

AIBased Race Strategy Assistant and Car data Monitor

Simon Pontin

**Datateknik, master
2023**

Luleå tekniska universitet
Institutionen för system- och rymdteknik

[Denna sida har avsiktligt lämnats tom]

“A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.” — Alan Turing

ABSTRACT

In the world of motorsport, it is not only the driver's skill that determines the outcome of a race. Race strategy and car setup are two main factors that determine if the driver is competitive or not. Thus, this project focuses on optimizing the strategy part. In some of the motorsport series racing strategy includes tyre compound choices and timings on when to do a pit stop in order to change tyres or repair the car. Optimizing these tasks allows to create an optimal race strategy. Such strategies are data-driven and rely on records of tire usage from practice sessions and modeled data using regressions. The constants produced from the model can then be used, as showed in the project, in quadratic optimization problem in order to create different base strategies. To automate the strategic choices that will be further used in a race can be used artificial neural networks (ANNs). In this project an ANN has been successfully trained on the generated data from the strategies in a simulation environment. Additionally, the whole process of creating a AI-based race strategy assistant for race engineers that starts from the creation of the regression model, followed up by obtaining an optimization formula and finally designing the ANN is introduced. The data collected for the race strategy assistant development have been generated using the driver-in-loop method. The outputs of the race strategy assistant in this project are decisions of tyre compound stint lengths/timings of pit stops and what combination of tyre compounds to use during a race in order to minimize the race time. The evaluation results of the race strategy assistant demonstrated the overall improvement in reducing the racing time by finding the optimal tyre race set.

CONTENTS

CHAPTER 1 – THESIS INTRODUCTION	1
1.1 General project introduction	1
1.2 Race simulation	2
1.3 Research scope and goal	3
CHAPTER 2 – LITERATURE REVIEW	5
2.1 Related work	5
2.2 Conclusion	6
CHAPTER 3 – METHODOLOGY	7
3.1 Data collection	7
3.2 AI data	8
3.3 Data pre-processing	12
3.4 Design	13
CHAPTER 4 – CAR DATA MONITORING	21
4.1 Car data collection	21
4.2 Car data monitoring	22
CHAPTER 5 – RESULTS	25
5.1 Regression model	25
5.2 Linear programming	28
5.3 Neural networks	29
CHAPTER 6 – DISCUSSION AND CONCLUSION	37
CHAPTER 7 – FUTURE WORK	39
REFERENCES	41
Appendices	43
CHAPTER A – SIMULATION DATA	45

ACKNOWLEDGMENTS

The creation of this thesis report is to complete my five year master study in civil engineering computer science at Lulea Technical University. It would not have been possible without my fantastic supervisors Anton Koval and Mats Wiklander. Also I would like to give a big thanks to Syntronic for the opportunity to come to them and do my master thesis.

Luleå, January 2023
Simon Pontin

Thesis Introduction

1.1 General project introduction

In order to be successful in Formula 1 (F1) racing it is required to have a really good racing car, a skilled driver and a good racing strategy. To be able to develop a car that will be overall the fastest on track, the drivers have to know where they lose/gain time on the track and for the strategist to be able to plan a maximized racing strategy. For this purpose F1 teams collect huge amounts of telemetry data which allow them to optimize their racing strategies. According to the Mercedes F1 team's website [2] the F1 teams use two main simulation approaches in order to prepare for a race. Driver-in-loop and computer simulations. The driver-in-loop approach is when a test driver is driving in a realistic simulator in order to generate data from the track and car while practicing his driving skills on the track. This approach is aimed at preparing both the engineering team with data and also the drivers for the upcoming race weekend. The computer simulations approach is mainly done for the engineering team to prepare strategies and the car setup before the race.

This project focuses on the driver-in-loop approach which allows to collect racing telemetry data from a virtual car model that is driven on different virtual race tracks. The project will focus on using the simulated telemetry data to prepare a maximized race strategy with the help of the *developed* AI-based race strategy assistant. The developed strategy builds on finding the best opportunities to do a pit stop in order to put on new tyres, to minimize the race time. The combination of tyre compounds that will be used during the race are also considered as a big part of a race strategy. The different compounds will degrade at a different pace and will provide different capabilities in grip to enable desired pace. For example, one compound is thus the fastest but will wear out faster, which means that after a certain time it will no longer give enough grip to provide good lap times. In opposite, another compound will be the most resistant compound for degradation but will initially be the slowest. The stint length of the tyre compounds that are used during the race will reflect on the race duration time. A stint length is

the amount of laps driven on a tyre compound before doing a pit stop (change to other tyres). The developed race strategy assistant will help the racing team to predict timings of the pit stop, how many pit stops will be present and which tyre compounds will be used to achieve the most beneficial trade off between long lasting compounds, pit stop time and faster tyre compounds.

1.2 Race simulation

The simulation environment considers many aspects that will define a driver's lap times and finally the total race time. The cars have unique chassis design, engines, engine modes (can be set to give more effect or be set in a more sustainable way in order to have a longer life time). The cars are also expected to be slower with added mass like fuel. The mass could differ under a race due to amount of fuel that is consumed during the race. F1 car's mass is varying during the race as a function of fuel consumption, e.g. at the beginning of the race all cars have full tank which is consumed during the race, meaning that the during the final laps cars are very light. The drivers have unique driver styles and are differently experienced, while tracks have different characteristics as well, including different length of the pit lane.

In general a simulated race time can be described as [9]:

$$t_{race}(l_{tot}) = \sum_{l=1}^{l_{tot}} t_{lap}(l) \quad (1.1)$$

where

$$t_{lap}(l) = t_{base} + t_{tyre}(a, c) + t_{fuel}(l) + t_{car} + t_{driver} + t_{grid}(l, p) + t_{pit, in-lap/out-lap}(l), \quad (1.2)$$

t_{base} is the reference lap time achieved under the best possible condition thus the optimal weather, optimal tyres, the best driver and the best car for the specific track. t_{tyre} describes the increasing time that tyre degradation will give to the lap time and how fast the tyre pace is initially with respect to the tire age a and compound c . Depending on how much fuel there is in the car that will add mass t_{fuel} gives the added fuel mass time. The skill off the driver t_{driver} will also have an affect the lap time. The driver can drive in a faster way through the track in a lap but also drive in a way that saves the tyre from degrading to fast, which will conserve faster lap times longer. A driver's starting position sets the conditions of how much traffic there will be ahead of the driver. Cars in front can block optimal race lines in curves or other parts of the track, which will negatively affect the lap time. This is denoted by t_{grid} which considers a driver's start position. The time gained/loosed due to the ability of the car is described as t_{car} . $t_{pit, in-lap/out-lap}(l)$ is the time that is added to the lap during a pit stop. The laps before and after the pit stop will be affected from the pit stop. In the first lap after the pit stop, the tyres will not have the optimal temperature to produce the expected compound grip and the expected lap time. The last lap before the pit stop a driver would max out the tyre compound

and not drive in a degradation economic way.

The race simulation can be utilized before the race in order to generate tyre data such as possible tyre pace and degradation model in order to calculate a base strategy. The simulation is also useful for confirming the basic strategy and generate data for the race strategy assistant AI to train on. In the case of this study, the simulation will also be used to simulate the actual race, since the author does not have access to test the AI on a real race. The simulation of the race will show how the base strategy will be re-evaluated by the AI (in case of different events).

1.3 Research scope and goal

Currently the simulation environment is not considering any calculated real world data from track tyre degradation and pace. The simulation has some recommended strategies that are based on unknown quantities. They could be hard coded in the simulation, that is unknown and then not applicable for F1-teams, since the underlying factors of a strategy are based on many different things. The recommended strategies are not bound to a certain driver's driving style since the recommended strategies are always same and not based on previous track data from a specific driver. It is also interesting to know how the strategy based on previous data can change during the race depending on factors such as higher degradation than expected due to example temperature change or a current situation based driving style on the race day. This would not be possible to determine with just a pre-defined hard coded race strategy. This also eliminates the use of probability models such as different stochastic mathematical models like Markov and Gaussian processes. These models have no repeatability that is specified against the specific situations that could occur during the races. This would make it hard to verify the underlying reason for the decisions of the model. Meaning that some possibilities will be over represented by forehand. The idea in this research is to have x base strategies and during the race depending on the factors get dynamic strategy recommendations. The base strategies are based on data that is generated during practice. Which means that they are adopted to driving style and different track factors that affect the racing conditions. The strategies can be controlled in the simulation with tyre choices and which laps the driver should do the pit stops. It is also possible to control factors like weather and car setups in the simulation. No other inputs are possible. Which means that it is not possible to use algorithms like the Monte Carlo approach in order to evaluate different strategy choices. This means that the strategies can only be tested and generated with a driver driving in the simulation. This is highly time consuming and makes it harder to evaluate strategies. This also makes it harder to get strategies that considers randomness such as safety car deployment during the race. In order to then be as efficient as possible, race strategies should be able to be accomplished with as few sessions as possible due to time limitations. This means that instead of doing hundreds or thousands strategy attempts with a Monte Carlo approach that is not possible in the current simulation, the best predicted strategies for different conditions are evaluated.

The base for the strategies only requires a few session in different conditions in order to give representative track and tyre data.

To the background of the above mentioned, this project is aiming to create a optimal dynamic race strategy assistant to give strategy suggestions considered the fact of limited control of the simulation environment. The idea is to create the race strategy assistant based on machine learning methods. In that way it is possible to map relations between inputs such as status in race, driving style and car telemetry data. Which otherwise with other models as mentioned above can be hard to do. The idea is to fetch a batch of data once or twice per lap in order to get strategy decisions such as, timings on pit stop and what tyres to put on, suggested. Being not dependent on the size of the batch size is another advantage with the usage of a machine-learning (ML)-algorithm based race strategy assistant. Its predictions could work simultaneously for different drivers where it will consider each individual data input differently and independently and produce specified individual outputs. The solution can work both in the simulation environment (pre-race weekend) and with data from the real world race weekend, which gives good possibilities for a race team to prepare for a race and generate valuable data for strategies. The race strategy assistant can be tuned and retrained so that it could work for other racing series, but this report focuses on the F1 series. Assuming that refueling under pit stops is not included during the series, which is the case in F1.

