Final Year Dissertation

Predictive race strategy modelling using Machine Learning-Integrated Simulation

Author

John-Louie Halog

Supervised by

Dr Fariba Sharifian

Abstract

With Formula 1 and motorsport increasingly relying on advanced technologies, machine learning (ML) has become an essential tool in race strategy optimisation. Traditional simulation models, such as Agent-based, account for probabilistic factors but struggle to capture the complex interactions influencing tyre performance and race strategy. Accurately predicting race strategies requires a model that can dynamically adapt to these factors, ensuring optimal and efficient decision-making in constantly evolving track conditions.

Therefore, this paper presents a machine learning-integrated simulation model to predict race strategies by focusing on tyre degradation using historical race data.

Using lap time data from the 2022-2023 seasons, an XGBoost model is trained to predict Ferrari tyre performance based on track conditions. These predictions are then incorporated into a Monte Carlo Simulation, which generates and optimises race strategies by accounting for probabilistic influences such as track conditions and tyre degradation trends.

Findings indicated that the simulation output was within ±4% error margin of the real-world performance, despite the limited data available for training and testing. However, further testing can be done to increase the accuracy to a more ideal 1%

This approach benefits race engineers, strategists, and teams in all forms of motorsport that require a race strategy by providing more accurate and adaptable predictions while utilising a more traditional ML approach.

Acknowledgement

I would like to express my gratitude to everyone who has supported and guided me throughout this project. Credit goes to my supervisor Dr Fariba Sharifian for the valuable guidance, constant support and insightful feedback throughout the year. Her expertise within the computer science domain has helped me navigate through any issues/difficulties I had.

I would like to show my appreciation to the university who have given me this opportunity to study computer science here, the last few years has been the best few years I've experienced academically and socially. I would also extend my appreciation to the computer science faculty as well for providing relevant and useful academic guidance/modules which played a role in the successful completion of the project.

Special mention to Studio Liverpool (defunct in 2012) and Sony who developed and published the F1 Championship Edition game, which first sparked my interest in Formula One and motorsports in the first place. As well as the race strategists who work hard every race weekend, they are inspirations behind this project.

A special thanks to my closest friends, Dara, Ethan, Sam, Harry, Nathan, as well as the other people I have met in and outside of the course who have supported and encouraged me over the last year or so, even if it is just taking slightest bit of interest in my project, it means a lot.

Lastly, I would like to thank my family for their unwavering support and understanding throughout this academic journey. Their support and love have been a persistent source of motivation over the last year.

Without everyone mentioned here, doing this project would have been an impossible task and I am sincerely grateful for your contributions.

Table of Figures

Table of Equations

List of Tables

Table of Contents

1. Introduction

1.1 Background Information

With the current world of Formula One becoming increasingly reliant on innovative technologies and software, the involvement of Machine Learning (ML) and Artificial Intelligence (AI) has become a go-to when it comes to race strategy as its ability to react quickly and effectively to changing conditions is invaluable in the high-stakes environment of Formula One racing and motorsports generally. Often, teams will have a race strategy for an upcoming race to produce the best possible outcome and there will be multiple variables that can come into play: weather, track condition, type of tyre compound, setup, position on track, among others. Said strategy is also altered in real time because of the aforementioned factors that are considered during critical decision-making moments. Though this does not just apply in Formula One, other racing series such as the WEC (World Endurance Championship) could best leverage models like this to decide the amount of fuel needed/tyres needed for the next stint as well as what driver (as they have driver swaps in that series).

Human error often comes into deciding when to come in, errors in strategy are not uncommon but one could think that it is down to the lack of planning. For example, Charles Leclerc's 2022 title chances were very slim coming into the Belgian Grand Prix, from the beginning of the 2022 up until Round 13 in Hungary, he had lost 108 Points (Morlidge, 2022). Proper preparation with predictive models by utilising and leveraging historical race data can mitigate these mistakes.

A real-world example of this was Lewis Hamilton's 2019 British Grand Prix win, where a perfectly timed pitstop during a safety car period led him to victory, this was through analysis of data related to tyre wear, fuel consumption, and the relative positions of competitors to determine the optimal timing for pit stops (Ambler, 2024a). Internet of Things (IoT) systems, AI and machine learning are all working together on track to help the team succeed and win championships, without that, the top level of motorsport would not be optimising and extracting the most performance of not just the drivers, the cars and the team but also the decisions made on track.

Despite its advantages, the application of AI in motorsport is not without challenges as the unpredictable nature of racing means that models need to be highly adaptable, and even the most advanced AI systems can have limitations in considering unpredictable events i.e. sudden changes in weather or accidents on track.

However, integration of AI within motorsport analytics does highlight the importance of AI-driven strategies, though not in real-time as this is done by the strategists, utilising the power of AI gives the baseline of most strategies.

1.2 Problems Statement

Although the use of ML in motorsport is continuously developing, the accuracy of predictive race strategy models is somewhat inadequate. The solution currently used in Formula One is the Monte-Carlo simulation, which does effectively cover one of the existing issues when conducting strategy prediction being probabilistic influences.

However, traditional race strategy models are limited in their ability to simulate the complex interactions between multiple uncertain factors, which can dramatically affect the race outcome. Moreover, the "perfect" tyre model cannot be accurately replicated and even F1 teams struggle to do so due to the lack of track time with Pirelli tyres on a race weekend. Alongside the uncertainty of tyre behaviour, there is also the added complexity as the teams need to know when to pit and change tyres at a point where the performance loss is too great; and this is due to the numerous variables that affect tyre wear:

- Tyre compound
- Track characteristics.
- Driving Style
- Weather Conditions
- Car Setup

Therefore, the aim is to enhance the accuracy of race strategy models by integrating traditional ML techniques with simulations, this alternative approach should improve predictive race strategy modelling. By incorporating real-time factors such as weather, tyre wear, unpredictable events (e.g., accidents or safety car deployments), the model can provide more accurate predictions on when to pit and optimise tyre usage.

It is vital that we consider the following key research questions:

1. How can machine learning be used to model tyre data?
2. How can machine learning be used to predict race strategies?
3. What are the best simulation methods for handling probabilistic influences?
4. How can the model consider with probabilistic influences?
5. How can we integrate machine learning in simulation.

1.3 Motivation

The motivation of this study stems from the constant change in technologies in and around the motorsport world. By integrating machine learning with race simulations, teams can gain deeper insights into race dynamics, enabling more informed decisions and potentially improving overall race outcomes; moreover, this project can aid in minimising risk and maximising performance by providing more accurate/faster race strategies.

Additionally, Integrating ML within simulations should allow for increased accuracy when race simulations are performed, which should aid predictive race strategy. Therefore, the goal of this project is to create a predictive race strategy model that combines the power of machine learning and simulation techniques to address these challenges.

This study is driven by the need for motorsport teams to stay ahead of the competition in a constantly evolving technological landscape. As race strategies become more complex, integrating machine learning into simulations offers a unique opportunity to optimise decision-making, making it faster and more accurate thus ensuring that teams are better equipped to handle the unpredictability of the race.

2. Background Research and Domain Analysis

2.1 Background Research

This section reviews the literature on race strategy in motorsports, with a primary focus on the predictive modelling approaches used in Formula One. Insights from other motorsports will also be explored to provide a broader understanding of relevant methodologies. By examining existing studies, this review aims to identify key strategies and techniques that can inform and guide the development of this project.

2.1.1 Predictive Machine Learning Methods

Related Sports

The application of predictive AI and ML has demonstrated significant value in motorsport, enhancing areas such as car development, performance optimisation, and driver analysis. Formula One, Formula Student (FSAE), and NASCAR are notable examples within the motorsport domain that can benefit from such techniques. For instance, Tonoli et al. (2024) employed various machine learning methods, namely linear regression, decision tree regression, boosted decision trees, and gaussian process regression (GPR), to devise optimal strategies for specific car configurations and provide driver feedback to maximise performance for the formula student team. The data was collected from a university's Formula Student team and analysed using a simulator to capture each driver's unique driving style. A standard dataset split of 80% training and 20% testing was used for the machine learning models. Though it is uncertain if this model was utilised in the following FSAE seasons, the results accurately imitated driver behaviour, demonstrating the model's potential. An additional use of cross validation against other teams may further prove its effectiveness. This example highlights the flexibility of machine learning techniques in motorsport strategy, offering valuable insights for consideration during the project's requirement analysis phase.

Instead of focusing on the driver, machine learning can be used in race strategy, Choo (2015) conducted research on real-time analytics that improves race car strategy. This experiment also involved the testing of the predictive capabilities of the system by using historic race data, though this is more in the context of tyre strategy as opposed to car set up. It was concluded that tracks with low tyre wear have higher ratios of 2:4 tyre change decisions compared to that of tracks that have a higher tyre wear. Choo had considered the external factors that may affect race strategy using system dynamics, this same system was implemented using track characteristics; therefore, it does produce accurate results with considerations. A notable criticism (which was also mentioned in the paper) was that tracks used as an example where primarily oval circuits as opposed to circuits with varying cornering/track length. Furthermore, the lack of validity cannot prove that this method can be repeatable, no comparison had been made to any real-life results which would be ideal to prove the experiments validation.

Race strategy is not limited to tyres, it can be in the form of energy management; *Formula-E race strategy development using artificial neural networks and Monte Carlo tree search* (Liu and Fotouhi, 2020) had experimented with how the battery should be managed throughout a Formula E race. Unlike other studies, synthetic data is produced based around the model of the driver and track using IPG/Carmaker and MATLAB/Simulink, where driver inputs from IPG is processed to MATLAB, where the torque outputs are produced, this generates lap times with the relevant data such as battery temperature and consumption. The integration of Monte Carlo Search Tree (MTCS) within the Artificial Neural Network (ANN) algorithm has allowed for a combined result of accurate simulated lap times from ANN and optimised simulated races using MCTS. Though the data is synthesised, validation is conducted by comparing the ANN results with the simulated lap times from the Carmaker-Simulink program. The biggest strength is the fact that probabilistic influences are included, i.e. when weather changes i.e. the temperature increases, the battery output is decreased to avoid not finishing (DNF). Further inclusion of driving styles also allows for battery management to adjust based of the driving style of the driver i.e. the more aggressive, battery output must be decreased at some point to avoid a DNF. Additional factors such as driving style can be applied when considering predicting race strategy; this allows for a more personalised strategy catered to the driver. Minor improvements could be made in terms of validation, should the lap times be compared to that of real-world results to ensure the pit wall viability of the project – meaning it can be used on trackside. Results also

Formula One
ML in Formula One has become the heart of research and development. However, race strategy has proven to be one of the most important parts of a team's performance; many solutions to increase accuracy have been viable such as the use of an embedded deep neural network (EDNN). Sitara, S., Fatima, W. and Johrendt, J. (2023) conducted an EDNN model to predict the 'winning' strategy. The model gathered data from the Ergast API, which, according to the Ergast developer API (n.d.), would be deprecated at the end of the 2024 season. By considering numerous historical factors, such as recent race results, maximum driver speed, and tyre performance to determine the optimal pit stop, both EDNN models were able to generate strategies for 1, 2, and 3-stop races. However, what the model did not consider was the wet weather as well as other probabilistic influences such as virtual safety cars or safety cars (VSC/SC). According to Sitara, S., Fatima, W. and Johrendt, J., "544 pitstops happened without the algorithm detecting them, which was caused by multiple factors, including car retirement or when the team principals' experiment with different strategies on their two available cars", which further implies that external factors were not considered.

Likewise, Zhao (2024) experimented with a different type of deep neural network – fully connected neural network (FCNN). Though not in the context of race strategy, this focused on lap time prediction from qualifying sessions, which is not properly representative of a drivers race pace. The experiment concluded with "relatively accurate predictions but some predictions have considerable deviations", Silverstone and Circuit Gilles Villeneuve are prime examples of massive outliers with the deviation being 12.9% and 16.9% respectively; this was further explained as the model did not account for realistic factors. Reducing massive deviations could be done by reducing the dataset down to the years where the regulations are the same. Context must always be considered as it must be noted that the regulations of the sport continually change, for example, an article by Dale (2017) stated that the difference between the 2016 and 2017 generation of cars were "2.45 seconds quicker over a pole lap across the season." Despite this, the use of the lap time parameter does provide insight to a driver's pace, i.e. a massive performance loss of 1-2+ seconds a lap (known as "falling off a cliff") can be noticeable in the data thus indicating a need for a change of tyres, though the use of the lap times from historic race data would be used in this case rather than qualifying times. Hence it is useful to consider a lap time parameter approach when developing the solution.

A further case of lap time and strategy use was when Loreto, G. T. (2023) forecasted F1 race outcomes. This time, the experiment was conducted with three different machine learning algorithms (SVM, Random Forest (RF) and ANN) like Tonoli but in a different context to determine which of the three would produce the most accurate results. Regarding use of datasets, this model utilised and combined two API's: Ergast F1 and FastF1 as they both complement each other for this model. It was concluded that SVM outperformed the other algorithms, not only in predicting pit stop timings but also in forecasting race scenarios. This comparison of machine learning models highlights the importance of evaluating different algorithms to identify the most suitable technique for specific racing strategy applications. Such a comparative analysis is critical in determining which method provides the most reliable insights for optimising race strategies in Formula 1.

It is important to recognise that the various approaches explored in the literature utilise various aspects of a Formula One race, such as lap time parameters, maximum speeds through corners, and historical race data. Additionally, applications of machine learning techniques in other motorsports and domains beyond race strategy offer significant insights into methodologies that could potentially be applied to Formula One race strategy. Techniques such as FCNN, RF, SVM and Gaussian Process Regression (GPR) have been utilised in these contexts. However, a common limitation in many of these studies is the lack of consideration for probabilistic influences, which is understandable given the unpredictable nature of such factors. Finally, most experiments trained their datasets using only a training set; ideally, a validation set could also be incorporated to identify and optimise the best-performing version of the machine learning model.

2.1.2 Race Strategy Simulation

Predicting race strategy involves the use of race simulations, which allow teams to gain an understanding of what to expect based on the parameters provided to the simulation model. There are four types of models that can be used for race strategy, as it relies on both data and data analysis: Monte Carlo simulations (MCS), Agent-Based Modelling (ABM), Discrete Event Simulation (DES), and System Dynamics Modelling. This section explores the various papers that have conducted experiments using different simulation models.

Related Sports

A Race Simulation for Strategy Decisions in Circuit Motorsports (Heilmeier, et.al, 2018) utilises the idea of DES use in motorsport (proposed by Bekker). Instead of dividing the track into a multitude of parameters, lap time calculations where calculated based off the factors affecting a driver over a lap. Separating the drivers into racing their own lap to conceive a base lap time for the model allowed for the experiment to produce the fuel loss, tyre wear loss, pitstop time loss data amongst numerous factors; this temporary race data would then be used in the competitor interaction model. Much like Bekker, the source of the data used in this model was unknown because of a "lack of publicly available information from the FIA". Furthermore, this experiment is outdated due to the tyre compound rules from 2018 are different now due to the removal of the rainbow tyres (*Hypersoft* all the way to *Superhard*, which in total was 7 compounds as seen in **Fig.1**) and replaced with the much more simpler C0-C5 tyre compound range (C0 is the softest, C5 is the hardest compound); which "requires less tyre management" i.e. longer stints on the newer compounds compared to their predecessors (Eagles, J., 2019). Despite this, the model did improve on Bekker's idea, including tyre degradation, simplified overtaking mechanics - including DRS and a simplified lap time simulation using the total (as seen in **Appendix A.1**).

Formula One

The experiment by Bekker, J., Lotz, W. (2009) was to plan Formula One race strategies via DES. By simulating a race by dividing a racetrack into nominal sectors and allowing the entities (drivers in this case) to interact with activities i.e. pitstops and overtakes etc., relevant data would be collected by the entities and the model calculated the simulated time the car spends in all sectors. It is not clear as to where the data was sourced from as the author had stated that the data was determined from "freely available data". Despite this, they did validate these results against real life grand prix, namely, Australia, Hungary and Italy. Interestingly, results of this experiment suggested that the simulated race strategies were accurate; deviations were not massive compared to the real-life positions of the cars. Again, like most models mentioned previously, probabilistic influences have not been considered and more importantly, the emphasis on tyre degradation is not present which is especially important to race strategy. Furthermore, this is outdated as Formula One cars nowadays have a drag reduction system (DRS) which helps with overtaking and lessens the stress on the tyre, resulting in longer stints (The F1 insider, 2020). While that is the argument made, the validation of this experiment still stands as minor tweaks to fit with the current regulations can be made.

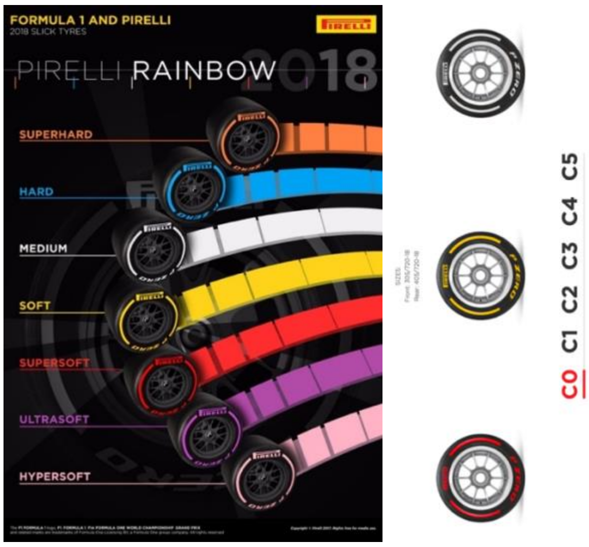


Figure 1: Comparison of F1 2018 compounds and current generation of tyre compounds (Left Image Source: Pirelli Motorsport on X (formerly Twitter) (2017).) (Right image Source: Pirelli (2019)).

Application of Monte Carlo Methods to Consider Probabilistic Effects in a Race Simulation for Circuit Motorsport by Heilmeier A, et. Al (2020a) considered the probabilistic influences unlike the other experiments. The model simulated pitstops (including in & out lap), lap times, overtaking while including probabilistic effects and improves on the methods of Bekker by generating lap times by sector rather than through numerous points on a circuit. However, it only focused on car damage resulting varying lap times and the deployment of VSC and SC – which is particularly important as it does impact a race. The lack of focus on another crucial factor which can affect strategy (weather/track condition), could be a result of the complex nature of weather occurrence and track conditions. Although it also could be argued that the variation of lap times could also be done to simulate the effect of weather/track conditions.

It is vital to know that race simulation is important to race strategy as it allows us to understand the number of stops that could be made while factoring in overtaking manoeuvres, pitstop time etc. Moreover, the use of race simulation allows use to optimise race strategies and allow teams to get the best chance of winning and getting the best race result while incorporating factors that affect a race. Although, the scarce research on general race strategy with the incorporation of probabilistic effects results in the research being limited within this domain and leaves a gap for possible further exploration.

2.1.3 Machine Learning Integrated Simulation

Related Sports and General Use

A combination of the ML and Simulation models is thought to increase the accuracy of the model. The healthcare sector for example utilised this method of predictive analytics, more specifically DES; Atalan, A., Sahin, H., Atalan, Y.A (2022) estimated the number of resources employed and analysed the cost of the resources depending on the cost coefficient in the emergency department. Their method had defined the variables for the patients, waiting time, physicians etc. and divided them into output and input variables. Output data (patients treated and waiting time) was put in the DES model, which was filtered and pre-processed before being used for ML prediction methods (RF, Gradient Boosting (GB) and AdaBoost (AB).) They concluded that this method obtained “accurate and sharp prediction data in advance of uncertain situations in a fast and inexpensive way”. This conclusion suggests that the utilisation of machine learning integration is an inexpensive way of producing accurate and precise results. Application of this towards race strategy could prove to be extremely useful, even if DES specifically is not used.

Boettinger and Klotz (2023) integrated reinforcement learning (RL) into a simulation framework for GT racing on the Nordschleife, employing a Deep Q-Network (DQN) to generate lap times. These times were based on baseline sector times and a Gaussian distribution. The races were simulated through an agent-based approach, allowing the model to respond to a range of factors like overtakes, traffic, and pit stops in real-time. This model, inspired by Heilmeier et al. (2018), provides a notable advantage in its ability to simulate a wide range of race dynamics, making it more comprehensive than other models focused mainly on lap times or pit stop decisions. The inclusion of factors such as traffic and overtakes enhances the realism of the simulations, though the model may benefit from incorporating additional probabilistic influences such as weather conditions or mechanical failures, which can also significantly impact race strategy.

Formula One
Heilmeier, A et.al (2020b) incorporated deep learning ANN to develop a virtual strategy engineer, results from the deep learning system were then processed into a MCS. Unlike Boettinger, this alternative approach employed a feedback model using an artificial neural network to make decisions as opposed to using reinforcement learning. Experiment resulted in accurate positions compared to real life as well as some improved positions which this model intended to do, with the inclusion of probabilistic influences such as Full Course Yellow flags (FCY) and Safety Cars (SC) as seen in **Appendix A.2**. A notable strength in this model is Heilmeier pointed out that “teams cannot choose freely from the tyre compounds available in a particular season” and the fact that the cars evolved within the time frame of which the dataset was chosen (2014-2019) means that acknowledgment of the rules have been kept in mind during development. Further strengths included the inclusion of probabilistic influences during the simulation, which was also used in Heilmeier, A et.al (2020a).

Greasley, A. et. al (2022), investigated a similar problem to Boettinger and Heilmeier but instead utilised an agent-based decision support system (DSS) alongside a digital twin. Though the project was not developed and the paper in question had provided the framework only, it did consider the many influential factors that could affect a race strategy. Moreover, the ideas presented here that explore the use of ML, optimisation, digital twins and reinforcement learning (RL), do offer a significant insight on the capability of a prediction program using ML integration. Nonetheless, without results, it is unknown whether this system would output highly accurate predicted race strategies.

As a result of this research, the lack of machine learning-integrated simulation experimentation within this domain highlights a clear gap, presenting an opportunity for further exploration. Specifically, there is a need to incorporate alternative machine learning algorithms or simulation methods. Additionally, most existing applications are confined to industry-specific cases, with little similar work done beyond these areas. While comparable approaches have been applied in other sports and sectors, the use of these techniques in Formula One and related motorsports remains largely unexplored, offering a distinct opportunity for further investigation.

2.1.4 Summary

Papers within this field have taken differing approaches to the problem during different periods. More recent papers that conducted predictive analytics and used lap time data as a basis often fail to consider the various regulations that affect lap times and strategy; however, this domain has been explored more thoroughly compared to simulated strategies. For example, massive regulation changes between 2021 and 2022 meant that cars became heavier and slower compared to their predecessors (Chris Medland, 2022). Naturally, limiting such data may result in unexpected or inaccurate outcomes though can be solved by selecting algorithms that are compatible with limited datasets. Interestingly, the implementation of the two-compound rule has not been mentioned in the papers reviewed particularly in years prior to 2022.

Further points include the failure to include probabilistic influences that affect race results in most other studies suggests that further improvement can be done here, via an alternative strategy for example. Although this is difficult to integrate, it remains an important consideration when designing the methodology and framework for such an application. Another oversight is the lack of inclusion of the qualifying rule in studies conducted between 2014 and 2022, which required those qualifying in the top 10 to start the race on the same tyre used in the second qualifying session, adding complexity to race strategy. However, its inclusion is no longer necessary, as this regulation was removed at the beginning of the 2022 season.

In addition to the detailed discussion denoting key gaps and strengths in research, a comparative analysis of the various models and techniques used in the reviewed papers is provided in **Table 1** (**Appendix A.3**), highlighting the key differences in their approaches, strengths, and limitations.

Key research gaps identified:

- Lack of probabilistic influences in most race strategy models.
- Lack of predictive race strategy models
- Very few integrate ability factors such as driving style.
- Limited ML integration in simulation models in the motorsport domain, requiring more exploration.
- Many studies fail to account for evolving FIA regulations, affecting strategy models.
- Machine learning models need more real-world validation to confirm their accuracy.

2.2 Domain Analysis

2.2.1 Identifying Key Concepts

When developing a predictive race strategy model, it is important to note that we must understand that the domain being analysed involves:

a) Data collection i.e. what data is going to be used.
b) Race simulation to simulate a race based off the predictive parameters.
c) Predicting tyre performance.
d) Risk Management i.e. predicting when the tyres will go "off the cliff" and need to change tyres.
e) Selection of tyre choice for the best possible race result.
f) Including probabilistic influences that may affect the strategy i.e. Safety Cars, Rain, Red Flags etc.

This is vital as it allows us to solve the problem that was imposed, stating the relevance of the concepts within the domain. Additionally, we must consider the key concepts used i.e. the structure of a programme, the programming language used, possible range of techniques and simulation models.

An existing method applied in industry that is widely used to simulate race strategies provides us with vital key concepts that we can re-use when considering the method, design and development of the programme. This process mentioned by former race strategist Gemma Hatton (n.d.) provided a general method that all formula one teams use and is made up of three parts:

1. Tyre Model Data
2. Free/Clean air Optimisation
3. Monte Carlo Simulation

This way of predicting and simulating a race strategy is optimal as it has real-life applications; furthermore, an adaptation of this structure can form a fundamental basis for the project. Though free/clean air optimisation is not fully viable with the datasets that exist online and thus will have to rely on an assumed value, though a thorough implementation of 1 and 3 is still possible.

2.2.2 Domain-Specific Challenges

Before exploring any solutions, domain-specific challenges must be emphasised as they will significantly influence the direction and approach of the project.

As mentioned consistently throughout the paper, probabilistic influences pose a massive problem when predicting a race strategy. Not only this, other factors on a racing car i.e. fuel, dirty air, car condition can all affect tyre wear and degradation.

In this domain, the magic formula (Pacejka tyre model) would be used to determine the behaviour of the tyre based on parameters such as lateral load, slip angle etc. Unfortunately, recreating a realistic tyre model with the magic formula is complex and exceedingly difficult to execute as it requires numerous parameters – this alongside recreating the conditions that tyre is in at a specific point in a race would be far too complicated.

Careful selection of the ML model must be made as the data such as lap time and weather are only publicly available. The lack of data also adds another challenge where the accuracy of the predictions produce may vary from perfectly accurate to not accurate. Though the introduction of bias could also be an issue as the ML model could heavily rely on the dataset and introduced unwanted data.

2.2.3 Handling Data

All research papers either used a database API (Ergast or FastF1), or a Kaggle dataset, though the use of synthesised data can also be considered. Utilising a pre-made dataset would reduce the time needed to extract the data as it is already provided. In some other cases, the publicly available dataset can also be ready for ML training and testing so processing the data would be minimal. It must be noted that many of the papers had a train/test split of at least 70-80%, this is a common practice in numerous experiments which ensures representative evaluation of model performance on unseen data.

Database APIs are rather advantageous, this can enable real time data and predictive analytics which is crucial in a domain such as motorsport. Furthermore, data can easily be selected based off the user needs, extracted and automatically cleaned which is convenient. However, it does have its own downsides, one of which being the quality of the data – it is only as good as provided, it relies on an internet connection which can be an issue if the artefact is a live service, though this can be resolved by caching the data on the local machine or within the program itself.

Numerous issues would arise if the use of synthesised data was conducted in this project as it could introduce overfitting to the model. Although the introduction of it in the dataset can be a viable solution should there be missing data or lack of data for certain tyre compounds for example.

Tyre Model Data

Tyre data is the focus of the project as it is the centre piece of the system. It will allow us to predict tyre performance and predicting the lap until a change of tyres. However, the current lack of F1 specific tyre data availability to the public and the magic formula's complex nature, a simplified version does suffice – this is by utilising historic lap time data which can be used to model the tyre (as shown in numerous papers mentioned in the literature review).

This is broken down into three basic principles:

- Base Pace – how fast a tyre is (in seconds)
- Tyre Degradation – the decrease in tyre performance per lap (directly affects pace)
- Tyre Life – how many laps can the tyre last (directly affects performance and pace)

It can be determined that the base pace of any tyre can be extracted from lap times as this is as close one can get without recreating an entire tyre model. Tyre degradation and tyre life can also be determined from the lap times where there is a noticeable difference between the current lap and the previous lap time.

2.2.4 Existing Solutions – Machine Learning

Multiple ML models can be considered to satisfy the predictive concepts involved within this domain, namely:

- Predicting tyre performance based off lap times.
- Predicting the lap to change tyres.
- Forecasting race outcomes

Deep Learning Models

A combination of multiple models or running the same model for both concepts do have potential to increase accuracy. The EDNN model by Sitara, S., Fatima, W. and Johrendt, J. (2023) has potential for usage due its complex nature, Two EDNN models were created to target two different metrics:

- The rank position of the driver – how high they finish
- The optimum lap to make a pitstop to finish as high as possible in the race. – ideal pit stop window

An EDNN model is a type of deep neural network, it can capture complex relationships such as delta times or finishing positions (as used in Sitara's paper). Leveraging complex models through numerous layers allows for more precise race strategy models as proven by Sitara, generating 1-2-3 stop strategies. It is advantageous to use this approach as EDNN is low latency when it comes to real-time analytics thus proving them useful in situations such as motorsport strategy which means they are less reliant on cloud-based system, thus making the system adaptable. As stated earlier, "544 pitstops happened without the algorithm detecting them", which is an issue, this was the direct result of unpredictable factors such as rain which can be rectified if combined with a stochastic model/simulation. Furthermore, the dependence on high-quality data is quite high thus the data being inputted into the EDNN must be pre-processed adequately to ensure accurate predictions. Not only this, EDNN is heavily dependent on the hardware, which means it can be limited if the hardware is lacking computational power.

However, despite its potential, the objective is to predict the race strategy, like Sitara, though with the limitations in place such as data and possibly hardware, this may not be applicable to producing an accurate solution. Though a hybrid approach may address the issues that were present in Sitara's solution.

Traditional Machine Learning Models
A comparison of ML techniques was conducted in paper by Loreto, G. T. (2023) with support vector machines performing the best of the three selected techniques (SVM, ANN and RF). They forecasted F1 race outcomes based off predicting pitstops. Despite this being a classification problem, this does not change the advantages and disadvantages of the algorithms used as aspects such as the flexibility of a model can be similar or the same when applied to a regression model.

SVM or Support Vector Regression (as this is a regression problem) does support smaller datasets which is very valuable for this problem. Its predictive capabilities are very good despite the class imbalances as mentioned in the paper; this must be considered as this may be an issue during development. Many advantages of this are the fact that numerical predictions work very well with this model, additionally, the lack of data fits with this algorithm as predictions can be conducted on small datasets; the avoidance of overfitting also works here as the regularisation parameter (C) controls the complexity of the model, ensuring that generalisation done well on unseen data rather than simply memorising the data. However, despite this, with the given data, there were generalisation issues which was a direct result of the class imbalance, which was "good pitstop" and "has pitstop" being the majority classes. This impacted the overall result, producing the higher number of false positives (pitstop prediction when there should not be one). Class imbalance can also apply to regression problems should there be more data points in one category compared to another.

