Insert the Project Title here

Project Number: 22-1-2-2546

Project Report

Student: Sela Dai ID: 205846488

Supervisor: Gilad Agam

Project Carried Out at: Griiip

Contents

List of figures

List of tables

List of equations

Abstract

If you have seen a motorsports event as a car race, you might have noticed that it can be quite challenging to follow for a non-experienced viewer.
While massive engineering efforts in the motorsport industry are driven by a large and varied amount of data, leveraging this data for in-depth analysis has not yet reached the average viewer, as have been done in many other types of sports.
This project aims to tackle this challenge with a data-driven solution of using telemetry data transmitted from the race cars, to create insightful *Race Events Map.* Those events will be ranked and be accessible live to fans and racing teams through an engaging application.
Facing multiple engineering challenges of signal processing, data engineering, algorithms, research, hypothesis, testing and visualization.
The essence of the project is using signal processing methods and race logic to implement three racing data analysis algorithms, shown in the following diagram, and will be described in detail in this book: Overtakes, Battles, and Errors (Time Gains and Loss)

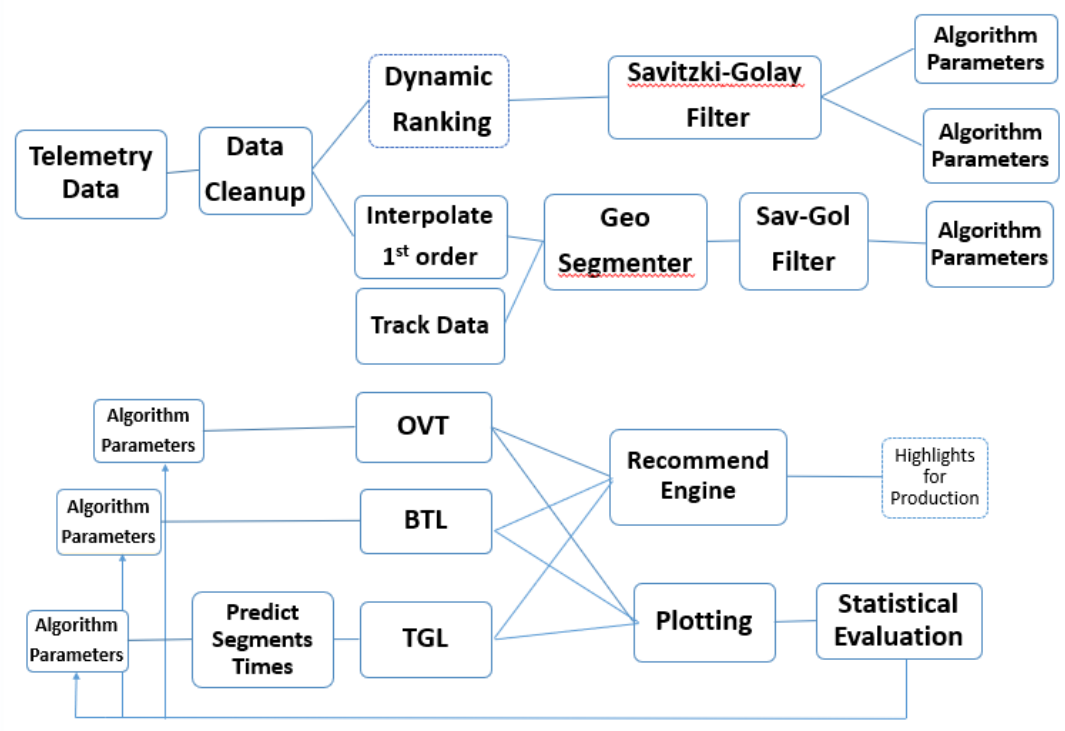

Figure 1: Block diagram of the project workflow

1 Introduction

In the world of motorsports, understanding the intricacies, tactics, and layout of a race from a live feed can be challenging and limited to experienced individuals. To engage a wider audience the *Racing Mapper* project has come to live. This project aims to unravel the complexity of races by providing a detailed significant racing events layout, that will be followed with a dynamic visual representation with the insightful analysis of the tactics and factors that influence outcomes.

The project strives to influence three main groups:

Viewers

By bridging the gap between the live feed and viewers, this project seeks to captivate and involve racing enthusiasts, by making racing more interactive, engaging, insightful, and educational.

Race production

Automating the highlighting process can save many hours of man work for talented TV sports production workers, editors, sport broadcaster, and enable the to provide much better and focused racing production.

Drivers and racing professionals

The drivers and racing coaches and engineers strive to perfect their vehicles and their team's performance. They need a thorough understanding of the race, which a designated map of events can provide, helping them identify and work on weakness, identify good and bad driving patterns, and attention areas to enable constant improvement.

1.1 Project goals

The project goals are achieving quality identification, and mapping of three distinct race events: Overtakes, Battles, and Errors (Time gain and loss), by designing algorithms using signal processing methods. The quality of the mapped event should achieve at least 80% hit rate and at more 10% false positive, when compared to a tagged raced event.

1.2 Approach to solving the problem

The data that will be used in this process is generated by the company's hardware device called RedBox. It is a raspberry-pi based Linux module, with GPS, CAN, UMI and communications modules (4.3).

The live telemetry raw data is transmitted and stored in AWS cloud database.

This project takes the following approach to achieving the above goals:

- Preprocess - transform the raw data into a desired dataset for analysis.
- Hypothesis → research → implementation → testing → incremental revisions (cycles)
- implementation – writing Python algorithms using `pandas, numpy, scipy`
- Testing – using visualization assessment and following matrixes of Hit Rate, and False Positive rates, when comparing to classified data.

1.3 Existing work

The insights accesibily in the racing industry is still in his crawling stages. Although, while they haven't publicly released their algorithms, Formula 1 have created a platform of insight that is similar in the recent years, with a wide range of analysis.

The extent of this platform is wide and the implementation is done well, it confines only to the Formula 1 racing league, and excluds thousands of over racing leauges worldwide. We are trying to brigde this gap.