CHAPTER 2

Literature review

2.1 Related work

In the recent years, the demand for statistics and analyzing data in sports has increased drastically [3]. The reason is the refinement of helpful technology combined with the demand of visualizing strength and weaknesses among opponents in order to plan strategies. Or look at the data internally in order for development to go in the right direction. The data is also for the fans that participate in watching and betting. Except for visualizing the data, there are possibilities to use modern methods such as regression or ML-algorithms to predict future outcomes from what has happened in the past. Most of the past research in this area is not public due to commercial interest. But there are some non-commercial works that uses ML-algorithms that can be investigated and have a further deeper meaningful evaluation of data in sports.

2.1.1 Motorsport

The authors of [9] researched that a neural network (NN) can be trained to take decisions on when to pit and what tire compounds to use. The training data is gained from [8] which is based on historic real F1 data from [5]. In their work they used a self made simulation instead of a more realistic environment to apply the AI to it. However, the database that the author is using lacks detailed information that would be essential in reality in order to make a strategy. Moreover, the model is not optimized for maximal accuracy. Liu and Fotouhi [10] used an artificial NN and Monte Carlo tree search to replace lap time simulations and predict solutions to unexpected scenarios that could occur during the race. Also pre-race decisions are handled by the approach. By applying the used techniques they are able to reach faster and better quality decision making. The problem of managing energy to the best in the Formula E cars still withstands.

The resulting positions in a NASCAR race were predicted with the help of a probabilistic model in [11] and the resulting positions in the last 4 races of the F1 season

were predicted in [13] using a NN and data from the earlier races of the season. Both rely heavily on a big portion of earlier results of the season, which means they are only relevant at the end of the season when enough statistics of the winning conditions have been generated. Tulabandhula and Rudin [14] demonstrated a real-time decision system for NASCAR rank changes after changing different number of tyres in a pit stop. Many different machine learning techniques were tested, where the regression analysis method LASSO was the best performing technique followed by support vector regression. Their research only considers one part of a complete race strategy and can be supplemented with more detailed race data. The prediction accuracy could also have been higher. Authors in [4] investigate machine learning software to predict track position changes, tyre change decisions, pit crew performance and tyre wear variation in order to achieve a better finishing position. They found that a driver's performance in the earlier stages of the race correlates to the performance in the later stages of the race. The article does not consider data from tyre wear, since sensor for measuring this is prohibited. Instead they calculate tyre wear based on lap times. The researchers in [1] were predicting NASCAR finishing positions using a statistical model by considering race start position and drivers experience as most important model variables. The article is not considering any data from the cars in the model, which in reality is highly relevant when predicting a outcome of a race. Pit-stop strategies are also a major factor in a race which is not either considered in the model used.

2.1.2 Other sport areas

It is showed in [7] that different data mining techniques can be used to predict results in sports. The techniques examined are: Bayesian method, support vector machines, artificial NNs, logistic regression, decision trees and Fuzzy system. The pros and cons for each respective technique is presented for comparison of each prediction model used. Some disadvantages mentioned is the variation and lack of details in the data sets used when comparing models. Also lack of high accuracy suggested that further research was needed for some of the models. Purucker [12] examined a variance of artificial NN strategies to predict winners in NFL matches. The article concludes that a model with back propagation will predict better on the selected data compared to self-organized map model. The article is written 1966 and since then the machine learning techniques and algorithms have been developed a lot and the collection of sports data has been increased.

2.2 Conclusion

From the literature survey it can be concluded that research that compares machine learning methods achieves the best prediction result by using artificial NNs. Most of the earlier work studied in the field focuses on predicting results in the form of win or loose. Only one out of all earlier work studies articles applies the predictions under the event. While the majority makes model predictions after the event has finished.

CHAPTER 3

Methodology

3.1 Data collection

The easiest and best method to get realistic and trustworthy data would be to get collected telemetry data from F1 teams. Unfortunately that is not publicly available due to high secrecy between the teams. The sport relies on the difference in cars and tactical performances. A team would not want to share detailed data in order to be better than opponent teams. This makes it a harder task to produce a strategy that would work in reality. Since a F1 car is really expensive, it is not an option to buy one for data collection. The best approach is to by simulation come as close as possible to the reality. There is a publicly available [8] database with some timings data collected from real races. The API contains for example information about drivers, finishing result, positions during the race, lap times and timings of pit stops. Any telemetry data describing the state of the car and tyres are not present in the API. And the data points are collected at just a few timing points, meanwhile a realistic data collection would be with at least with a couple of seconds as interval. Since F1 teams have a strict budget to follow and rules about length of track time usages, they also need to use simulation methods in order to plan strategies and prepare the drivers for the upcoming tasks. As presented in [2], drivers take part in simulations with the driver-in-loop approach. The simulation used in this project is the F1 2021 game. The game is developed to be as close to the reality as possible. The cars simulate their reality strengths and weaknesses. The game has many options for different car setups, example break bias settings, wings setup for more or less down force etc. The different impacts on the cars engine and parts during the race are also considered and simulated. Differences in tyre compounds are simulated in a way so they are effected by driving style, track, temperature and other external events. The real structure of a race weekend is also simulated, meaning that it is possible to prepare for a race with practice sessions in the same way as in reality.

Telemetry data from the simulation are gathered by opening a port and send data via

a UDP stream to a client [15]. The client collects different categories of data packets example motion packet, event packet car setup packet etc. The client is also recognizing the different sessions of data that are recorded (race, practice, qualify) and saves the data. The data can then be extracted in the form of a CSV file from the client, and be stored locally or in the cloud. The CSV file that is produced by the client contains 193 data entries. Every row with entries/columns is gathered by the UDP stream on different frequencies. In some parts of a race there are 20 rows of data entries per second and in some parts 10 rows per second. Some junk data are also produced. All entries that can be retrieved from the UDP stream from the simulation can be found in Table 4.1 in appendix. The data collection with the client has been tested about 50-100 times and has every time been producing a valuable and good data flow that can be use for the race strategy assistant.



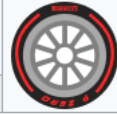


No.	Compound details					Tread	Driving conditions	Grip	Durability	
C1		Hard (White)		Medium (Yellow)		Soft (Red)	Slick	Dry	5 – Least grip	1 – Most durable
C2									4	2
C3									3	3
C4									2	4
C5									1 – Most grip	5 – Least durable
–				Intermediate (Green)		Treaded	Wet (light standing water)	—		
–				Wet (Blue)			Wet (heavy standing water)	—		

Figure 3.1: Figure showing the different tyre compounds that was present in the 2021 season of F1 [16] and in the simulation. The last two compounds in the figure were not used during the research.

3.2 AI data

When constructing a NN it is important to carefully select input features in order for the AI to predict in the best possible way the output features. The input features should be representative for in this case the current state of the race and the status of the car and tyres. Then the AI will learn the indications of data when it is time to pit and which tyres to choose. We want to create a relevant relationship between the input and the output features for the NN. To be able to produce and clean the collected data to get the selected and relevant input features, we have to pre-process the data. Which can be read about in section 3.3.

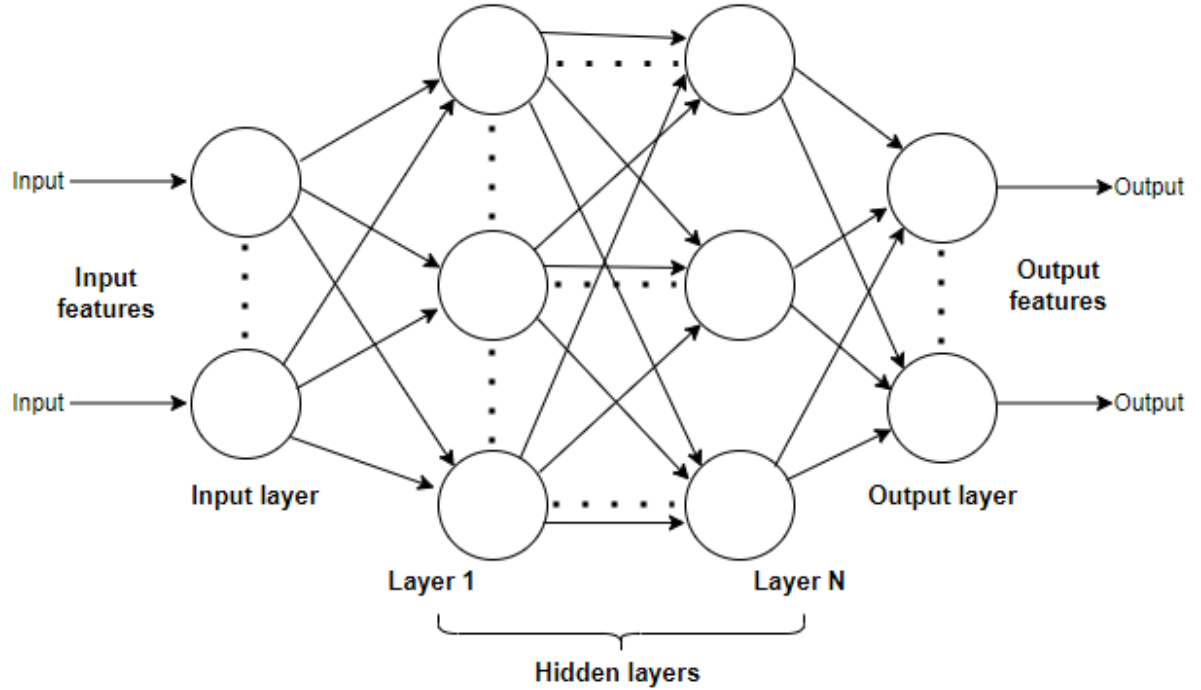


Figure 3.2: The figure shows how the input features are used as input to the NN and which layers it goes thorough in order to accomplish output in form of output features. The dotted lines describes that there is a set of units.

In section 3.2.1 and 3.2.2 the input features and the output features for the two NNs that are created in this project in order to get race strategy decisions.

3.2.1 Pit stop decision

The pit stop NN features was defined from testing and observing F1 races. Front right and left wing damage are not tied to the tyre quality and the need to pit for that reason. Those features are relevant since the damages effect the lap times in the way that the car cannot function properly through especially corners. The safety car feature is relevant since when safety car is deployed, drivers lose much less time when doing a pit stop. This is because of the speed limit during a safety car.