Much like SVM, RF is also suited to this as the performance indicated precise results, however its lower performance compared to SVM may be the result of the interpretability of feature importance i.e. overemphasising majority class features. Furthermore, the regularisation on RF is not as strong compared to SVM which results in a poorer generalisation due to the class imbalance. Avoidance of skewed results may have to reduce the bias towards the majority class through class weighting or using oversampling/under sampling. Though this experiment that does address the fact that possible class imbalances such as tyre compound may be present and must be considered if such data is dealt with within the solution.

Reinforcement Learning

Further experimentation conducted by Boettinger and Klotz, 2023 integrated reinforcement learning (RL) into a simulation framework for GT racing on the Nordschleife, employing a Deep Q-Network (DQN). Reinforcement Learning benefits complex problems such as this one as the agents learn from the data, especially since the deltas and strategies are dynamic; furthermore, the real-time application for this technique is useful for motorsport as real-time data is very important for strategic decisions when it comes to selecting the tyre and the pit window. Additionally, the adaptive correction mechanism RL has allowed the agents to correct any errors made, which minimises the mistakes outputted in the results. Much like Boettinger, RL is very flexible and can combine with other techniques such as DQN, this leverages the strengths of both methods which can strengthen the overall result.

However, like neural networks, a large amount of data is needed to be able to output precise and accurate results. The reason it did work for Boettinger is that the Nordschleife is a timed event, 6 hours of constant laps allows for a large amount of data to be recorded unlike Formula 1 where it is limited to a fixed race distance i.e. 50 laps. It can also be argued that the computational requirements are demanding, which may be an issue for smaller organisations/teams that do not have the resources to power the program. Finally, debugging the algorithm can be an issue as understanding why the agent does certain actions may not be as straight forward, which can create difficulties in ensuring accountability and transparency (Birchwood, 2024).

2.2.5 Existing Solutions – Simulation

Modelling processes allow for a better insight, especially when it comes to strategy development. Especially if applied in areas where the next step in a situation is important i.e. Formula One for example, where the timing of the strategy and the next set of tyres the driver should be on can make or break a race. It is important to consider all types of simulation before choosing one that would provide the most suitable results.

Discrete Event Simulation

Bekker's DES model for example manipulated and recorded data over a track broken down into a multitude of sectors; though this is not efficient at all and would require extensive work, this type of discrete model would still provide accurate results due to the nature of each corner/sector on a racetrack. Another advantage of this approach is purely the fact that the DES system can capture each unique characteristic at each point in the track resulting in data such as speed, cornering, tyre degradation easier to understand on how it impacts lap/tyre performance.

There are some benefits to using DES, namely the reduced uncertainty when simulating, though does not consider random events. Doing so allows for a more concrete decision to be made on the track, suggesting that the system is deterministic. Bekker does try to do this with DES, though the implementation of the sector times to capture the complexity of the problem. Furthermore, as the time is the target, DES relies on sequential events, meaning the model is only updated at a particular instance in time i.e. an overtake or changing tyres as shown in Bekkers experiment. Outcomes using this system allows for a higher confidence result compared to stochastic models such as MCS.

However, DES is quite complex to develop, it must require a deep understanding of the model itself before development. The biggest drawback is the lack of scalability and flexibility, which makes this an ineffective option should the dataset grow increasingly larger with that amount of data; this in turn increases the complexity of the model which can make it difficult to apply this system in real time strategy calculations.

Monte Carlo Simulation
Monte Carlo Simulation (Tom McCullough, 2024) is currently used in Formula One, models like this are used to predict the outcomes of uncertain events, i.e. in this case, when to bring the car in for a pitstop. Numerous positive implications are included when using this model, whereby a wide range of values can be factored in by the model for various inputs, it is then passed into a mathematical model where a result would be produced.

A massive advantage when utilising such a model is the fact that the inclusion of probabilistic influences through variation methods allows for simulating incidents/VSC/SC periods (Heilmeier, A et.al, 2020a). A Virtual Strategy Engineer (Heilmeier, A et.al, 2020b) used Neural Networks combined with Monte Carlo Simulation to create a more powerful simulation, though this is catered towards the "real time" strategy decisions, a combined ML-model and simulation to produce such strategies is a viable solution to the problem, though the large amount of data used to train the neural network may be a possible issue that must be dealt with, given that not all the data used is recent. Relevant methods used in the VSE include utilising a gaussian distribution for varying lap times, which assumes that most the generated lap times in the simulation would fall within the mean average – this is valuable to use in the solution of the project as it allows one to see the tyre progression over course of the stint while emulating it in a realistic way through standard deviations from the dataset.

Unlike DES, developing MCS can be easier as it can easily account for probabilistic influences using stochastic modelling, which is something DES cannot provide and handles random events better which is why this is used in industry. In addition, the scalability for this model is much better compared to DES as it allows for both small and large datasets to be utilised with this simulation model.

However, its greatest strength also so happens to be its weakness as the number of inputs must be large enough to produce quality results; not only this but the output of the model is very heavily dependent on the values inputted into the system. Any values that are not included in the dataset must be programmed into the simulation file such as averaging the mean of pit stops. Finally, since it is a stochastic model, it heavily relies of probability assumptions meaning the confidence level may be lower than DES.

Agent-Based Simulation
An ML-integrated solution utilising agent-based simulation (Greasley, A. et.al, 2022) included influential factors unlike DES through a DSS. The DSS was comprised of a supervised machine learning algorithm, mathematical optimiser, reinforcement learning and agent-based simulation. As mentioned in the literature review, the paper was all framework and theoretical, it is not known whether the results outputted would be accurate or not. However, this does strengthen the argument that hybrid models such as ML integrated simulations will be needed to produce accurate race strategy predictions.

Agent-based models are cost effective and time saving, it is stated that it is "an effective tool used for reproducing and analysing a broad variety of complex problems…" (Bazghandi, 2012). The same author for that paper had also explained that the use of agent-based models allows for scalability and flexibility which is not offered in DES; this means that the more data that is being simulated, the performance will remain consistent which is needed if teams need to input more information which could output a more accurate result.

Contrary to this, modelling the competition between agents (drivers) is very difficult and would require heavy debugging when balancing out the interactions between each other (such as pace and what their pit strategy would be). Much like many of the models, mentioned before, complex race conditions (probabilistic influences) are very difficult to model and most likely need to be stochastic. Furthermore, ABM has limited predictive power hence the influence of the simulation comes from the ML and RL, which leads to the conclusion that they offer a more adaptive and precise approach to race strategy optimisation.

2.2.6 Summary

The domain analysis focuses on creating a predictive race strategy model in motorsports, centred on key elements like tyre performance, data collection, race simulation, and risk management. The main challenges include accounting for probabilistic influences such as safety cars, weather, and tyre degradation, all which impact strategy decisions. Current methods in the industry, including tyre model data, free air optimisation, and MCS, provide a foundation for developing the model. However, accurately modelling tyre behaviour and managing the limited availability of public data present significant issues that must be acknowledged.

The research also explores the potential of combining machine learning techniques with simulation models to enhance the accuracy of predictions. Existing studies have shown that such integrations, particularly through Monte Carlo and agent-based simulations, could offer valuable insights. Nevertheless, the lack of experimentation in this domain, especially with machine learning integrated into simulations, creates an opportunity for further research and development.

From the analysis conducted, SVR, RF and XGBoost emerge as candidates and are viable options to use as they are capable of handling smaller datasets. Thorough model comparison testing will be needed to determine the best model for this problem. Neural networks may not be a viable choice as the data set used may be limited and the solution does factor in compatibility with less powerful systems.

3. Requirement Analysis and Methodology

3.1 Requirement Analysis

When performing a requirements analysis, it is vital to evaluate the options discussed against the problem statement in **Section 1.2** are met alongside the aims and objectives outlined here, which will be evaluated against in **Section 7.1**.

Aim:

- Enhance the accuracy of predictive race strategy modelling by integrating machine learning techniques, providing an alternate approach to the problem.

Objectives:

- Develop a Machine Learning-Based Tyre Model
- Optimise Race Strategy Decisions
- Integrate ML Predictions into Simulations
- Consider complex interactions between tyres and racing conditions

This section explores potential machine learning techniques and the selection of an appropriate simulation model. Further discussion includes functional requirements i.e. the software/techniques used which allows the model to serve its intended purpose; and non-functional requirements such as assessing the performance and efficiency of the model.

3.1.1 Technical Requirements

The programming language must support robust machine learning libraries, seamless integration with simulation frameworks, and efficient data handling.

- Extensive ML libraries (e.g., TensorFlow, Scikit-learn).
- Compatibility with simulation tools.
- Large availability of resources.
- Readable syntax, enabling efficient prototyping and debugging.

Since machine learning and simulation is the core functionality of the project, a powerful PC/laptop would be required to run the numerous simulations. Though there is also a consideration to optimise the system, so it can be run on lower end systems if possible.

3.1.2 Functional Requirements

Data Acquisition

Data collected from relevant sources must meet the specific requirements for improving the predictive race strategy model. The scope of the data must be carefully considered when selecting the primary source(s). Some databases provide highly relevant data, such as lap times, sector times, and telemetry information, which are critical for analysis. Conversely, other databases may include irrelevant data that do not align with the project's objectives. A thorough evaluation of the available sources will ensure that reliable data is utilised.

Mandatory features for the machine learning model:

- Lap Number
- Lap times
- Tyre Compounds
- Pit Duration times.

Numerous sources such as Ergast F1, FastF1, OpenF1, Kaggle and other open-source databases will provide access to the necessary information, including lap times, sector times, and telemetry data. These sources will serve as the foundation for the analysis, compensating for the lack of detailed tyre data. It is, however, essential to recognise that reliance on publicly available data may impose certain limitations, particularly with the depth of tyre performance evaluation.

Machine Learning
Based on the research conducted, the problem can be categorised as a regression problem. This involves predicting quantitative data, such as lap times, based on input variables including lap number, and tyre compound. Contextually, the relationship between lap time and lap number is non-linear and becomes more distinct as tyre performance drops off, causing lap times to increase significantly. Thus, the need for an algorithm to predict and imitate the complex trends such as SVM, random forest, decision trees or neural networks could be considered.

The ML algorithm must meet the criteria:

- The model must be capable of handling complex, non-linear relationships between the variables, in this case: Lap Number, Lap time, Tyre Compound.
- Must be able to accurately predict the lap times using a small dataset – to allow future tyre characteristic predictions.
- It must integrate with the existing simulation framework, providing accurate predictions of race strategy (e.g., when to pit and which tyres to use).

Integration of the ML model into the simulation can be done in numerous different ways i.e., a feedback loop, a batch model or real-time integration.

Simulation Requirements
Simulation selection needs to be able to run race simulations because of various inputs. The chosen model must:

1. Generate realistic lap time fluctuations around the mean average on the selected compound.
2. Allow for adjustments to reflect different race conditions (e.g., weather, tyre performance).

To ensure the simulation accurately reflects real-world racing scenarios, it is essential to introduce variation in lap times and incorporate unpredictable events, such as safety cars. This requires identifying a suitable simulation model that allows for variability. For this purpose, simulations that can model lap time variability and unexpected race events must be considered to model these variations effectively. Further refinement will determine how key parameters (e.g., mean and variability of lap times) are set to align with real-world race dynamics.

3.1.3 Non-functional Requirements
The following non-functional requirements must be met to ensure that the project is high quality:

- Data quality – Data collected and pre-processed must be consistent and cleaned up, additional effort must be done to ensure the data is complete.
- Scalability – Performance of the program must not degrade should more data be loaded into the program. Though there is a limitation here as the selected machine learning algorithm may not work with larger datasets.
- Reliability – The inclusion of fail safes i.e. dealing with error or empty data that is not cleaned when loaded into the program.

- Accuracy – Results from the program will be compared to real life results as well as in-depth analysis to assess the performance of the program.
- Ethics - Data utilised is extracted from an open-source database API, therefore ethical compliance with relevant computing laws such GDPR and Copyright is not an issue. Additionally, ethical concerns regarding the use of AI and ML include the assurance that the output of results are not biased in any way and the program is not being used for any malicious intent.

To conclude, the outlined non-functional requirements aim to create a project that is technically sound and socially responsible for the actions undertaken within this project. By ensuring data quality, system reliability, scalability, accuracy, and ethical compliance, the project can contribute valuable insights without compromising integrity or user trust. These considerations will be continuously assessed and refined throughout the development process to ensure the project remains of the highest standard.

3.2 Methodology

Methodology in other studies had utilised a large amount of data from different eras of the sport, thus this approach allows for a more realistic approach to predictive race strategy by using data within a relevant time frame. The purpose of this new approach is to utilise lap time information as the basis for a "tyre model" and therefore used to predict a strategy. The artefact methodology in question must answer the following research questions mentioned in **Section 1.2.**

The ML model will be used to predict the lap time difference lap by lap (delta) which will act as the tyre performance across a stint based off numerous factors i.e. track conditions.

The simulation model will generate numerous race simulations and strategies, selecting the best ones according to differing situations as well as the inclusion of probabilistic influences and realistic race dynamics. Selecting the best fit simulation as well as the machine learning models to meet the criteria are vital to the success of this project.

Implementation of both machine learning and simulation will be carried out in Python, using libraries such as NumPy and Pandas for data manipulation, Scikit-learn for pipeline and machine learning predictions.

3.2.1 Data Collection

Collected the lap time data will be taken from a combination of practice sessions and historical race results from OpenF1 and FastF1. The dataset will ensure that the data is within the same time period as the current regulations as using data pre-2022 may not be reliable due to the difference in tyre regulations. Therefore, to ensure consistency and reliability, the dataset used for the project are races from 2022-2023, though the free practice session data was limited to 2022-2024. However, additional data may be introduced to aid model robustness and generalisation, though thorough evaluation must be conducted to ensure that the additional data aligns with the methodology.

Given that the tyre data is limited, acknowledging the data used will heavily rely on lap times – this will act as a foundation for our "tyre model." This addresses the issue of how the tyre is modelled in the simulation by performing manipulation techniques on the lap times, we can emulate tyre characteristics through various assumptions.

3.2.2 Feature Selection

The main features extracted from the API database:

- Team
- LapTime_Seconds
- Session Lap time:
    - Free Practice Sessions 1,2,3
    - Race
- Tyre Compound
- Track Conditions (Humidity, Air Temperature, Track Temperature and Rainfall)
- FreshTyre (Whether the tyre used was fresh or not)
- StintNumber (The sequence of tyre stints)
- StintLap (The number of laps done on that tyre during a tyre stint)
- Delta (or $\Delta Laptime_{i-(i-1)}$, see **Equation.1**)

$$\Delta Laptime_{i-(i-1)} = Laptime_i - Laptime_{i-1}$$

- $Laptime_i$ – Current Lap time

- $Laptime_{i-1}$ – Previous Lap time

Equation 1: Delta Calculation

It is important to pay special attention to the selected races for this data as all circuits over during the time span will vary in characteristics. This ensures that the model is not overly catered to specific track types, such as downforce-heavy or top-speed-oriented circuits. When extracting lap times, particularly for Race Sessions, lap-by-lap time differences ($Delta$) will be calculated and will be known as 'Delta'.

The 'circuit' feature is excluded as the ML model should only learn from the conditions on the track and not by name.

See **Appendix B.1** for an example of the raw dataset.

3.2.3 Dataset Analysis

An analysis of one of the datasets (**see Fig.2**) has been conducted to extract key insights and identify patterns. Understanding this relationship provides a good foundation for the methodology.

Certain **outliers** can be identified in the data, due to:

- Safety Car Laps
- Mistakes during the lap
- Traffic
- In lap and Out lap

These anomalies provide insight into external factors that influence lap times and must be considered when interpreting tyre degradation trends. Thus, a simple equation for tyre degradation can be calculated for each compound. Since tyre wear typically follows a non-linear pattern, lap times progressively increase over a stint. Performance eventually deteriorates rapidly should the driver carry on with the same stint —often referred to as "falling off a cliff."

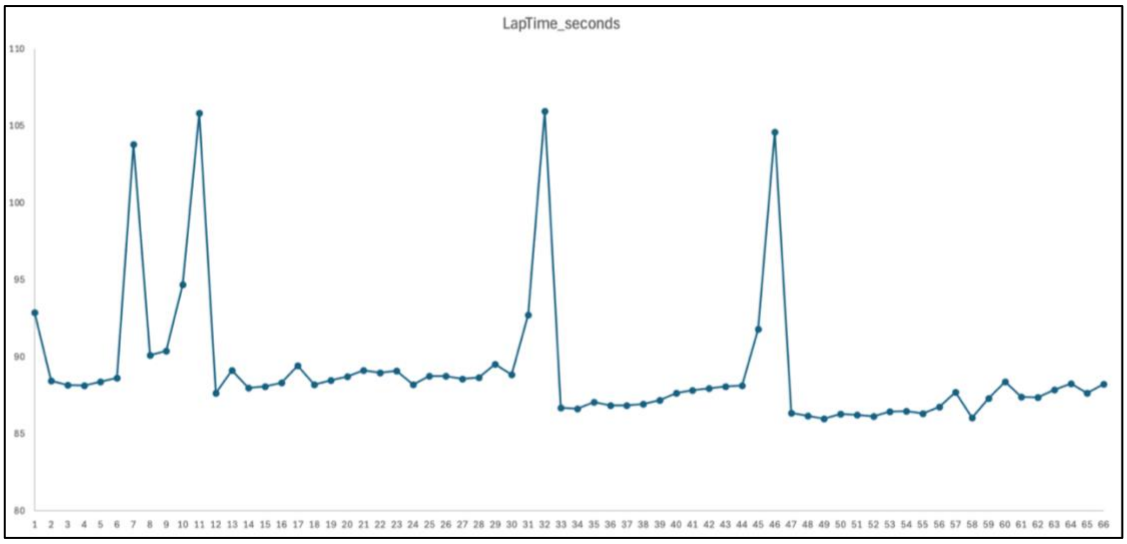


Figure 2: Leclerc, Spain 2022. Lap progression against lap time.

Further analysis on the target variable (delta) in **Fig.3** suggests that most points lie around 0, though the normal distribution of delta does suggest that when processing and preparing the data for ML, the datapoints passed through must be within the range of 0 for accurate predictions. Consideration must be taken to remove unwanted 0 values as they will influence the model predictions.

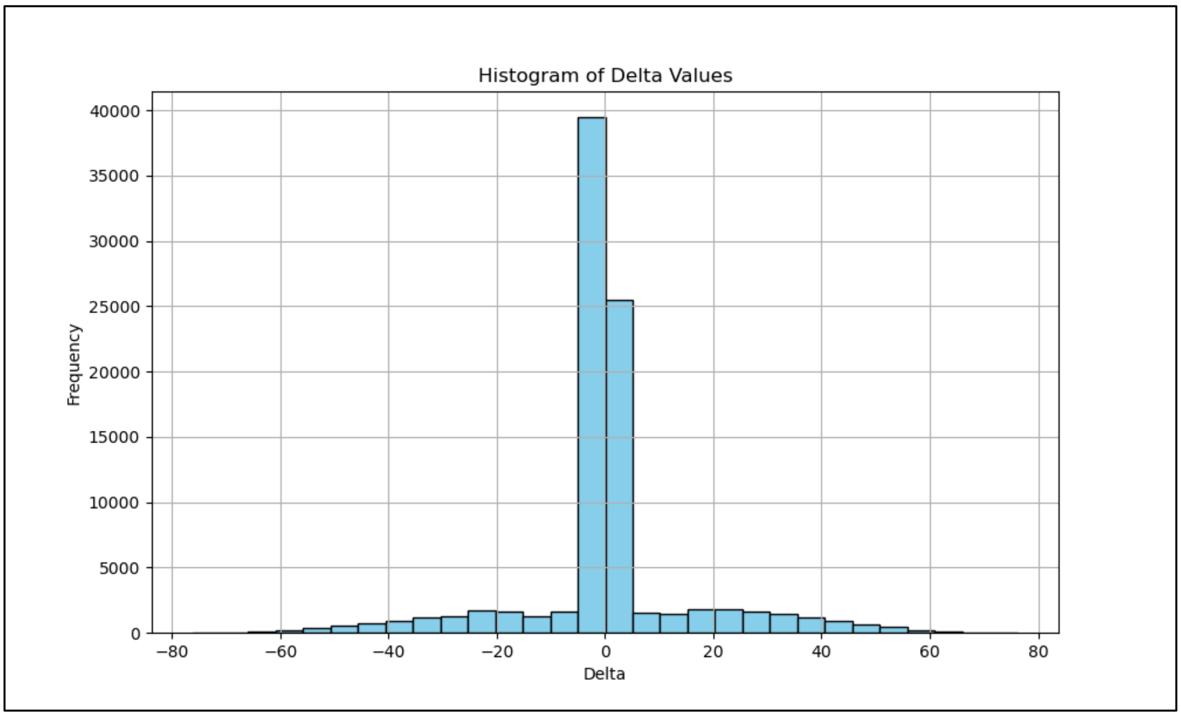


Figure 3: All delta points in the set.

3.2.4 Data Cleaning and Preprocessing

Data will be normalised where applicable to ensure consistency and prevent model biases due to differing magnitudes in variables, this may apply to track conditions like air, temperature and humidity. A multitude of cleaning techniques can be used to filter out any missing data, or alternatively, replace null values with synthesised data. Additional techniques such as deduplication may be applied should there be any redundant data during extraction. Manual removal of data may also be carried out to address inconsistencies, which can be easily identified with sufficient background knowledge of the domain. Further emphasis is placed on the data remaining consistent, where the collected data must be within the same generation of cars ensuring consistency in future predictions. A pipeline will be implemented to split the data accordingly and one-hot encode categorical features to ensure that all relevant variables are transformed into a format suitable for machine learning models, enabling efficient training and consistent evaluation across different datasets.

Cleaning

Each dataset is separated into two: session lap times and pit duration. For this project, data from Scuderia Ferrari will be utilised, this is so the ML can learn patterns between the track conditions and the delta. An additional column to identify stint numbers will also be incorporated, particularly in race sessions; this will be important in identifying the length of the tyre stint. Cleaned data will be stored in CSV format before being fed into the Scikit-Learn pipeline, ensuring consistency in formatting and transformation before model training.

Furthermore, it is important to consider these points when cleaning lap times:

- How do we know what lap times in practice sessions were done as "Race simulation runs"?

While it is challenging to definitively identify which practice laps were conducted as "race simulation runs" throughout the weekend, all lap times from the practice sessions will be considered. The same pre-processing steps will be applied to all data before being fed into the machine learning algorithm.

- What do we do if we have no data for wet/dry weather?

No data for wet/dry weather in races/sessions lead to two actions that could be done:

1) Utilise synthesised data based on lap times set in the session and the tyre compound the driver was on, this can be either through KNN, over/under sampling or random number generation.

Or:

2) Omitting unavailable data altogether as this could increase noise which can affect the outcome of the ML model.

- How do we determine that probabilistic influences were having an impact on the lap time?

Lap times will be processed for a selected team to account for differences in driving style. External influences such as DRS use, dirty air, and yellow flags cannot be explicitly identified, but lap time variations will be considered in preprocessing.

Preprocessing
A scikit-learn pipeline will be implemented to structure the preprocessing steps systematically, ensuring efficient handling of data transformations before applying machine learning models. This will include:

- Missing Values: Estimated lap times will be used to fill in missing times for deleted lap times (this typically happens during a race due to track limits or other reasons).
- Feature Engineering: Categorical variables will be transformed using one-hot encoding to allow the models to process them numerically.
- Data Optimisation: Operations such as deduplication and normalisation will be applied, removing duplicate data for improved model accuracy.
- Datatype Conversions: The original time format (dd:hh:mm:ss.sss) will be converted to seconds for numerical consistency.

By structuring these preprocessing steps within the Scikit-Learn pipeline, a reproducible and efficient workflow is established, ensuring consistency in data transformation before feeding it into machine learning models. Each tyre compound and fresh tyre columns would be one-hot encoded (i.e. categorised by unique identifiers 1, 2, 3 etc.), as well as other categorical features that may be inputted in the ML model such as the circuit name. By implementing this cleaning and preprocessing approach, we ensure that the data is in the optimal form for training and evaluating machine learning models, leading to more reliable and accurate predictions.

3.2.5 Machine Learning Implementation
Loreto (2023) showcased the effectiveness of support vector-based models, particularly Support Vector Machines (SVM), Random Forest (RF), and Artificial Neural Networks (ANN), for forecasting race scenarios and pit stop timings. While these three algorithms were used for classification tasks, this study will compare the predictive performance of traditional ML models, namely: SVR, RF, and XGBoost, to identify the most effective approach for tyre performance modelling. By evaluating these models, the study aims to determine which provides the best balance of accuracy and interpretability in predicting tyre degradation and optimising pit stop

timing before ultimately selecting the most suitable model. Results from this model would then be utilised in the simulation.

The implementation was carried out in Python, using libraries such as NumPy and Pandas for data manipulation and Scikit-learn for machine learning predictions. A mock-up of the expected tyre data and baseline lap times is provided in **Appendix B.3**, illustrating how the results will be structured in the output.

Tyre Degradation Model
The tyre degradation model was implemented based on the outputs of the machine learning algorithm, specifically using lap time deltas to estimate the rate of degradation for each tyre compound. Implementation of the model included:

- The total calculated lap time difference ($Laptime_{max} - Laptime_{min}$) to identify the maximum allowable time loss before pitting ($StintThreshold_{max}$).
- Averaging delta ($\Delta Laptime_{i-(i-1)}$) over multiple stints to derive the tyre degradation (TD_c) for each compound. This value is calculated using lap-by-lap differences and aggregated to determine the average time loss per lap.

Predicted tyre deltas would be totalled up to produce a maximum threshold from the difference of the minimum and maximum lap time, this would act as a **maximum** lap time loss before the tyres "fall off the cliff" for each compound known as $StintThreshold_{max}$. It is essential to have this variable as it provides clarity for the timing of the pitstop. Further justification and explanation of $StintThreshold_{max}$, including its assumptions and implementation within the simulation, can be found in **Section 4.4.3**.

Data produced from the ML model will be passed into the simulation to aid lap time generation.

Cross-Validation
Validation was performed by comparing the derived degradation rates to known patterns from historical race data.

Initially, the setup of the cross validation being Leave One Out Cross Validation, where a sample of each circuit will be used for the validation, and the rest is used for training; this is vital as generalising the model across each circuit allows for the test circuit (unseen data) to have accurate predictions and prevents overfitting. The same K-Folds (5) and iterations (100) were used, as well as the data selected (all time deltas between 0.5 and -0.5 seconds) to compare model performance.

However, consideration was then focused on Group K-Fold during development as the ability to generalise better than the initial approach was better without having to sacrifice extra computational power. Further adjustments made to the delta until the filter -0.2 to 0.2 was settled on, this allowed for more reasonable predictions that sit close to real life tyre performance; however, this did not capture the temporal relationship within the data despite the complex variable.

This led to the discovery of the time-split where 2022-2023 data would be used for all teams, then a specific circuit with team Ferrari (2022-2025) would be used as a test set. Doing so meant the removal of the cross-validation methods altogether in favour of a more robust data splitting method, which avoids data leakage completely.

3.2.6 Simulation Implementation

MCS was chosen to simulate the various race strategies, primarily due to its robust ability to handle uncertainty and incorporate probabilistic variables, factors that other methods in other papers often fail to include or incorporate.

The simulation will be executed over 10,000 iterations, with each iteration representing a distinct potential race scenario. In each iteration, race conditions such as track temperature, tyre degradation, pit stop durations are all generated using normal distribution and the pit decision logic relies on the difference between the minimum and maximum lap time using the historic race data. The predicted delta value from the ML model will aid in lap time generation as it will add that lap time loss/gain from the previous lap using gaussian distribution, this ensures that the simulation can emulate real-world tyre performance as closely as possible over the course of the stint.

The occurrence of Safety Cars or Virtual Safety Cars are randomly generated according to user input. These conditions are based on both the historical data gathered and predefined distributions that reflect the inherent variability found in real-world racing.

During development, major changes made to the simulation that specifically effected the pit decision, initially a stint threshold was used to determine pit stops, but this was replaced with "TyreLife" values, which more accurately reflects the remaining lifespan of the tyre under varying race conditions rather than relying on the cumulative delta. This change allows for a dynamic approach to degradation, where track temperature directly impacts tyre wear, ensuring that pit stop timing aligns with realistic tyre performance throughout the race.

Key features used in the simulation are taken directly from past race data and tyre degradation values from the output from the machine learning models are used. These outputs help to more accurately model tyre performance, which in turn influences lap times during the race simulations. This approach enables the simulation to produce a wide range of potential outcomes, capturing the unpredictability of race events and allowing for the generation of strategies that emulate the dynamic nature of motorsport.

The simulation process includes:

- Race parameters such as tyre selection, pit stops, and weather conditions are input into the model.
- The simulation progresses lap by lap per race simulation, recalculating tyre degradation, lap time adjustments, and pit stop timings based on the distributions in historic data.
- Once completed, the simulation generates a set of strategies, which are then ranked by overall race time using a sorting algorithm. This ranking identifies the fastest strategies under both normal conditions and conditions where Safety Cars or Virtual Safety Cars are present.

The top strategies for each scenario are then selected based on their performance. A mock-up of the expected simulation output can be found in **Appendix B.4**, which shows the predicted average lap times on the stint and tyre degradation during the stint.

3.2.7 Model Evaluation

The overall prediction results from the simulation will be evaluated by comparing the predicted race strategy to real-life results and the strategies used by teams for the same driver. For example, Max Verstappen’s 2024 British Grand Prix strategy would be compared against the model’s predictions using his actual race data via basic statistical methods, including the mean, standard deviation, and average, to assess variability of the model.

Additionally, comparisons with the results can also be made with Pirelli's tyre predictions before the race or generic tyre expectancy where according to Blackcircles (2023):

- SOFT – Last around 20-30 Laps
- MEDIUMS – Last around 30-40 Laps
- HARD – Last around 40-50, though can last a whole race but due to the mandatory pitstop rule, doing so would not be possible.

Error metrics will be utilised through numerous measurement methods such as R^2 score, MAE (Mean Absolute Error), RMSE (Root Mean Square Error), MSE (Mean Square Error). This is both applied to the machine learning algorithm(s) and the simulation results to determine the accuracy of both models combined. Visualisation techniques such as histogram plots and line charts will also be used to compare distributions and identify trends.

3.2.8 Considerations

Pit duration data is limited within both databases, pit duration time was invalid as cars do not take over an hour in the pit lane; this was further proved by the fact that null values in pit out time was also prevalent even for the rest of the drivers. OpenF1 lacked pit duration data before 2023, hence any pit duration data will solely rely from this source only.

When calculating the degradation constant, it is important to note that the degradation constant for each tyre compound can change depending on the circuit, weather conditions etc. However, it must be highlighted that the constants are based off historic race data and practice sessions and not considering conditions that affect degradation. Moreover, attention must be drawn towards the fact that probabilistic influences could play into it i.e. timing of ERS (Energy Recovery System) use/DRS (Drag Reduction System) deployment use, engine mode use in race that could affect lap time. It is important to consider this and factor such variance when processing this in the MCS.