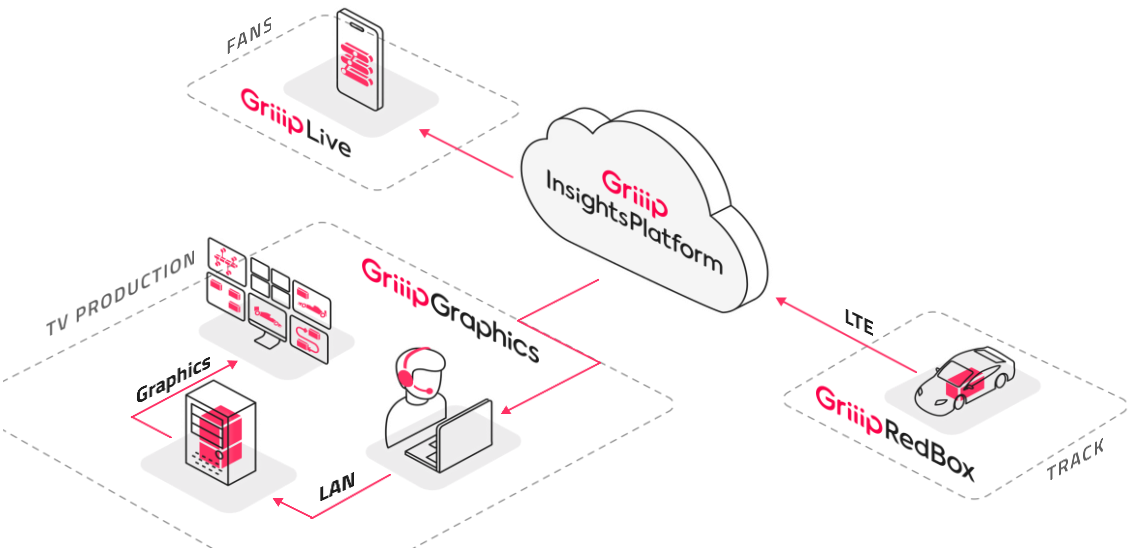

Figure 2: Griiip Insights System Design

2 Theoretical background

In this section, the theoretical background and relevant algorithms will be described. Alternative algorithms for project implementation should also be referenced.

2.1 Racing data collection and challenges

Receiving stable live data transmission from a 250 km/h moving vehicle cannot be taken for granted.

The project faces challenges of measurements that vary sensitivity, smoothness, measurements ranges, and offsets.

2.1.1 CAN (Controller Area Network) bus

A robust and widely used communication protocol in automotive and industrial applications, enabling real-time data exchange among electronic control units (ECUs) in a distributed network.

Used in our system to receive real-time data from the vehicles control as throttle, brake, gear, etc.

2.1.2 Interferences

Sensor signals are affected by noise resulting from vibration or electromagnetic fields from the ignition system. In addition, temperature, pressure, mounting of the sensor and contamination by fluid, and dirt can influence the output signal of the sensor[1].

2.1.3 Networking and data loss

With the rate of transmission and the amount of data from ~25 race care during a live race, a problem of data loss has occurred.

This is due to network congestion, failed hardware components, low GPS and cellular reception. The data had to be verified and completed using interpolation.

2.2 Signal Processing

2.2.1 Interpolation

Interpolation is a type of estimation in the field of numerical analysis, which constructs new data points based on a discrete set of existing data points.

---

[1] see referencing details in Jorge Segeres Analysis Techniques for Racecar Data Acquisition book ch.19. [1]

Linear (1st order) Interpolation:

For estimation the y value for a giving x, we will find the closest two x values, where $x \in (x_1, x_2)$, and estimate using the following equation:

$$y = \mathrm{y}_1 + (\mathrm{x} - \mathrm{x}_1)\frac{(\mathrm{y}_2 - \mathrm{y}_1)}{(\mathrm{x}_2 - \mathrm{x}_1)}$$

Equation 1: 1st order Interpolation

2.2.2 Filtering

Savitzky–Golay filter is a digital filter with the purpose of smoothing the data and increasing the precision of the data without distorting the signal tendency. Used with an odd symmetrical window the filtered value is calculated as such:

$$Y_j = \sum_{i=\frac{1-m}{2}}^{\frac{m+1}{2}} C_i y_{j+i}\,, \qquad \frac{m+1}{2} \leq j \leq n - \frac{m-1}{2}$$

Equation 2: Savitzky-Golay filter

Where C_i are the convolution coefficients (see Appendix A).

2.2.3 Regressions and Algorithms

2.2.3.1 Linear Regression

Linear regression is a statistical technique used for modelling the relationship between a dependent variable and one or more independent variables. straight that minimizes the differences between the observed data points, and the predicted values on the line, in terms of least square.

$$y_i = x_i^T \beta + \epsilon_i$$

Equation 3: Linear Regression

2.2.3.2 One hot encoding

One hot encoding is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions.

We convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector.

2.2.3.3 Outliers Identification

Outliers are observed data points that are far from the least-squares line. They have large errors, where the error or residual is not very close to the best-fit line.

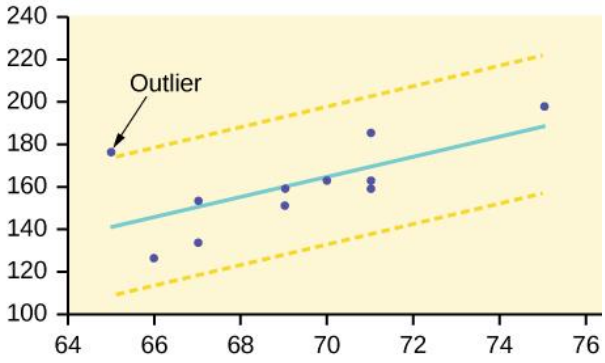

Figure 3: Outliers Identification with regression

In the project we will use outliers identification method that is based on the distance from the linear regression fit.

In Fig.3, the light blue line is the linear fitting line, and the yellow lines are the confidence interval.

s is the standard deviation of all the $y_i - f(x_i) = \epsilon$ values, where n is the total number of data points.

$$s = \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2}{n-2}}$$

Equation 4: Standard deviation for the fitting model

We then classify outlier as being with an error ϵ ,greater than K standard deviations for the model fit $f(x_i)$.

$$\epsilon \geq 2 \cdot K$$

Equation 5: Outlier classification rule

3 Simulation

As the project is software based, the simulations are part of the assessment and progress and act as testing of the algorithm.