Input features		
Feature	Type	Value range
Progress	Numerical	[0.0,1.0]
Position	Categorical	{0,1,..,20}
Track	Categorical	{Austria,..,Suzuka}
Tyre Compound	Categorical	{16,17,..,20}
Wheel speed0	Numerical	[1,2,..,100]
Wheel speed1	Numerical	[1,2,..,100]
Wheel speed2	Numerical	[1,2,..,100]
Wheel speed3	Numerical	[1,2,..,100]
Wheel slip0	Numerical	[-1,..,1]
Wheel slip1	Numerical	[-1,..,1]
Wheel slip2	Numerical	[-1,..,1]
Wheel slip3	Numerical	[-1,..,1]
Track temperature	Numerical	[0,1,..,40]
Safety car status	Categorical	{0,1,2,3}
Tyre surface temperature0	Numerical	[0,1,..,160]
Tyre surface temperature1	Numerical	[0,1,..,160]
Tyre surface temperature2	Numerical	[0,1,..,160]
Tyre surface temperature3	Numerical	[0,1,..,160]
Tyre inner temperature0	Numerical	[0,1,..,110]
Tyre inner temperature1	Numerical	[0,1,..,110]
Tyre inner temperature2	Numerical	[0,1,..,110]
Tyre inner temperature3	Numerical	[0,1,..,110]
Tyres age	Numerical	[0,1,..,25]
Tyre wear0	Numerical	[0,1,..,100]
Tyre wear1	Numerical	[0,1,..,100]
Tyre wear2	Numerical	[0,1,..,100]
Tyre wear3	Numerical	[0,1,..,100]
Tyre damage0	Numerical	[0,1,..,100]
Tyre damage1	Numerical	[0,1,..,100]
Tyre damage2	Numerical	[0,1,..,100]
Tyre damage3	Numerical	[0,1,..,100]
Fuel in tank	Numerical	[0,..60]
Lap Time	Numerical	[0,..150]
Start grid position	Categorical	{1,2,..,20}
Front left wing damage	Numerical	[0,..,100]
Front right wing damage	Numerical	[0,..,100]

Table 3.1: Table showing the input to the the NN deciding whether the driver should pit or not pit this lap.

Output		
Feature	Type	Value range
Pit/Not pit	Categorical	{Yes,No}

Table 3.2: Table showing output from the NN deciding when to pit stop.

3.2.2 Tyre change decision

The tyre change NN features almost only contains data points that are directly or indirectly affecting the tyres. Position and progress are the features that are not directly affecting the tyres. They were selected since they can describe a state of the race that can indirectly be connected to the tyre choice.

Input features		
Feature	Type	Value range
Progress	Numerical	[0.0,1.0]
Position	Categorical	{0,1,..,20}
Track	Categorical	{Austria,..,Suzuka}
Track temperature	Numerical	[0,1,..,40]
Tyre surface temperature0	Numerical	[0,1,..,160]
Tyre surface temperature1	Numerical	[0,1,..,160]
Tyre surface temperature2	Numerical	[0,1,..,160]
Tyre surface temperature3	Numerical	[0,1,..,160]
Tyre inner temperature0	Numerical	[0,1,..,110]
Tyre inner temperature1	Numerical	[0,1,..,110]
Tyre inner temperature2	Numerical	[0,1,..,110]
Tyre inner temperature3	Numerical	[0,1,..,110]
Tyres age	Numerical	[0,1,..,25]
Tyre wear0	Numerical	[0,1,..,100]
Tyre wear1	Numerical	[0,1,..,100]
Tyre wear2	Numerical	[0,1,..,100]
Tyre wear3	Numerical	[0,1,..,100]
Tyre damage0	Numerical	[0,1,..,100]
Tyre damage1	Numerical	[0,1,..,100]
Tyre damage2	Numerical	[0,1,..,100]
Tyre damage3	Numerical	[0,1,..,100]
Lap Time	Numerical	[0,..150]

Table 3.3: Table showing the input to the NN deciding which tyre to put on next.

Output		
Feature	Type	Value range
Next tyre compound	Categorical	{soft,medium,hard}

Table 3.4: Table showing output from the NN deciding what tyre to put on next.

3.3 Data pre-processing

In this project data sets were generated and used for the further analysis (see Table 1.1 in appendix for data points). The stored data files both contain finished races and unfinished races (due to crashes). This is also to make the data collection as efficient as possible and thereby maximize the use of the research time. The unfinished races are still representative as data to the AI since they describe sub parts of a whole race, and thereby track data and car data that is representative for a sub part of the race. The data files are saved in a common folder with the track name followed by a index. The data that is inputted to the regression model in order create constants for tyre pace and tyre wear is generated during a few laps in a practice session. The idea is to generate a representative amount of lap data (around 10 laps) to get as good constants for the tyres as possible. The constants will be generated from a regression model that takes as input the tyre data from the practice session (see Figure 5.2). The NN inputs are generated in a full race simulation using the optimal strategies that are calculated by the linear programming formula that uses the constants from the regression model.

All the data that are collected and stored from the client are not necessary for the NNs and regression model. As shown in table 3.1 and 3.3 there are only certain defined data features that are needed for input to the NNs from the CSV files. In the pre-processing of the regression model combined with the linear programming formula we start with stripping out all features that are not necessary from the CSV file. Then we remove all invalid data such as rows that have -1 as values (client error rows) and also remove duplicate data. For the NN models we do the same as for the regression model. In the NNs we also refactor the laps into race progression. This is because we want to generalize the race progression since tracks have different many laps and different lengths of laps.

Before fetching the NNs with the input features, we split the data set into 80% training data, 10% validation data and 10% test data. Before the data set is split, it is randomized. The data split is done in order to have some portion of data to train the NNs on and some for validating the model on with metrics during training. The test data are used for testing the NNs on new data for final model validation. The data is randomized to improve the learning quality of the NNs.

3.4 Design

The project was divided into several individual connected parts in order to prove the concept of a fully functional and complete system that can be used in reality. The system parts of this project can be seen in Figure 3.3. The yellow area in the Figure 3.3 represent the storing and visualization part of the project, where there is an API that controls the data flow to and from the database. There is also a frontend and a interface to communicate with a real car using a OBD-2 reader. The frontend both visualize the real car data and also the data from the NNs, thus their predictions. The database can store necessary data to be used in the NNs. More about this part of the project is described in section 4. The interaction and data flow between the blue and the red area can be seen in Figure 3.4 and is described more in detail in the subsection 3.4.1.

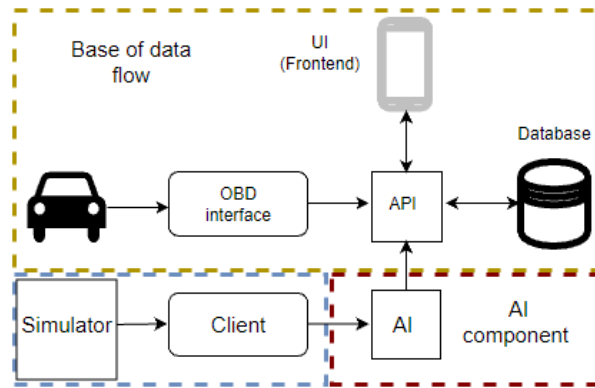


Figure 3.3: Overall project design showing the whole project in different parts. The frontend part is connected API, simulator connected to AI and AI connected to simulator.

3.4.1 General AI design

The data flow in Figure 3.4 is representing the blue and the red part in the overall design Figure 3.3. The figure shows the data that is collected as described in section 3.1 and then is preprocessed as described in 3.3, before being used in the regression model that is described in section 3.4.2. The regression model will then produce data constants that are being used in the linear programming formula, that is described in section 3.4.3. From the linear programming formula, different strategies are produced, which are used in the simulation in order to create strategy data for the NNs that are both described in Section 3.4.4. Before the strategy data is used by the NNs, they are preprocessed and some features are selected. The features of the different NNs is shown in 3.2.2 and 3.2.1.

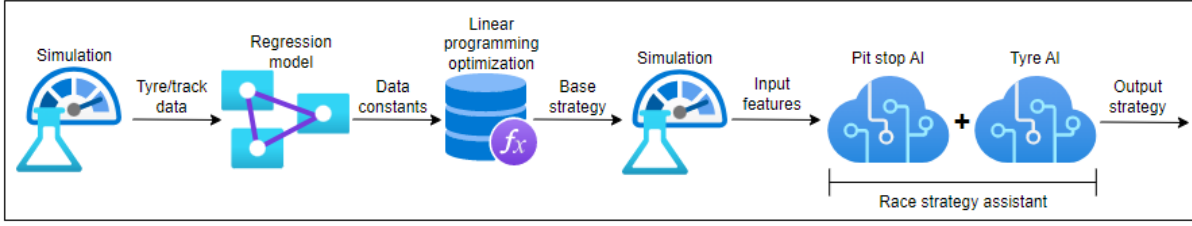


Figure 3.4: Data flow and sub parts that are present in the race strategy.

3.4.2 Regression model and tyre constants

For the linear programming formula we need a constant for the tyre pace. The pre-processed data consists of x and y velocity vectors that have to be recalculated into common xy velocity vectors. We do this by looping through all the x and y values and calculate the resulting vector using:

$$v_{tot} = \sqrt{v_x^2 + v_y^2} \quad (3.1)$$

We then use a linear regression model on the resulting pace vector for each lap. This is done by splitting the data rows at a start and a end of the lap from the laps column and then apply those index on the velocity column. The different sub parts of the velocity column representing an individual lap is then one by one used as input to the regression model together with the same indexed sub part from the distance data column. The distance has to be reshaped and transformed in order to fit the model. The model used is a ordinary least squares (OLS) 7th degree polynomial linear model. This model produced the best mapping and prediction of the velocity vs distance curve, compared to Lasso, LassoLars, Bayesian and Ridge regression. OLS method for linear regression can be described as:

$$y = \beta_0 + \sum_{j=1}^p \beta_j x_j + \epsilon \quad (3.2)$$

where y is the target (dependent variable), x_j is the independent variable, β_0 is the intercept, β_j is the slope and ϵ is the error. The goal for the method is to minimize the sum of difference between the predicted value and the actual value:

$$\sum (y_i - \hat{y}_i)^2 \quad (3.3)$$

where y_i is the actual value and \hat{y}_i is the predicted value.