The specific C0–C5 tyre compounds are not accounted for in this model, as each circuit utilises three of the possible six compounds selected by Pirelli. For example, Spain uses C3, C4, and C5, whereas Silverstone could have C1, C2, and C3. Factors such as high track temperatures may influence compound selection (e.g., preferring C1 over C5), thus the reliance on a specific circuit as a test set will be used to compensate. Moreover, the "C" labelled tyre compounds were not available in the data and are listed under the standard SOFT, MEDIUM, and HARD dry compound labels.

Finally, while the methodology outlines the initial approach, further exploration and experimentation such as utilising various cross validation methods/increasing the dataset size during development can lead to better results.

Section 5 outlines the development with changes that may deviate from the original method.

4. Artefact Design

4.1 Artefact Overview

A data-driven approach will be used to predict race strategy by integrating predicted lap times with the MCS. The data collected will serve as inputs for a machine learning (ML) model, which will out put the relevant features used in the simulation. Multiple scenarios would then be run, generating probabilistic strategies by accounting for variations in lap times, and other factors. The simulation would be expected to output the best strategies for different scenarios, accounting for probabilistic influences.

4.2 Framework, Tooling & Technology

4.2.1 Tooling & Technology

All data extracted will be loaded into an API as a live connection would cause major issues due to the 500 API Call limit that the package fastf1 has, thus loading this data into CSV datasets would be more viable due to the data being structured. Furthermore, the removal of the need for a database means that there is a significant reduction in overhead due to the simplicity of the storage. In addition, use of CSV files is compatible with the packages used within python such as Pandas, Sci-Kit Learn and NumPy, each of these will be used in both ML and Simulation components.

Sci-kit learn will be used for the tyre degradation model as it is lightweight and versatile, this extensive library allows me to pick from numerous different machine learning models available without any additional/heavy dependencies. Though this is not the main library for one of the ML models (XGBoost), it can still be used as the interface. – this is an advantage as it allows for standardised workflow like any other ML model that is available in Sci-kit. Additionally, integration with Pipeline and CV functions within Sci-kit also make this an intuitive package to use. Finally, the ease of experimentation allows me to switch from one model to the other given that parameters match up with the relevant ML model.

The final ML model will be saved as a `.pkl` file using a package called pickle. Since it is a built-in python module, this reduces the need for additional dependencies. Additionally, since the machine learning model is not exceptionally large, using joblib will not be necessary. Utilising this module allows for a seamless integration of ML model within the simulation by simply loading the `.pkl` file up and proceeding to conduct prediction values which are returned to the MCS model.

Monte Carlo Simulation will be powered with NumPy, since it is powered using numerical computations. The library is simple and flexible for use and uncertainty modelling using this library is easy to execute due to the amount of control you have on random distributions. Finally, performance costs are kept at a minimum with this package as it is fully optimised for numerical computational problems such as this one.

4.2.2 System Architecture

Since the overall system follows a **modular architecture**, it allows for the overall system to be flexible, scalable and maintainable by enabling independent development of each component of the system. This design will follow Single Responsibility principle (**SRP**) which gives each script a single task to be responsible for:

- Simulation

- Preprocessing
- Predictions
- Visualisations
- Results Output
- Error Handling
- Loading data
- Sorting data

Handling the code and organising the scripts this way allow for easier debugging as well as improving code clarity, reusability and collaboration with other people. Any modifications made to any specific component can be made if the dataflow and dependencies between modules are maintained.

Fig.4 provides the diagram that shows how the components interact with each other, showing what script/functions will be called in which script. Each script is called accordingly in the diagram, showing the dependencies in the program.

Each major component of the model is made up of three parts:

- **main.py** – the main script consisting of a menu called in the terminal, this will call both **sim.py** to process, predict and simulate the data in the csv files.
- **sim.py** – calls the tyre_model.py and loads the predictions. It then simulates the races using the predicted values and models uncertainty, this produces the predicted/recommended ideal pit in as well as visualising pit windows and lap time traces
- **preprocessing.py** – processes the data and applies relevant feature engineering
- **tyre_model.py** – the script that will predict the lap time loss of each compound, will preprocess the data and use it in the ML pickle (.pkl) file. Predicted values are then returned.
- **vis.py** – visualises the data and outputs the results, this allows for easier comparisons and validate results against real life or predicted performance from official partners of F1 such as Pirelli.

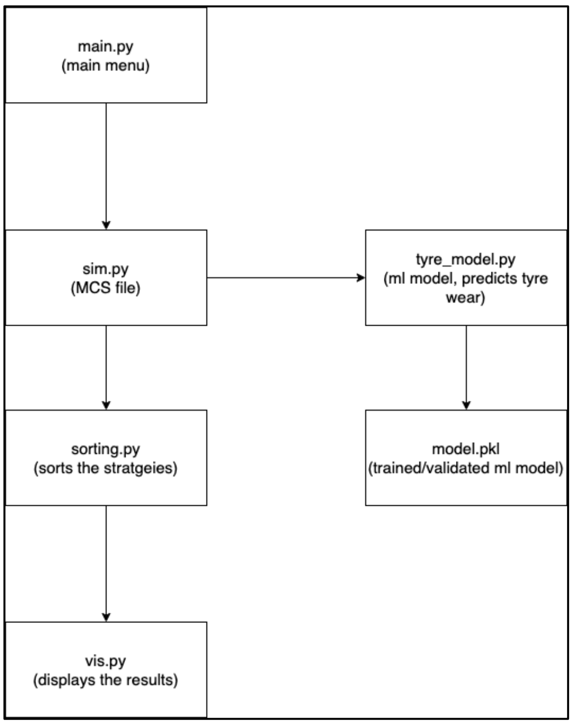

main.py (main menu)
sim.py (MCS file)
tyre_model.py (ml model, predicts tyre wear)
sorting.py (sorts the stratgeies)
model.pkl (trained/validated ml model)
vis.py (displays the results)

Figure 4: Dependency diagram

Careful tweaking of the code must be done however to ensure the system maintains its integrity and functionality, as the interdependency between components requires careful management. Failure to address this can result in a dysfunctional system and possible difficulties to further debug the system. It must also be addressed that as the system scales (if needed to include more functionality), managing the dependencies are critical to preserving the overall functionality.

4.2.2 Project Structure

The planned directory tree diagram (**Fig.5**) aims to separate the different components of code to ensure clarity, as well allowing the programme to be easily maintained and flexible to any additions that it may need in future work while following the planned design of the program. This ensures that the SRP is being followed and ensures that the scripts remain in one folder to allow easy access when importing each script.

```
Project_folder
└── src/
    ├── main.py
    ├── sim.py
    ├── sorting.py
    ├── tyre_model.py
    ├── vis.py
    └── MLModel.pkl
```

Figure 5: Directory Tree diagram

4.2.4 Overall System Data Flow Diagram

The machine learning model can be replaced with a different model, although for the purposes of this project, the best model will be utilised. A simple data flowchart in **Fig.6** represents the overall data flow of the system, doing it this way allows for an efficient and elegant handling of the data.

- **FastF1 & OpenF1 API** – Both API sources are called to extract the data into CSV files, as mentioned earlier API call limits can cause issues, hence storing this in CSV files are necessary.
- Pre-processing is done to the race data, transforming it and making the data suitable for ML training and validating
- **Machine Learning Model** – the script trains code for the selected ML model, training and validation is done here. This does include fine tuning parameters and splitting the data accordingly to achieve satisfactory predicted results. This is then saved as a `.pkl` file.
- **Load PKL** – the data is processed and is used in the loaded pickle file to produce predictions to the tyre compounds, this data is then passed to the simulation model.
- **Monte Carlo Simulation** – this simulates a set number of races and determines the best strategies by filtering out by number of pitstops and race time, this will then identify unique strategies that the driver can use in the race/the engineer can use as a baseline.

Once the data is pre-processed and the ML model trained, the model's predictions are passed to the Monte Carlo simulation to generate optimal strategies.

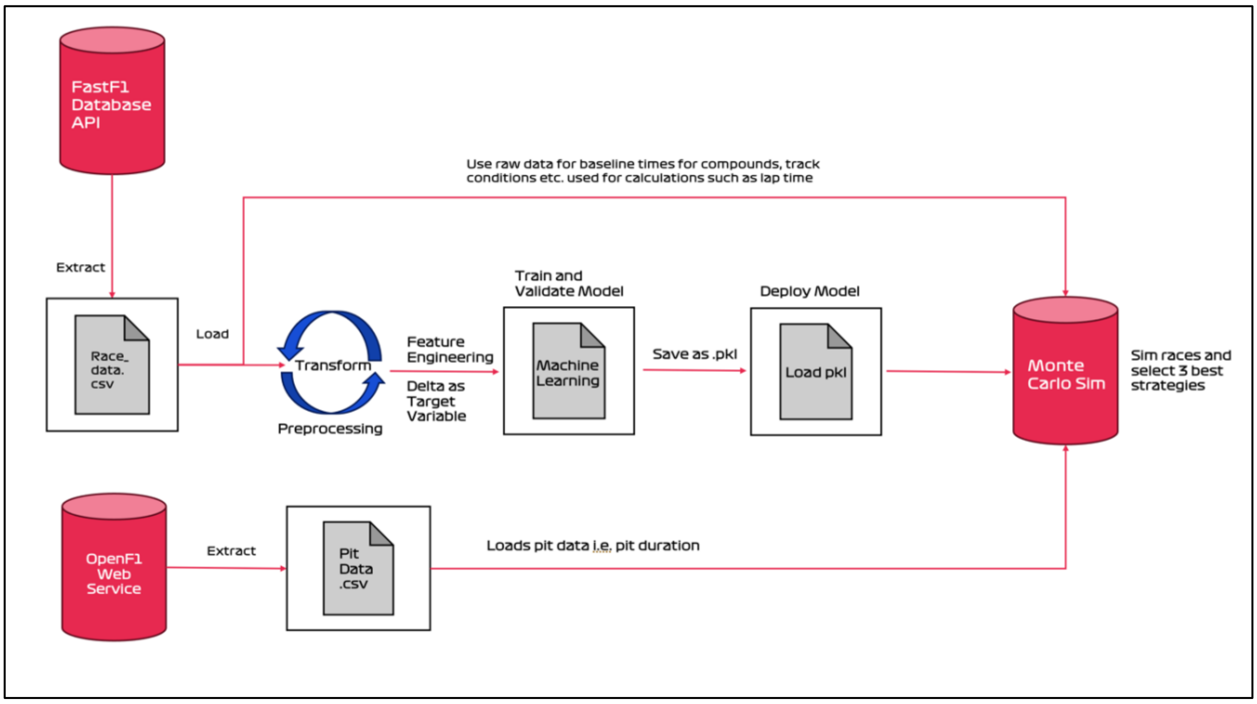


Figure 6: Data flow of the overall system

4.3 Tyre Degradation Machine Learning Model

The tyre degradation model will be used to predict the lap time loss as well as the variation of the deltas in each lap for every compound. This section will discuss the structure and each component of the model, and how this will achieve the objective of the problem imposed on the project.

Leave One Out Cross Validation (LOOCV) will be used, where a single circuit is excluded for validation in each iteration, while the remaining circuits are used to train the model. This process involves training the model on all circuits except one, with the excluded circuit serving as the validation set. The model's performance is then evaluated on the left-out circuit, and this is repeated for each circuit in the dataset. LOOCV ensures that each circuit is tested individually, providing a strong assessment of the model's ability to generalise to unseen data. This method is particularly useful when the dataset is limited, as it allows for thorough evaluation, making sure the model can handle the variability of different circuits in real-world applications.

From the analysis conducted, SVR, RF and XGBoost emerge as candidates and are viable options to use as they are capable of handling smaller datasets. Thorough model comparison testing will be needed to determine the best model for this problem by evaluating their performance based on factors such as accuracy, training time, and computational efficiency or through metrics such as R^2, MSE, RMSE and MAE.

Neural networks may not be a viable choice as the data set used may be limited and the solution does factor in compatibility with lower-end systems where possible. Considering this allows for smaller teams in the motorsport domain a chance to have a solution without investing in more powerful hardware.

4.3.1 Model Structure

The machine learning in model will always be structured into the following components:

- Input Features

The selection of input features plays a vital role in determining the quality and relevance of the data, which influences the model's overall performance. By carefully choosing the most relevant features, it effectively reduces the presence of noise, allowing the model to focus on learning meaningful patterns that contribute to accuracy. This process ensures that the model is not burdened with unnecessary or irrelevant information, making it both efficient and precise. Furthermore, selecting the right features enables the model to focus on the most important relationships within the data, which becomes particularly critical when working with complex models that rely on feature interactions to deliver accurate results.

- Preprocessing and Feature Engineering

The preprocessing pipeline ensures that raw session data is properly cleaned and structured for model training:

- Data Cleaning: Irrelevant columns are removed, missing values are handled, and session consistency is ensured.
- Feature Selection and Engineering: Relevant inputs such as tyre degradation, lap times, stint lengths, and track conditions are extracted. Additional features may be engineered to better capture race dynamics.
- Handling Outliers: Extreme lap times caused by pit stops, accidents, or unusual track conditions are filtered out to maintain data quality, this is done through filtering the relevant target variable range to eliminate outlier influence.

- Normalisation and Scaling: Tree-based models are capable of handling unscaled numerical data effectively, though may still need to apply it if necessary. As for other ML models, normalisation and scaling will be necessary through various means i.e. Robust Scaler or Standard Scaler.

Preprocessing will also involve one-hot encoding the variables to ensure that the ML model can process any categorical data as numerical. It will be made clear in the code as separating the features into categorical and numerical allows clarity.

- Evaluation and Model Tuning

The unseen set will not be tested until the final model is tested and ready to produce results. Therefore, evaluating the ML model will rely on cross-validation and comparison of model scores between the training set and the validation set. Hyperparameter search techniques such as random grid searching will be used to allow for effective hyperparameter search as it can explore a large space, and visualisations will be made to see any trends that may indicate over/under fitting. Once finalised, the model will be deployed in the form of a `.pkl` file. **Fig.7** diagram provides a visual of the data flow during training and validation, it also shows the data flow when loading the pickle file is used to predict unseen data in the `tyre_model.py` script.

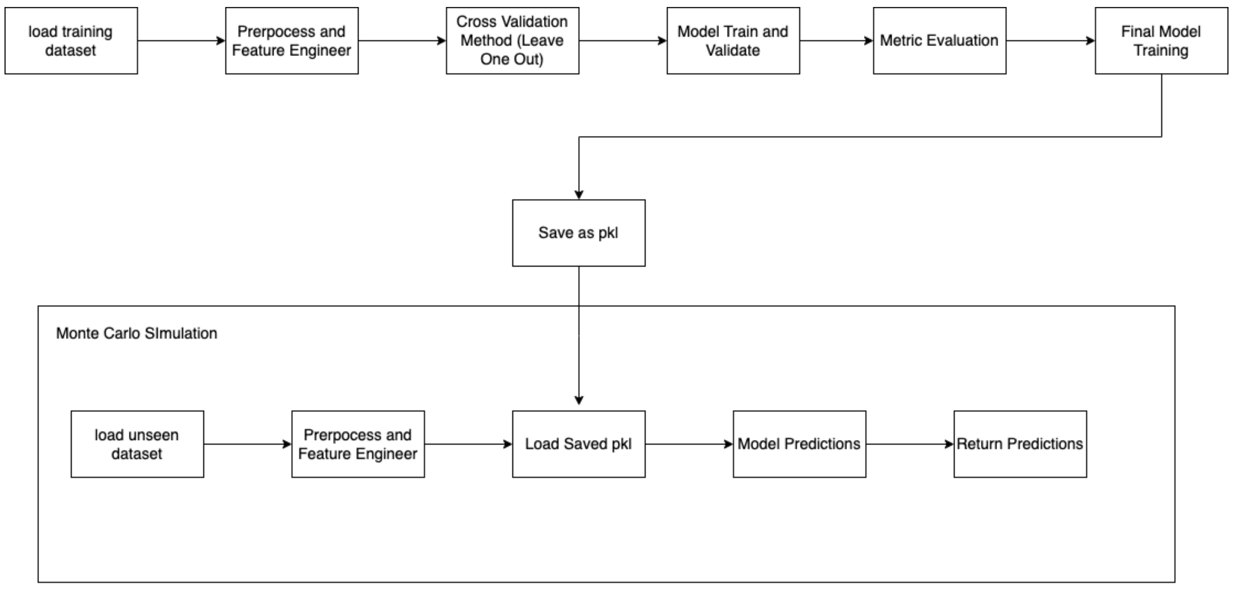


Figure 7: Data flow of the Machine Learning Model

4.4 Monte Carlo Simulation

MCS will be used as the basis of our predictive analysis model to cover the uncertain variables problem. It would allow us to simulate different scenarios based on the parameters provided from the ML models. The following can be used as uncertainty variables in the simulation:

- Track conditions: Surface condition, temperature, etc.
- Tyre properties: Tyre temperature, lateral load, slip angle etc. anything that affects the wear of the tyre besides track and weather conditions.
- Weather conditions
- Chances of incidents/crashes/safety cars which could force a strategy change.
- Probabilistic factors that affect pit duration.

Though the use of any of the simulation choices may be viable (particularly DES and MCS), the selection of simulation must meet the criteria as outlined in the requirements analysis.

Initial exploration of simulation models focused on Discrete Event Simulation, as would be ideal for use in this project. However, the main issue was the fact that the inclusion of probabilistic influences could prove difficult as there are far too many to consider, hence a model that can simplify the instances while also showing that the effect is happening would be needed. A further argument for the choice to use MCS is the fact that it is used in industry thus flexibility and robustness when it comes to modelling uncertainties, which DES cannot do as efficiently.

Fig.8 illustrates the Monte Carlo Simulation (MCS) model, providing a visual representation of the race simulation workflow. This workflow determines when lap times are generated, when pit stops occur, and how ML values are integrated into the simulation.

The race simulation is structured around stints, where each stint consists of multiple consecutive laps driven on a single tyre compound. Lap times are generated within these stints based on the tyre degradation model. Once a tyre reaches its maximum lifespan or another condition triggers a pit stop, the stint ends, and a new stint begins with a fresh set of tires. This process loops through multiple stints, separated by pit stops, until the full race distance is completed. Each complete simulation run produces a possible race strategy, helping to evaluate different tyre and pit stop combinations under varying conditions**.**

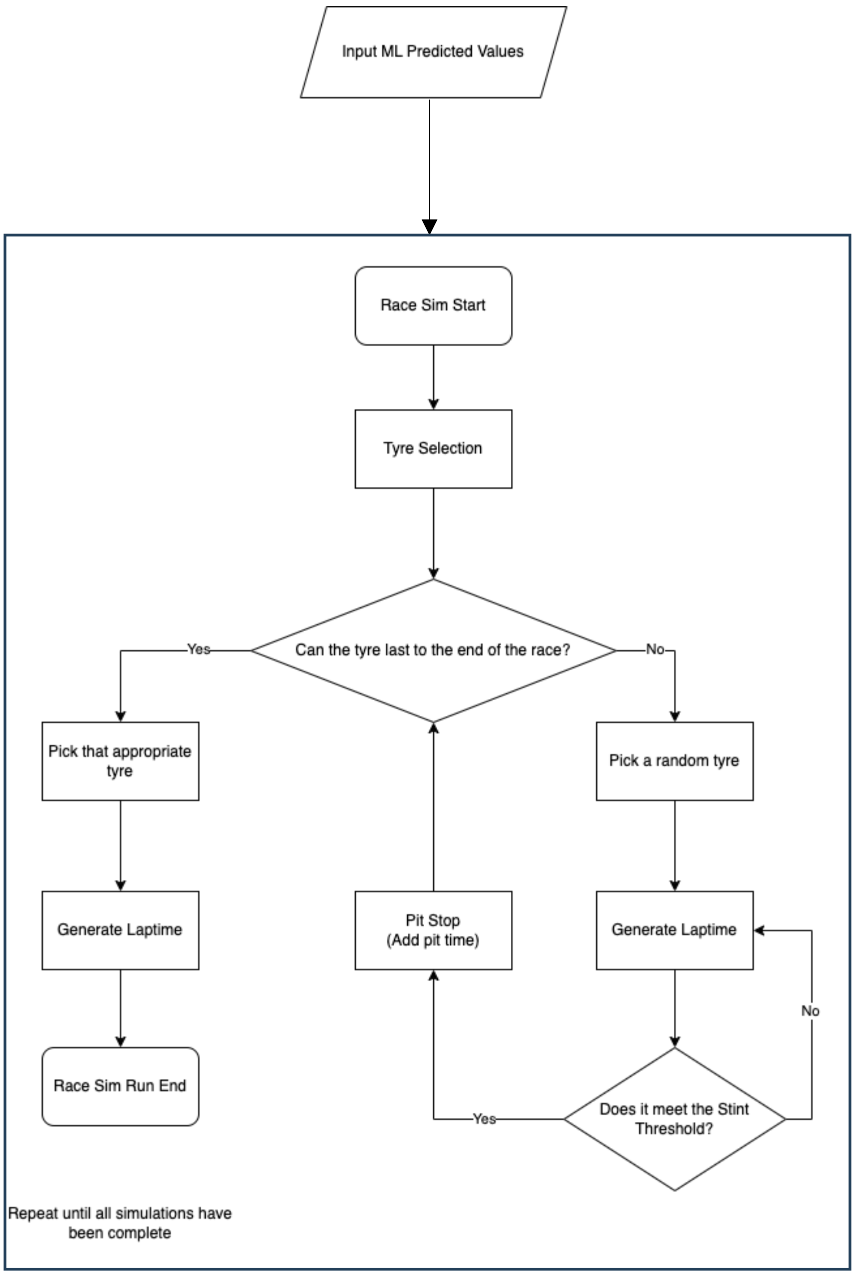

Input ML Predicted Values
Race Sim Start
Tyre Selection
Can the tyre last to the end of the race?
Yes
No
Pick that appropriate tyre
Pick a random tyre
Generate Laptime
Pit Stop (Add pit time)
Generate Laptime
No
Race Sim Run End
Does it meet the Stint Threshold?
Yes
Repeat until all simulations have been complete

Figure 8: MCS flowchart

4.4.1 Baseline Tyre Degradation

Output data from the machine learning model would be used to determine the assumed degradation rate for each compound. Averaging the degradation rate for each compound is necessary as this gives us a general idea of the rate of degradation for each compound. Input features utilised in the ML model would be:

- i (number of laps) as the independent variable
- $\Delta Laptime_{i-(i-1)}$ or Delta acting as the target variable.

All the data points for each Compound and FreshTyre combination in the machine learning model would then be averaged out with *each stint* which will determine the tyre degradation for each compound known as μ_{ML}. Furthermore, a standard deviation would be produced to reflect the varying tyre degradation lap on lap giving a rough estimation of the lap time loss as opposed to using a fixed value to emulate a racing situation– this will be useful when implementing the simulation. Use of $Laptime_{i-1}$ is always used as we want to monitor the lap-by-lap performance of the tyre during the stint (see **Section 4.5.1**). All the values would then be returned into the dictionary in the MCS model which would be used for processing and simulating the race scenarios.

4.4.2 Lap time Modelling (Lap time generation component)

Lap times simulated would simply be generated from the previous set lap time ($Laptime_{i-1}$) with the additions of a time delta to emulate the race progression i.e. tyre wear/tyre saving/driver pushing. However, it is important to note that the tyre will gradually get worse as it will be in real life, thus scaling with this equation $\left(1 + \frac{i}{xT_{max}}\right)$ would simulate the relationship between the tyre wear and the degradation.

Considering that tyre degradation and the track temperature have a relationship as they affect each other. A higher track temperature means that the tyre life is shortened due to the increased tyre degradation, whereas a lower track temperature can have the opposite effect. To model this relationship, the following takes the current lap track temperature and the mean which is randomly generated. This is then used to account for the ratio of the current lap track temperature to the average lap temperature as represented by:

$$\frac{TrackTemp_i}{TrackTemp_{mean}}$$

This demonstrates the effect of the track surface onto the tyre to make up for the lack of tyre data, this is further supported by Ambler (2024b) which states "Higher temperatures increase tire grip but can lead to overheating and degradation issues. Lower temperatures reduce tire flexibility, making it harder to generate friction."

This will be accounted for alongside the additional scale factor value of $\left(1 + \frac{i}{xT_{max}}\right)$ in $DegFactor_i$ which accounts for the increase of wear over the course of the stint against the expected maximum tyre life (known as xT_{max}) taken from the dataset.

See **Appendix C.1** for the full break down of **Equation 2**.

Lap times are then generated using the full equation:

$$Laptime_i = Laptime_0 + TD_c \times DegFactor_i$$

Equation 2: Lap time generation equation

Thus, the model ensures that:

- More aggressive tyre degradation leads to faster lap times initially, followed by a steeper performance drop-off.
- Less aggressive tyre degradation results in slower initial lap times, but with better longevity in the stint.

4.4.3 Pit stop modelling

Pit Decision

The pit max degradation threshold is determined by the difference of min and max lap time on the dataset (**Equation 3**) plus the baseline lap time. Since the exact race conditions can vary from stint to stint, historical race data is used to calculate the delta, allowing for a more accurate reflection of tyre degradation over time. This method ensures the simulation captures realistic tyre life cycles and degradation patterns. By incorporating $StintThreshold_{max}$, we account for the gradual increase in lap times as the tyres wear, defining the point at which they become unsustainable for continued performance, thus triggering the need for a pit stop.

$$StintThreshold_{max} = Laptime_0 + (Laptime_{max} - Laptime_{min})$$

- $Laptime_0$ - Baseline lap time

Equation 3: Stint Threshold Equation

Notably, "all drivers must use at least two different slick compounds during the race, providing the track is dry" (Seymour, 2023). Since this is a predictive model, the simulation will attempt every tyre combination in the simulated 10,000 runs, this will then be filtered out when the strategies are sorted and selected. The driver would only then pit should the difference between $Laptime_0$ and $Laptime_i$ plus the baseline lap exceeds the threshold by following the simple logic in **Equation 4**.

$$Laptime_i \geq StintThreshold_{max}$$

Equation 4: Stint Threshold Logic

Additionally, pitstop durations will be varied to emulate possible holdups due to traffic or pit crew mistakes, lockup during the pitting in phase or various other factors that play into a pitstop as seen in **Equation 5**.

$$PitDuration \sim (PitDuration_{mean}, PitDuration_{std})$$

Equation 5: Pit duration distribution

Tyre Selection

In the model, tyre selection follows a random process while ensuring the full race distance is covered. Initially, a tyre is randomly chosen for the opening race stint, the model runs the stint until the tyre is no longer viable (based on xT_{max}). At that point, another random tyre is selected, and the process repeats until the final tyre allows the driver to reach the end of the race, which will not trigger a pitstop. The logic in **Fig.8** illustrates how tyre changes occur dynamically based on stint limitations.

Strategy Selection

It must be noted that the strategies selected are unique based off compound use order, and the fastest times. Unique strategies are identified using a cumulative sum and rounding method, where the number of laps done on the stint is recorded and how many laps were completed overall during the race - showing race progression.

For example, if a driver completes 10 laps before the first pit stop, 12 laps for the second stint, and 9 laps for the third, the cumulative sum will result in pit stops occurring at laps 10, 22, and 31, respectively.

The number of laps done on the stint is rounded, for simplicity this model will use to the nearest 10 to ensure a balance between generalising the strategies. This process is designed to reduce the precision of lap numbers, making the strategies more general and easier to categorise. For example:

- A pitstop at lap 22 would round to 20.
- A pitstop at lap 12 would round to 10.
- A pitstop at lap 24 would round to 20.

Same compound usage i.e. SOFT to MEDIUM but with different pitstop laps like 12 and 13 will still be considered unique and will be implemented. Each produced race strategy will be stored as a key-value pair in a dictionary, this will make it easy to access when selecting strategies.

When selecting the strategies, we will store all the fastest race time of each unique strategy into a heap and instead of sorting them, the fastest strategies will be selected from the tree. There will be a filter that omits any strategies that do not use at least two different slick compounds to ensure compliance with the sporting regulations. Doing it this way is more efficient as it allows the model to quickly identify and retrieve the best possible strategies without unnecessary computation, ensuring a more effective and streamlined selection process.

See **Appendix C.2** for the full breakdown and visualisation of the algorithm.

4.5 Simulating Race Dynamics

Accurately capturing the dynamics and conditions of each lap on track is essential to producing accurate and reliable predictions. By incorporating the possible probabilistic influences such as the track condition, the simulation can better reflect the real-world racing conditions, providing valuable insights which can lead to enhanced performance optimisation.

This section outlines the components that are involved in emulating race dynamics as closely as possible using stochastic modelling.

4.5.1 Emulating realistic tyre behaviour

Utilising normal (gaussian) distribution to emulate realistic behaviour is necessary as it reflects real-world dynamics (as seen in **Equation 6** and **7**), following a normal distribution for tyres allows for effective modelling of typical gradual wear and degradation patterns observed in Formula One tyre compounds. This done by generating a delta every lap with the given values from the ML model:

$$TD_c \sim (\mu_{ML}, \sigma_{ML})$$

Equation 6: Tyre degradation distribution

- TD_c – Tyre degradation (delta per lap)
- μ_{ML} – Predicted mean from ML model
- σ_{ML} – Predicted Standard Deviation from ML model

Furthermore, including variation in lap times due to the track condition evolving over time allows for realistic tyre modelling. The utilisation of the gaussian distribution applies to TD_c – this directly affects the lap times generated as mentioned in **Equation 6**. Additional tyre behaviour can be influenced through the track conditions, Track temperature for example is calculated for each lap using **Equation 7**:

$$TrackTemp \sim (TrackTemp_{mean}, TrackTemp_{std})$$

Equation 7: Track temperature distribution

By modelling the track temperature this way, natural variation of the conditions on the circuit are captured; during races, track temperature constantly changes but will remain the same unless there is reduced sunlight due to cloud coverage or wind. Variability must be accounted for in the simulation alongside TD_c to capture the dynamic lap time predictions.

4.5.2 Safety Car Deployment

The inclusion of a chance of a Safety Car/VSC would be incorporated into the simulation, though it would be difficult to implement, the system will select the most viable strategy to follow assuming a safety car will come out (as seen in **Fig.9**.). It will follow the condition that if a safety car probability was $< 50\%$ it will be classed as low probability SC rate and follow a more aggressive strategy which will be indicated by the system. Whereas a probability of $\geq 50\%$ will still be valid but recommended as the main strategy to follow. Doing so allows for a more defined decision as to which strategy to select during a race weekend.