3.1 Cloud Simulation Environment

In accordance with Griiip backend architecture, the simulations and tests in the project were preformed using AWS infrastructure. As it is specific and not critical for the understanding of the project, the detailed explanation of the environment can be found in Appendix B.

3.2 Plotting

After the executing of the algorithm has completed, plots of the results were made and analysis matrixes were implemented to assess, revise and improve the results. As we will see later 5).

4 Implementation

The implementation of the project was done via python software implementation with a signal processing approach, and racing logic specifications.

We will start by breaking down and describing the system's implementation in two stages: the pre-processing, and the racing event detection.

Following the above, we will take a detailed view into the hardware[2] and software architecture.

4.1 Pre-processing

Let's first take an overview of the first part of the system, the pre-processing, followed by a detailed description of each component.

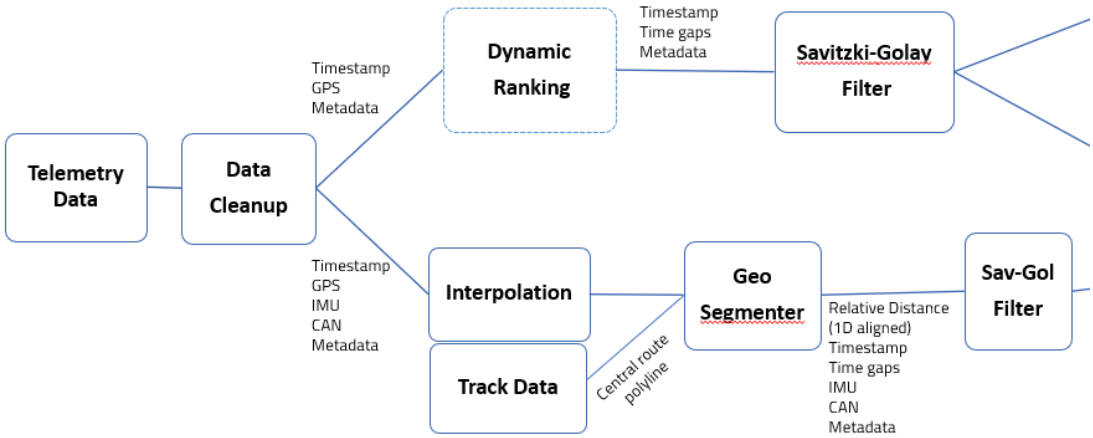

Figure 4: The data pre-processing flow chart

4.1.1 Telemetry Data

The fundamental input of the system is the telemetry data. Which includes the GPS, gyroscope, CAN and metadata data that is streamed live from the race cars through the RedBox (4.3).

With the most relevant attributes in the telemetry data being:

timestamp, lapname (id), carnumber, gpslat/long, lat/long acc (acceleration), speed, roll/pitch/yaw rate, and the throttle and brake attributes which originated through the CAN and are found in the extras.

[2] The HW component is outside the scope of the project but will be described to gain better understanding of the data input.

timestamp	lapname	gpslat	gpslong	speed	thingname	extras
2022-10-08 08:...	1509100822...	49.32644	8.56671	175.79925	RB-EU-17089	{"memotecAccX":0.48,"tea...
2022-10-08 08:...	1510100822...	49.33234	8.58427	74.16149	RB-EU-17107	{"memotecTintake":28.7000...
2022-10-08 08:...	1514100822...	49.33150	8.57351	223.29194	RB-NA-17045	{"teamTtRr":9.8,"teamWspfl"...
2022-10-08 08:...	1486100822...	49.33164	8.58023	251.10898	RB-EU-17078	{"teamTtFr":5.0,"teamPfuel":...
2022-10-08 08:...	1487100822...	49.33163	8.58013	259.94116	RB-EU-17094	{"teamTpFl":1.72,"memotec...
2022-10-08 08:...	1489100822...	49.33054	8.57697	199.89006	RB-EU-17091	{"memotecTmanifold":25.20...
2022-10-08 08:...	1491100822...	49.32947	8.56448	181.26820	RB-NA-17060	{"brakepresRear":21.0,"team...
2022-10-08 08:...	1494100822...	49.32907	8.56478	229.68874	RB-EU-17096	{"teamTcactive":0.0,"teamW...

Table 1: Telemetry Data sample

4.1.2 Data Cleanup

Preparing the telemetry data collected from live racing cars is crucial before it can be used for any data analysis process. The data is collected in a semi-structured format, from different sources which requires organization and transformation into a usable format.

The data cleaning process involves few steps:

- Selecting and extracting the relevant attributes.
- Data exploration: determining patterns, relationships, and trends in the dataset.
- Removing outliers and missing values.
- Handle noisy or inconsistent data points by filtering, and/or scaling the data appropriately, to ensure consistency across the different cars' sensors and time intervals.

In addition, data synchronization and alignment are also critical in this stage, as we want to analyze only the relevant racing data. Hence, we have filtered data, and laps which are not relevant to the analysis as: formation laps[3], pit stops[4], laps which are not competitive. By applying time filters (for the race time, as cars transmit data when the car ignition is on, before and after the race), by integrating the pit stop external indicator [5] , and by filtering the laps by the average acceleration to remain with only the competitive driving time. By preprocessing and preparing the data, we can now ensure better data quality, and lay the foundation for more accurate and meaningful insights.

---

[3] A brief practice lap taken by race cars before the start of a race to warm up tires and familiarize drivers with the track conditions.

[4] brief stops where cars come into the pit lane for tire changes or adjustments.

[5] Integrating the data from the official race timekeeping, measured by optical on-track sensors

4.1.3 Interpolation

For handling missing data that occurs, a time-based linear interpolation (2.2.1) is used, for missing data of up to 3 seconds per time frame per car.
As the regular telemetry transmission rate is 10Hz, and larger missing data windows seems to have thrown a bigger exception that cannot be overcome with interpolation.

4.1.4 Track Data

Mapped data of the racetrack's central polyline[6]. This will allow to flatten the coordinates data into a single dimension attribute 'Relative Distance' that represents a floating point with the closest perpendicular point along the central polyline.