For each pace constant representing each lap of data we calculate an average resulting pace constant from the mean. This is done for every compound which is stored in different session files.

The tyre degradation constants are calculated by the gradient of the tyre wear:

$$y = y_n - y_0 \quad (3.4)$$

where y_n is the last tyre wear measured. This is done for every tyre tyre0,...,tyre3, and the tyre that has the highest degradation constant is used, since it will be the bottle neck for the tyre set. It could differ from track to track which tyre has the most wear, due to the track layout. A constant is calculated for every tyre compound like the pace constant.

3.4.3 Linear programming

In order to give the NNs as maximized and good data input as possible, so that they will learn as optimal races strategies as possible, we aim to determine the optimal stint lengths for each tyre set. When calculating a strategy with optimal stint lengths we also have to consider the fact that only three of five tyre compounds are available at the race and that every driver needs to use at least two different compounds before completing the race. In order to determine the optimal stint lengths we need to first determine all possible and relevant combination of tyre sets during a race. From the possible combination we can calculate the optimal stint length for each combination. This approach will give the racing engineers a comparisons between different possible tyre strategies for the AI to train on. Since it gives an insight in how many pit stop and what tyres combination to use in order to minimize the race time. The possible tyre sets combinations (without repeated values) can be defined as:

Relevant pit stop strategies	
Strategy	Tyre combinations
1-stop strategies	{SM, SH, HM}
2-stop strategies	{SSM, SSH, SMM, SMH, SHH, MMH, MHH}
3-stop strategies	{SSSM, SSSH, SSMM, SSMH, SSHH, SMMM, SMMH, SMHH, SHHH, MMMH, MMHH, MHHH}

Table 3.5: Table is showing all the relevant tyre combinations that can be used with respective pit stop strategies. H stands for hard, M for medium and S for soft

The optimal stint lengths for a tyre set combination will give the optimal race time for that tyre set. From equation 1.1 it can be observed that a driver's race time is defined from the sum of the lap times. Observing equation 1.2 it shows that t_{tyre} , t_{grid} , t_{fuel} and $t_{pit,in-lap/out-lap}$ are the time components that the lap time is dependent on, when excluding driver skills, the capacity of the car and lap time achieved under best conditions. t_{fuel} and t_{grid} are occurring independently of what the race strategy is, so they have not to be considered when calculating a optimal stint length. $t_{pit,in-lap/out-lap}$ will represent a fixed time since the amount of pit stops per set is known. Thus our race strategy formula is only dependent on the t_{tyre} time parameter which is considering the

tyre pace and degradation, which we calculated in section 3.4.2. For the determination of the optimal stint length for tyres in a tyre set combination we will use linear to solve a Mixed-Integer Quadratic Programming (MIQP) optimization problem. For that we will use the python *cvxpy* package which is a language model for solving convex optimization problems [6]. The MIQP formula that is used for calculating the optimal stint length is derived from:

$$\min t_{\text{race}}(l_{\text{tot}}) \hat{=} \min \sum_{l=1}^{l_{\text{tot}}} t_{\text{tyre}}(l). \quad (3.5)$$

where the goal shows that the objective is to estimate a minimal sum of t_{tyre} . If we reformulate the formula by splitting the race into certain amount of stints N we get:

$$\min \sum_{i=1}^N \sum_{a=1}^{\alpha_i} t_{\text{tyre}}(a, c_i) \quad (3.6)$$

where

$$\sum_{i=1}^N \alpha_i = l_{\text{tot}} \\ \alpha_i \in \mathbb{N}^+ \quad \forall 1 \leq i \leq N.$$

Stints are represented by a stint index i where $\text{pit stops} = N - 1$, compounds are represented as c_i for stint i and stint length as α_i .

In order to derive the equation so that t_{tyre} will be dependent on the tyres characteristics for the current track and use the tyre constants from section 3.4.2, we introduce a linear model. Which shows t_{tyre} dependency to the tyre age a , k_0 which describes the tyre pace and k_1 which describes the tyre wear:

$$t_{\text{tyre}}(a, c) = k_0(c) + k_1(c) \cdot a. \quad (3.7)$$

It is expected that a softer tyre will initially be faster than a harder compound thus k_0 will be smaller, but will have a greater tyre wear which will reflect on a higher k_1 than for a harder compound. If equation 3.7 is used in equation 3.6 we get:

$$\sum_{i=1}^N \sum_{a=1}^{\alpha_i} t_{\text{tyre}}(a, c_i) = \sum_{i=1}^N \left(k_0(c_i) \cdot \alpha_i + k_1(c_i) \sum_{a=1}^{\alpha_i} a \right) \quad (3.8)$$

By using the Gaussian sum we can rewrite equation 3.8 as:

$$\sum_{i=1}^N \sum_{a=1}^{\alpha_i} t_{\text{tyre}}(a, c_i) = \sum_{i=1}^N \left(k_{0,i} \cdot \alpha_i + k_{1,i} \left(\frac{1}{2} \alpha_i^2 + \frac{1}{2} \alpha_i \right) \right). \quad (3.9)$$

Since not all drivers have tyres that have been driven 0 laps we have to introduce a tyre

starting age $a_{s,i}$ into the formula:

$$\sum_{i=1}^N \sum_{a=1}^{\alpha_i} t_{tyre}(a, c_i, a_{s,i}) = \sum_{i=1}^N \left(\left(k_{0,i} \cdot (\alpha_i + a_{s,i}) + k_{1,i} \left(\frac{1}{2}(\alpha_i + a_{s,i})^2 + \frac{1}{2}(\alpha_i + a_{s,i}) \right) - \left(k_{0,i} \cdot a_{s,i} + k_{1,i} \left(\frac{1}{2} + a_{s,i}^2 + \frac{1}{2} + a_{s,i} \right) \right) \right) \right) \cdot \left(\right. \quad (3.10)$$

The equation in vector form:

$$\vec{\alpha}^T H \vec{\alpha} + f^T \vec{\alpha}^T \quad (3.11)$$

which follows the expression presented under equation 3.6, where

$$H = \begin{bmatrix} 0,5 \cdot k_{1,1} & 0 & \cdots & 0 \\ 0 & 0,5 \cdot k_{1,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0,5 \cdot k_{1,N} \end{bmatrix} \left(\right. \quad (3.12)$$

and

$$f = \begin{bmatrix} k_{1,1}(0,5 + a_{s,1}) + k_{0,1} \\ \vdots \\ k_{1,N}(0,5 + a_{s,N}) + k_{0,N} \end{bmatrix} \left(\right. \quad (3.13)$$

3.4.4 Neural networks

For the design and construction of the neural networks we use Keras API in TensorFlow. It offers the ability to easily have a trial and error approach when constructing the networks in order to achieve the best possible models. It is easy to add additional layers to the models and switch around with different activation functions, loss functions and optimizers.

Pre-processing model

As can be observed in Tables 3.1 and 3.3 we have different types of data in our input features for the NNs. Some features are integers, some numerical and some strings. They also have different value ranges. For the integers they have a finite length of value range that follow:

$$I_i \in \mathbb{N}^+ \quad (3.14)$$

in the defined value range $\{I_0, \dots, I_N\}$. The categorical string features are defined in the finite value range as $\{S_0, \dots, S_N\}$. The numerical float values follows:

$$F_i \in \mathbb{R} \quad (3.15)$$

in the defined value range $[F_0, \dots, F_N]$. When constructing a neural network it is necessary to construct a input tensor that holds the input features. This can typically be done in

the models input layer that can be seen in Figure 3.2. A tensor is a multidimensional array with a single data type. In our case as mentioned above, we have input features that differ in data type. Therefore a pre-processing model was constructed and used in the NN models as input layer.

We start by constructing a symbolic tensor for each input feature. The tensors are defined with their respective data type, feature name and expected shape. The shape defines the dimension of the input batch. The symbolic tensors are structured in a dictionary. We then go through all the tensors and take out only those that are numerical floats. A mean and variance are computed for all the numerical tensors and they are then passed through a normalization layer. The layer normalizes the tensors by shifting and scaling them into a distribution with standard deviation of 1 centered around 0. The normalization layer uses the calculated mean and variance to accomplish this. By normalizing the numerical float tensors we give them a common scale where no information is loosed and the value ranges are preserved. This helps the algorithm to model the data correctly and avoids any problems with combining different features. After the normalization all the numerical float tensors are concatenated into one tensor.

For the categorical tensor which is both strings and integers we one-hot encode them. For the strings we map them to integer indices. The integers maps to continuous ranges. Both methods are performing a table based lookup. We then create category encoding (one-hot) on both string and integer tensors where the sizes of the vocabularies are the number of tokens. Which means that we get x tensors with N in dimension where the dimension N is the vocabulary size.

The one-hot encoded tensor and the numerical float tensor are then concatenate into a single tensor with a single data type, which can be used as input to the NNs.

Pit stop decision

The input features for the NN for the pit stop decision is showed in Table 3.2.1. For the evaluation of the pit stop decision model, F_1 scored was used, which is defined as:

$$F_1 = 2 \cdot \frac{p \cdot r}{p + r} \quad (3.16)$$

where p stands for precision and r for recall. Precision describes the amount of true positives divided by the amount of all positives. Recall is the amount of true positives divided by the amount of positives that are supposed to be classified as positives. The value range of the F_1 score metric is $[0, \dots, 1.0]$ where 1.0 is the highest possible value of the score. All the true positives, false positives, true negatives and false negatives are used as metrics in a confusion matrix in order to visualize how the model has done its classification. Accuracy is also used as a metric, which counts the frequency of predicted y and true y matches and divides it by the total.

In order to avoid getting the model over trained and to dependent on the training data set (overfitted), the model was implemented to use early stopping. The early stopping functionality was chosen to monitor the validation data sets precision and recall value. A patience was set on the max of the validation precision and recall, if the max value has not become better after the x patience epochs set, the training of the models stops. Which means that the model could stop training before reaching the initial set training epochs. The early stopping also restores the best weights. To be able to train the model as carefully and good as possible, learning rate reduction was used. This enables the model to have a initial learning rate set in a way were the optimizer can update the weights "harder" in the beginning of the training. The more the weights get optimized the more in detailed they should be updated with smaller steps adjustments. The learning rate reduction functionality will monitor the validation data sets precision and recall values with x epochs patience. If not a new precision and recall max have been achieved during the patience epochs, the learning rate is reduced with a set factor. A minimum learning rate is also set.