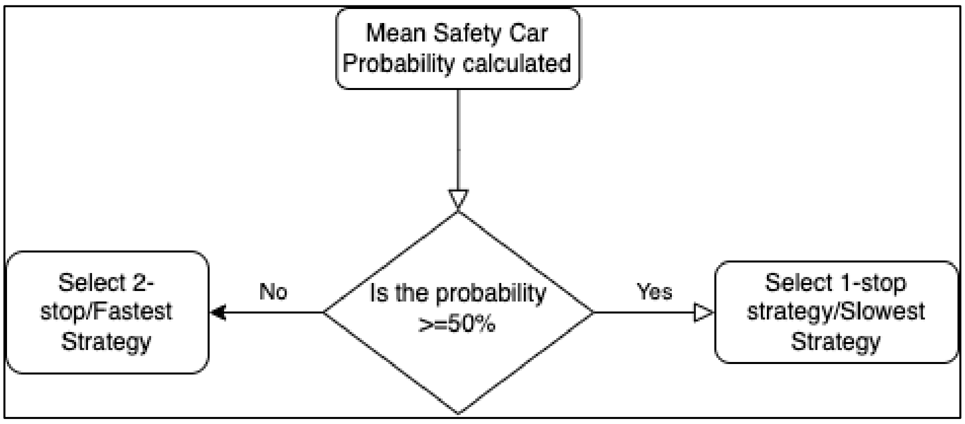

Mean Safety Car Probability calculated
Is the probability >=50%
No
Select 2-stop/Fastest Strategy
Yes
Select 1-stop strategy/Slowest Strategy

Figure 9: Probability Diagram

4.5.3 Clean/Free Air Integration

What numerous studies did not mention was the incorporation of dirty air, this is effectively the same as slipstream down the straights except, the dirty air effect occurs in corners, where "the lead car effectively uses up the energy of the oncoming air and leaves behind air with low total pressure", this reduces the aerodynamic performance of a car as well as the aero balance which results in an increased lap time (Ono, 2020).

Upon further research, Halliday (2024) simulated the effects of slipstreaming (or following a car within a ½ a car length), it was concluded that a 22% loss of drag meant alongside a loss in downforce negatively affects the cornering speeds of the car resulting in slower lap times. While different car designs and setups may try to reduce the dirty air affect, an assumption can be made to model effects on the tyres, which also corresponds to strategy where if you were held up by the car ahead, it would make sense to pit and change tyres to gain an advantage, this is called *undercutting*.

Implementing this into the MCS can be done by forcing the lap times to be decrease when there is clean air, which suggests that there there is a fixed 4.77% encounter rate of clean air (1 driver encountering 1 of the 21 other drivers hence 1/21). However, in the simulation, the initial probability will be doubled as drivers will not always be stuck behind another driver and will end up having clean air. This value can be higher in more spaced-out tracks where the gaps between drivers are bigger, though it is very difficult to determine the encounter rate, hence the encounter rate is doubled to **9%** for each circuit tested.

4.5.4 Interchangeable conditions

Implementing interchangeable conditions can be difficult, but I propose a probability system to handle this challenge. The system would assess the likelihood of rain during any given iteration of the simulation. For example, if there is a 20% chance of rain, the system would ensure that, at least, 2000 out of 10,000 simulations experience rain. This probabilistic approach would add variability to the simulation, making it more representative of real-world racing conditions where weather changes are unpredictable.

Additionally, the system would incorporate key environmental data such as track temperature, air temperature, and humidity to more accurately simulate the likelihood of rain. By factoring in these environmental features, the system could dynamically assess whether the rain would fall during the race, increasing the realism of the simulation. For example, high humidity or low air temperature could increase the likelihood of rain, while track temperature might influence how quickly the rain affects the racing conditions.

When the rain is triggered in the simulation, it would randomly occur within a specific lap range, i.e. lap 30-50, mimicking the unpredictability of rain during a race, forcing the simulation to adapt by changing to INTERMEDIATE or WET tyres. In this way, the simulation would integrate weather variability and environmental factors, ensuring a more complex and realistic race outcome. By relying on the probabilistic approach, we can simulate different scenarios and outcomes, providing a more realistic model of how weather influences race strategy and tyre choices.

The timing and nature of the tyre change would still rely on the Tyre Selection System, as described in **Section 4.4.3** "Tyre Selection."

4.6 Summary

In this section, the design of the artefact was discussed while prioritising the focus on understanding tyre degradation and its effects on lap times, pit stops and general race strategy.

Key Components:

- **Tyre Degradation model** - A machine learning model that predicts the rate of lap time loss for each compound, while incorporating possible factors that affect the tyre over the course of a stint such as track temperature, air temperature, humidity. This ML model will then influence the MCS to simulate lap time changes as realistically as possible.
- **Monte Carlo Simulation** - This is the probabilistic approach that simulates numerous races with varying race strategies with the given data from both the dataset and the predicted data. This will determine optimal strategies under various conditions.
- **Race Dynamics** - MCS will also stochastically model these dynamics to emulate varying/evolving track conditions, incorporating varying lap times as a direct result of the conditions on track. Furthermore, the inclusion of dirty air and clean air impacts as well as safety car/virtual safety car deployments allows for race strategies to be optimised in all potential conditions.

Overall, the approach integrates complex factors to emulate real-world conditions to produce accurate and reliable predictions that can aid and enhance performance of a team during a race. The combination of ML and MCS allows for a more robust system which reduces the uncertainty as much as possible while boosting the confidence in the system. A stochastic model such as this one ensures that strategies are adapted accordingly as numerous strategies are produced as well as the probability, this fallback allows for teams to make decisions based off predictions made by historic data.

5. Artefact Development

5.1 Development Overview

During development, numerous changes were made throughout to ensure an ideal and satisfactory final product. Initial use of LOOCV was tested before changing the validation method to Group K-Fold, but after through testing and exploration, it was discovered that time-split by testing on 2022-2023 data using all teams except Ferrari and training on Ferrari (2022-2025 data) provided a better output. Furthermore, MCS model had some changes regarding logic, more notably changes made to the pit decision logic and improving the way the model simulates lap times.

5.2 Tyre Degradation Model

5.2.1 Model Selection

It is stated that, "NNs performed much better… than traditional methods, especially in terms of prediction quality, robustness, and comprehensibility." (Heilmeier, 2020b). Even though the utilisation of this approach is viable for this problem such as , FCNN, ANN and EDNN, the issue lies in the amount of data needed to produce such results which would not be possible given the limited data collected; therefore, not meeting all the criteria.

Given the constraints, models that can better handle smaller datasets such as the use of SVM (Support Vector Machines) could be considered over Random Forest, as claimed by Loreto G.T (2023), though a model comparison could be conducted to ensure the correct model is chosen.

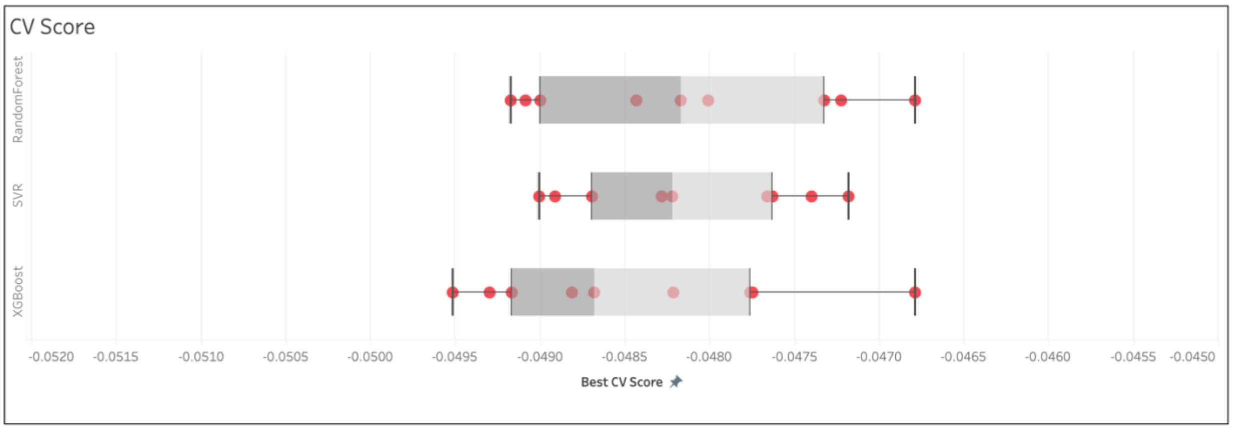
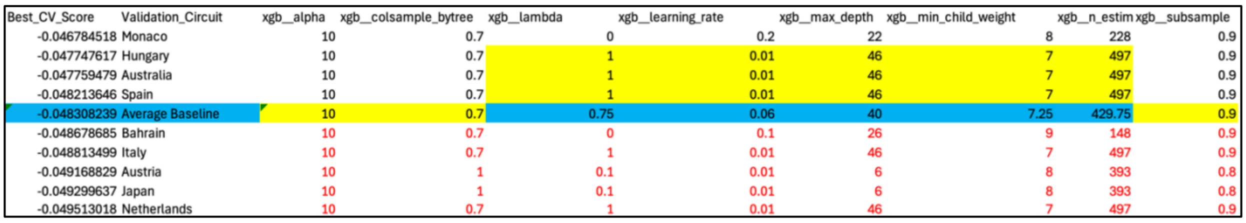


Figure 10: Comparison Of Models Box plot

As seen in **Fig.10** the "Best CV score" is the negative MSE (used with sci-kit learn) to determine a metric for regression model evaluation, the more increasingly negative the number, the better the performance. XGBoost shows promising results in terms of generalisation, as the majority of its data points lie within the interquartile range (IQR), despite a large standard deviation. This suggests that XGBoost has a robust generalisation performance compared to other models.

Despite the standard deviation being so large compared to the other models, this majority of the points lie within the IQR range which shows robust generalisation across the models, furthermore MSE, MAE and RMSE loss should be as close to 0 where possible. See **Appendix D.1** for the full metrics table, visualisation and analysis.

Hyperparameter Tuning and Preprocessing
When finding the correct parameters, taking the average CV score will help find the baseline, using this also eliminates the hyper parameters that have a CV score further away from 0. However, the parameters that appear most are more likely to be better at generalising, thus they will be chosen; in the case where there is not common parameter value, the average will be used.



Best_CV_Score	Validation_Circuit	xgb__alpha	xgb__colsample_bytree	xgb__lambda	xgb__learning_rate	xgb__max_depth	xgb__min_child_weight	xgb__n_estim	xgb__subsample
-0.046784518	Monaco	10	0.7	0	0.2	22	8	228	0.9
-0.047747617	Hungary	10	0.7	1	0.01	46	7	497	0.9
-0.047759479	Australia	10	0.7	1	0.01	46	7	497	0.9
-0.048213646	Spain	10	0.7	1	0.01	46	7	497	0.9
-0.048308239	Average Baseline	10	0.7	0.75	0.06	40	7.25	429.75	0.9
-0.048678685	Bahrain	10	0.7	0	0.1	26	9	148	0.9
-0.048813499	Italy	10	0.7	1	0.01	46	7	497	0.9
-0.049168829	Austria	10	1	0.1	0.01	6	8	393	0.8
-0.049299637	Japan	10	1	0.1	0.01	6	8	393	0.8
-0.049513018	Netherlands	10	0.7	1	0.01	46	7	497	0.9

Figure 11: Highlighted initial Hyperparameters

Hyper-parameters highlighted in yellow in **Fig.11** will be the basis of the testing, further tweaks will then be made to ensure that the model is performing optimally. This is then saved as a `.pkl` file using the pickle package where it can be re-used when this is integrated into the simulation.

All models used `StandardScaler()` to normalise the value, except with XGBoost as that can handle small values particularly well, thus normalisation is not needed; this is done by using the `‘passthrough’` rather than the `StandardScaler()` (**see Fig.12**).

It must also be highlighted that one-hot encoding is used to ensure that the ML model can process the categorical values (cat_f),s specifically the tyre compounds.

```
    preprocessor = ColumnTransformer (
        transformers= [
            ('one_hot', OneHotEncoder(handle_unknown='ignore'), cat_f),
            ('num', 'passthrough', num_f)
        ]
    )
```

Figure 12: Preprocessing pipeline

5.2.2 Initial Model – Initial Parameters
Despite the initial testing on the current parameters highlighted in the comparison model table above as well as the standard feature engineering (utilising the features in the given dataset) as seen in **Fig.13**. **Table 1** showed poor capture in data variability despite the relatively low MSE, RMSE and MAE scores. The model had already been using LOOCV (Leave-One-Out Cross Validation) to allow for better generalisation.

```
#num + cat
num_f = ['TyreLife', 'LapNumber', 'LapTime_seconds', 'AirTemp', 'Humidity', 'TrackTemp']
cat_f = ['Compound', 'Session', 'FreshTyre', 'Stint', 'Rainfall', 'Driver', 'Year']
```

Figure 13 : Original Features

Metric	Training	Test
MSE	0.05	0.05
RMSE	0.22	0.22
MAE	0.17	0.18
R^2 Score	0.17	-0.05

Table 1: Results from machine learning model 1

5.2.3 Version 2 – Additional Feature Engineering

Additional Feature engineering was needed to model the relationships between the features in each column (**Fig.14**). It included the temperature difference between the track and the air, how the humidity and the track temperature interacted with each other and the average tyre life of the compound.

This would help the ML model to learn the interactions between the features and help better predict the target variable (Delta).

Furthermore, the evaluation of CV score shifted the focus from MSE to R^2 score to capture the variance in delta, doing so allows for an improved MAE, MSE and MAE values which allowed for a more accurate model performance (**Table 2**).

```
df_filtered['TrackCondition'] = df_filtered['Humidity'] *
df_filtered['TrackTemp']

df_filtered['Temp_Range'] = df_filtered['TrackTemp'] - df_filtered['AirTemp']

df_filtered['Compound_FreshTyre'] = df_filtered['Compound'].astype(str) +
df_filtered['FreshTyre'].astype(str)

df_filtered['Avg_TyreLife'] =
df_filtered.groupby('Condition_Compound')['TyreLife'].transform('mean')
```

Figure 14: Enhanced Feature engineering

Metric	Training	Test
MSE	0.02	0.04
RMSE	0.013	0.021
MAE	0.011	0.015
R^2 Score	0.04450	-0.01185

Table 2: Results of machine learning model 2

5.2.4 Version 3 – Refining the filter and change of CV method

Further improvements were made by refiltering the delta between -0.1 and 0.5 seconds, instead of the former -0.5 to 0.5 range, to more accurately represent meaningful tyre performance.

Additional changes were made to the features, I felt that the feature engineering in model 2 did not show the true complexities of the interactions between features and needed altering by using previous historical data - this would be an attempt to capture the variability in the data and enhance the accuracy of the predicted values. Furthermore, to reduce computational expense, group K-Fold was used instead of Leave One Out Cross Validation as it produced irregular R^2 values.

Using Group K-Fold (with 8 splits) also allows for all the datapoints to be grouped together and in equal sizes, which would increase the accuracy of the model while still avoiding data leakage. Overall changes to the model should capture the tyre performance better.

Finally, further optimisation is made by retesting the set with RandomSearchCV (parameters seen in **Appendix D.2**, this will find the optimal parameters and return better results. The introduction of predicted deltas against residuals was also used (**Fig.15**) to see the outcome of the results in a visual form as well as identify any overfitting/systematic issues.

```
    for prev in range(1, prev + 1):

        df_filtered['TyreWearRate'] = df_filtered.groupby(['Driver', 
'Session'])['Delta'].shift(prev).cumsum() / df_filtered['TyreLife']

        df_filtered[f'TrackCondition_Prev_{prev}'] = 
df_filtered['TrackCondition'].shift(prev)

        df_filtered[f'Delta_TrackTemp_Prev_{prev}'] = df_filtered['Delta'].shift(prev) * 
df_filtered[f'TrackCondition_Prev_{prev}']

        df_filtered[f'Delta_LapChange_{prev}'] = df_filtered['Delta'].shift(prev) - 
df_filtered['Delta'].shift(prev + 1)


    df_filtered = df_filtered.ffill().dropna()

    return df_filtered
```

Figure 15: Additional Feature engineering

Noting that `.ffill()` and `.dropna()` was used to drop any empty fields during the rolling window as it utilised previous values. This to fill empty values with the previous observed value, if there is no last valid observed value, it will be dropped from the data frame; doing so allows for null values to be filled in with the relevant data as opposed to ignoring it altogether and reducing the number of datapoints.

Additional model improvements did see positive results (**Table 2**). Despite this, the R^2 Score had a high difference which suggests underfitting, it should be noted that delta is a very complex variable that requires variable than just the available data. Though R^2 Score is not fully representative as MAE, RMSE and MSE are more critical metrics for evaluation.

However, after this experiment, it was noted that there were issues with the residual data (**Fig.16** shows an unexplainable line). This suggests that the results are not accurate, indicating that the experiment needs to be repeated with adjustments in data processing to ensure more accurate residuals.

Metric	Training	Test
MSE	0.0119	0.0181
RMSE	0.1092	0.1324
MAE	0.0790	0.0965
R^2 Score	0.4086	0.1310

Table 3: Results of machine learning model 3

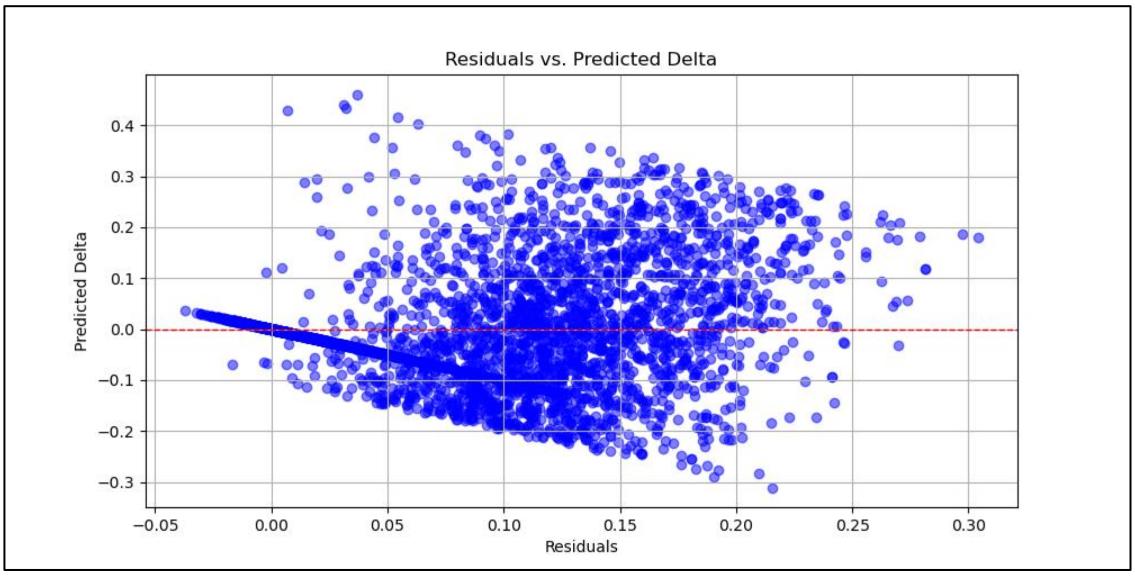


Figure 16: Model 3 residual graph

5.2.5 Version 4– Further filtering and parameter adjustments

Further improvements were made by refiltering the datapoints to -0.2 to 0.2 seconds as this is where the majority of valuable datapoints lie (see **Appendix D.3**). Furthermore, datapoints where the TyreLife = 1 were removed as the first lap of the stints are not representative of tyre performance; this reduced the number of 0 values drastically and removed unnecessary noise (see **Fig.17**).

Finally, further optimisation measures were taken to ensure the maximum performance is extracted from the model as the use of Bayesian optimisation using the Optuna framework wase more beneficial (see **Appendix D.4**) as opposed to RandomSearchCV to find parameters that produced a more accurate outcome.

Further improvements included optimisation of feature engineering by using a standard rolling window to balance simplicity and complexity while capturing variance. (see **Appendix D.5**). This model will be utilised in the test because of time constraints.

Though there is some massive underfitting shown in **Table 4,** the elimination of systematic errors meant that excessive noise no longer influences the model, some improvements in areas such as MAE for test prove this, although this is possibly due to the influence of the stricter delta filtering. **Fig.17** does show that the there are no systematic errors which does suggest that the noise has been removed which was influencing results previously.

Metric	Training	Test
MSE	0.0105	0.0119
RMSE	0.1026	0.1089
MAE	0.0872	0.0928
R^2 Score	0.1219	0.0127

Table 4: Results from machine learning model 4

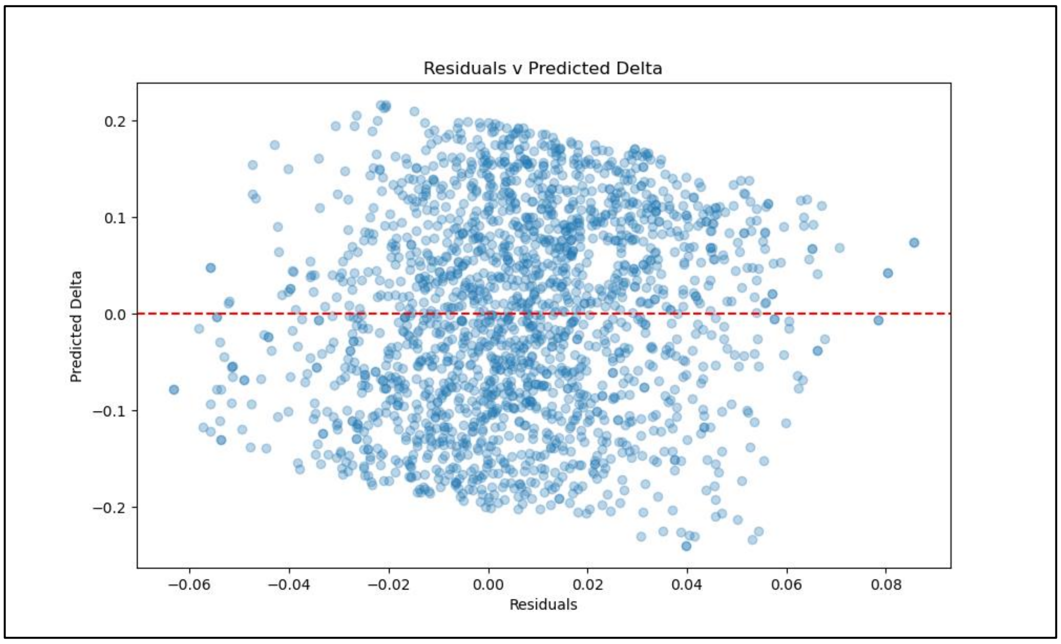


Figure 17: Model 4 residual graph

5.2.6 Final Model - Overhaul (Final Iteration)

An entirely new approach to the ML model was then considered by altering method of splitting the training and test sets for better generalisation.

Instead of using all circuits and excluding an entire circuit, all team and circuit data from 2022-2023 (excluding Ferrari) is used to allow the model to learn the patterns of delta and the track conditions better.

Feature importance conducted in **Fig.18** showed that numerous features that were related to tyre wear were essential, however the removal of all features in `cat_f` list but Compound was necessary to ensure that the model is not being hindered by unnecessary noise/features that had contributed nothing to the model.

"Session" variable, though potentially useful, acts as a categorical identifier rather than a true predictive feature. Including it in the model could lead to overfitting, where the model memorises session-specific patterns instead of learning generalisable trends. By omitting "Session," the model focuses on more meaningful relationships, improving its ability to generalise to new, unseen data. Driver was removed completely from the features due to the similarities between all of them/provided no added importance as seen in **Fig.19**, thus adding noise that may negatively affect the model

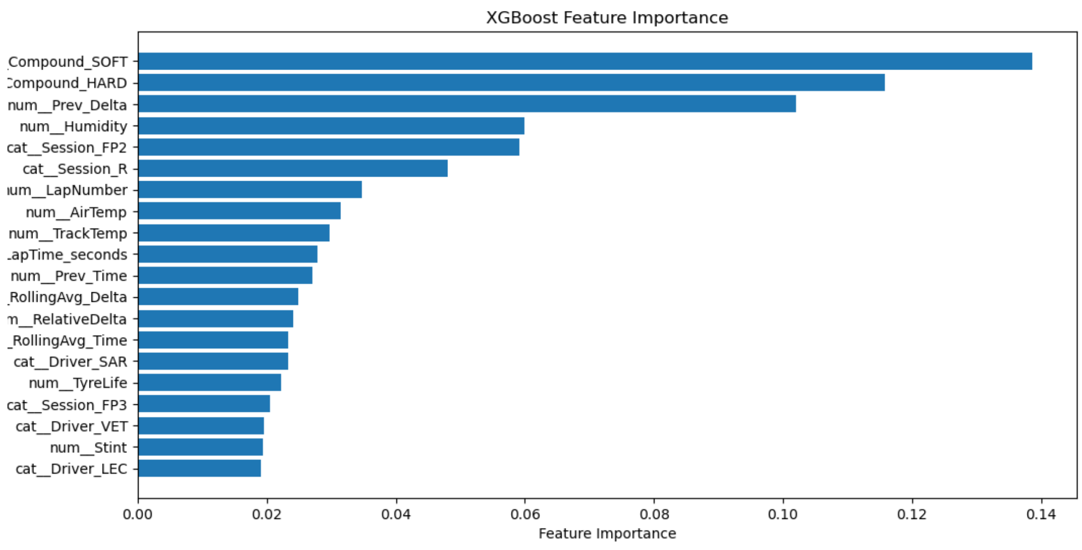
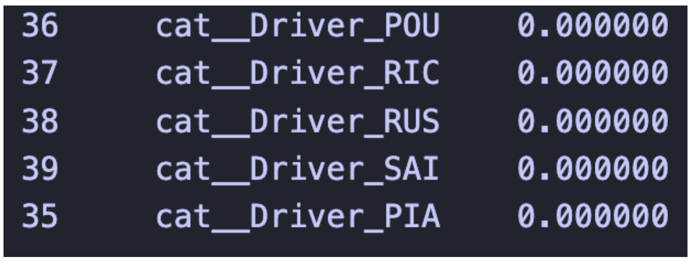


Figure 18: Feature Importance



```
36      cat__Driver_POU     0.000000
37      cat__Driver_RIC     0.000000
38      cat__Driver_RUS     0.000000
39      cat__Driver_SAI     0.000000
35      cat__Driver_PIA     0.000000
```

Figure 19: Driver features that provided no usefulness in the model

All Ferrari data from FP1, FP2, FP3, and sprint sessions from 2022-2025 will be used for the unseen test set, while race data will be limited to the 2022-2024 season (when available); this ensures that the ML model is not overfitting by seeing the data before predictions via data leakage.

All the separate Ferrari data will be track specific i.e.:

- ferrari_australian_sessions.csv
- ferrari_bahrain_sessions.csv

They will contain the FP1,2,3 sessions (2024-2025) and if occurred, the sprint race sessions as well, doing so will increase R2 score and decrease the other evaluation metrics.

Notably, the rest of the current ML model will remain the same with the created features added in and the removal of the Group K-Fold in favour of a time-split (see time-split in **Appendix D.6**) and hyper-parameter tweaks to fit the refinement of the dataset via Bayesian optimisation. Scaling and normalisation to the `num_f` features also were added to increase the overall test score. Time-split was more appropriate for a problem like this as the data is sequential, previous lap performance may influence the next lap i.e. a mistake causing extra degradation can affect the tyres in the next lap.

Adopting this over LOOCV and Group K-Fold ensures that the model accounts for the temporal dependencies within the data, providing a more realistic representation of the dynamics at play during a race.

Furthermore, the preprocessing step of the pipeline is further refined from 'passthrough' to normalising as much of the data as possible (**Fig.20**), because normalisation ensures that all features contribute equally to the model, preventing any single feature from disproportionately influencing the results.

By standardising the range of the data, especially for features like num_f, the model can learn more effectively, as it removes any bias caused by differences in scale. This step is crucial for improving the model's performance, as it allows for better convergence during training and can lead to a more robust and accurate predictive model.

```
preprocessor = ColumnTransformer([

    ('one_hot', OneHotEncoder(handle_unknown='ignore'), cat_f),

    ('robust', RobustScaler(), ['LapTime_seconds', 'RollingAvg_Time', 'RollingAvg_Delta',

                                 'Prev_Time', 'Prev_Delta',
'RelativeDelta','LapTime_TrackTemp']),

    ('standard', StandardScaler(), ['TrackTempMean', 'TrackTemp', 'AirTemp',
'Humidity','Delta_TrackTemp']),

    ('passthrough', 'passthrough', ['LapNumber', 'TyreLife', 'Stint', 'AvgTyreLife'])

])
```

Figure 20: Enhanced Preprocessing pipeline

This version will be used in the final testing of the model as seen in **Section 6.1.1**

5.2 Monte Carlo Simulation

5.2.1 Initial Proof of Concept

Initial development followed the core ideas outlined in the methodology, with the prototype only using hard coded data, however this acted more of a proof of concept as it does validate the equations mentioned in **Section 4.4.2**

Prototyping the MCS allowed for the predicted data and dataset values to be utilised. **Appendix D.7** shows a prototype which only predicted the pit window and average lap time on the stint. A graph was also plotted, plotting how far the drivers could go on that tyre in a single stint. Though this model is not the intended final artefact, the simulation proves the concepts of the equations mentioned in **Section 4.4** regarding the mathematical generation of lap times for each compound as seen in **Fig.21** and **Fig.22**. This was a base outline for the final product where 10,000 simulations would be run, then the optimal lap times would be selected for each tyre in the stint. This does show a good idea of tyre degradation trends on the track, however generating a strategy that showcases the overall race (which is originally intended) would require a different approach.

Key Points drawn from initial product:

- HARDTrue compounds are close, though the pit windows and the average times on other compounds need work despite being faster.
- Difference between the baseline lap and the lap they are due on each compound to pit is ±6 seconds is not realistic and should be within 2-3 of baseline seconds before pitting.

```
lap_time = lap_times[i-1] + deg[i] * deg_factor
```

Figure 21: Lap time generation implementation

```
deg_factor = (1 + (i / xTmax)) + (track_temp / track_temp_mean)
```

Figure 22: DegFactor equation implementation

Finding a solution to mitigate the large difference between the baseline lap time and the pit in lap time was the way the *pit in* decision was programmed. It utilised only the condition of the stint threshold, meaning the pit decision will only be made once that condition was exceeded.

Even though the stint length was realistic for the three tyre compounds, the constantly increasing lap time between the pit in time and the baseline time was unrealistic. Therefore, an additional condition where if it exceeded xT_{max}, the pitstop is forced as seen in **Fig.23**.

```
if stint_laps >= xTmax or lap_time >= stint_threshold:

    print(f"Pit stop required for {current_compound} at lap {i+1} (stint laps: {stint_laps}, xTmax: 
{xTmax}, lap time: {lap_time}, max lap time: {max_lap_time})")

…
```

Figure 23: Updated pit decision condition

5.2.2 Improved Model

Stint Loop

Though the current code shows the current projection of the tyre stint over the course of the race, the current approach is not what was initially thought. The improved prototype implements the core functionality of the equations provided in **Section 4.4** but shifting the focus of the core race simulation logic to simulating stints within a race simulation as mentioned in the artefact design, this is iterated until the full race distance is completed as per the loop shown in **Fig.24**.