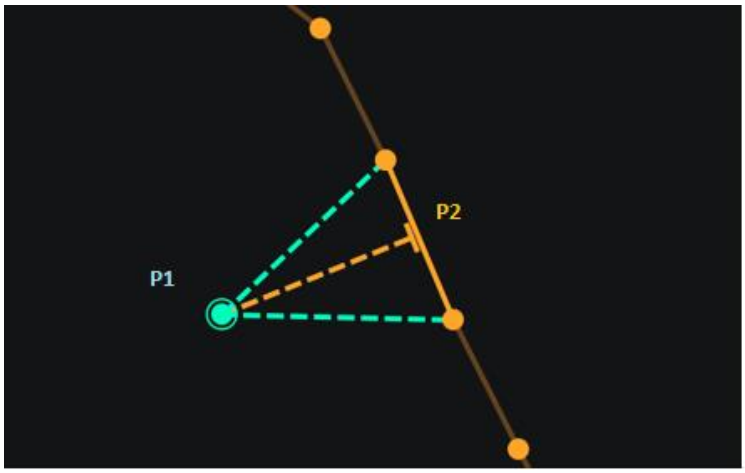

Figure 5: Central polyline and relative distance

Where p_0 is the start of the central line, L is the total length, p_1 is the telemetry data point and p_2 is the closest perpendicular point, we will calculate:

$$0 \leq \ relative\ distance\ = \frac{Distance(p_0, p_2)}{L} \leq 1$$

Equation 6: Relative distance calculation

4.1.5 Geo Segmenter

The Geo Segmenter is an analysis module that enables synchronization and alignment of the telemetry data across different drivers and laps, by segmenting it in respect to an input relative distance interval.

[6] A series of ordered points along the center of the race track.

For example, if we would like to have 100 segments along the track for each lap to enable aligned comparison and analysis, we will feed the module with $\mathrm{RELATIVE\_DISTANCE\_INTERVAL} = 0.01$.

The module will interpolate only the numerical and time data columns, as the metadata as car and driver id will naturally remain static.

4.1.6 Savitzki-Golay filter

After alignment by interpolation the Savitzki-Golay filter (2.2.2) is applied to provide a smoother interpretation and reduce noise.

Figure 6: Example of Savitzki-Golay filter for speed attributes

4.1.7 Dynamic Ranking

As it is implemented in Griiip, not part of the project's implementation, but some of the project work build upon the dynamic ranking data, it will be explained in Appendix C.

4.2 Racing events detection

Now that we have covered the pre-process stage, we can investigate the event detection process and algorithms in more depth.

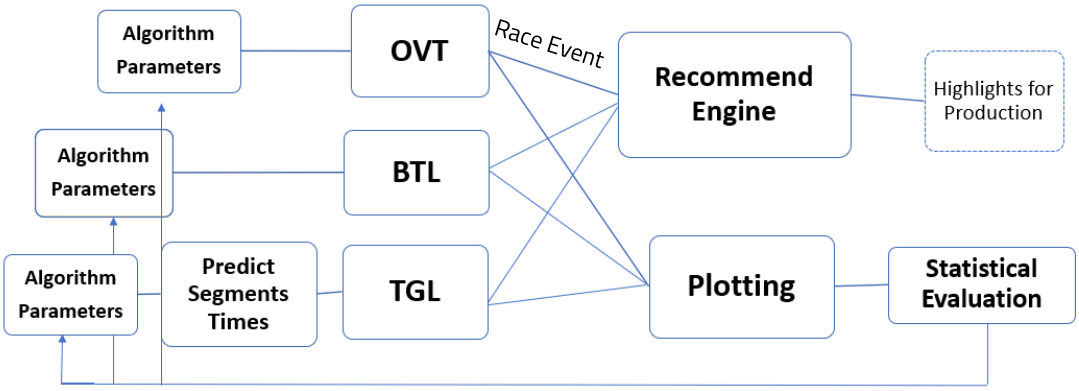

Figure 7: Racing events detection flow chart

4.2.1 Predict Segments Times

The prediction of segments times module serves the purpose of identifying the significant outliers in time performances of drivers with respect to the *race group pace*.

Race group pace is a term we have generated, as we would like to compare the performance of the driver in relation to the other drivers that are in similar conditions to his. Therefore, we need a strategy to dynamically categories the drivers' timing during the race segment to a set of comparable past segment's performances.

For this purpose, we have come up with two parameters that will help define the above group:

- N - The number of closest drivers to consider.
- M - The number of look behind laps to consider.

After defining and filtering the race group pace for each driver's segment, we will use this set as the input for a linear regression fitting, and identify the outliers as described in the 2.2.3.3.

4.2.2 TGL – Time Gain / Lost

Using the recent predicted segment time generated from the previous module, we could now identify the appropriate outliers and determine event of time gain or lost in a near real-time approach.

For this final classification we have defined three algorithm's parameters:

- SIG [s] - Filter outliers by delta (offset) > SIG standard deviation from the fitting line.

- MIN_TIME_DELTA [s] - minimum total time gained or lost compared to referenced segment[7], to qualify as a significant time gain or loss.
- MAX_NEUTRAL_DISTANCE [relative distance] (optional) - the maximum relative distance (by segments) between possible TGL mark windows for registering a new separate mark.

After the outlier identification filter, we will aggregate events by the MAX_NEUTRAL_DISTANCE and filter again by MIN_TIME_DELTA for significant events only.

4.2.3 OVT - Overtake

An overtake segment is marked when a driver overtakes another in a competitive setting. The parameters below describe the settings of the "competitive overtakes".

[*Definition*] "OVT window" - a segment that is a 'candidate' for being an OVT mark. The window can be open, temporarily closed, finished or false (Fig. 8). The OVT window is opened when the ranks of two drivers changes.

For this classification we have defined three algorithm's parameters:

- MAX_GAP_CHANGE_RATE $\triangleq mgcr\ [\frac{s}{s}]$ - calculate the average gap change rate (slope) within a certain timeframe around the overtake and define a threshold rate for competitive OVT (too high slope means that the overtake was "too easy" and not competitive).
- OVERTAKE_MARGIN $\triangleq mr\ [s]$ - a timeframe window around the overtake start, to calculate the gap change rate by. Currently, the window is symmetrical before and after the overtake.
- RELEASE_TIME $\triangleq rt\ [s]$ - to validate that the 'overtaker' wasn't overtaken back shortly after.