The NN is using a optimizer function called root mean square propagation (RMSprop) in order to back propagate and update the weights. The RMSprop function is defined as:

$$s = \beta s + (1 - \beta) \delta_{\theta} J(\theta) \odot \delta_{\theta} J(\theta) \quad (3.17)$$

and

$$\theta = \theta - \eta \delta_{\theta} J(\theta) \oslash \sqrt{s + \epsilon} \quad (3.18)$$

where η is the learning rate, β is the momentum (discount gradient factor), ϵ is a factor preventing a zero division (eliminate the weights), $\delta_{\theta} J(\theta)$ is the gradients cost function and s is the moving average of the quadratic gradients.

The loss function used (binary cross entropy) in the NN in order to calculate how far the predicted value is from the actual is described as:

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \quad (3.19)$$

where y is the label, $p(y)$ is the predicted probability of the positive class and $\log(1 - p(y_i))$ is the log probability of it being the positive class. N is the number of rows.

Tyre change decision

The input features for the NN for the tyre decision is showed in Table 3.2.2. For evaluation metric categorical accuracy is used. Which counts the frequency of predicted y and true y matches. Then the total is divided with the count. The tyre change model also used learning rate reduction and early stopping functionality (mentioned in above section).

The NN is also using the same RMSprop optimizer as the pit stop NN, described by Equation 3.18. The loss function used is the categorical cross entropy. It is the same

modified loss function as the binary cross entropy described by Equation 3.19 but used for multi-class classification. It is defined as:

$$H_p(q) = -\frac{1}{N} \sum_i^N \sum_j^M y_{ij} \log(p(y_{ij})) \quad (3.20)$$

where N is the number of rows and M is the number of classes.

CHAPTER 4

Car data monitoring

The goal of the car monitoring part in this project is to showcase the result of the predictions of the AI and show that it can be integrated into a user interface that can be monitored by a engineering crew. Its purpose in this project is also to show that live data from a real car is possible to collect and analyze. The reason for the live data not participating in the AI is that there are only a few possible data points to collect from a regular car and those data points are not close to be representative for F1 purposes.

In order to capture, save and visualize data from the car a API was built. The API was created in python Django as a restful API. To control the data flow, endpoints for getting and saving the data in posgre-SQL database was created. Also endpoints for sending the stored data. A front end for displaying the car data from a real car and the AI in a user interface was created in Android Studios as a mobile application. The front end was written in react-native, which gives the alternative to build the app as a IOS application or a Android application. The front end was built as mobile application since it gives a mobility to the users, thus they can walk around on different places while monitoring the real car data and AI data.

4.1 Car data collection

For the collection of the data from the car a on-board diagnostics (OBD)-2 scanner was plugged into the cars data port. The OBD-2 scanner connects to the cars electronic control units (ECU)s and extracts the data with different frequencies. A computer with a OBD-2 interface is connected via Bluetooth to the OBD-2 device. The interface define which parameters to capture and builds a json frame and sends it to the API.

Real car data points	
Measure	Unit
Coolant temp	°C
Engine load	%
Intake pressure	PSI
Intake pressure	kPa
RPM	revs/min
Speed	kph
MAF	g/s
Throttle position	%
Fuel pressure direct	kPa
Fuel level	%
Barometric pressure	kPa
Ambient air temperature	°C
Accelerator position x	%
Accelerator position y	%

Table 4.1: Table showing the data that the project extracts from the car and monitor.

4.2 Car data monitoring

The monitoring of the real car data is shown when the user enters the live data page from the menu showed in 4.2(a). The user will then be met by the live data view showed in Figure 4.2(b). In Figure 4.1 the data flow from the car via the OBD-2, the API and the frontend is illustrated. It can be observed that the OBD-2 interface is posting the data to the API in a certain frequency and that the frontend that is illustrated in 4.2 is getting the data and updating the frontend. The user can also access a live location map that is using the Google API, showed in Figure 4.2(c).

```

08/Nov/2022 11:55:30] "GET /api/car-data-latest HTTP/1.1" 200 420
08/Nov/2022 11:55:31] "POST /api/car-data HTTP/1.1" 201 417
08/Nov/2022 11:55:32] "GET /api/car-data-latest HTTP/1.1" 200 417
08/Nov/2022 11:55:34] "GET /api/car-data-latest HTTP/1.1" 200 417
08/Nov/2022 11:55:37] "POST /api/car-data HTTP/1.1" 201 418
08/Nov/2022 11:55:38] "GET /api/car-data-latest HTTP/1.1" 200 418
08/Nov/2022 11:55:40] "GET /api/car-data-latest HTTP/1.1" 200 418
08/Nov/2022 11:55:42] "GET /api/car-data-latest HTTP/1.1" 200 418
08/Nov/2022 11:55:44] "POST /api/car-data HTTP/1.1" 201 418
08/Nov/2022 11:55:44] "GET /api/car-data-latest HTTP/1.1" 200 418
08/Nov/2022 11:55:46] "GET /api/car-data-latest HTTP/1.1" 200 418
08/Nov/2022 11:55:48] "GET /api/car-data-latest HTTP/1.1" 200 418
08/Nov/2022 11:55:50] "GET /api/car-data-latest HTTP/1.1" 200 418
08/Nov/2022 11:55:50] "POST /api/car-data HTTP/1.1" 201 420
08/Nov/2022 11:55:52] "GET /api/car-data-latest HTTP/1.1" 200 420
08/Nov/2022 11:55:54] "GET /api/car-data-latest HTTP/1.1" 200 420
08/Nov/2022 11:55:56] "GET /api/car-data-latest HTTP/1.1" 200 420
08/Nov/2022 11:55:56] "POST /api/car-data HTTP/1.1" 201 418

```

Figure 4.1: Shows the the data flow between the OBD-2 interface, the API and the frontend.

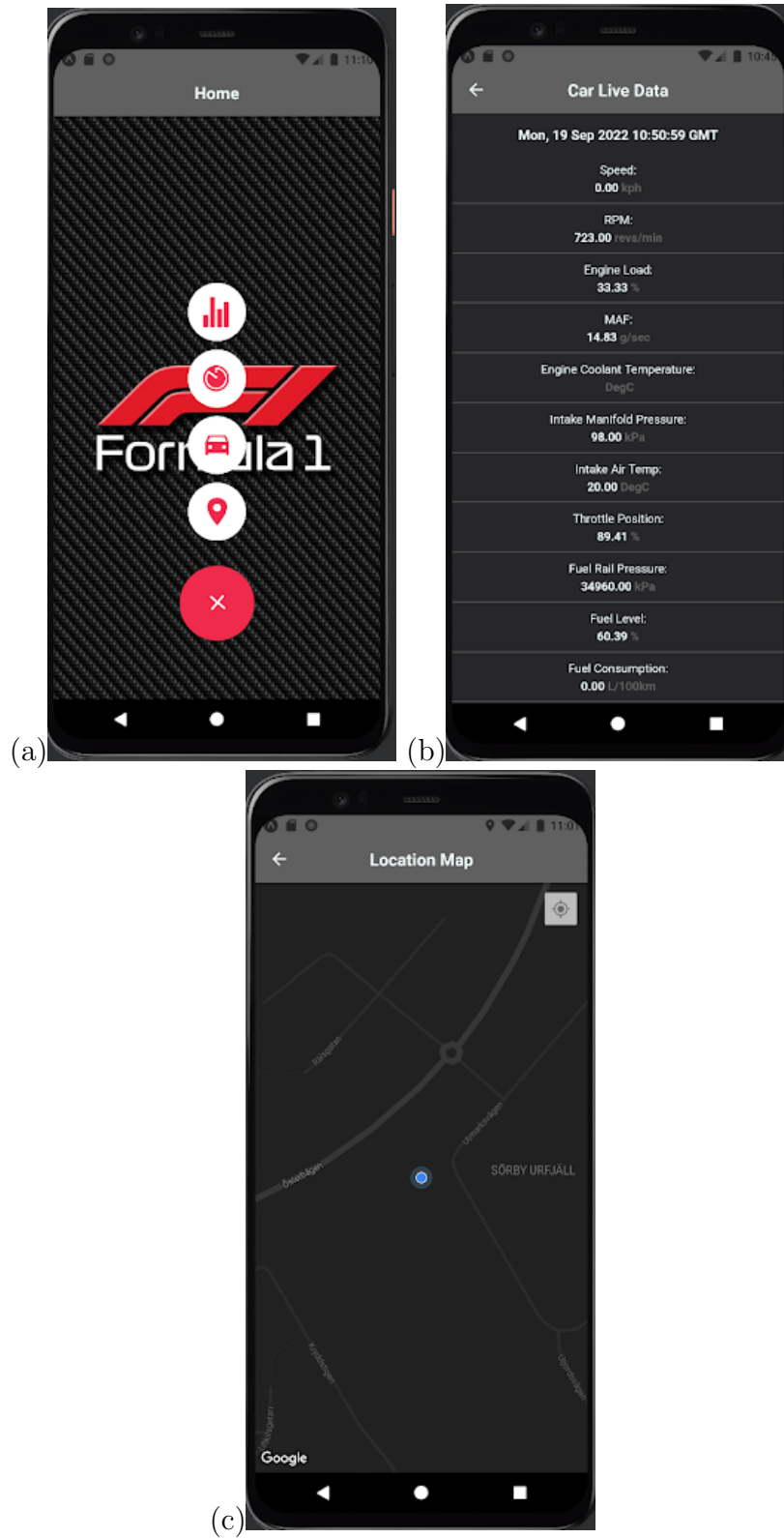


Figure 4.2: (a) Figure showing the app front-end with different navigation alternatives. (b) Figure showing the live data from the car that is monitored. (c) Figure showing the live location map that can be displayed in the application.

5.1 Regression model

By collecting tyre data from practice session we can observe the differences in the intercept coefficient per lap for each available tyre compound for example track Paul Ricard in Table 5.1. Initially the softer tyre is the fastest one per lap. But after 5 laps the medium tyre becomes the fastest tyre as the softer loses speed and the medium one slightly increases/have same pace. It can also be observed that on lap 12 the harder tyre is faster than the soft.

