Estimating Training Stress Scores for Mountain Biking. (2011). Retrieved 31 May 2020, from *https://www.coachcox.co.uk/2011/04/08/estimating-training-stress-scores-for-mountain-biking/*

[22] Friel, Joe. "Managing Training Using TSB - Joe Friel". Joe Friel, 2015, *https://joefrielsblog.com/managing-training-using-tsb/. Accessed 28 May 2020.*

[23] Friel, J. (2016). The Triathlete's Training Bible: The World's Most Comprehensive Training Guide. Velo-Press.

[24] What is FTP in cycling and how do I test and improve it?. (2020). Retrieved 30 Aug 2020, from *https://www.cyclingweekly.com/fitness/ftp-cycling-363865*

[25] Estimating Training Stress Score. Retrieved 30 Aug 2020, from *https://www.trainingpeaks.com/blog/estimating-training-stress-score-tss/*

[26] Ridge and Lasso regression: L1 and L2 Regularization. (2018). Retrieved 05 Sep 2020, from *https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b*