Classification algorithm:

We will merge overtakes windows within rt from each other, and consider:

$$gcr = \frac{interval(t+mr) - interval(t-mr)}{2mr} < mgcr$$

Equation 7: Overtake classification

[7] Comparing to the segment time that was predicted with a linear fit.

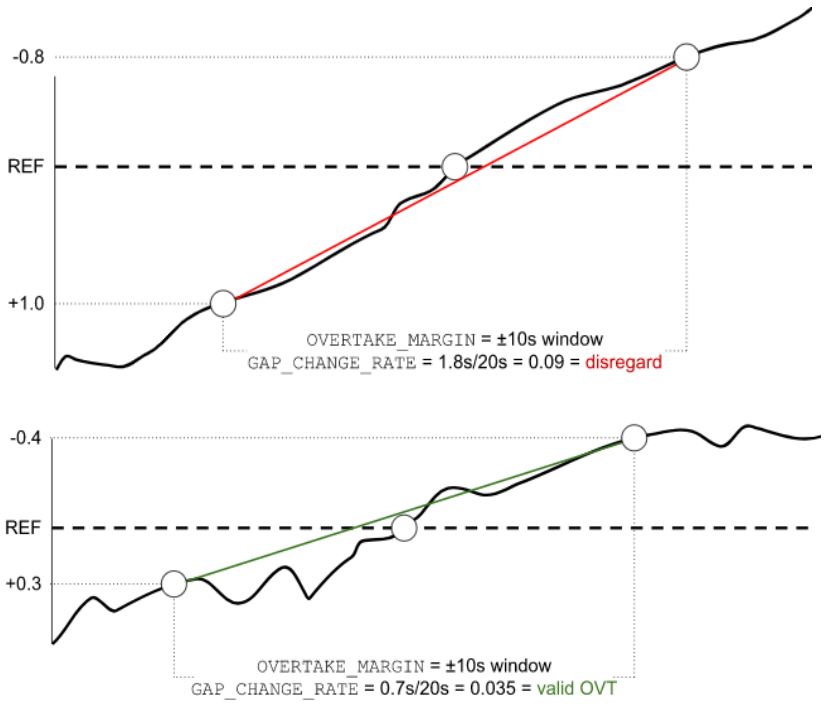

Figure 8-9: Overtake classification

4.2.4 BTL – Battle

A battle segment is marked when two or more drivers are within a certain proximity to each other, closer than a defined threshold, for a significant time during a race session.

Declaring a battle can be done while it's still ongoing but closing a battle segment can only be done some time after the battle has seemingly ended.

[*Definition*] "BTL window" - a segment that is a 'candidate' for being a BTL mark. The window can be open, temporarily closed, finished or false.

The BTL window is opened when the interval gap between drivers is smaller than a defined threshold.

For this classification we have defined three algorithm's parameters:

- TIME_GAP_THRESHOLD $\triangleq gt\ [s]$ - maximum time gap between drivers (interval) to count as active battle.
- MIN_MARK_TIME $\triangleq mt\ [s]$ - minimum BTL mark length time, to exclude "touch and go" battles or easy overtakes.
- RELEASE_TIME $\triangleq \delta_t\ [s]$ - the maximum time window between two BTL windows, under which the two are combined into one.

Classification Algorithm:

$$\exists\ t_{start}, t_{end},\ \delta_t : \forall t\ \in (\ t_{start}, t_{end} - \delta_t) :\ \exists t' \in (t, t + \delta_t) :\ interval(t') < gt$$

$$and\ t_{end} - t_{start} > mt$$

Equation 8: Battle classification

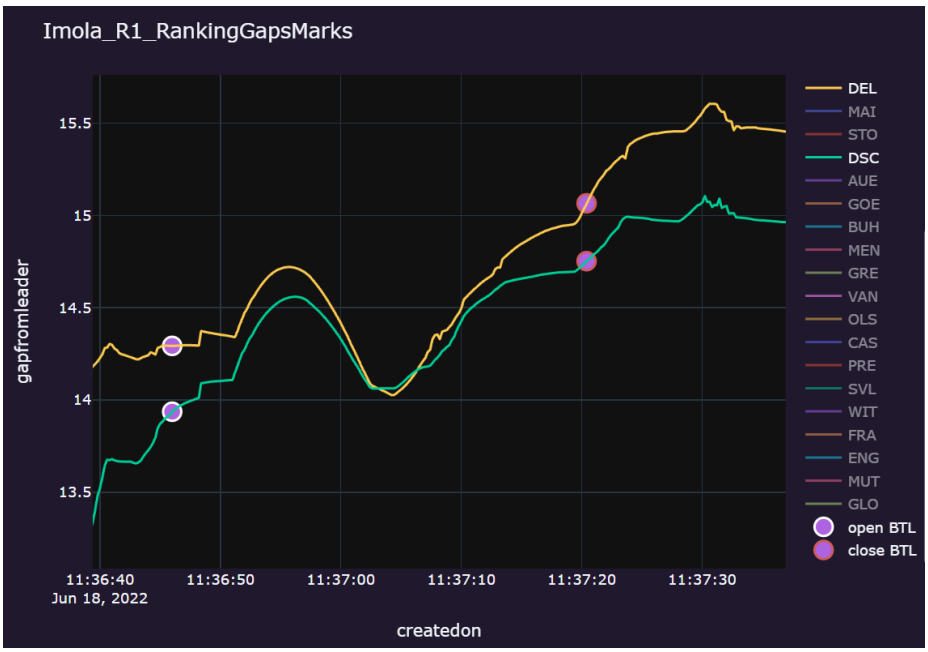

Figure 10: Battle classification

4.2.5 RecommendEngine

For racing highlights purpose, a recommendation engine was implemented to rank the race events, with consideration of:

- Event type score $\triangleq es$: $0 < es \leq 1$ – score per type of event, reflection the relative importance of it to the race.
- Current participants race rank $\triangleq cr$ – the current rank of the participation drivers during the event.
- Championship rank importance $\triangleq$ sri: $0 < sri \leq 1$ – indicate how important is an event is for the seasonal ranking of its participant[8].

$$event\_score = es \cdot \frac{N - cr}{N - 1} + (1 - es) \cdot \text{sri}$$

Equation 9: RecommendEngine

4.2.6 Plotting

To review the algorithm results and the events detection in relation to the input parameters and accuracy several types of plots have been made.