By driving the same amount of laps on each compound in a practice session, these intercept coefficients for each compound was summarize and the mean was taking. Then the lap length was divided by the mean pace, in order to achieve a time based constant. The resulting coefficients that was used in the linear programming formula for example track Paul Ricard is showed in Table 5.2. It can be observed that the medium tyre gives the smallest and best constant for 12 practice laps, followed by the soft. The hard tyre was showed to be the slowest. It can be seen in 5.1 that the soft tyres almost have twice as high tyre wear compared to the hard compound. The medium has slightly higher tyre wear than the hard compound but not significantly and when considering the pace of the medium as well, the medium can be assumed to be a really good tyre for the example track Paul Ricard. The data points in the Figure are the tyre wear for each compounds bottle neck tyre. An example of tyre wear of a specific compound for all four tyres can be seen in Figure 5.3. It can be observed that the tyre wear can have a big difference when comparing tyre to tyre on the same compound but it can be assumed that the pit stop decision will rely on the worst tyre, since the whole set will perform as good as its worst tyre. The difference of the tyre wear from tyre to tyre depends on the outline of the track. The constants calculated that are used in the linear programming formula from the tyre wear data from the practice session on Paul Ricard is shown in Table 5.3.

Tyre pace comparison first 12 laps			
Lap	Soft	Medium	Hard
1	65.652	65.540	64.923
2	65.707	65.439	64.655
3	65.582	65.405	65.273
4	65.711	65.512	65.259
5	65.381	65.782	64.494
6	65.458	65.635	65.264
7	65.324	65.225	65.166
8	65.087	65.502	64.960
9	64.701	65.343	64.912
10	65.058	65.112	64.828
11	64.759	65.484	64.151
12	63.900	64.638	64.361

Table 5.1: Table showing the tyre pace (m/s) difference result from the regression model for the 12 practice session laps on Paul Ricard. The fastest compound for each lap is in bold.

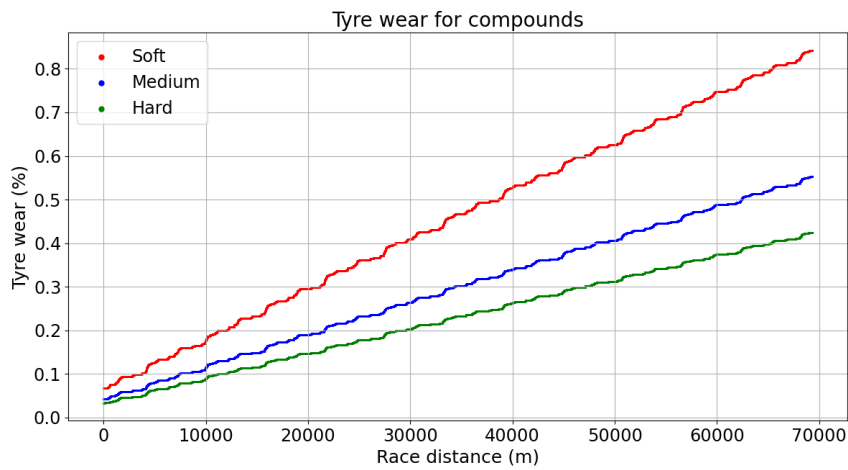


Figure 5.1: Figure shows the difference of the tyre wear between the soft, medium and hard compound on a practice session of 12 laps on the track Paul Ricard.

Tyre pace constants		
Soft	Medium	Hard
89.206	88.935	89.663

Table 5.2: Table showing the final calculated tyre pace constants from the regression model on example track Paul Ricard. The constants are based on 12 practice laps.

Tyre wear constants		
Soft	Medium	Hard
0.77469	0.50938	0.39127

Table 5.3: Table showing the different tyre wear constants (the bottle neck tyre) for each tyre compound from a practice session on the Paul Ricard track.

As described in section 3.4.2, the OLS regression model was chosen since it was performing the best compared to the other alternative models. By testing it was concluded that the model gave the best mapping when constructing a 7-th degree polynomial:

$$P(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \quad (5.1)$$

where $\{a_0, \dots, a_7\}$ are constants that are given for each individual lap respectively. An example of the polynomial with its lap and compound specific constants of a_i is showed in Figure 5.2.

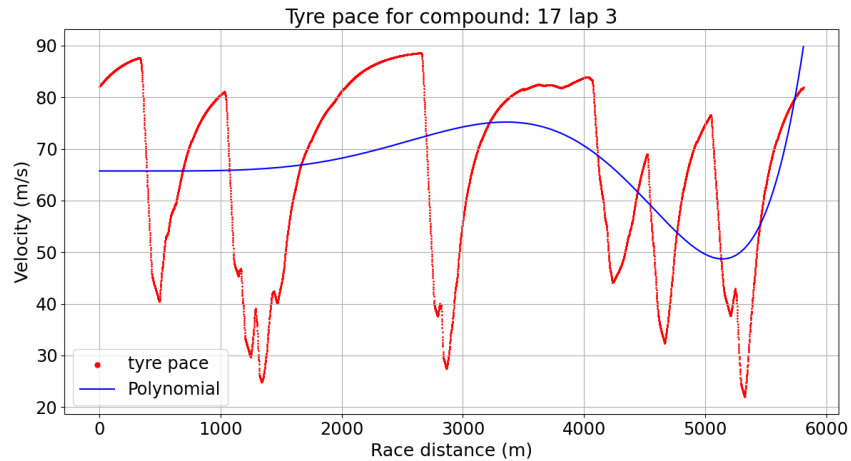


Figure 5.2: Figure showing an example practice session lap where tyre data is collected. The red dots denotes the actual speed in m/s and the blue continuous line shows the mapping of the pace from the regression model. The x-axis denotes the lap distance.

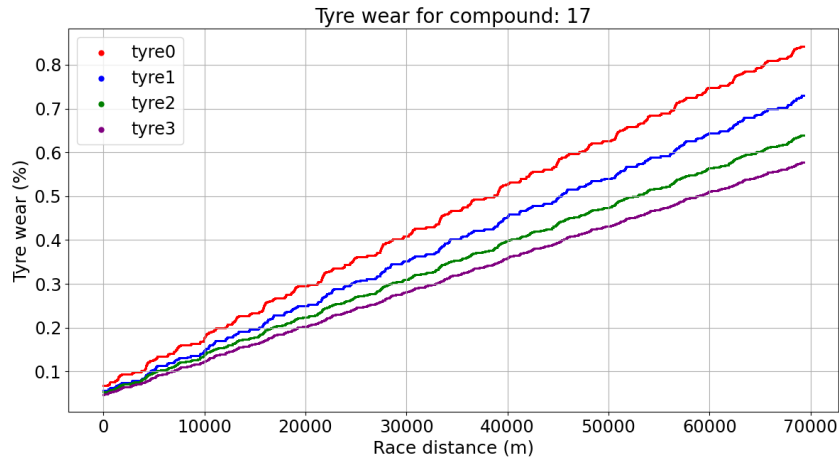


Figure 5.3: Figure shows the plot of how the tyre wear for all four tyres change during a race stint where the tyre wear increases and differs from tyre to tyre. This plot shows the soft tyres on the Paul Ricard track.

5.2 Linear programming

To be able to get as effective and relevant base strategy data as possible for the NNs to train on, the linear programming formula can be set to only calculate some tyre combinations optimal stint length. As can be seen in Figure 5.4 the relevant amount of stops on Paul Ricard was decided by testing to be 1 or 2-stop strategies. As can be seen in 5.4 it is expected in the different tyre combinations that the hard tyre could handle longer stint lengths than the medium and that the medium can go for longer than the soft.

Optimal stint lengths Paul Ricard		
Strategy	Tyre combination	Stint lengths (Nr of laps)
1-stop	SM	10 17
1-stop	SH	9 18
1-stop	MH	12 15
2-stop	SHM	6 11 10
2-stop	SSM	7 8 12
2-stop	SHH	5 11 11
2-stop	SSH	7 7 13
2-stop	SMM	7 10 10
2-stop	HHM	9 9 9
2-stop	MMH	9 9 9

Table 5.4: Table shows the resulting stint lengths that was calculated by the linear programming formula on the example track Paul Ricard. It was inputted that stint lengths for 1-stop and 2-stop strategies only would be calculated.

It should be mentioned that the 1-stop strategy of soft having a stint length of 10 laps and medium for the stint length of 17 laps was proven by testing to be faster than the simulation environments best recommendation of a base strategy. The best recommendation of the simulation was a 2-stop strategy of soft for 10 laps, new soft for another 10 laps and a finishing stint with medium for 13 laps. The best time done on this strategy during testing was a finishing time of 43:49,451. The 1-stop strategy of soft and medium gave a time of 43:48,031. For the simulation recommended tyre combination, the linear programming formula gave the stint lengths of soft for 8, new soft for additional 8 laps and finishing with medium for 11 laps. This tyre combination gave a time of 43:52,228.

5.3 Neural networks

5.3.1 Pit stop decision NN

The resulting hyperparameters that was used in the pit stop deciding NN is showed in Table 5.5. They were all carefully chosen by trial and error testing. The batch size of 200 000 data sample was chosen since it gives a enough big of chance to contain some samples of the positive class (pit). The amount of labeled data points in training data was 3688947 labels in total, where only 17925 (0,49%) were positive labels (pit) and the rest negative labels (no pit). Due to the imbalance in the data set we modify and set the class weights to 0,502 for negative labels and 102,810 for the positive labels. The class weights will compensate for the imbalance in the data set.

When achieving the best resulting model for prediction, the learning rate reduction was set to have a patience of 3 epochs, mode max (monitoring the max value of precision and recall), a reducing factor of 0,5 and a minimum learning rate of 0,000001. The early stopping parameters was set to patience of 8 epochs, mode max and to restore the best class weights. The best models score with the showed hyperparameters in Table 5.5 and above mentioned functionally that resulted in graphs 5.4 achieved and **F1-score of 0,986**. Example predictions made by the resulting NN during races can be seen in Figure 5.5 and 5.6.