Doing it this way allows for the simulation of each individual stint until the xT_{max}. When that stint ends, where it exceeds the condition noted in **Fig.25**, a pitstop is *simulated* (**Fig.26**) which signifies the start of a new stint.

```
while total_laps < num_laps:

        remaining_laps = num_laps - total_laps
```

Figure 24: Race distance tracker

```
while (len(lap_times) <= adj_xT_max and total_laps + len(lap_times) < num_laps):
```

Figure 25: Stint continues until exceeding the expected tyre life or race distance.

```
if total_laps > 0:

    lap_times[0] += pit_stop_time

    total_pit_stops += 1
```

Figure 26: Begin stint by adding pit stop time to lap time and pit counter

Tyre Selection

The tyre selection in **Fig.27** checks if the compound chosen can last the remainder of the race, if it does not a random compound is chosen. A specific tyre will not be chosen until the if condition is met, this provides flexibility in the model, allowing to try and test different tyre choices, and find the optimal tyre. Once the tyre is selected, the stint counter will be incremented by one, this determines how many tyres (or different compounds) are used throughout the race.

```
viable_compounds = [

          comp for comp in df['Compound']

          if xT_max[comp] >= remaining_laps

          ]

          if not viable_compounds:

                    viable_compounds = list(df['Compound'])


          #pick any compound

          compound = np.random.choice(viable_compounds)

          stint_counter[compound] = stint_counter.get(compound, 0) + 1)
```

Figure 27: Tyre Selection Logic

5.2.3 Min-heap strategy selection

Strategies generated in each race simulation are stored in lists, these are then accessed in the sorting.py file by passing the argument `sort_strategies(pit_stop_count, total_race_times, pit_stop_strategies)`. Both pit_stop_count and total_race_times are zipped to create tuples and then sorting by fastest race time by identifying the second element in the tuple of each strategy index, then ordering them as seen **Fig.28**.

```
Race_strategy = [(pit_stops, race_time) for pit_stops, race_time in zip(pit_stop_count, total_race_times)]

sorted_by_fastest = sorted(range(len(race_strategy)), key=lambda idx: race_strategy [idx][1])
```

Figure 28: Creating tuples, then sorting by fastest total race time

Despite not being explicitly stated, each index has the strategy attached to it. This will be incorporated into the tuple in **Fig.29**, which will then be passed to filtering out the strategies that do not follow the two-compound rule.

```
for idx in sorted_by_fastest:

    sorted_strategy = tuple(strat_no for strat_no, _ in pit_stop_strategies[idx])

    if len(set(sorted_strategy)) < 2:

        continue
```

Figure 29: Filter to fit two compound regulation.

All sorted strategies are then identified as unique or not through the rounding and cumulative (**Fig.30**), where it will then be stored in a dictionary, the code in **Fig.31** shows the unique strategies being selected and if the strategy already exists, it will compare race times, if the new race time is faster, the old one will be replaced. When all strategies are selected and the fastest times are selected, a min-heap sort is done using the heapq python package to determine the four fastest strategies.

```
for _, stint_laps in pit_stop_strategies[idx]:

            cumulative_sum += stint_laps

            pit_lap_distribution.append(cumulative_sum)


unique_key = (sorted_strategy, tuple(round(lap / 10) * 10 for lap in pit_lap_distribution))
```

Figure 30: Cumulative and rounding to identify unique strategies

```
if unique_key not in unique_strategies or total_race_times[idx] < total_race_times[unique_strategies[unique_key]]:

            unique_strategies[unique_key] = idx


selected_indices = heapq.nsmallest(4, unique_strategies.values(), key=lambda idx: total_race_times[idx])
```

Figure 31: Select fastest unique strategies and store in heap.

Appendix C.2 explains min-heap sort for further clarity and understanding.

Visualising Strategy
The main visualisation would be the visualising the ‘race story’ or the strategy using stacked horizontal bar charts to show the progression of the race as well as visualising the lap times in a race trace. **Appendix E.1** shows the outputs of the functions below.

A set of visualisation functions were made to help look at and compare different pit stop strategies. `display_strategy` prints out a breakdown of each picked strategy, showing the total race time, when the ideal pit in lap it, and how each stint performs across the race (see **Appendix E.1**).

`plot_strategies` function is used to make a visual representation of tyre stints and pit windows using horizontal bars, making it easier to see how the strategy plays out during the race (see **Appendix E.2**).

`plot_race_trace` creates a lap-by-lap comparison of lap times for different strategies, showing performance trends and how pace evolves in each stint (see **Appendix E.3**). These tools help make it clearer what strategies work better/how it measures up to the predicted and where improvements could be made.

5.3 Integrating Machine Learning with MCS

To integrate the ML within the MCS, the model was saved as a .pkl file (**Fig.32**), which allowed for easy portability and reusability within the system, this was then loaded into the simulation, where it would be stored into a dictionary for efficient access after calling the function (**Fig.33**). To make up for the class imbalance for FALSE/TRUE versions of the tyre compounds, the compounds were grouped together to SOFT, MEDIUM and HARD, which helped streamline the process and made the integration of the predicted values more manageable. This decision to group the compounds simplified the overall structure, allowing the model to function more smoothly within the MCS framework.

```
#load the trained model

def load_model(model_path="src/xgb_pipeline_model.pkl"):

    print('loading model...\n')

    with open(model_path, "rb") as f:

        loaded_model = pickle.load(f)

..preprocessing

…predictions

    return loaded_model
```

Figure 32: Loaded Pkl file, in model.py, run predictions.

```
    predicted_tyre_data = tyre_model(unseen_csv)

    print('loading tyre data...\n')

    df = pd.DataFrame({

        'Compound': list(predicted_tyre_data.keys()),

        'Predicted Mean': [predicted_tyre_data[key]['Predicted Mean'] for
key in predicted_tyre_data],

        'Predicted Std': [predicted_tyre_data[key]['Predicted Std'] for key
in predicted_tyre_data]

    })
```

Figure 33 : Call model.py function and store in dictionary in sim.py

5.3.1 Initial Testing and Discovery

Upon initial testing, the lap times (as shown in Appendix E) where not as accurate as they looked. Therefore, additional improvements and additions that were added into the code to closely emulate the lap times closer compared to the initial testing, the most notable being:

- The inclusion of dirty/clean air was not implemented, slowly constant decreasing lap time trends were not realistic, they should be lowly increasing with improvements occurring every now and then.
- Heavy reliance on xT_{max} (maximum expected tyre life) was then noted as the pit stint lengths were only going as far as meeting the maximum tyre life as opposed to meeting the $StintThreshold_{max}$.

- Furthermore, graphs generated for some circuits such as the Australian Grand Prix (**Fig.34**) suggested that the pit time was not being added correctly, showing a very short pit stint added.

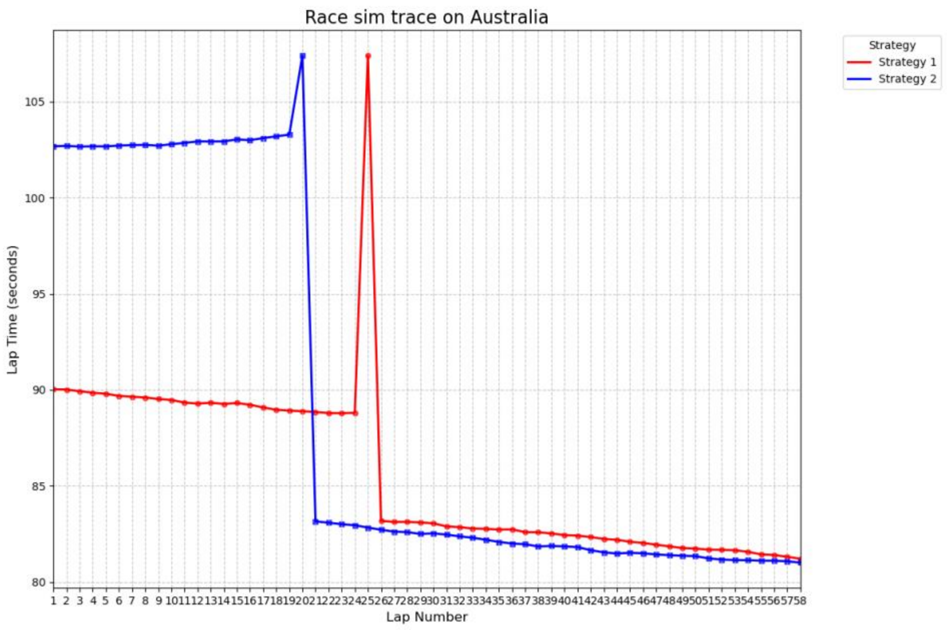


Figure 34: Original Visualisation

5.3.2 Improved Lap time generation

The `weighted_avg` function calculates a weighted average of lap times, giving greater importance to more recent laps. This is achieved by applying exponentially decreasing weights, where the most recent lap times have the highest influence on the final average.

The lap times are arranged in ascending order, weights are then assigned using an exponential decay function, meaning that as lap times become older, the less influence is given to the mean. Using `alpha`, the `np.exp` function computes the weighted average, ensuring that recent lap times contribute more significantly, while older laps have progressively less impact. `alpha` is negative though; ensuring that the significance given to later lap times decreases.

This method provides a more accurate reflection of a driver's current performance by prioritising recent lap times. Doing so mitigates the bias of outliers that increase the lap time and balances between the fastest possible race time and the slowest race time, **Fig.35** shows the code snippet that ensures accurate driver performance – this is applied to both race and practice times shown in **Fig.36**. Through multiple tests, a value of 0.8 is the optimal value which balances the fastest and slowest lap times to create an ideal mean baseline lap time.

```
#weighted avg

def weighted_avg(lap_times, alpha=0.8):

    lap_times = np.sort(lap_times)

    weights = np.exp(-alpha * np.arange(len(lap_times)))


    return np.average(lap_times, weights=weights)
```

Figure 35: Weighted average of lap times in data set.

```
#weighted avg to both fp and race

compound_weighted = (r.groupby('Compound')['LapTime_seconds'].apply(lambda x:
weighted_avg(np.array(x))).to_dict())

race_compound_weighted =
(r_filtered_race.groupby('Compound')['LapTime_seconds'].apply(lambda x:
weighted_avg(np.array(x))).to_dict())
```

Figure 36: Weighted average distributed to other variables.

5.3.3 Refinements: Reliance on xT_{max} values

It was then noted that artefact over-relies on xT_{max} and ignored $StintThreshold_{max}$, further experimentation had also concluded that the use of $StintThreshold_{max}$ forced pitstops too many times forcing it to be unrealistic. This issue was not identified until later during testing as it turned out that the $StintThreshold_{max}$ was not even being used as all stints were meeting the xT_{max} conditions but not the $StintThreshold_{max}$.

Reasoning behind this was that the $StintThreshold_{max}$ variable was too large to be met, due to the way it was programmed, and if it was changed to a certain threshold like 0.5s total lap time loss, numerous pitstops could occur, many of which would be unnecessary in real life. To solve this issue, xT_{max} will become dynamic, changing upon temperature changes as seen in **Fig.37**.

The maximum tyre life `xT_max` is adjusted based on track temperature changes. First, the difference between the current track temperature and the average `temp_change` is calculated. This difference is then scaled using a fixed 0.25 factor, this is just an assumption that needs to be noted. The adjusted tyre life is then found by subtracting this scaled value from the original maximum tyre life for the selected compound. Finally, the value is converted to an integer to keep calculations consistent.

It is assumed that higher track temperatures reduce tyre life, while lower temperatures extend it (Ambler, 2024b), which is logical as high temperatures increase tyre degradation and make it difficult to keep the tyres in the optimal temperature window without overheating. Whereas lower temperatures decrease tyre degradation, which can extend the `xT_max` as it prevents the tyres from overheating.

```
factor = 0.25

temp_change = track_temp - track_temp_mean

adj_xT_max = xT_max[compound] - temp_change * factor

adj_xT_max = int(adjusted_xT_max)
```

Figure 37: Track temperature affecting the tyre

5.3.4 Dirty/Clean Air Integration

The improved model now utilises the probability of running in clean or dirty air logic, which affects lap time trends during a stint (**Fig. 38**). If the scenario meets the 9% chance, the lap time improves due to reduced degradation, otherwise it forces the `deg` (or $Delta$) to be positive, meaning lap times increase.

Further refinements were relatively minor such as conversion of lap times to allow for easier comparison when validating the model.

```
if np.random.rand() > 0.09:

       deg = abs(deg)
```

Figure 38: Dirty/Clean air logic

6. Testing and Evaluation of Artefact

6.1 Artefact Testing and Evaluation

Full system testing is conducted on Python 3.9.18 alongside the following package versions:

- Scikit learn 1.2.2
- NumPy 1.23.5
- xgboost 1.7.3
- Pandas 2.2.3
- Matplotlib 3.9.2

These were the latest version at the time of testing and are compatible with the current system framework. Every test conducted was done under these configurations for consistency and reproducibility of results. Furthermore, this final test does include the refinements and improvements suggested in **Section 5.3.3**.

6.1.1 Machine Learning

Testing was done during development, with constantly improving iterations of the model. The final model below showcases the best version of the model. The test set for these results was the 2024 Australian Grand Prix, with data between 2022-2024 and free practice data as well as the race/sprint races that may have taken place on that weekend.

Other versions of the model that were with results can be seen in **Section 5.2.6**

Though the results in **Table 5** may not be as satisfactory, the vast improvement in MSE, RMSE, MAE and R^2 scores shows significant improvement due to the refined approach and feature engineering. The larger training dataset in **Model 5** allows the model to generalise better, leading to improved error metrics. The improvement in R^2 and the increased training data contribute to a more reliable and robust model overall, though room for improvement is still there to close the gap between the training and test metrics and allow to model to generalise better and mitigate underfitting.

Metric	Training	Test
MSE	0.0066	0.0108
RMSE	0.0815	0.1039
MAE	0.0692	0.0861
R^2 Score	0.4516	0. 1587

Table 5: Results from machine learning model 5

Systematic errors were not an issue as shown in **Fig.39**, however the complex target variable was very difficult to predict, though an improvement, more steps could be done to mitigate the overfitting further and improve generalisation.

This is further proved in the residual distribution in **Fig.40** and the actual v predicted graph in **Fig.41**, where the distribution is a very rough bell curve, though with a somewhat symmetric distribution indicating no model bias in the predictions. The absence of heavy skewness or heavy tails suggests that the model is capturing the general structure of the data effectively. and the comparison graph also shows the large spread of points with a handful of predicted points around the area of the ideal line, suggesting that improvements and further fine tuning can be made.

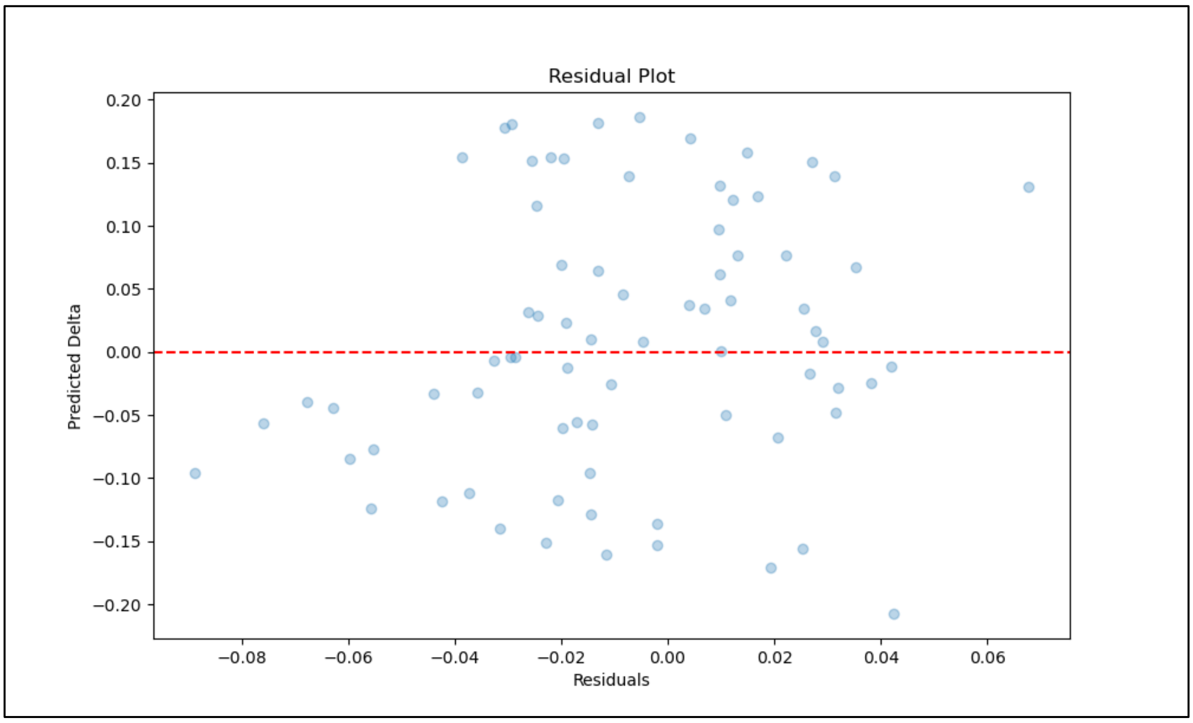


Figure 39: Residual Plot

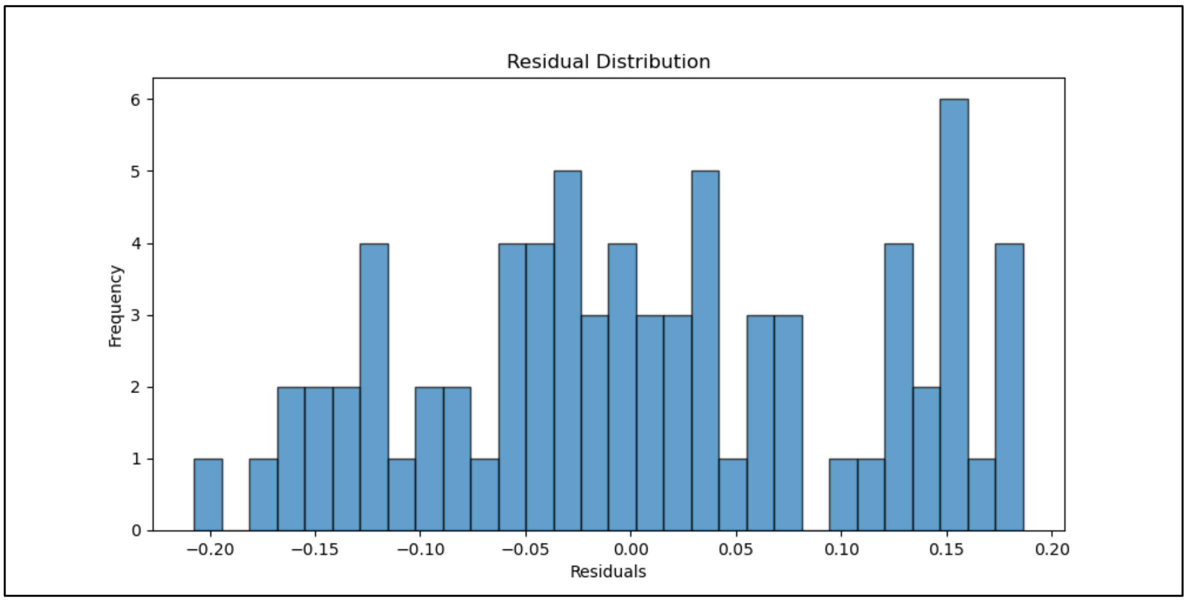


Figure 40: Residual Distribution

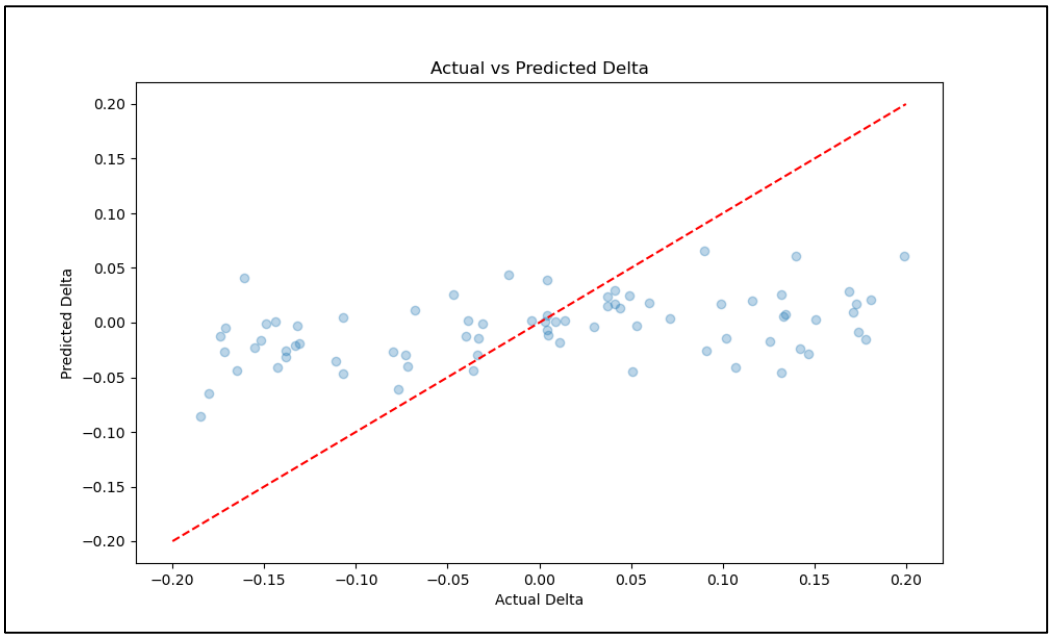


Figure 41: Actual v Predicted Data

True comparison between the actual data and the predicted data in addition to the metrics can be seen in the histogram in **Fig.42**. As all the points lie within half the range of the intended target delta, it does suggest that the model had some bias, favouring the mean around 0, because of either class imbalance for each compound or heavy regularisation.

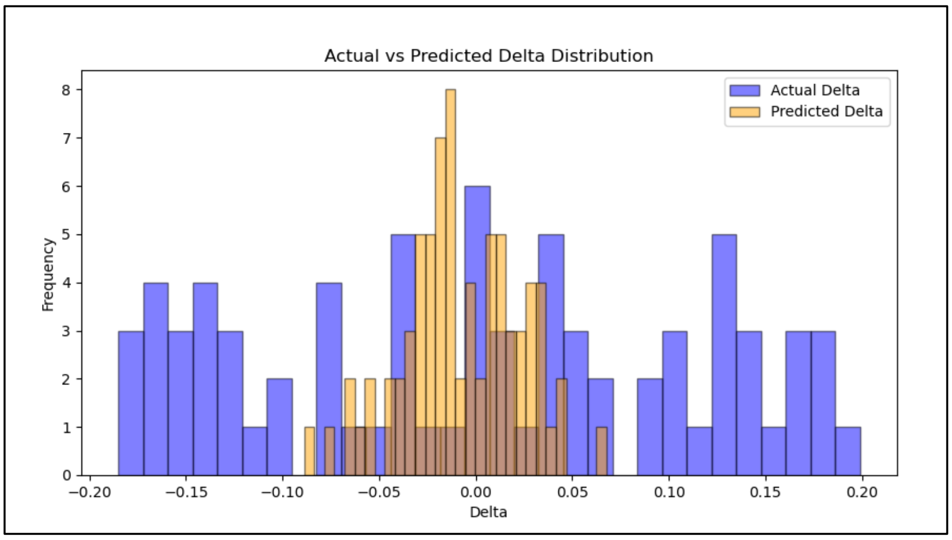


Figure 42: Actual v Predicted Delta Distribution

Due to the time constraints of this project, the ML model has been developed to a functional state, but further growth is needed to fully enhance its ability to fully capture the variance of the data. While the current model provides useful insights and predictions, there are areas that require refinement, such as incorporating more complex features, improving accuracy through further optimisation via Optuna. Alternative models could also be considered if provided more time to research such as LGBM (Light Gradient Boosting) or RF – though other solutions can include stacking/blending ensemble models to capture the variance better.

Given more time and resources, the model could be expanded and refined to better address the dynamic and real-time factors that influence race strategy.

6.1.2 MCS Test with Integrated Simulation

The actual 2025 Australian Grand Prix cannot be a valid comparison due to the constant interchangeable conditions that took place, and since this model does not factor in wet conditions. It resulted in completely skewed lap times, not showing an accurate representation of the accuracy of the model. Therefore, the comparison was made with the 2024 Australian Grand Prix.

To test the model, the histogram in **Fig.43**, suggests that some of the lap times simulated were faster than the actual simulated race data, given that most of the simulated times fell below of the that year's race it does suggest that may be overestimating the lap times, though it does not include the VSC factor as that was deployed at the end of the race which the simulated model does not consider. This same reasoning can be applied to the metrics in **Table 6**, though the lower variability in the model could be due to the minuscule of effect of the implemented probabilistic influences i.e. dirty/clean air, while actual races experience more fluctuation.

Section 6.2.1 elaborates on the same reasoning further in the context of strategy and simulated race outcomes.

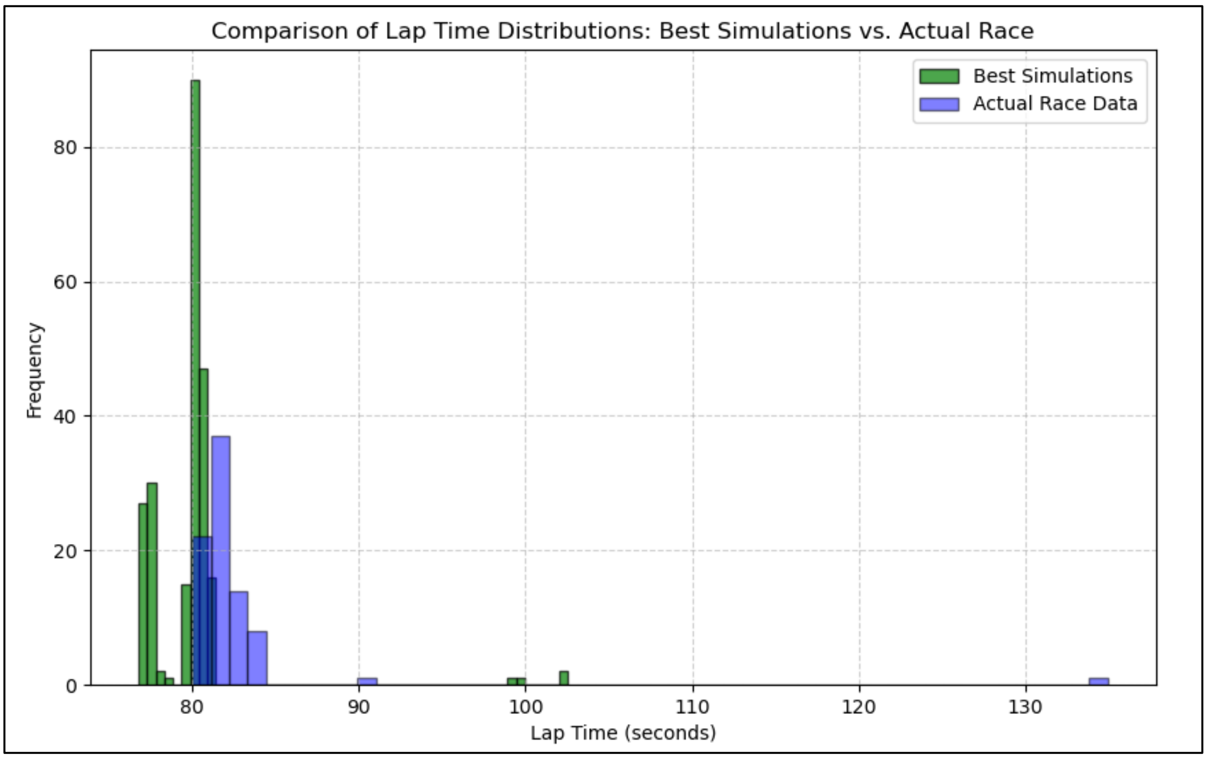


Figure 43: Actual v Predicted Lap time distribution

Metrics	Simulated Lap times (s)	Actual Lap times (s)
Q1	78.096	81.090
Mean	79.914	82.542
Q3	80.469	82.410
Std Dev.	3.113	5.946

Table 6: Australian Grand Prix Metrics Comparison

6.2 Model Results and Analysis

Results will be compared to the predicted strategy from Pirelli, or with the race strategies employed during the race, this is taken from the armchair-strategist.dev site. It is expected that the simulation race times will have a be expected to be within a ±5% error margin because of the unpredictable events that cannot be accounted solely from data alone such as when exactly will the safety car come out; though this model does not account for all conditions therefore results will rely on all dry races.

For comparison, the simulation race time will be benchmarked against the fastest Ferrari race time. If this is unavailable, the winner's race time will be used instead. Please note that various factors, such as the deployment of Safety Cars or rain, may have influenced the actual race times, which means the total race time in the simulation could differ from the actual race time.

All actual total race time data shown in tables are taken from https://www.formula1.com

6.2.1 Race time Comparison

Australian Grand Prix

The sim predicted that the race times would be faster than the actual race time. Despite the differences, the late race VSC had an impact on lap times at the end of the race, therefore, dry lap times without the VSC means that the simulation time could be even better than the -3.98% average error margin suggests.

Strategy	Sim Race Time	Actual Race Time	Sim Difference to actual in seconds +/-	Error Margin (%) +/-
1	01:16:51.352	1:20:26.843 Carlos Sainz	-215.491	-4.46
2	01:17:00.406		-206.437	-4.28
3	01:17:32.799		-174.044	-3.61
4	01:17:35.440		-171.403	-3.55

Table 7: Australian Grand Prix Simulation Results Comparison

China 2025

Note that both Ferrari drivers were disqualified at the end of the race, therefore Esteban Ocon's race time was used for comparison as he finished behind of both Ferrari drivers. Comparisons made to the recent 2025 Chinese Grand Prix shows the true potential of the ML Predicted system regarding race time. All strategies produced were within 3% of the actual race time as the average error margin was 2.53% slower than the actual race time, proving the validation of the equation to some extent. This result highlights the accuracy the model can be despite the limitations of data, even when external factors such as rain or Safety Cars do not interfere with the race.

Strategy	Sim Race Time	Actual Race Time	Sim Difference to actual in seconds +/-	Error Margin (%) +/-
1	1:33:58.971	1:31:44.995 Esteban Ocon	+133.976	+2.43
2	1:34:09.223		+144.228	+2.62

Table 8: Chinese Grand Prix Simulation Results Comparison

6.2.2 Pit strategy Comparison

Pit stop strategy comparison is done by either comparing all drivers from the race or the predicted Pirelli strategy, regardless of the conditions that occurred. It is important to reiterate that the simulation heavily depends on dry, ideal conditions while factoring possible influences that affect the stint length. Therefore, analysis will only be done where valid dry stints are done i.e. laps are completed according to the standard tyre life lengths as suggested by Blackcircles (2023):

- SOFT – 20-30 Laps
- MEDIUMS – 30-40 Laps
- HARD – 40-50 laps

If actual race strategies are available, they will be visualised with https://armchair-strategist.dev, this provides useful insights and allows for the actual race and simulation to compare strategies visually.