---

[8] For example: if two drivers, ranked 2nd and 3rd in the season overall rank with 2 points delta, fight for the lead of the race and an overtake occurs, the difference in the final position will can change the overall championship ranking with 5 points delta for 1st and 2nd places.

Some are written in the python library matplotlib but mostly written with plotly that enables rather easy for dynamic html plotting.

4.2.7 Statistical Evaluation

After the resulting event identification, we can compare to a tagged event table to validate our result, measure the accuracy.

We use two types of measurements when comparing to tagged events:

- Hit Rate [%]: the rate of tagged events for a particular race and event type that were identified by the mapping algorithm.
- False Positive [%]: the rate of identified events that were not tagged for a particular race and event type.

In addition, to evaluate the regression's predicted segment times, we use measurements such as:

compared_to	r2	rmse	avg_abs_error	medianabs_error	avg_error	median_error
predicted_seg_time	0.977	0.751	0.306	0.099	0.112	0.008
avg_seg_time	0.872	1.787	0.827	0.116	-0.481	-0.050
median_seg_time	0.797	2.252	0.727	0.078	-0.237	-0.013

Table 2: Regression fitting evaluation

The measurements include r squared, RMSE, average absolute error, median absolute error, average error and median error.

4.3 Hardware Description

The telemetry, the main source of data that has been used in this project is generated by Griiip's hardware device called the "RedBox".

As it is not part of the implementation, it will be explained in Appendix CD.

4.4 Software Description

The software in the project can be categories to four categories:

- Main application: processing data from a racing session and executing selected, or all, event mapping algorithms. Automatically triggered or executed on demand.
- Analysis Modules: The classification and specific processing modules: TGL, OVT, BTL and inherited classes.
- Jupyter Notebooks: coding notebooks used for research and initial data exploration, and testing hypothesis.
- Code utilities: for shared functionality between modules and notebooks: data analysis functionality, time handling, database, and plotting.

5 Analysis of results

5.1 Event detected and highlights database

Running the algorithms on 3 races data, resulted in the following number of racing events detected:

Event Type	Av. #Events
BTL - Battle	211
OVT - Overtake	178
TGL - Time Gain Loss	262

Table 3: Results - events detected

The following numbers are the average events per race session.

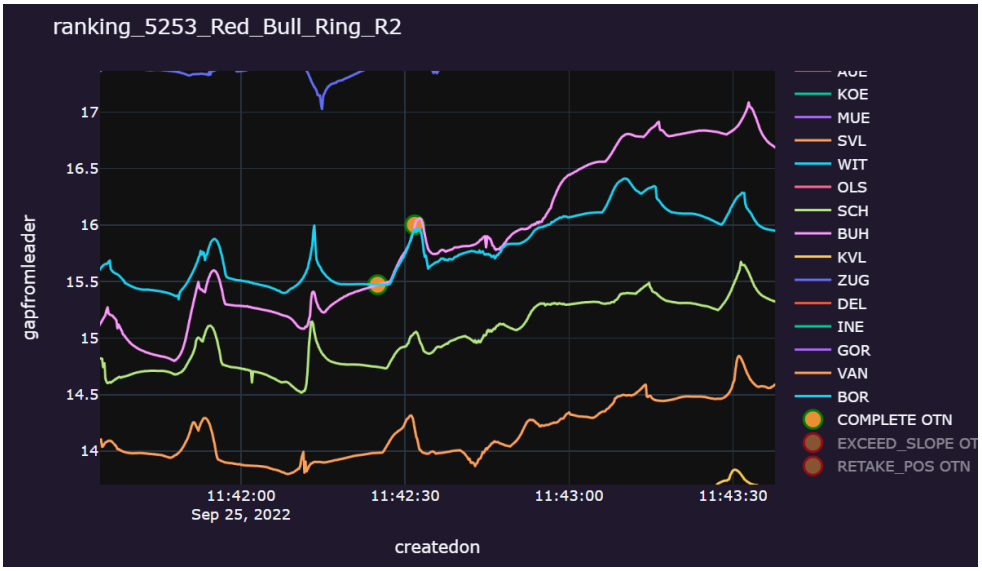

Figure 11: Overtake classification

5.2 Accuracy and comparison

Now we can assess the algorithms' results by two measurements hit-rate, false-positive, with respect to the tagged events data.

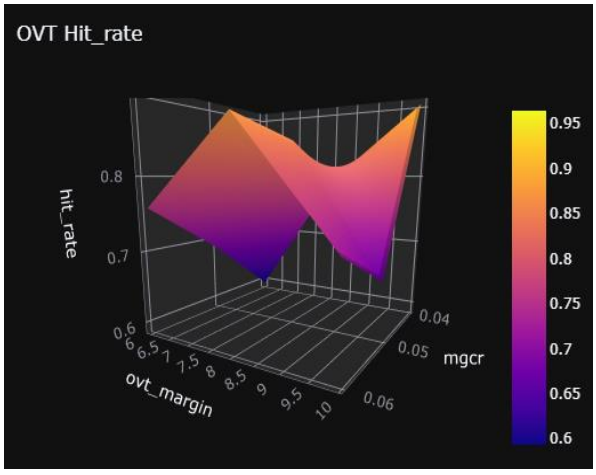

Fig. 12: Results – hit-rate by parameters

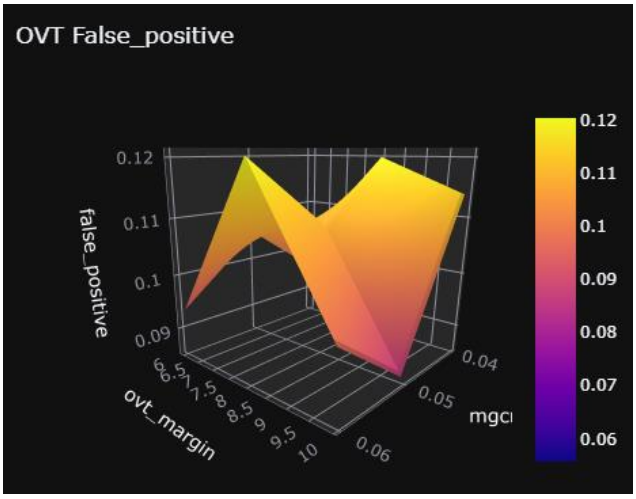

Fig. 13: Results –false-positive by parameters

Event Type	Av. #Events	Hit Rate	False Positive
BTL - Battle	211	93.2%	9.7%
OVT - Overtake	178	90.4%	8.9%

Table 4: Results – hit-rate and false-positive

5.3 Alternative algorithms

Unfortunately, there are no public algorithms for this specific purpose.