As can be seen in Figure 5.4(a) the loss of the pit stop deciding NN becomes very small at the end of the training for both the training and the validation data set. Which means that the NN is predicting with predictions that in the big majority are accurate, since the difference between actual and predicted is small. This can be confirmed by looking at Figure 5.4(b) which shows that the accuracy during the training for both the data sets are close to 100%. The precision and recall curves that the F1-score is based on can be seen in Figure 5.4(c) and 5.4(d). It can be observed that these measures increase in a positive trend throughout the training without coming into a stage where it can be suspected that the model is overfitting.

NN model hyperparameters	
Hyperparameter	Value
Initial learning rate	0.01
Batch size	200 000
Initial epochs	50
Initial learning rate	0.01
Loss function	Binary cross entropy
Optimizer	RMSprop
Epsilon	1e-08
Rho	0.9
Number of neurons per layer	64
Number of hidden layers	3
Activation functions	ReLU (hidden layers), sigmoid (output layer)

Table 5.5: Table shows the resulting hyperparameters that was chosen by testing and evaluation for the pit stop NN, in order to create as good and accurate predictions as possible for the NN.

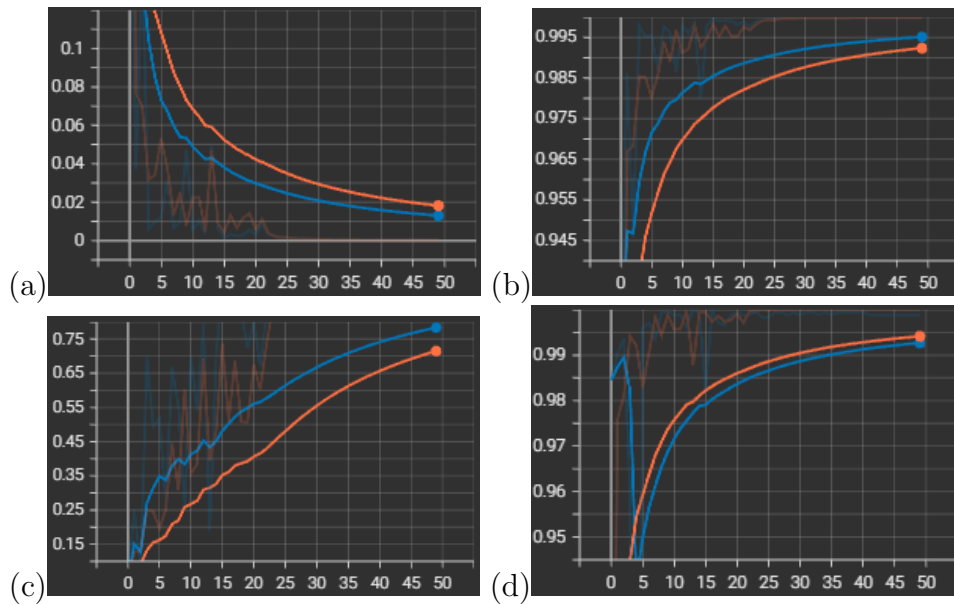


Figure 5.4: (a) Figure shows the curve of the loss (y-axis) per epoch (x-axis) when training the pit stop deciding NN. (b) Figure shows the curve of the accuracy (y-axis) per epoch (x-axis) when training the pit stop deciding NN. (c) Figure shows the curve of the precision (y-axis) per epoch (x-axis) when training the pit stop deciding NN. (d) Figure shows the curve of the recall (y-axis) per epoch (x-axis) when training the pit stop deciding NN.

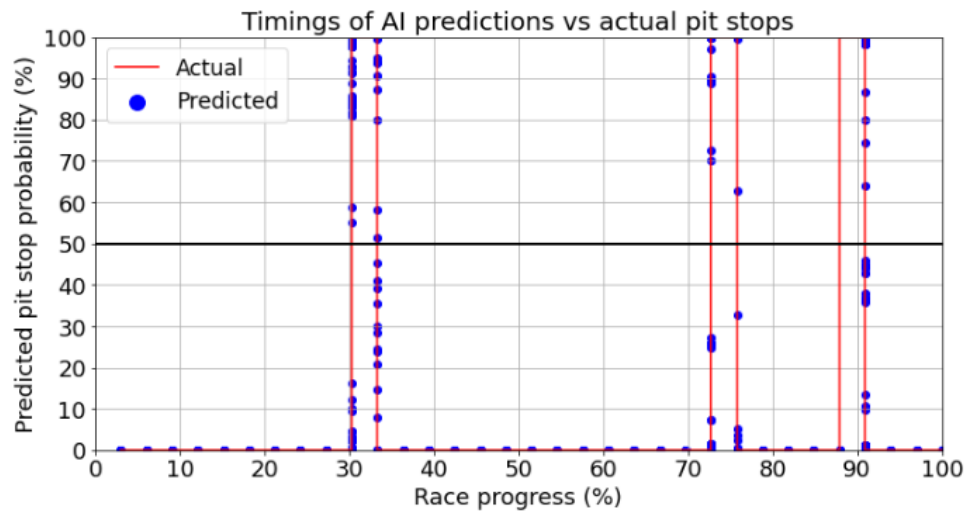


Figure 5.5: The predictions for the pit stop NN is shown in the figure. The NN model has been retrained on a new strategy at the Portimao track. Prediction values are shown in blue and red indicated the start and stop pit stop interval.

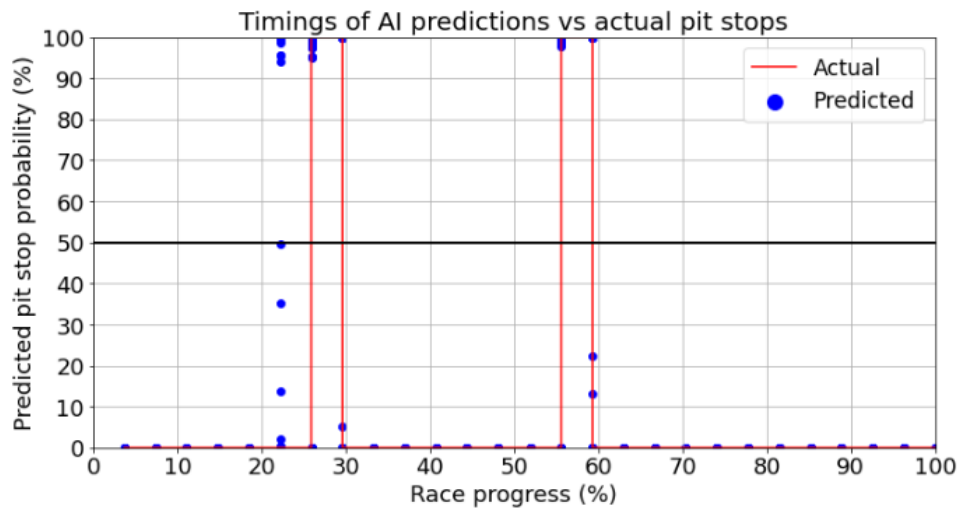


Figure 5.6: The figure is showing the pit stop deciding NNs prediction on test data on the Paul Ricard track. Prediction values are shown in blue and red indicated the start and stop pit stop interval.

By observing Figure 5.5 and 5.6 it can be seen that the NN can adapt to different strategies and be retrained in order to be more optimized. It is showed that the predictions (blue dots) made by the NN are over 50% probability in the strategy pit stop interval (red area). In Figure 5.6 the NN indicates to pit slightly before the set strategy. This could indicate that the driver's driving style or some other environment parameters have change more than expected. Like for example a higher tyre degradation due to higher track temperature.

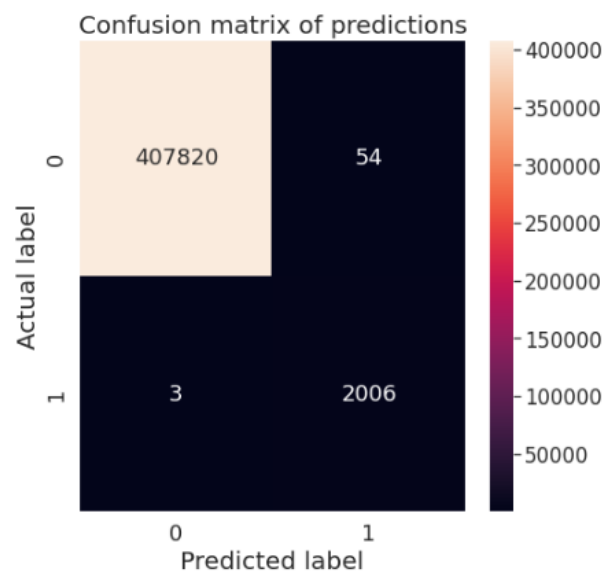


Figure 5.7: The figure is showing the fractions (number of samples) of the predictions from the pit stop deciding NN model. 1 is the positive class (to pit) and 0 is the negative class (to not pit).

As can be observed in Figure 5.7 the NN predicts wrong on the positive classes (pit) in 3 out of 2009 and 54 wrong out of 407874 negative classes (no pit). For the negative wrong guesses it could be assumed that the NN predicts the pit stop earlier than it takes place (like in Figure 5.6). Predictions earlier on the lap before the pit stop should occur does not matter practically, it can be seen as good since we don't want too late predictions during the lap either.

5.3.2 Tyre change decision NN

The resulting hyperparameters used in the tyre deciding NN model is showed in Table 5.7. The same batch size as the pit stop NN of 200 000 was tested to be the best for tyre model was well. Which also makes the training and retraining easier for practically reasons. For this model, there was no problem with imbalance in the data between the classes. Therefore no class weights was set. The fraction between the classes during the training are showed in Table 5.6. The classes could represent different tyre compounds as shown, depending on which race track it is. The output layer has 21 units instead of 1 (for the pit stop model), since the model refers to the class integers starting at 0. Since we then have classes in the range of 16-20 it becomes 21 when starting from 0. This does not effect the result.

Training data class fraction	
Tyre ID	Fraction (Nr of samples)
16 (soft)	705145
17 (soft, medium)	1336470
18 (soft, medium, hard)	887542
19 (medium, hard)	591972
20 (hard)	167818

Table 5.6: Table shows the fraction of the classes in the training data of the tyre deciding NN model. The tyre ID can represent different tyre compounds depending on the track.