Australia 2025

Due to rain in the real-life race, it was not possible to predict the dry stints, thus relying on the predicted strategies Pirelli generated is the best available method for analysing dry race conditions. Where Pirelli predicted two stop strategies all round, except for strategy 1, the simulation predicted one stop strategies only.

Assuming that the race is dry and in ideal racing conditions with no interruptions, the windows of the simulations remain realistic, a SOFT stint lasting anywhere between 6-12 laps in **Fig.44**, compared to the simulated strategy where it can last between 7-12 laps using Strategy 3 in **Fig.45**.

Simulated Strategy 1 have similar pit windows compared to Pirelli's Strategy 1, implying that the tyre wear and stint lengths in the simulation closely match Pirelli's projections for an optimal dry-race strategy. Though other predicted strategies such as the one stop suggested by the simulation are still possible to minimise pit stop loss unlike the other strategies provided by Pirelli.

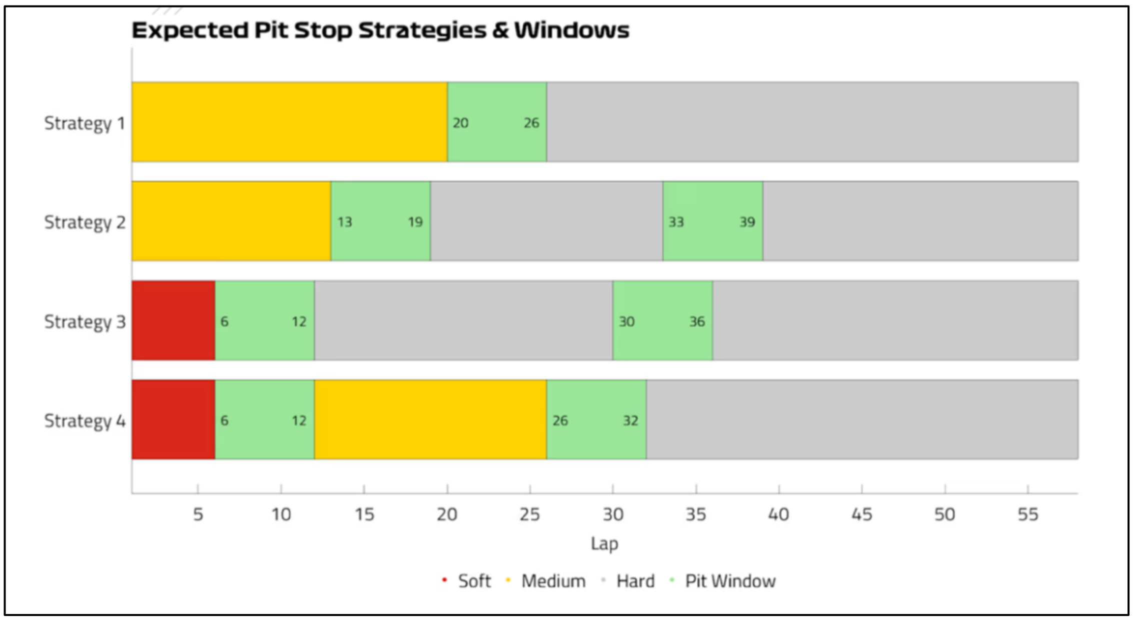


Figure 44: Expected Pit Strategy Prediction (Youson, 2025)

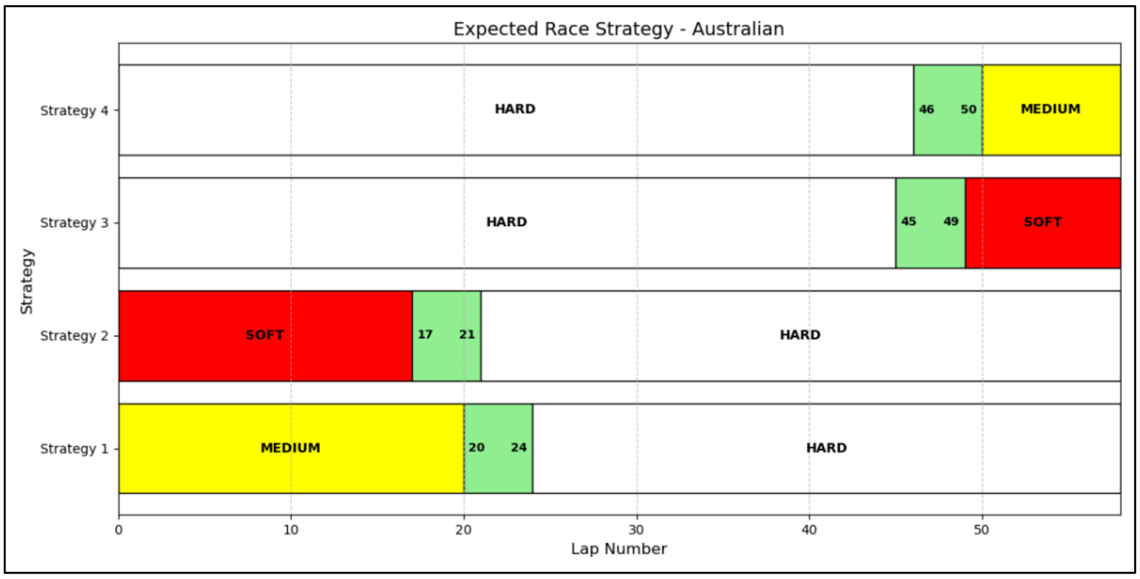


Figure 45: Simulated 2025 Australian Grand Prix Pit Strategies

China 2025

As mentioned in **Section 6.2.1**, the simulation (**Fig.48**) accurately predicted the number of stops relative to the actual race (**Fig.47**). The two stop was not the optimal strategy in the actual race as Pirelli predicted as seen in **Fig.46**; the simulation, in comparison, predicted a one-stop strategy which many drivers during the race opted for.

However, the MEDIUM compound stints had not lasted as long as the simulation thought - where the simulation had predicted a pit window of lap 22-24 in Strategy 2, many drivers did not get far on that set of tyres, with the exception being Nico Hulkenberg (HUL) and Alexander Albon (ALB) who did 20 laps on the MEDIUM tyres which is close to the pit window of Strategy 2 in the simulation.

As for the HARD compound tyre, many drivers did have stints that lasted between 30-40 laps which the sim expected. Interestingly, Lance Stroll (STR) did 36 laps on the HARD compound tyre before continuing to finish the race on MEDIUM compound tyres, which falls within the close to the expected pit window of Strategy 1.

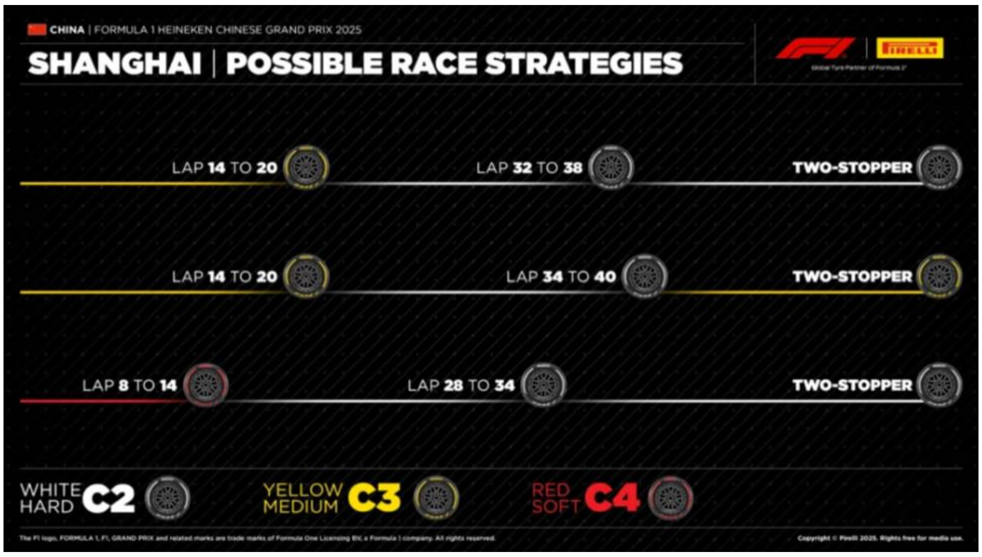


Figure 46: Pirelli Expected Pit Strategy (RaceStaff365, 2025)

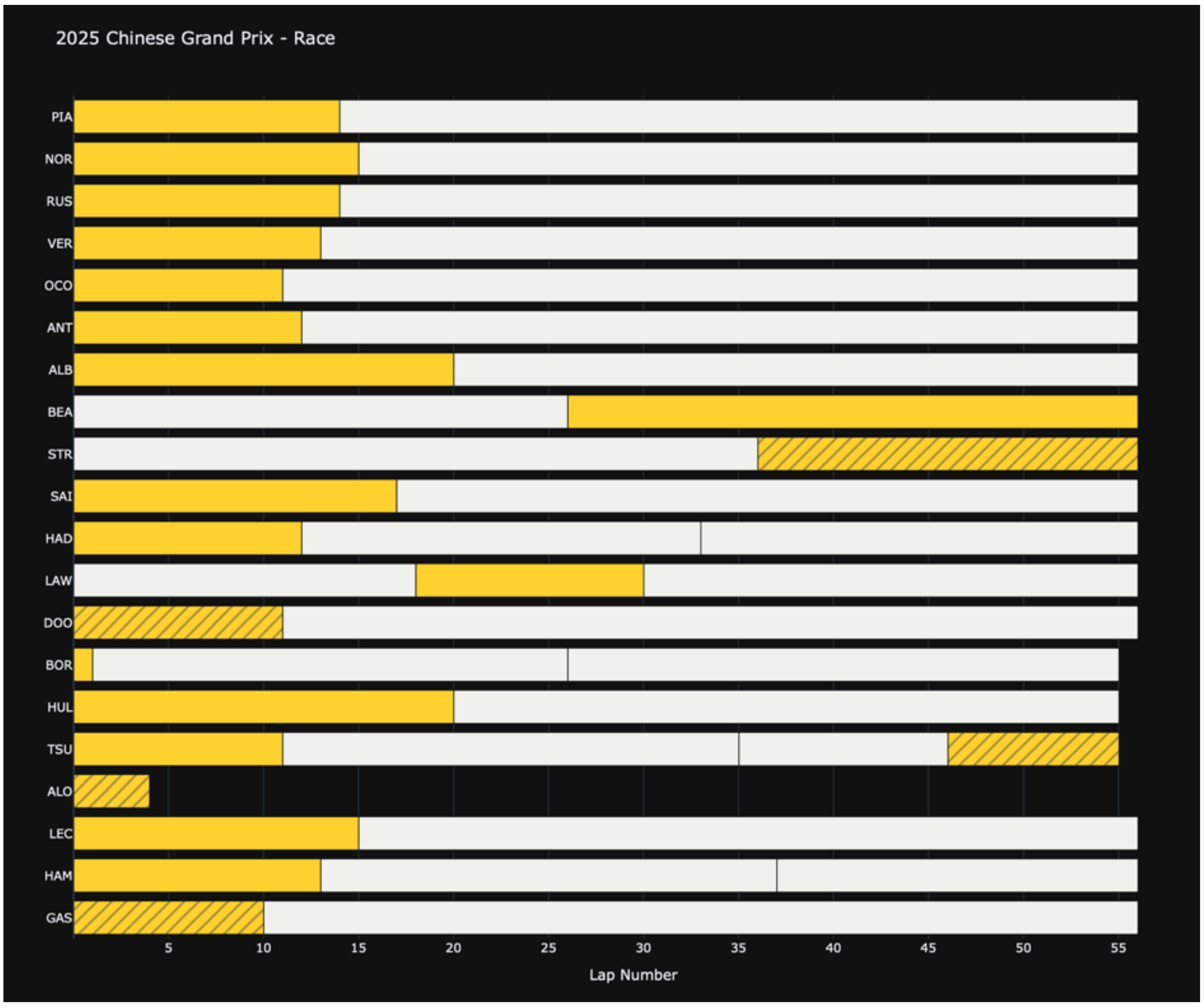


Figure 47: Actual 2025 Chinese Grand Prix Race Strategies

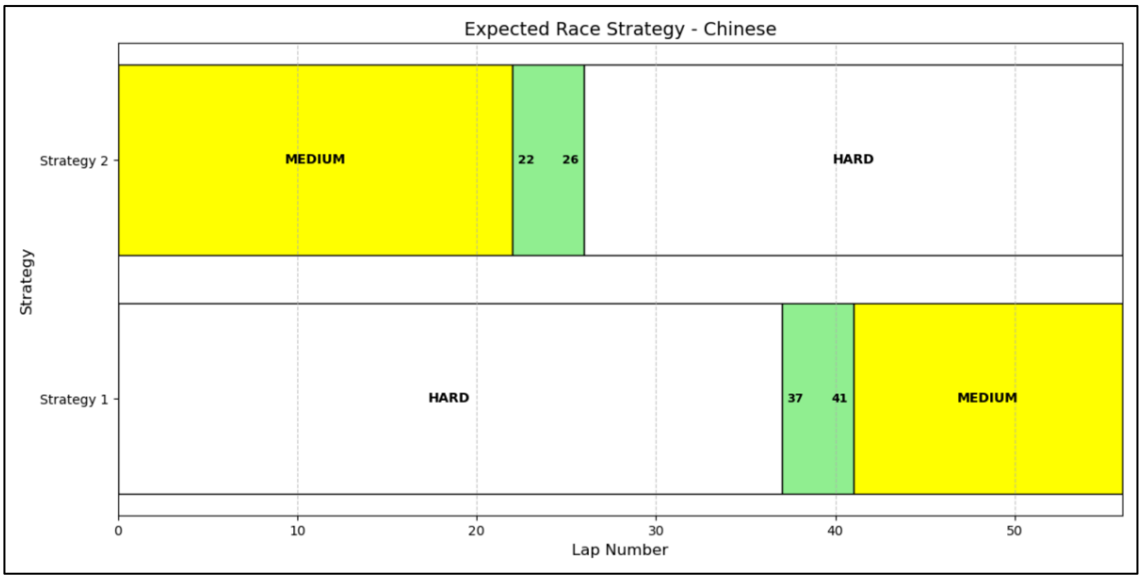


Figure 48: Simulated 2025 Chinese Grand Prix Pit Strategies

6.2.3 Overall Analysis

Across all races, the simulation consistently predicted faster overall race times than the actual results, with error margins varying depending on external factors such as weather conditions, race interruptions, and deviations in strategy execution. When conditions remained stable (China 2025 for example), there were no major external disruptions, which showed that the model demonstrated a high level of accuracy, with all predicted strategies being within 4% of the actual race time.

This confirms that the core structure of the predictive model is solid but requires further refinement to account for real-world race variables that disrupt expected performance trends and bring the predicted race time closer to the actual race time as the ideal simulation performance should fall within ±1%.

It was noted that the simulation strategy ended up being quite different from Pirelli's predicted tyre strategy, but interestingly it matched up much better with the real-life race strategies, which suggested that the simulation was somehow better at picking up on all the real-world conditions and those unpredictable race dynamics, like tyre wear, track changes, and how the car handles under different circumstances, which does not always follow the expected predictions. Pirelli's predictions are usually based on these generalised models that aim to cover ideal or controlled conditions but often miss out on the little details and complexities that can make a race go in all sorts of directions, which possibly explains the noticeable difference between the two strategies. This difference shows that while Pirelli's model relies on these perfect conditions that may not really happen in a live race, the simulation's approach seems to be more flexible, picking up on all minor changes during the race. Therefore, while Pirelli's predictions are useful for a starting point, the strategy that comes from the simulation provides a more realistic outlook when it comes to the race due to the reliance on historical race data.

The most significant limitation of the model remains its inability to dynamically adjust for changing race conditions, particularly in scenarios where weather fluctuations, Safety Cars, or Virtual Safety Cars impacted race pace. As a result, in races where these factors were present, the predicted times were skewed, leading to higher error margins than what would be expected in a fully controlled environment. Future improvements should focus on enhancing the model's adaptability to changing conditions, particularly by incorporating adjustments for weather, Safety Cars, degradation patterns that evolve differently in dynamic scenarios and further improvement in the delta prediction tyre model to increase further degradation accuracy. Addressing these limitations would allow for a more robust and adaptable prediction system, reducing the disparity seen in cases where external factors significantly impact race pace.

7. Project Evaluation

7.1 Evaluation

Looking back to the initial aims and objectives outlined in **Section 3.1**, it is evident that I met almost all the objectives:

Aim:

- Enhance the accuracy of predictive race strategy modelling by integrating machine learning techniques, providing an alternate approach to the problem.

Objectives:

- Develop a ML-Based Tyre Model
- Optimise Race Strategy Decisions
- Integrate ML Predictions into Simulations
- Consider complex interactions between tyres and racing conditions

An agile approach was employed throughout the project, breaking the work into manageable sections that allowed for continuous refinement and adaptation to new insights as they emerged. To support this, a Gantt chart was used to help visually break down the project into key phases, ensuring tasks were organised and deadlines were clearly defined and met. This allowed for flexibility, enabling the model to be developed bit by bit with each version building on the previous one or cutting back and omitting parts of the project, to ensure that the project could remain on track despite the changing nature of the work.

The model itself evolved step-by-step, beginning with a simple machine learning model, to a proof-of-concept simulation, to gradually incorporating more complex elements as additional data was gathered and more knowledge was learned. Data-driven decision-making was a fundamental aspect of the project, with a concentrated focus on filtering and analysing historical race data to extract key patterns and improve the predictive capabilities of the model and enhance accurate simulations.

Despite developing a functioning ML-integrated predictive race model, some objectives were only partially met. The model successfully incorporates the tyre model machine learning predictions into race simulations, allowing for the identification of optimal pit stop strategies and the comparison of various race scenarios. Validation against real-life predictions made by Pirelli as well as real world races proves that this model can produce race predictions and does optimise them to a certain point that they could potentially be used.

However, even though it can produce results, its accuracy is inconsistent, this is due to the limited time and data constraints – due to limited testing time, thorough testing and validation was not effective. Furthermore, implementation of external factors, such as fuel load, track evolution, and tyre life, is not fully captured which means the final objective was not fulfilled. Further testing and validation against real-world race data are required to improve the model's predictive capabilities. Despite these challenges, the model lays a solid foundation for future enhancements with additional data sources and advanced predictive techniques.

7.2 Limitations

7.2.1 Complex Target Variable

The primary target variable in this model (delta) was used to predict tyre performance, which was used to predict race outcomes. This was very complex as the target is influenced by multiple dynamic and uncertain factors, such as tyre wear, weather conditions, track evolution, and driver-specific behaviour. Such complexity made it challenging to accurately predict tyre wear (lap time loss) due to the interactions between the tyres and the external factors, this proved difficult to model with high precision, given the constraints in data and computational resources.

7.2.2 Limited Availability of Features

Another significant limitation was the unavailability of crucial real-time race features, such as detailed weather data, precise driver behaviour metrics, and specific team strategies. Many of these factors, essential for enhancing the model's accuracy, are not publicly available, which restricted the model's ability to fully replicate the real-world conditions that influence race strategy. For instance, not having access to exact tyre data like the slip angle, or the forces acting on the tyre.

7.2.3 Free Air Optimisation Challenges

Free air optimisation, a crucial aspect of race strategy to predict the effect of clear track conditions, was not achievable within the scope of this project. The computational complexity required to simulate accurate free air scenarios, combined with the limited data available, made it infeasible to model this factor effectively, hence falling back onto the 4.5% assumption. Free air conditions, which heavily impact race strategy (such as overtaking and lap time gains), could not be accurately incorporated into the model under the given constraints.

7.2.4 Literature Review Challenges

The lack of extensive literature review within ML integrated simulation was a limitation. Despite this, broader studies conducted within motorsports have an influence on the project, which aided with the development of the concept and while some benchmarking was performed against historical race strategies and Pirelli recommendations, a more detailed review of similar methodologies in sports analytics or real-time decision-making systems could provide further validation and refinement.

7.2.5 Time challenges

Despite the core functionality of the model being developed to a satisfactory standard, some features in the initial design such as the strategy selection based off SC probability were not implemented. Increased depth in this project would benefit from more time to effectively optimise the ML model and add/enhance any dynamic features in the MCS model.

8. Conclusion and Future Work

8.1 Conclusion

This project has explored the development of predictive race strategy modelling using an alternative ML integrated simulation approach to optimise race strategy. While we aimed to tackle key challenges that have limited other models, such as including probabilistic events like Safety Car deployments that can alter strategy, this aspect was not fully implemented in the final submission though it is seen in the artefact design. (see **Section 4.5**)

Integration of real-world performance factors allowed for predicted data to be more accurate and aligned with real world conditions. This approach adds complexity but enhances the model's adaptability, enabling more robust predictions in dynamic race environments. While this increases the model's ability to handle real-world uncertainties, it also introduces challenges in validation and data limitations.

Ultimately, the model offers some value by providing teams with a tool that leverages limited race data for race strategies, which could evolve alongside new race data, impacting future decision-making processes in motorsport and beyond.

Key points explored in this project:

- **Tyre degradation model** – utilised a traditional machine learning method (XGBoost) as an alternative to deep learning. This was used to predict the rate of lap time loss across tyre compounds.
- **Monte Carlo Simulation** – Used to simulate numerous race scenarios while incorporating the impact of probabilistic factors and identifying the ideal strategies under differing racing conditions.
- **Incorporation with real-world dynamics** despite limited data was seen with the use of standard deviations in the MCS, further influences such as track temperature influence the lap times. With higher standard deviation values suggesting the race was more dynamic.

Key lessons learned:

- Effective preprocessing and cleaning if data has a massive impact on ML results and can easily influence them.
- More Complex models do not necessarily produce better results, through Bayesian optimisation and feature engineering optimisation, the model was developed in a way to be simple to ensure at least some data variance was captured
- Benchmarking/Validation is always important, this enables results such as mine to be valid and find any areas to improve. This directly links to the opportunity to develop the model further in the future, as this alternative approach is novel in the field.
- Project required a massive blend of software engineering and data analysis which we had not expected.
- Finally, the model must be not only mathematically accurate to an extent but also practical in some sense. This model has a purpose to improve/provide an alternative approach to strategy generation.

Integrating ML within simulation models provides a robust approach to race strategy modelling, like Heilmeier's VSE approach. Though with the use of a more traditional ML model, less computational power is needed as well as the need for data – which was limited to only current

Formula One regulations to emulate realistic predictive modelling. Despite this, there were notable limitations such as the lack of alternative features such as driver behaviour (throttle usage over the course of a lap for example), ERS and DRS deployment over the course of the lap could be utilised if available. Further development and research can be done to model these dynamic factors in a more complex manner.

Overall, this work provides an alternative approach to the development of predictive race strategy modelling, the intention is not to replace possible the best solution offered but provide a different approach to the problem.

8.2 Future Work

There are numerous features that can be included to improve this model and increase the robustness and improve accuracy and precision. Further development in these areas can enhance the final product and results in a more complete model.

8.2.1 Incorporating Interchangeable/Wet Conditions

The complex nature of strategy prediction featuring interchangeable/wet conditions may be incorporated via the use of weather data within the model, this can be extracted from the API. This mainly applies to the race simulation where it begins or rain or dry up – this can be done with a different model, allowing for two models to predict different things. Furthermore, the lack of consistent wet weather data means that synthetic wet data could be used to estimate delta times for INTERMEDIATE and WET compounds.

8.2.2 Driver behaviour

Exploration of driver behaviour can influence the predictions to perhaps cater the strategy towards the drivers driving style. This can be through various means such as throttle% (throttle input), braking%, steering angles; all of these affect the tyres over the course of a lap, inclusion of this could provide more accurate predictions through the machine learning component of the artefact.

8.2.3 Compound and Tyre Freshness Predictions

The ML model does already predict by compound and fresh tyre, however there is limiting data which prohibits accurate generation of data, thus forcing the project to generalise the tyres by SOFT, MEDIUM and HARD only.

8.2.4 Real-time implementation

If possible, with more resources, a real-time implementation could be experimented with this model. Doing so can allow maximum flexibility and test the true application of a simple model like this in any racing situation.

8.2.5 Exploration of other ML models & Model Selection Logic

As outlined in Section 5.2.2, system is designed for easy insertion of alternative ML models, if the CSV file is structured the same way, the system can have no problem with experimenting with different ML models. This could be by using Light Gradient Boosting, RF, Neural Networks, if they are in a `.pkl` file and processed correctly, there will be no issue in testing other models to improve accuracy. Further improvements can also include the use of Model selection logic, where the system decides the best model for predictions on each circuit by evaluating the relevant regression scores (RMSE, MSE, MAE and R^2).

8.2.6 Additional Data Incorporation

The model successfully predicted Ferrari's performance by training on its 2022-2025 data and testing on the 2022-2023 data from other teams. Though, further improvements involve expanding the model to incorporate data from all teams across the 2022-2025 period except Ferrari to improve the generalisation towards them specifically.

Bibliography

Atalan, A., Sahin, H., Atalan, Y.A (2022) Integration of Machine Learning Algorithms and Discrete-Event Simulation for the Cost of Healthcare Resources [online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9601943/pdf/healthcare-10-01920.pdf

Ambler, W. (2024b). The Effect of Track Surface on Race Strategy. [online] Catapult. Available at: https://www.catapult.com/blog/race-strategy-f1-track-surface.

Ambler, W. (2024a). F1 DATA ANALYSIS & TECHNOLOGY: HOW DATA IS TRANSFORMING RACE PERFORMANCE. [online] Catapult. Available at: https://www.catapult.com/blog/f1-data-analysis-transforming-performance

Bazghandi, A. (2012). *Techniques, Advantages and Problems of Agent Based Modeling for Traffic Simulation*. [online] Available at: https://www.ijcsi.org/papers/IJCSI-9-1-3-115-119.pdf.

Bekker, J., Lotz, W., (2009) 'Planning Formula One race strategies using discrete event simulation' Department of Industrial Engineering, Stellenbosch, South Africa. [online] Available at: https://link.springer.com/content/pdf/10.1057/palgrave.jors.2602626.pdf

Birchwood (2024). *What are the Reinforcement Learning Advantages and Disadvantages*. [online] Birchwood University. Available at: https://www.birchwoodu.org/reinforcement-learning-advantages-and-disadvantages/.

Boettinger, M. and Klotz, D. (2023). *Mastering Nordschleife -A comprehensive race simulation for AI strategy decision-making in motorsports*. [online] Available at: https://arxiv.org/pdf/2306.16088.

Blackcircles (2023). F1 Tyres Explained 2023. [online] Blackcircles. Available at: https://www.blackcircles.com/news/f1-tyres-explained-2023.

Choo, C.L.W. (2015). Real-time decision making in motorsports: analytics for improving professional car race strategy. [online] dspace.mit.edu. Available at: https://dspace.mit.edu/handle/1721.1/100310.

Dale, W. (2017). *Watch: F1's massive leap in speed in 2017*. [online] Fox Sports. Available at: https://www.foxsports.com.au/motorsport/formula-one/f1-sidebyside-comparison-of-how-much-faster-formula-1-cars-were-in-2017/news-story/df5240caa795eb55f8ef46aba2197544# [Accessed 20 Dec. 2024].

Eagles, J. (2019). *Pirelli: 2019 C5 tyres require less management compared to '18 hyper-softs - The Checkered Flag*. [online] The Checkered Flag. Available at: https://www.thecheckeredflag.co.uk/2019/05/pirelli-2019-c5-tyres-require-less-management-compared-to-18-hyper-softs/ [Accessed 21 Dec. 2024].

Ergast Developer API (2024). Ergast API website: https://ergast.com/mrd/

FIA (2024) 2025 FORMULA ONE SPORTING REGULATIONS PUBLISHED ON 31 JULY 2024 Issue 1. (n.d.). Available at: https://www.fia.com/sites/default/files/fia_2025_formula_1_sporting_regulations_-_issue_1_-_2024-07-31.pdf [Accessed 3 Jan. 2025] [Online]

Greasley, A., Panchal, G. and Avinash Samvedi (2022). The Use of Simulation with Machine Learning and Optimization for a Digital Twin-A Case on Formula 1 DSS. 2022 Winter Simulation Conference (WSC). [online] Available at: https://doi.org/10.1109/wsc57314.2022.10015299.

Halliday, N. (2023). Drafting (Slipstreaming) in Racing | Blog. [online] SimScale. Available at: https://www.simscale.com/blog/drafting-slipstreaming-in-racing/. [Accessed 19 Mar. 2025].

Heilmeier, A., Graf, M. and Lienkamp, M. (2018). A Race Simulation for Strategy Decisions in Circuit Motorsports. 2018 21st International Conference on Intelligent Transportation Systems (ITSC). doi: https://doi.org/10.1109/itsc.2018.8570012.

Heilmeier, A., Graf, M., Betz, J. and Lienkamp, M. (2020a). Application of Monte Carlo Methods to Consider Probabilistic Effects in a Race Simulation for Circuit Motorsport. *Applied Sciences*, 10(12), p.4229. doi: https://doi.org/10.3390/app10124229.

Heilmeier, A., Thomaser, A., Graf, M., & Betz, J. (2020b). Virtual Strategy Engineer: Using artificial neural networks for making race strategy decisions in circuit motorsport. Applied Sciences, 10(21), 7805. [online] Available at: https://doi.org/10.3390/app10217805

Liu, X. and Fotouhi, A. (2020). Formula-E race strategy development using artificial neural networks and Monte Carlo tree search. *Neural Computing and Applications*. doi: https://doi.org/10.1007/s00521-020-04871-1.

Loreto, G. T. (2023). Applying machine learning to forecast Formula 1 race outcomes. [online] Available at: https://aaltodoc.aalto.fi/items/5848c100-478d-45dd-b2e8-5caf3a3114fb

Morlidge, M. (2022). *Hungarian GP: Explaining Ferrari's latest blunder and Charles Leclerc's dwindling Formula 1 title dreams*. [online] Sky Sports. Available at: https://www.skysports.com/f1/news/12433/12663464/hungarian-gp-explaining-ferraris-latest-blunder-and-charles-leclercs-dwindling-formula-1-title-dreams. [Accessed 27 March 2025]

McCullough, Tom (2024). Aston Martin F1 Team. [online] Available at: https://www.astonmartinf1.com/en-GB/news/feature/strategically-speaking-f1-strategy-explained-with-tom-mccullough. [Accessed 3 December 2024]

Ono, A. (2020). *Slipstream and 'dirty air' explained*. [online] Racecar Engineering. Available at: https://www.racecar-engineering.com/tech-explained/slipstream-and-dirty-air-explained/. [Accessed 19 Mar. 2025].

Pirelli Motorsport on X (formerly Twitter). (2017). *Here's our 2018 #Fit4F1 tyre-spotting guide!* [online] Available at: https://x.com/pirellisport/status/933679440214810624/photo/1 [Accessed 30 Nov. 2024].