As mentioned, the insights in the racing industry is still in its crawling stages. Formula 1 have created a platform of insight that is similar in the recent years, with a wide range of analysis, but it is confines only to the Formula 1 racing league, and excluds thousands of over racing leauges worldwide, that this project andf Griiip are aiming to help.

6 Conclusions and further work

6.1 Project goals and results

The project goals of quality identification, and mapping three distinct race events: Overtakes, Battles, and Time gain and loss, with >80% hit rate and <10% false positive were met.

The resulting race event map and the software infrastructure implemented in this project, provides a strong base for a racing production's highlights engine for increasing viewers engagement.

6.2 Improvement suggestions

As this project goals were met, it was still limited by the allocated research and development time. The most impactful improvement suggestion for increasing the automation and quality of the race events' classification would be adding a learnable framework to the process, using larger structured dataset of tagged racing events, to gradually learn the algorithms hyperparameters. Utilizing this ML and/or DL technics can increase the accuracy and scalability of the algorithms, making it more robust.

6.3 Looking forwards: possibilities for future work

This project is just the first steppingstone for a greater vision of a motorsport data revolution described 1.There is much more R&D efforts that can be allocated for this goal.

From research to production: the project code was written with decent practices in a modular and dynamic approach, but there is still much further testing and validation to be made for enabling the production of these insights and highlights engine.

Expanding the race coverage with more types of racing events classification: researching racing events like accidents, off road, oversteer, understeer, technical problems, top speed, fastest lap or corner, etc. can enrich the effect of the product and increase its value.

7 Project Documentation

Hence the project was conducted in a privately held startup company, the source code could not be entirely shared. Therefore, to provide I will share the detailed README file in a separate GitHub public repository, as well as an overview and description of the user guide, software, environment setup and project files in this section. Let's start with the software description:

7.1 Software Description: Classes, Relations, and Abstraction

To understand the project's software architecture, we will look at overview of the UML class diagrams, showing relations, abstraction, and modularity.

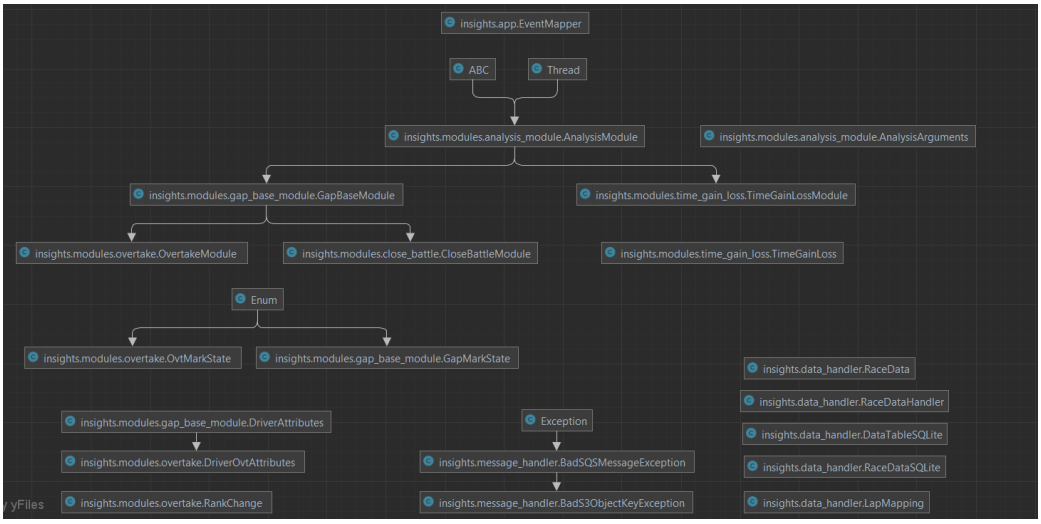

Figure 14: Project UML Class Diagram

7.2 Project Files

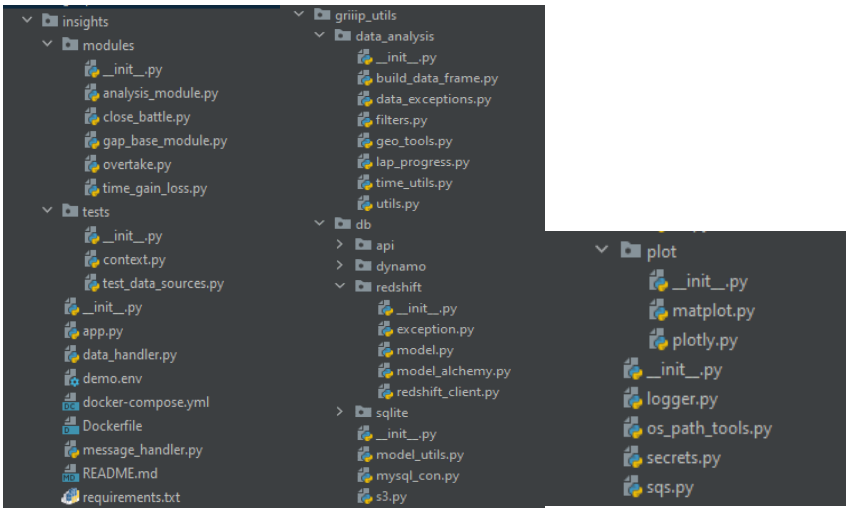

Figure 15: Project files

The insights repository holds the main application that executes the different event detection algorithms described above, while the utils repository provides generalization of common functionality.