Pirelli (2019). *F1 Tires*. [online] Pirelli.com. Available at: https://www.pirelli.com/tires/en-us/motorsport/f1/tires. [Accessed 21 Dec. 2024].

Pirelli (2024). Silverstone Possible Race Strategies. [online] X (formerly Twitter). Available at: https://x.com/pirellisport/status/1809662104959340851 [Accessed 25 Mar. 2025].

Raceteq.com. (2024). *F1 strategy: How Formula 1 teams determine the fastest race strategy*. [online] Available at: https://www.raceteq.com/articles/2024/07/how-formula-1-teams-determine-the-fastest-race-strategy. [Accessed 21 Dec. 2025].

RaceStaff365 (2025). What are the strategy options for the Chinese Grand Prix? [online] RacingNews365. Available at: https://racingnews365.com/what-are-the-strategy-options-for-the-chinese-grand-prix [Accessed 26 Mar. 2025].

Seymour, M. (2023). *F1 Tyres explained: the Beginner's Guide to Formula 1 Tyres | Formula 1®*. [online] formula1.com. Available at: https://www.formula1.com/en/latest/article/the-beginners-guide-to-formula-1-tyres.61SvF0Kfg29UR2SPhakDqd.

Sitara, S., Fatima, W. and Johrendt, J. (2023) 'Deep-Racing: An embedded deep neural network (EDNN) model to predict the winning strategy in Formula One racing,' International Journal of Machine Learning, 13(3). [online] Available at: https://doi.org/10.18178/ijml.2023.13.3.1135.

The F1 Insight. (2023). The Impact of DRS on Formula 1. [online] Available at: https://thef1insight.wordpress.com/2023/01/29/insight-1-effect-of-drs/ [Accessed 19 Nov. 2024].

Tonoli, A., Favelli, S., Salza, D. and Oldani, F. (2024). *POLITECNICO DI TORINO Data-Driven Vehicle Performance Optimization for Formula Student Racing Candidate Lal AKIN*. [online] Available at: https://webthesis.biblio.polito.it/secure/33043/1/tesi.pdf

Youson, M. (2025). *Strategy Guide for the Australian Grand Prix*. [online] Formula 1® - The Official F1® Website. Available at: https://www.formula1.com/en/latest/article/strategy-guide-what-are-the-tactical-options-for-the-season-opener-in.2bjujzRbKNnaNmEJl72vKh [Accessed 20 Mar. 2025].

Zhao, Zhixuan. (2024). Deep Neural Network-based lap time forecasting of Formula 1 Racing. Applied and Computational Engineering. 47. 61-66. 10.54254/2755-2721/47/20241191. [online] Available at: https://www.researchgate.net/publication/379012640_Deep_Neural_Network-based_lap_time_forecasting_of_Formula_1_Racing#

Appendix

Appendix A: Literature Review Material

Appendix A.1: Total Lap and Race Time Generation (Heilmeier et al., 2018)

The following formulae are used to calculate lap time and total race time based on various factors affecting a race performance.

Total lap time generation:

$$t_{lap}(lap) = t_{base} + t_{tire}(a_{lap}, c_{tire}) + t_{fuel}(lap) + t_{car} + t_{driver} + t_{grid}(lap) + t_{pit_2 inlap\&outlap}(lap)$$

Where:

- t_{base} – Baseline lap
- $t_{tire}(a_{lap}, c_{tire})$ – a logarithm denoting the tyre degradation lap time loss
- $t_{fuel}(lap)$ – lap time loss from fuel consumption
- $t_{car} + t_{driver}$ – lap time loss based off car and driver ability
- $t_{grid}(lap)$ – lap time loss due to starting grid position
- $t_{pit_2 inlap\&outlap}(lap)$ – pit stop lap loss time.

Total race time generation:

$$t_{race,currentlap} = \sum_{lap=1}^{current\ lap} t_{lap}(lap)$$

Appendix A.2: VSE Results with Probabilistic influences (Heilmeier, 2020b)

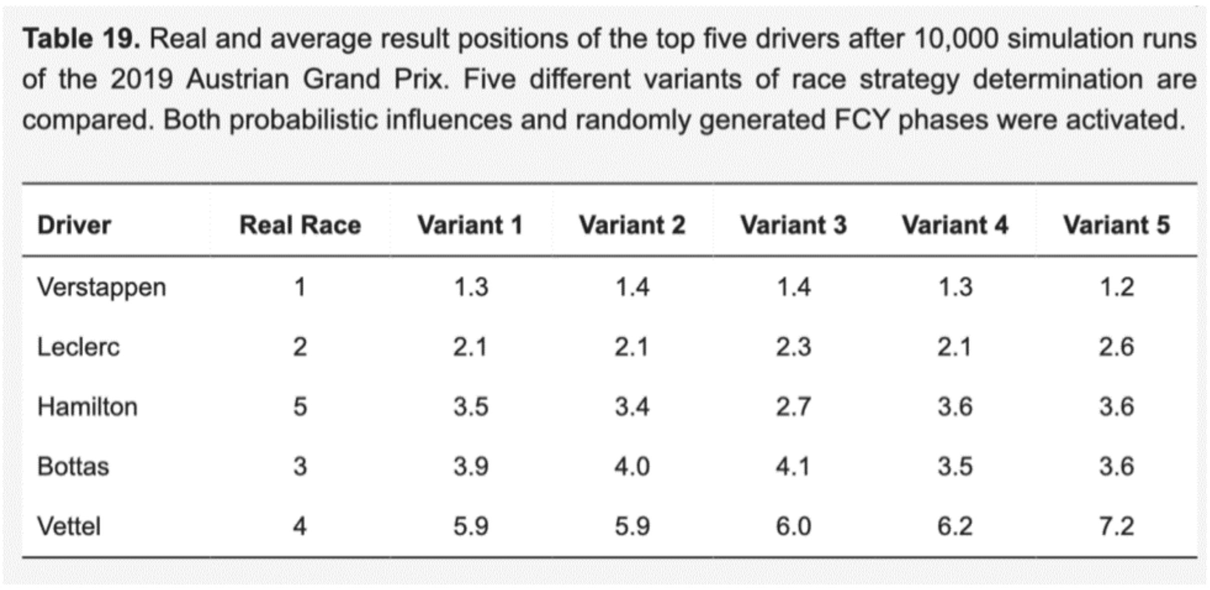


Table 19. Real and average result positions of the top five drivers after 10,000 simulation runs of the 2019 Austrian Grand Prix. Five different variants of race strategy determination are compared. Both probabilistic influences and randomly generated FCY phases were activated.

Driver	Real Race	Variant 1	Variant 2	Variant 3	Variant 4	Variant 5
Verstappen	1	1.3	1.4	1.4	1.3	1.2
Leclerc	2	2.1	2.1	2.3	2.1	2.6
Hamilton	5	3.5	3.4	2.7	3.6	3.6
Bottas	3	3.9	4.0	4.1	3.5	3.6
Vettel	4	5.9	5.9	6.0	6.2	7.2

Appendix A.3: Key ML and Simulation Strengths and Weaknesses in Experiments

Below are the summarised differences between each model proposed in the literature review, further in-depth discussion is provided in **Section 2.2**

Author	Model	Strength	Weakness
Atalan et al. (2022)	Random Forest (RF), Gradient Boosting (GB), AdaBoost (AB) integrated with Discrete Event Simulation (DES)	Cost-effective and accurate ML-integrated simulation method. Leave possible opportunity for research in motorsport	Healthcare studies often have larger datasets, whereas motorsports may have limited/noisy sets. This must be addressed going forward
Bekker & Lotz (2009)	Discrete Event Simulation (DES)	Simulated race strategies closely matched real-world results	Did not consider tyre degradation or DRS, outdated due to technological advancements now compared to 2009
Boettinger & Klotz (2023)	Reinforcement Learning (Deep Q-Network - DQN) with Agent-Based Modelling	Considered overtakes, traffic, and pit stops using RL	Lacked external probabilistic factors like weather & mechanical failures – they are crucial for strategy consideration.
Choo (2015)	System Dynamics Modelling	Effective tyre strategy analysis, Considered track characteristics	Focused on oval tracks, Lacked real-world validation
Greasley et al. (2022)	Digital Twin with Agent-Based Decision Support System (DSS)	Considered multiple race strategy factors using Digital Twin approach	No actual results or validation.
Heilmeier et al. (2018)	Discrete Event Simulation (DES)	Improved DES model with tyre wear & DRS considerations	Data source unknown, Outdated due to tyre rule changes and regulation changes.
Heilmeier et al. (2020a)	Monte Carlo Simulation (MCS)	Included probabilistic effects (SC/VSC) in race simulations	Lacked weather/track condition factors
Heilmeier et al. (2020b)	Artificial Neural Network (ANN) with Monte Carlo Simulation (MCS)	Incorporated probabilistic influences, Validated using F1 race data	Could benefit from further real-world validation
Liu & Fotouhi (2020)	Artificial Neural Networks (ANN), Monte Carlo Tree Search (MCTS)	Effective battery management strategy Included driver styles & weather effects	Used synthetic data which can lead to overfitting, Lacked real-world lap time validation

Loreto (2023)	Support Vector Machine (SVM), Random Forest (RF), Artificial Neural Network (ANN)	SVM outperformed other ML models in predicting pit stops & race scenarios	Further validation against real-world data needed
Sitara, Fatima & Johrendt (2023)	Embedded Deep Neural Network (EDNN)	Generated pitstop strategies using deep learning	Did not account for weather, VSC/SC, or external factors 544 pit stops were missed by the algorithm
Tonoli et al. (2024)	Linear Regression, Decision Tree Regression, Boosted Decision Trees, Gaussian Process Regression (GPR)	Accurately modelled driver behaviour, Applied multiple ML techniques	No cross-validation, Unclear real-world impact.
Zhao (2024)	Fully Connected Neural Network (FCNN)	Provided insights into driver pace and tyre wear effects	Large deviations on some tracks (e.g., Silverstone, Canada)

Appendix B: Datasets and Example Outputs

Appendix B.1: Raw Dataset

Lap Number	Time	Tyre	FreshTyre	Stint
1	90	SOFT	TRUE	1
2	90.2	SOFT	TRUE	1
3	90.1	SOFT	TRUE	1
4	91	SOFT	TRUE	1
5	93	SOFT	TRUE	1

Appendix B.2: Expected ML Output

Compound	FreshTyre	Average Delta/Lap	Delta Stdev	Baseline Lap(s)	Baseline Stdev
SOFT	TRUE	0.05	0.03	90.5	0.1
SOFT	FALSE	0.06	0.04	91	0.2
MEDIUM	TRUE	0.07	0.05	91.5	0.3

Appendix B.4: Possible Simulation Output

Strategy	Compound	Avg Lap time	Pit Window	Laps done
1	SOFTTrue	90	15-20	17
	HARDTrue	91	40-50	30

Appendix C: Mathematical Model and Algorithm Design

The following equations will be utilised to simulate lap times, derived from background knowledge, various papers during research and dataset analysis

Appendix C.1: Lap time model

Basic lap time simulation equation to generate lap times is:

$$Laptime_i = Laptime_{i-1} + TD_c$$

TD_c is generated from ML predictions of tyre deltas, and will vary lap by lap:

$$TD_c \sim (\mu_{ML}, \sigma_{ML})$$

- TD_c – Tyre degradation (delta per lap)
- σ_{ML} – Predicted Standard Deviation from Machine Learning Model

The $DegFactor_i$ equation then adjusts the lap time based on both the laps done on the stint (i) and the tyre life (xT_{max}), then $TrackTemp\ and\ TrackTemp_{mean}$:

$$DegFactor_i = \left(1 + \frac{i}{xT_{max}}\right) + \frac{TrackTemp}{TrackTemp_{mean}}$$

- xT_{max} is the maximum TyreLife on a race stint, this is taken from the dataset.
- $\left(1 + \frac{i}{xT_{max}}\right)$ represents generic scaling i.e. 1.1 times worse than the original degradation
- $TrackTemp \sim (TrackTemp_{mean}\ , TrackTemp_{std})$ represents the gaussian distribution of values of the temperature to represent the fluctuation for each lap

Then the Lap time Generation with the inclusion of tyre degradation factor:

$$Laptime_i = Laptime_{i-1} + TD_c \times DegFactor_i$$

Appendix C.2: Strategy Selection Algorithm

Assume that the following data is used in a dictionary.

```
unique_strategies = {
    ('strategy_1',): 0,
    ('strategy_2',): 1,
    ('strategy_3',): 2,
    ('strategy_4',): 3
}


total_race_times = [120, 110,
130, 115]
```

It is first ordered by indices:

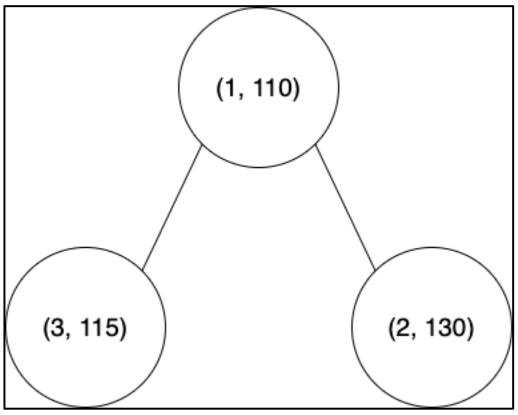
```
sorted _strategies = [(0, 120), (1, 110), (2, 130), (3, 115)]
```

Then the key-value pairs are sorted first by fastest time and then identify the unique strategies by using a combination of compound use and rounded lap distributions. The key-value pairs **(Strategy number, total race time)** in the dictionary are then stored in a min-heap, allowing for easy identification of the fastest race time. This is done because despite sorting the strategies by fastest total race time, there is no guarantee that the fastest laps from each unique strategy will still be sorted by race time, hence using min-heap or heap is useful here as it can easily navigate the nodes to identify the fastest strategies. This is by initially storing the fastest 3 strategies (in this example), then comparing the remaining strategy against the largest race time in that heap, if it does, then it gets replaced, and the min-heap will sort if the other nodes have a slower race time.

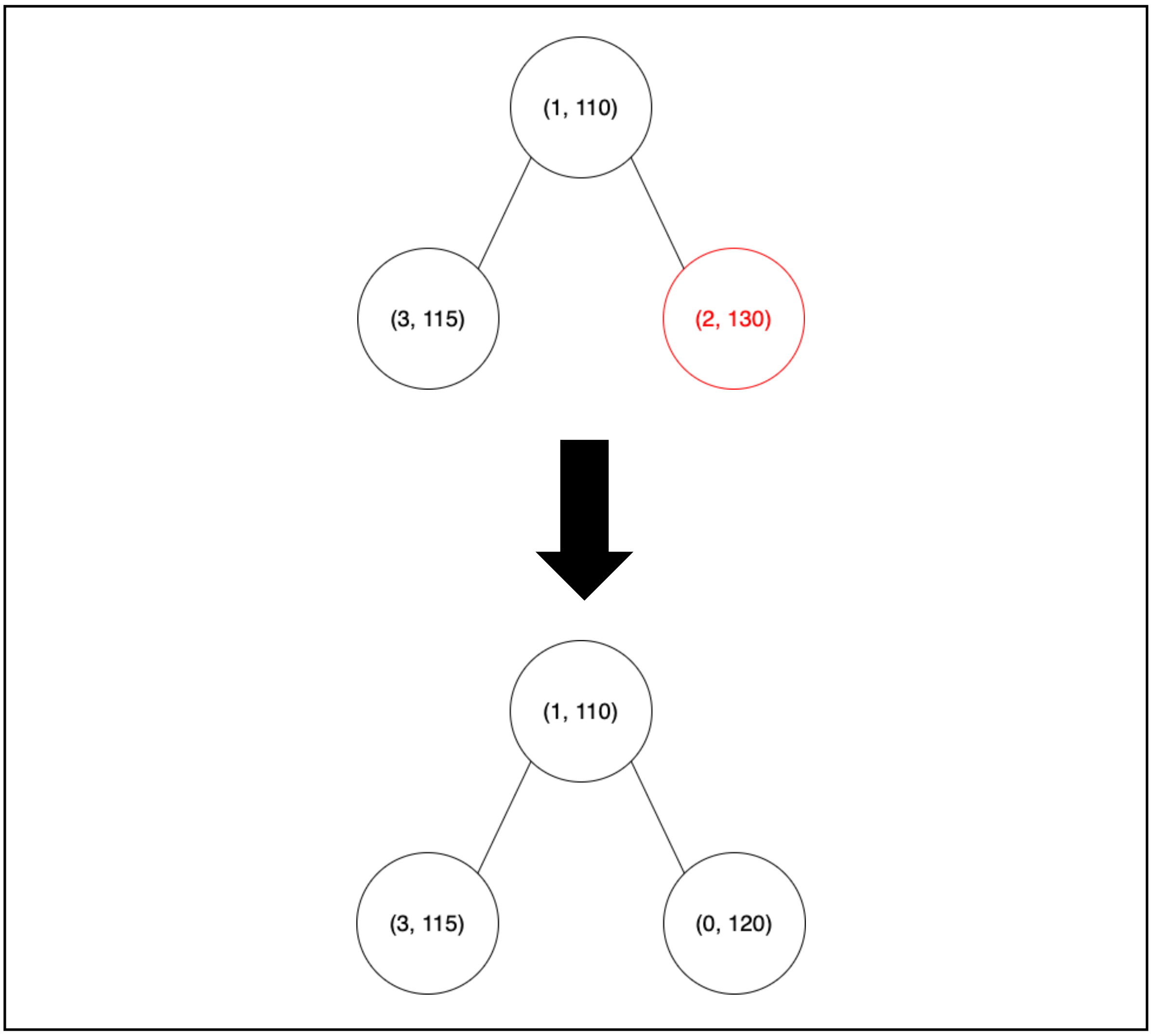
Below is the array version where the dictionary values are stored as tuples and sorted by fastest to slowest total race time in the array:

```
sorted _strategies = [(1, 110), (3, 115), (2, 130), (0, 120)]
```

Below is the visualised min-heap tree, inserted using priority queue and identified by strategy number:



Strategy 0 is the remaining strategy in the array, it is then compared to the largest node (slowest race time) in the min-heap tree, it is then replaced.

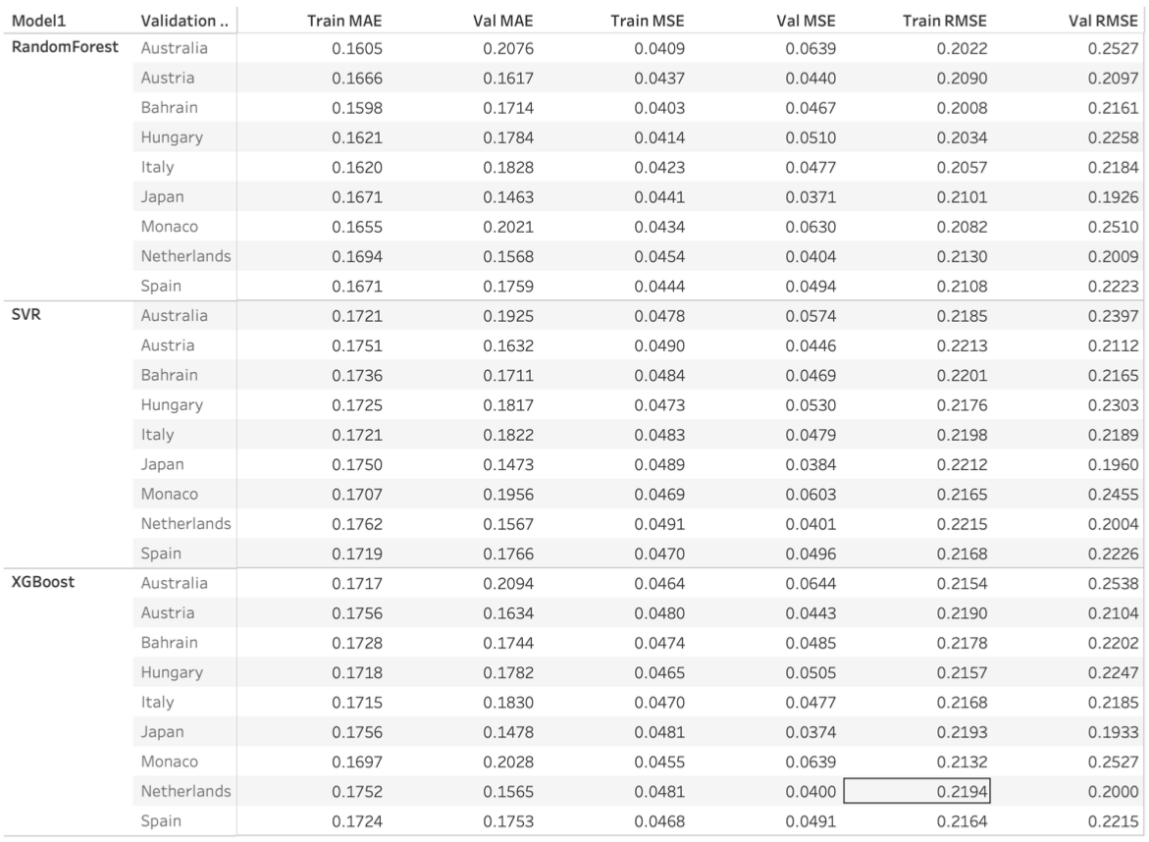


In the full system, it will compare more strategies, as in 10,000 simulations it is likely that there will be numerous combinations of strategies. This data structure allows us to quickly retrieve the fastest strategies without needing to repeatedly sort all the strategies and optimises the process by reducing the need for expensive sorting operations, making the selection of the fastest strategies more efficient.

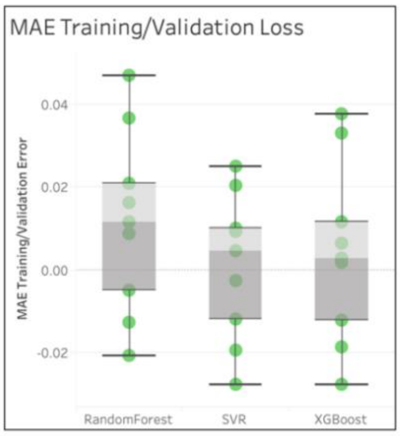
Appendix D: Approach

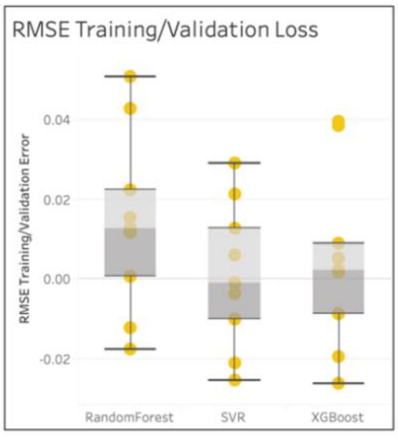
Appendix D.1: Machine Learning Model Comparison

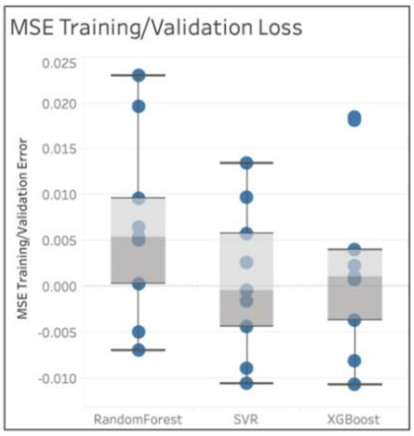
Below are the full metrics table for each circuit. The same LOOCV and features was conducted to ensure a fair test



Model1	Validation ..	Train MAE	Val MAE	Train MSE	Val MSE	Train RMSE	Val RMSE
RandomForest	Australia	0.1605	0.2076	0.0409	0.0639	0.2022	0.2527
	Austria	0.1666	0.1617	0.0437	0.0440	0.2090	0.2097
	Bahrain	0.1598	0.1714	0.0403	0.0467	0.2008	0.2161
	Hungary	0.1621	0.1784	0.0414	0.0510	0.2034	0.2258
	Italy	0.1620	0.1828	0.0423	0.0477	0.2057	0.2184
	Japan	0.1671	0.1463	0.0441	0.0371	0.2101	0.1926
	Monaco	0.1655	0.2021	0.0434	0.0630	0.2082	0.2510
	Netherlands	0.1694	0.1568	0.0454	0.0404	0.2130	0.2009
	Spain	0.1671	0.1759	0.0444	0.0494	0.2108	0.2223
SVR	Australia	0.1721	0.1925	0.0478	0.0574	0.2185	0.2397
	Austria	0.1751	0.1632	0.0490	0.0446	0.2213	0.2112
	Bahrain	0.1736	0.1711	0.0484	0.0469	0.2201	0.2165
	Hungary	0.1725	0.1817	0.0473	0.0530	0.2176	0.2303
	Italy	0.1721	0.1822	0.0483	0.0479	0.2198	0.2189
	Japan	0.1750	0.1473	0.0489	0.0384	0.2212	0.1960
	Monaco	0.1707	0.1956	0.0469	0.0603	0.2165	0.2455
	Netherlands	0.1762	0.1567	0.0491	0.0401	0.2215	0.2004
	Spain	0.1719	0.1766	0.0470	0.0496	0.2168	0.2226
XGBoost	Australia	0.1717	0.2094	0.0464	0.0644	0.2154	0.2538
	Austria	0.1756	0.1634	0.0480	0.0443	0.2190	0.2104
	Bahrain	0.1728	0.1744	0.0474	0.0485	0.2178	0.2202
	Hungary	0.1718	0.1782	0.0465	0.0505	0.2157	0.2247
	Italy	0.1715	0.1830	0.0470	0.0477	0.2168	0.2185
	Japan	0.1756	0.1478	0.0481	0.0374	0.2193	0.1933
	Monaco	0.1697	0.2028	0.0455	0.0639	0.2132	0.2527
	Netherlands	0.1752	0.1565	0.0481	0.0400	0.2194	0.2000
	Spain	0.1724	0.1753	0.0468	0.0491	0.2164	0.2215







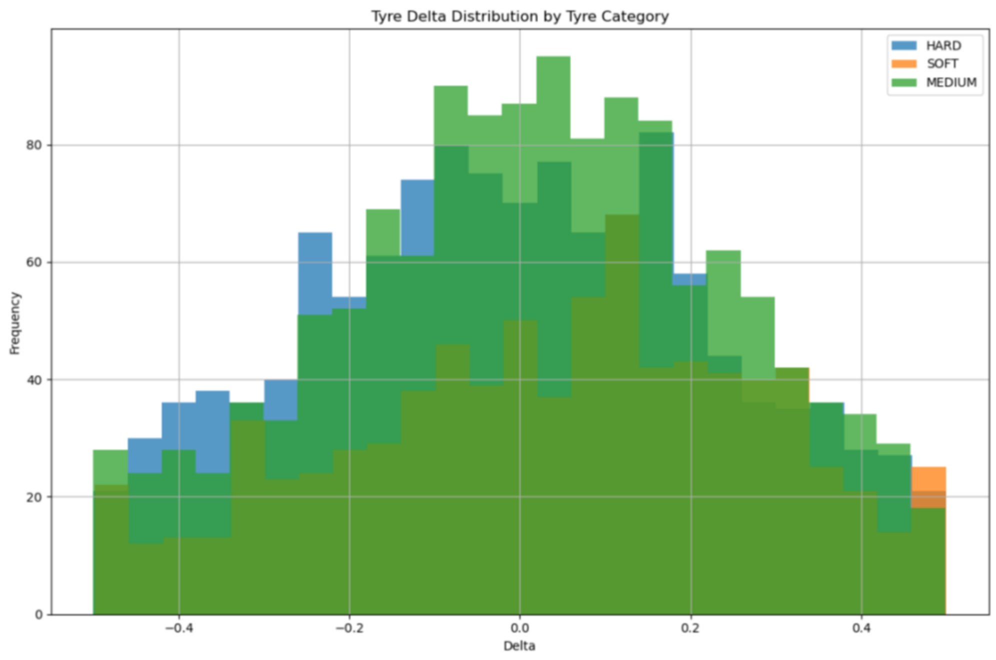
The median training/validation loss.

Appendix D.2: RandomSearchCV Parameters

```
xgb_pipeline   =   Pipeline([('preprocessor',   preprocessor),   ('xgb',
xgb_model)])

param_dist = {

    'xgb__n_estimators': randint(100, 1001),

    'xgb__max_depth': randint(1, 10),

    'xgb__min_child_weight': randint(1, 11),

    'xgb__learning_rate': uniform(0.001, 0.1),

    'xgb__subsample': uniform(0.6, 0.4),

    'xgb__colsample_bytree': uniform(0.5, 0.4),

    'xgb__alpha': uniform(0.001, 1.0),

    'xgb__lambda': uniform(0.001, 1.0)

}
```

```
Best Parameters:

xgb_model = XGBRegressor(

    random_state=42,

    alpha=4,

    colsample_bytree=0.7259,

    reg_lambda=0.0872,

    learning_rate=0.0812,

    max_depth=88,

    min_child_weight=8,

    n_estimators=384,

    subsample=0.6348

)
```

Appendix D.3: Distribution of Delta Points between 0.5 and -0.5



Appendix D.4: Optuna Search Method

Below is the search function for the parameters using Optuna

```
def objective(trial):

    params = {

        'n_estimators': trial.suggest_int('n_estimators', 100, 50000),

        'max_depth': trial.suggest_int('max_depth', 2, 100),

        'min_child_weight': trial.suggest_int('min_child_weight', 1, 100),

        'learning_rate': trial.suggest_float('learning_rate', 0.001, 0.05),

        'subsample': 1.0,

        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.82, 1.0),

        'alpha': trial.suggest_float('alpha', 0, 10),

        'lambda': trial.suggest_float('lambda', 0, 20),

        'gamma': trial.suggest_float('gamma', 0, 0.01),

        was notree_method': 'hist',

        'n_jobs': -1

    }
```

```
study = optuna.create_study(direction='maximize',
sampler=optuna.samplers.TPESampler())
```

Then the study is created using the TPE sampler as it learns from past trials by modelling good and bad results separately, then picks new parameters that look more like the good results which is computationally efficient as opposed to the random sampler which aimlessly searches the hyperparameters.

Appendix D.5: Optimised Feature Engineering

```
def create_features(df_filtered):

    df_filtered['RelativeDelta'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound_Fresh'])['Delta'].shift(1).diff()



    df_filtered['TyreLife_Lap'] = df_filtered['LapNumber'] * 
df_filtered['TyreLife']

    df_filtered['LapTime_TrackTemp'] = df_filtered['Delta'].shift(1) / 
df_filtered['TrackTemp'].shift(1)



    df_filtered['AvgTyreLife'] = 
df_filtered.groupby(['Compound'])['Delta'].mean()

    df_filtered['TrackTempStd'] = 
df_filtered['TrackTemp'].rolling(window=window_size).std()



    df_filtered['Prev_Time'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound_Fresh'])['LapTime_seconds'].shift(1)

    df_filtered['Prev_Delta'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound_Fresh'])['Delta'].shift(1)



    df_filtered['RollingAvg_Time'] = (df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound_Fresh', 'Stint'])


['LapTime_seconds'].shift(1).rolling(window=window_size, min_periods=3).mean())



    df_filtered['RollingAvg_Delta'] = (df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound_Fresh', 'Stint'])


['Delta'].shift(1).rolling(window=window_size, min_periods=3).mean())



    df_filtered = 
df_filtered.dropna(subset=['RollingAvg_Time','RollingAvg_Delta','Prev_Time','Prev
_Delta','RelativeDelta', 'LapTime_TrackTemp'])

    return df_filtered
```

Appendix D.6: ML Model Overhaul Time-split set up

Feature Engineering

```
def create_features(df_filtered):

    window_size = 5

    df_filtered['RelativeDelta'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound'])['Delta'].shift(1).diff()

    df_filtered['TyreLife_Lap'] = df_filtered['TyreLife'] / 
(df_filtered['LapNumber'])

    df_filtered['LapTime_TrackTemp'] = df_filtered['Delta'].shift(1) / 
(df_filtered['TrackTemp'].shift(1) + 1e-6)

    df_filtered['AvgTyreLife'] = df_filtered.groupby(['Year', 'Session', 
'Compound', 'Driver'])['TyreLife'].transform('mean')

    df_filtered['TrackTempMean'] = 
df_filtered['TrackTemp'].rolling(window=window_size, min_periods=1).mean()

    df_filtered['Prev_Time'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound'])['LapTime_seconds'].shift(1)

    df_filtered['Prev_Delta'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound'])['Delta'].shift(1)

    df_filtered['RollingAvg_Time'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound', 
'Stint'])['LapTime_seconds'].shift(1).rolling(window=window_size, 
min_periods=3).mean()

    df_filtered['RollingAvg_Delta'] = df_filtered.groupby(['Session', 'Driver', 
'Circuit', 'Compound', 'Stint'])['Delta'].shift(1).rolling(window=window_size, 
min_periods=3).mean()

    df_filtered = df_filtered.dropna()

    return df_filtered



#features

num_f = ['LapNumber', 'LapTime_seconds', 'TyreLife', 'TrackTemp', 'Humidity', 
'AirTemp', 'Stint', 'Prev_Time', 'Prev_Delta', 'RelativeDelta', 
'RollingAvg_Time', 'RollingAvg_Delta']

cat_f = ['Compound']
```

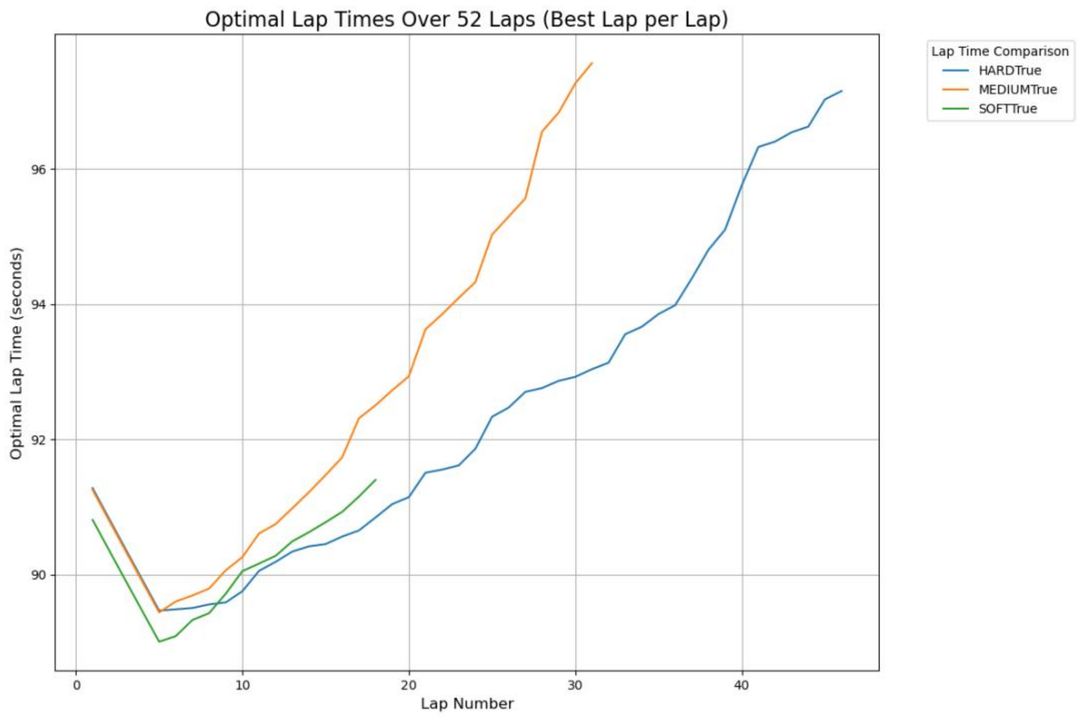
Time-split set up

```
df = pd.read_csv(was notyre_data/datasets/train/all_sessions_22_23.csv')

df_filtered = df[

    (df['TyreLife'] != df.groupby(['Session', 'Driver', 'Stint',
'Circuit', 'Year'])['TyreLife'].transform('min')) &

    (df['Delta'].between(-0.3, 0.3)) &

    (df['Team'] != 'Ferrari') &

    (~df['Compound'].isin(['TEST_UNKNOWN', 'UNKNOWN', 'WET',
'INTERMEDIATE']))

].copy()


#test

df_ferrari = pd.read_csv(was
notyre_data/datasets/test/ferrari_australia_sessions.csv')

df_ferrari_filtered = df_ferrari[

    (df_ferrari['TyreLife'] != df_ferrari.groupby(['Session', 'Driver',
'Stint', 'Circuit', 'Year'])['TyreLife'].transform('min')) &

    (df_ferrari['Delta'].between(-0.3, 0.3)) &

    (~df_ferrari['Compound'].isin(['TEST_UNKNOWN', 'UNKNOWN', 'WET',
'INTERMEDIATE']))

].copy()


df_ferrari_filtered = df_ferrari_filtered.dropna(subset=['Compound'])

df_ferrari_filtered = create_features(df_ferrari_filtered)

df_ferrari_filtered['Predicted_Delta'] =
xgb_pipeline.predict(df_ferrari_filtered[num_f + cat_f])
```

Appendix D.7: Proof of Concept MCS

Results of the initial model, this shows a proof of concept that the equations proposed in Section **4.4** are viable. Despite the large difference, between the sim and the actual sample numbers, further optimisation can be done to ensure that the mean value sits closer to the actual.

Compound	Pit Window	Sim Mean Lap time on Stint	ACTUAL Mean Lap time on Stint	Difference between sim and actual (seconds)
HARDTrue	Lap 46-51	92.380	96.211	+0.063
MEDIUMTrue	Lap 31-36	92.530	96.007	+0.655s
SOFTTrue	Lap 18-23	90.170	91.235	+1.065s



As mentioned in **Section 4.4.2**:

- More aggressive tyre degradation leads to faster lap times initially, followed by a steeper performance drop-off.
- Less aggressive tyre degradation results in slower initial lap times, but with better longevity in the stint.

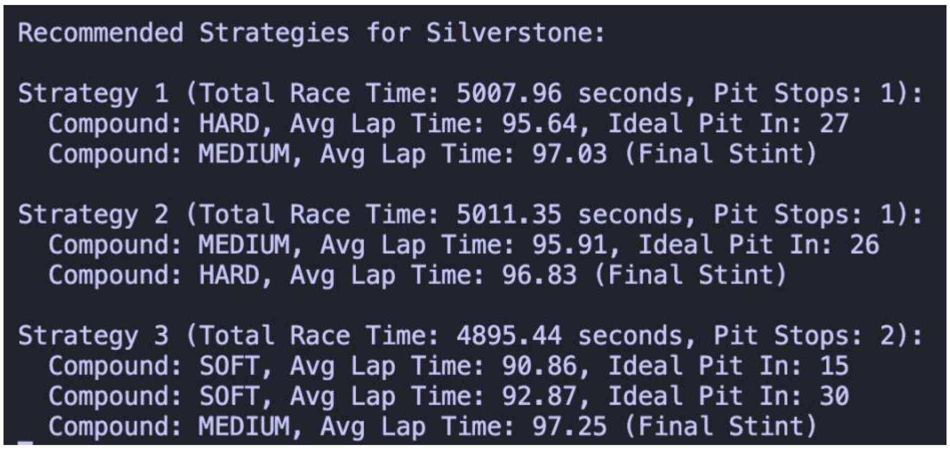
Which is somewhat accurately shown here, SOFTTrue has a higher degradation rate compared to MEDIUMTrue and HARDTrue which proves the bullet points stated. Thus, verifying the mathematical model presented for tyre degradation.

Appendix E: Visualisation Function and Outputs

Below are the ouputs of each respective function, they do follow the principles of SRP and will use Silverstone as an example.

Appendix E.1: display_strategies

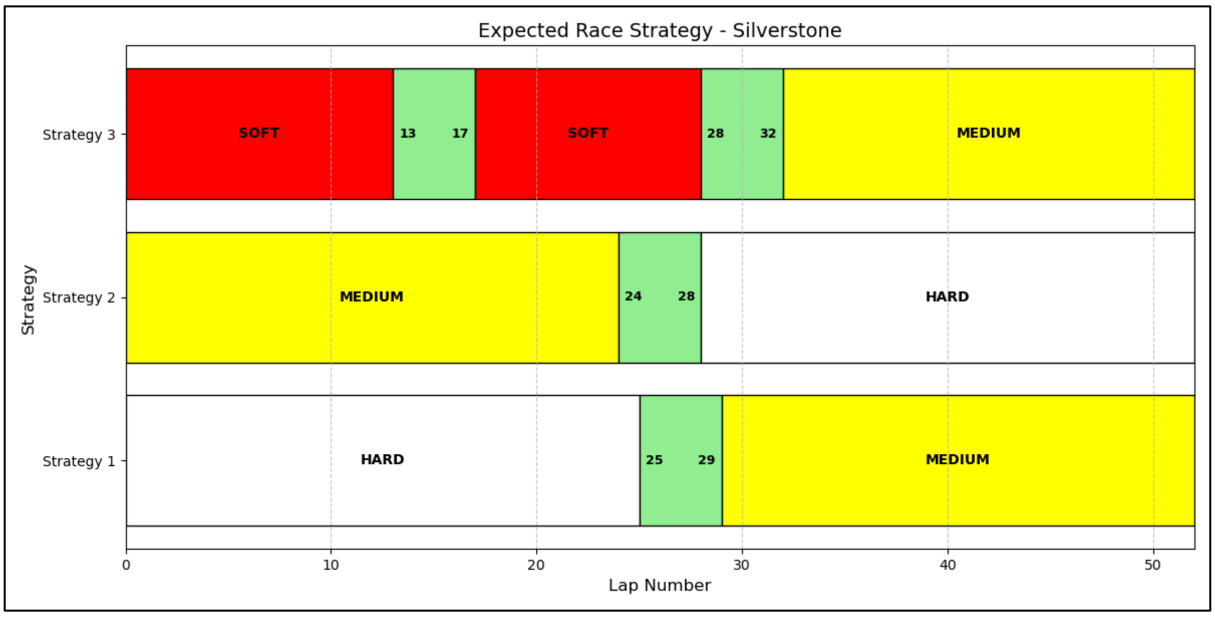
```
def display_strategy(selected_indices, pit_stop_strategies, total_race_times, pit_stop_count, all_lap_times, pit_stop_time):

print(f"\nRecommended Strategies:")

    total_pit_stop_time = sum(pit_stop_count[idx] * pit_stop_time for idx in selected_indices)

    total_pit_stops = sum(pit_stop_count[idx] for idx in selected_indices)

    avg_pit_stop_time = total_pit_stop_time / total_pit_stops if total_pit_stops > 0 else 0

    print(f"Overall Average Pit Stop Duration: {avg_pit_stop_time:.3f}")


    for rank, idx in enumerate(selected_indices, start=1):

        print(f"\nStrategy {rank} (Total Race Time: {format_time(total_race_times[idx])}, Pit Stops: {pit_stop_count[idx]}):")

        total_stint_time = 0

…
```



```
Recommended Strategies for Silverstone:

Strategy 1 (Total Race Time: 5007.96 seconds, Pit Stops: 1):
  Compound: HARD, Avg Lap Time: 95.64, Ideal Pit In: 27
  Compound: MEDIUM, Avg Lap Time: 97.03 (Final Stint)

Strategy 2 (Total Race Time: 5011.35 seconds, Pit Stops: 1):
  Compound: MEDIUM, Avg Lap Time: 95.91, Ideal Pit In: 26
  Compound: HARD, Avg Lap Time: 96.83 (Final Stint)

Strategy 3 (Total Race Time: 4895.44 seconds, Pit Stops: 2):
  Compound: SOFT, Avg Lap Time: 90.86, Ideal Pit In: 15
  Compound: SOFT, Avg Lap Time: 92.87, Ideal Pit In: 30
  Compound: MEDIUM, Avg Lap Time: 97.25 (Final Stint)
```

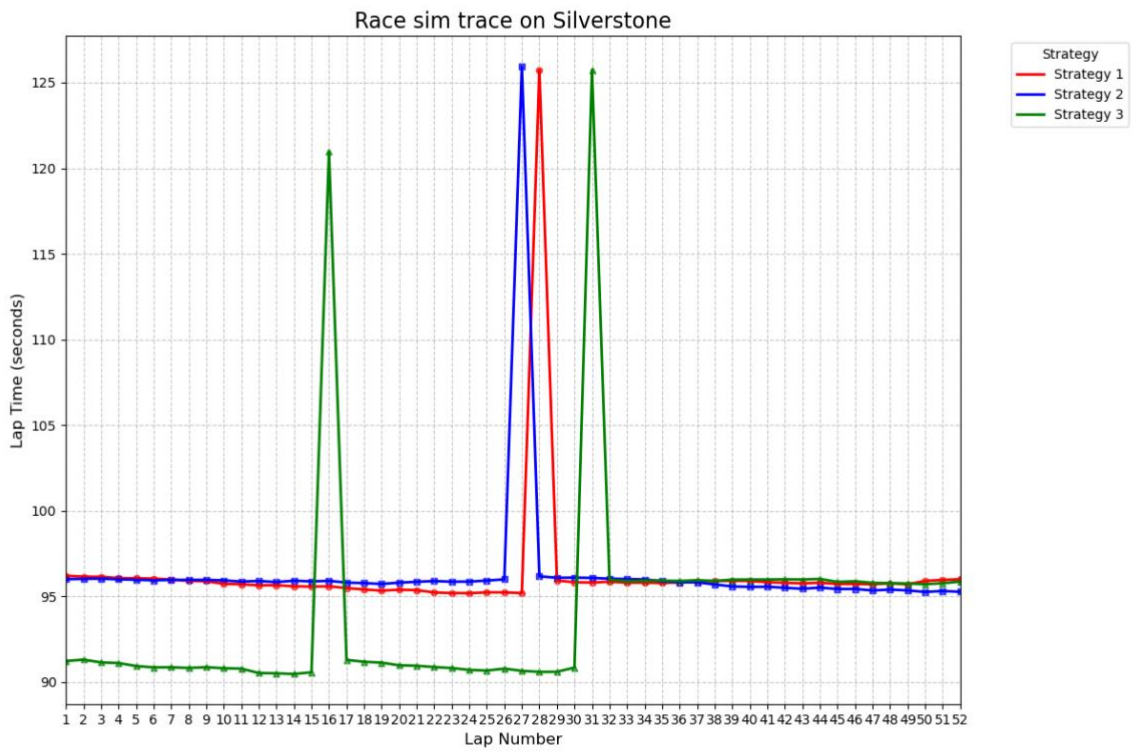
Appendix E.2: plot_strategies

```
def plot_strategies(selected_indices, pit_stop_strategies, num_laps, compound_c, r):
    plt.figure(figsize=(12, 6))
    y_labels = []

    for i, idx in enumerate(selected_indices):
        y_labels.append(f"Strategy {i+1}")
        x_start = 0

        for j, (compound, stint_laps) in enumerate(pit_stop_strategies[idx]):
            stint_start = x_start
            stint_end = x_start + stint_laps

            #adjust for pit windo, 2 laps before pit in and 2 laps after
            if j > 0:
                stint_start += 2
            if j < len(pit_stop_strategies[idx]) - 1:
                stint_end -= 2
```



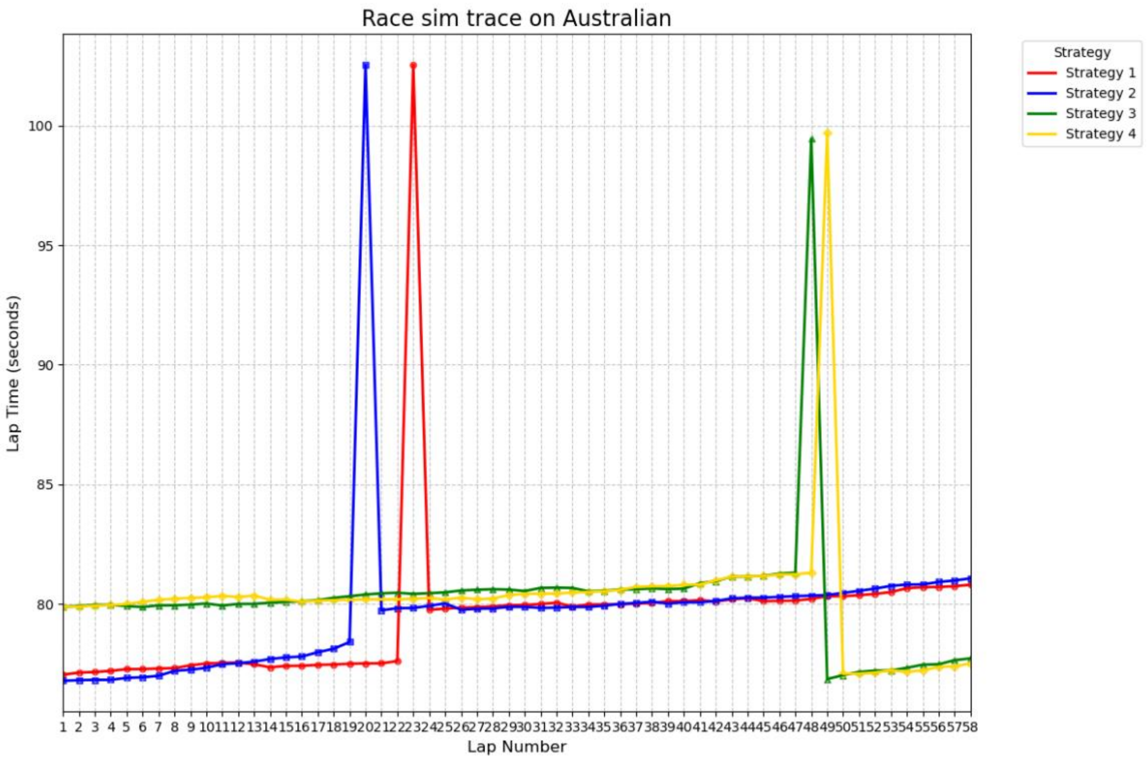
Appendix E.3: plot_race_trace

```
def plot_race_trace(selected_indices, all_lap_times, num_laps, r):
    plt.figure(figsize=(12, 8))
    colour = ['r', 'b', 'g','gold']
    markers = ['o', 's', '^','D']

    for i, idx in enumerate(selected_indices):
        lap_numbers = range(1, len(all_lap_times[idx]) + 1)
        plt.plot(lap_numbers, all_lap_times[idx], label=f"Strategy {i+1}", colour=colour[i], linestyle='-', linewidth=2)
        plt.scatter(lap_numbers, all_lap_times[idx], colour=colour[i], marker=markers[i], s=20, alpha=0.7)
```

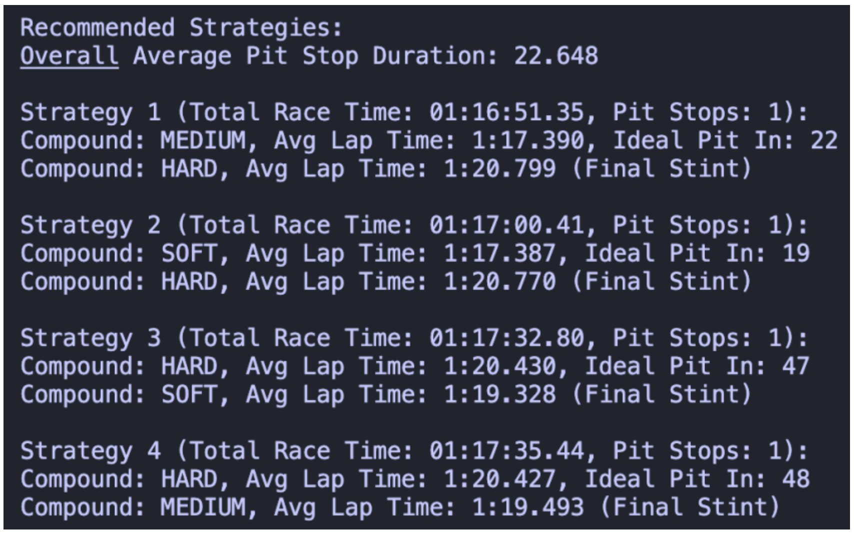


Appendix F: Race simulation trace

Appendix F.1: Race sim trace and detailed strategy breakdown Australia 2024

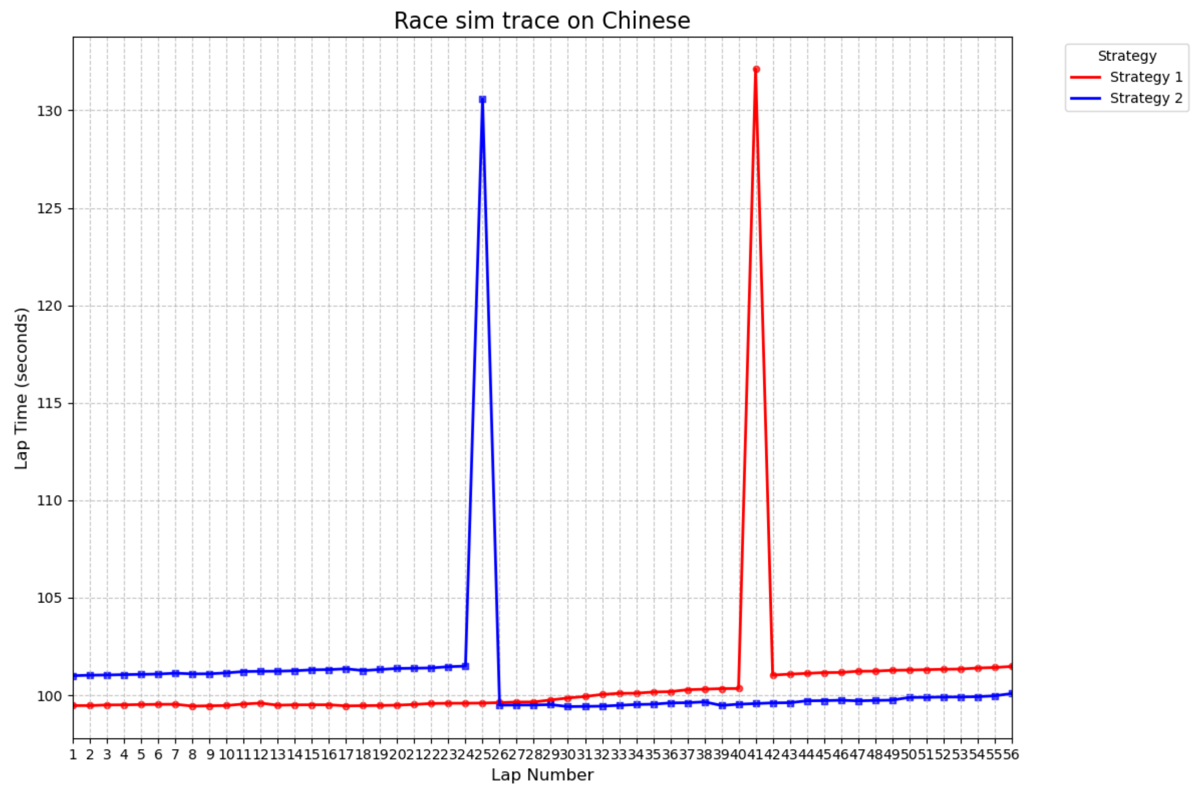


Strategy Breakdown

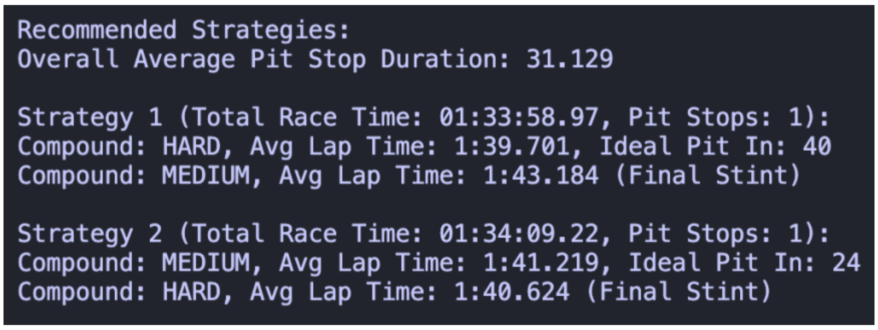


```
Recommended Strategies:
Overall Average Pit Stop Duration: 22.648

Strategy 1 (Total Race Time: 01:16:51.35, Pit Stops: 1):
Compound: MEDIUM, Avg Lap Time: 1:17.390, Ideal Pit In: 22
Compound: HARD, Avg Lap Time: 1:20.799 (Final Stint)

Strategy 2 (Total Race Time: 01:17:00.41, Pit Stops: 1):
Compound: SOFT, Avg Lap Time: 1:17.387, Ideal Pit In: 19
Compound: HARD, Avg Lap Time: 1:20.770 (Final Stint)

Strategy 3 (Total Race Time: 01:17:32.80, Pit Stops: 1):
Compound: HARD, Avg Lap Time: 1:20.430, Ideal Pit In: 47
Compound: SOFT, Avg Lap Time: 1:19.328 (Final Stint)

Strategy 4 (Total Race Time: 01:17:35.44, Pit Stops: 1):
Compound: HARD, Avg Lap Time: 1:20.427, Ideal Pit In: 48
Compound: MEDIUM, Avg Lap Time: 1:19.493 (Final Stint)
```

Appendix F.2: Race sim trace and detailed strategy breakdown China 2025



Strategy Breakdown



```
Recommended Strategies:
Overall Average Pit Stop Duration: 31.129

Strategy 1 (Total Race Time: 01:33:58.97, Pit Stops: 1):
Compound: HARD, Avg Lap Time: 1:39.701, Ideal Pit In: 40
Compound: MEDIUM, Avg Lap Time: 1:43.184 (Final Stint)

Strategy 2 (Total Race Time: 01:34:09.22, Pit Stops: 1):
Compound: MEDIUM, Avg Lap Time: 1:41.219, Ideal Pit In: 24
Compound: HARD, Avg Lap Time: 1:40.624 (Final Stint)
```

Glossary

Machine Learning Terms

Term	Definition
Cross Validation	A technique used to assess how well a machine learning model will perform on unseen data by splitting the dataset into multiple parts and testing the model on different combinations of those parts. This helps ensure the model is not overfitting to the training data. Techniques include time-split, group k-fold, stratified k-fold
Deep Learning	A subfield of machine learning that uses neural networks with many layers (hence deep) to model complex patterns in large datasets. It's especially useful for tasks like image recognition, natural language processing, and speech recognition.
Digital Twins	Virtual models or simulations of real-world systems, processes, or objects (like a car or a racetrack). These digital versions allow for real-time monitoring, analysis, and optimisation without affecting the real-world system.
Feature Engineering	The process of selecting, modifying, or creating new variables (features) from raw data (normally columns) to improve the performance of machine learning models.
Hyperparameters	Parameters in machine learning algorithms that are set before training the model and control how the model is trained (e.g., learning rate, tree depth, etc.).
Machine Learning (ML)	A type of artificial intelligence that allows systems to learn from data and improve including various models and algorithms, such as XGBoost, neural networks, and decision trees, among others.
Monte Carlo Simulation (MCS)	A simulation method using random sampling to simulate different possible race scenarios and predict outcomes.

Overfitting	A situation where a machine learning model learns not only the underlying patterns in the training data but also the noise, leading to poor performance on new, unseen data.
Reinforcement Learning	A type of machine learning where an agent (like a robot) learns to make decisions by interacting with an environment and receiving rewards or penalties based on its actions.
Simulation Model	A computer-based model used to replicate real-world situations (like a race) to predict possible outcomes.
Supervised Learning	A type of machine learning where the model is trained on labelled data, meaning the outcomes or answers are known and the model learns to map inputs to those outputs.
Testing Data	A data set that is used to evaluate the performance of a trained machine learning model. It helps ensure that the model generalises well to unseen data.
Training Data	The data used to train a machine learning model, allowing the model to learn patterns and relationships.
Underfitting	A situation where a machine learning model is too simple and fails to capture the underlying patterns in the training data, leading to poor performance on both training and new data.

Motorsport/F1 Terms

Term	Definition
Delta	The lap time difference between each lap
Drag Reduction System (DRS)	A system that reduces aerodynamic drag on a car by adjusting the rear wing, allowing it to go faster, usually used when a car is behind another to help with overtaking.
Driver Specific Behaviours	The unique driving style of each racer, affecting car performance, tyre wear, and lap times.
Energy Recovery System (ERS)	A system in Formula 1 cars that recovers energy during braking and stores it to be used later for extra power.
Free Air/Clean Air	A term used to describe the clear space on the track ahead of a car, without any other cars obstructing or affecting its lap time.
Lateral load	It is the side-to-side force that tyres experience due to steering input, affecting the car's grip and handling during a turn.
Overtaking	The action of one car passing another on the racetrack.
Pitstop	A brief stop made by a car during a race to change tyres, refuel, or make repairs.
Race strategy	A set of plans made by the team to optimise a car's performance during a race, including when to pit, tyre choices, and fuel management.
Safety Car (SC)	A car that temporarily drives on the track during a race to slow down the drivers in case of an accident or dangerous conditions.
Slip angle	The angle between the direction in which a tire is pointing and the direction it is actually moving.
Track conditions	The state of the racetrack, including surface type, weather, temperature, and any other factors that can affect performance.
Track evolution	The changes in track conditions during a race, such as rubber build-up, temperature, and grip, which affect lap times.

Tyre degradation	The process where a tyre’s performance decreases over time as it wears down, impacting grip and lap times.
Undercut	A race strategy where a driver pits earlier than a competitor, with the aim of taking advantage of fresher tyres. The idea is that the driver can set faster lap times while on newer tyres and gain position through the time lost by the competitor when they eventually pit. This tactic is normally used to overtake a competitor who is ahead on track.
Virtual Safety Cars (VSC)	A system used in some races where cars slow down to a specified delta time due to incidents on the track without the need for a full safety car deployment.