8 References

Books:

[1] Jorge Segers - Analysis Techniques for Racecar Data Acquisition-SAE International, 2014

[2] Matt Harrison - Effective Pandas Patterns for Data Manipulation, 2021

[3] Cole Nussbaumer Knaflic - Storytelling with Data A Data Visualization Guide for Business Professionals-Wiley, 2015

Papers:

[4] Paulo S. R. Diniz - Adaptive Filtering: Algorithms and Practical Implementation, 2nd Edition, Jan 2008

Appendix A. **Savitzky–Golay filter**

Where C_i are the convolution coefficients, the elements of the matrix

$$C = (J^T J)^{-1} J^T$$

Where J is the Vandermonde matrix, where the i-th row has values $1, z_i, z_i^2, \ldots$

$z = \frac{x - \bar{x}}{h}$, for example $m = 5 \rightarrow z = \frac{1-m}{2}, \ldots, 0, \ldots, \frac{m-1}{2} = -2, -1, 0, 1, 2$

$$\mathbf{J} = \begin{pmatrix} 1 & -2 & 4 & -8 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{pmatrix}$$

For example, calculation filtered value 2nd degree polynomial with 7 coefficients points:

$$Y_j = \frac{1}{35}(-3y_{j-2} + 12y_{j-1} + 17y_j + 12y_{j+1} - 3y_{j+2}),$$

Appendix B. **Cloud Environment Simulations**

In accordance with Griiip backend architecture, the simulations and tests in the project were preformed using AWS infrastructure.
After the algorithms have been written and tested locally on a smaller dataset, as we would like to test them on a larger scale in a dynamic and configurable way, with the help of the backend team we have planned and created a simulation environment for this purpose. The simulation environment architecture includes:

Figure 16: Simulation cloud environment

At the end of a race session, or upon requested, a dump file of all the race data is sent to S3 in sqlite format, triggering an SNS (Simple Notification Service) that transfer those notification to a SQS (Simple Queue Service), the algorithms' testing code is ready and running on a EC2 compute machine and configurable with a Docker file.
The SQS trigger and event that executes the desired algorithms on the recently dumped data and will write the result into a dedicated database.

The trigger is set to be executed on the finished race session data for batch post process, but for the testing and simulation purpose it was done manually, enabling a more robust test environment that utilizes better networking and computation that can be achieved locally.

Appendix C. **Dynamic Ranking**

The Dynamic Ranking is a process that was developed in Griiip to dynamically rank the drivers during the race based on telemetry and official timing data. The data is serial with constant time interval and uses the latest transmitted location of each driver with add race logic to determine the gaps at any time. Its main usage is to provide a backend for TV production gap graphics.

Figure 17: Dynamic ranking in TV production graphics

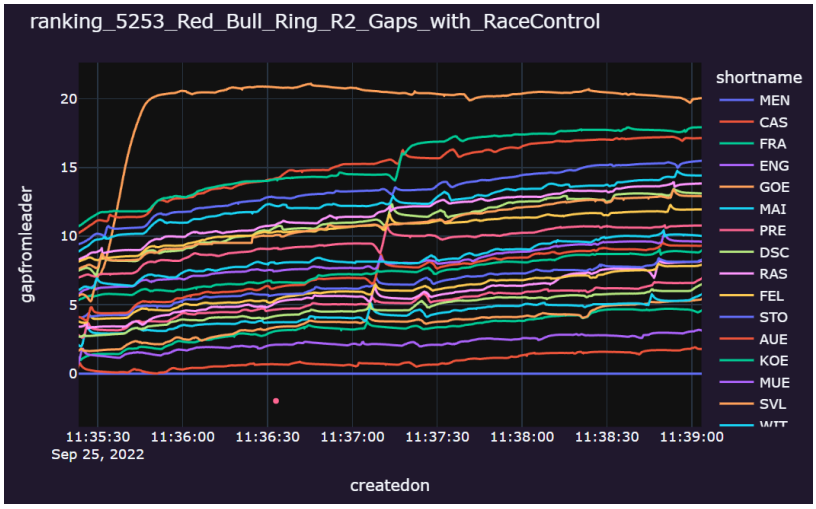

Figure 18: Dynamic ranking by timeframes

Appendix D. **Griiip's RedBox**

The 'RedBox' is a raspberry-pi based Linux module, with Internal 10-18Hz GNSS positioning system, high frequency 9DoF IMU (accelerometer, gyro, magnetometer), CAN Bus i/o input, 4G cellular modem, Bluetooth, and Wi-Fi connectivity.

The device is in charge of the communication of all racing vehicles' data in real-time to the AWS cloud platform, enabling further process and live visualization and analytics.

The main type of connection at recent races that have been researched in this project was cellular communication. This led to few challenges of instability, missing data, and more 2.1.3.

Figure 19: Griiip HW - RedBox

Appendix E. Code base and utilities description

Short description of the main classes and code utilities:

AnalysisModule(Thread, ABC)

The ‘AnalysisModule’ is an abstract python class serves as a base class for specific racing analysis modules that inherit from it.

The class extends the ‘Thread’ class.

It contains the following methods and attributes:

GapBaseMoudle (AnalysisModule)

The ‘GapBaseModule’ is designed to identify and analyze gaps between drivers during a racing session. This module utilizes the ranking data and shares common functionality of gap related analysis that is used with the battle and overtake modules.

In addition, there are some classes as the data handler that handle the processing and exceptions classes, as well as data classes as for driver attributes (metadata).

Code Utilities

Some general functionality that is used in different modules and parts of the project is share through a utilities repository.

Database Utilities

Those utilities include the DB objects models and pre-defined queries for several of Griiip’s databased. It is Mostly based on the SQLAlchemy library that provides a convenient way to interact with relational databases using the SQL. It offers a high-level Object-Relational Mapping (ORM) interface, as well as a low-level SQL expression language for constructing complex database queries. Also utilize the sqlite3, mysql, and boto3.

Visualization Utilities

This code provides utility functions for plotting data using the plotly and matplotlib libraries. Main plot types are scatter plots, line plots, histogram, in addition it includes some coloring functions, and generalized figures exports. Overall, these functions simplify the process of creating and exporting plots.

Data Analysis Utilities

This part of the code utility includes functions supporting datetime handling, filtering functions, and pandas' data frames operations.