NN model hyperparameters	
Hyperparameter	Value
Initial learning rate	0.01
Batch size	200 000
Initial epochs	50
Initial learning rate	0.01
Loss function	Categorical cross entropy
Optimizer	RMSprop
Epsilon	1e-08
Rho	0.9
Number of neurons per layer	64
Number of hidden layers	3
Activation functions	ReLU (hidden layers), softmax (output layer)

Table 5.7: Table shows the resulting hyperparameters that was chosen by testing and evaluation for the tyre NN, in order to create as good and accurate predictions as possible for the NN.

The early stopping functionality was set to have a patience of 4 epochs, mode max and to restore the best weights. The learning rate reduction had a patience of 2 epochs, mode max, a reduction factor of 0,5 and a minimum learning rate of 0,000001. This was conducted by trial and error in order to achieve result that gave the best metrics. The best resulting model that used the above mentioned settings and the parameters showed in 5.7 achieved a **categorical accuracy of 1,00**. Example of predictions made by the resulting NN can be seen in Figure 5.9.

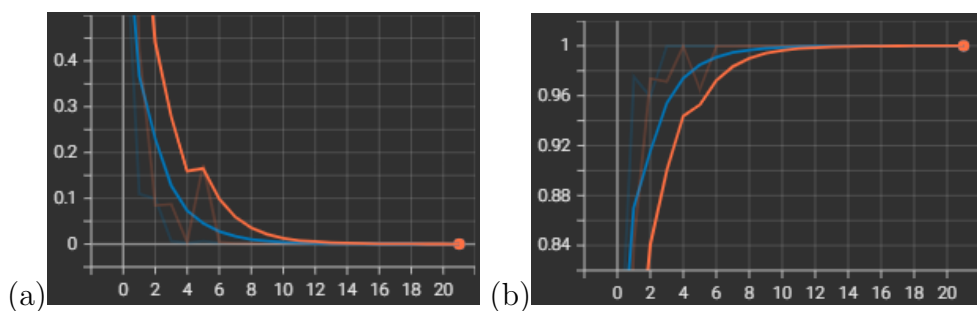


Figure 5.8: (a) Figure shows the curve of the loss (y-axis) per epoch (x-axis) when training the tyre deciding NN. (b) Figure shows the curve of the categorical accuracy (y-axis) per epoch (x-axis) when training the pit stop deciding NN.

The loss during the training of the tyre model can be seen in 5.8(a) to be decreasing steadily and reaches a value close to 0 at the end of the training. The categorical accuracy is meanwhile increasing during training and at the end reaches a value of 1,00, for both the training and the validation data set. This indicates that the NN is heavily improving its predictions and is learning to map the features to the right class.

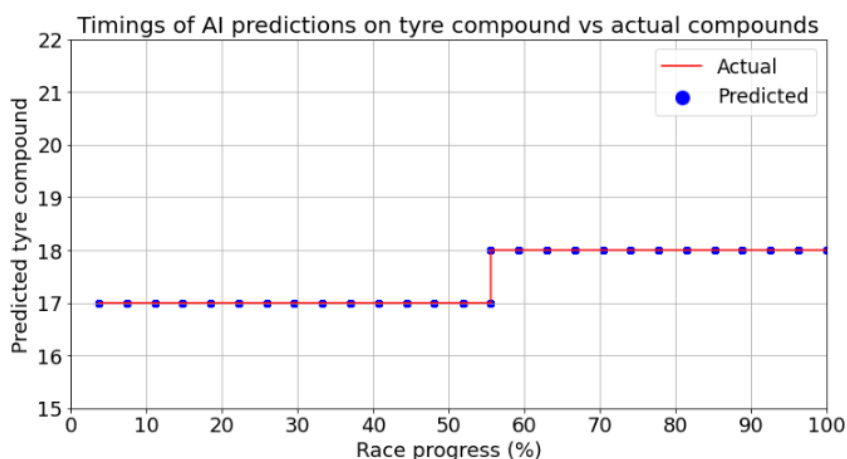


Figure 5.9: The figure is showing the predictions of the tyre deciding NN. The predicted values are in blue and the actual in red. The prediction is made on a test data set on the Paul Ricard track.

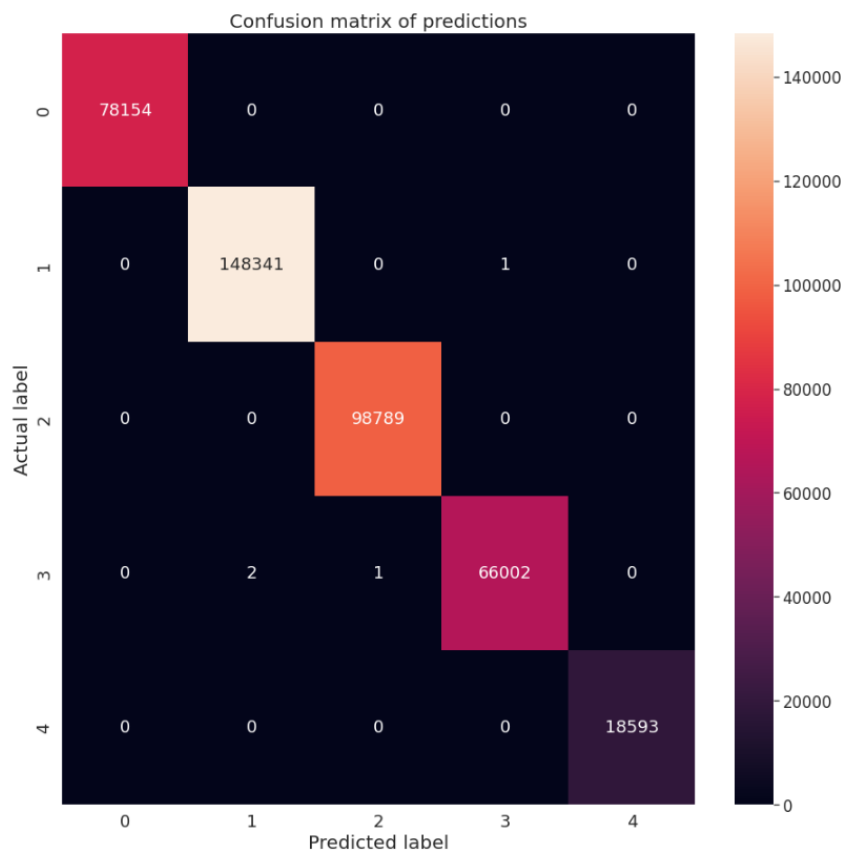


Figure 5.10: The figure shows the fraction (number of samples) of the class predictions made by the tyre deciding NN on the test data. The compounds that are represented by the integer classes are described in Table 5.6.

It is showed in Figure 5.10 that the NN only predicts wrong on 4 occasions. While the rest of the test samples are predicted right, even though there is 4 different classes and the classes represent different compounds.

CHAPTER 6

Discussion and conclusion

By collecting data in practice sessions, the tyres could be modeled after the specific track, car and weather conditions. The modeled tyre data could be used in the MIQP in order to create a initial base strategy for training the NNs. The downside of the practice data collection is the determination of the number of laps that actually best represent the tyre compounds. This has to be determined by the racing engineers in order to create arbitrary representative practice laps. The MIQP has the benefit to be computational cheap, but heavily depends on the quality of the modeled tyre data. However it is based on the assumption of a clear track and does not consider all random events that could occur during a race.

The race strategy assistant could be used for different number of drivers, in the assumption that it has been training on the specific car and driver circumstances, providing fast and tailored decision. Since the NNs will adapt to the cars race situation in order to accomplish a trained strategy. The race strategy assistant could also assume another strategy, if it has been trained on multiple strategies. That relies on the assumption that the drivers have generated data with multiple strategies for the AI to train on. When it is trained properly, it fulfills it purpose of providing automated race strategy decisions.

The race strategy assistant can adapt to the new season regulations and constant developments under the assumption that it is trained under the new circumstances (new tyre compounds, rules or new car setups). When generating data, it is of importance to generate as diverse race situations as possible, in order for the AI to learn as many random events and situations as possible. Which is time consuming, since the simulation environment does not allow a automated Monte Carlo approach. However could the AI work in such a simulated environment with additional reinforcement learning adaption. This approach would be beneficial to combine with the driver-in-loop approach. Then the automated part could contribute with data describing random events and the driver-in-loop with data tied to the specific driver's driving style and skills. When comparing the data that is possible to collect from the simulation, it is highly simplified compared

to the sensor data that is collected by the F1 teams in reality. But the AI could be reconstructed and adapted to use more and real sensor data. However the AI is not considering the tactics such as under or over cuts, which in reality would make drivers drive freely on the track and can be very beneficial.

CHAPTER 7

Future work

As part of the future work it would be interesting to add new data from the real world with more data points and more precise sensor measurements which would allow to improve the prediction quality of the AI. It would also make the AI more realistic and useful for the F1 teams. Additionally by also applying a simulation environment together with reinforcement learning in order to create more data automatically and data adapted to a wider spread of random events, the AI would generate more situation specific decisions.

REFERENCES

- [1] Mary Allender et al. Predicting the outcome of nascar races: The role of driver experience. *Journal of Business & Economics Research (JBER)*, 6(3), 2008.
- [2] Mercedes AMG. How does f1 simulation work?, 2020. Accessed: October 2022.
- [3] Fortune buisness insights. Sports analytics market size, share covid-19 impact analysis, by deployment (cloud and on-premises), by type (on field and off field), by solution (video analytics, bio analytics, smart wearable tecknology, and others), by technology (ai, big data, and others), by end-user (individual and teams), and regional förecast, 2022-2029, 2022. [Online; accessed 7-March-2023].
- [4] Christopher Ledesma Weisen Choo. *Real-time decision making in motorsports: analytics for improving professional car race strategy*. PhD thesis, Massachusetts Institute of Technology, 2015.
- [5] Ergast developer API. Ergast developer api, 2022. Accessed: October 2022.
- [6] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.
- [7] Maral Haghighat, Hamid Rastegari, Nasim Nourafza, Najafabad Branch, and Iran Esfahan. A review of data mining techniques for result prediction in sports. *Advances in Computer Science: an International Journal*, 2(5):7–12, 2013.
- [8] Alexander Heilmeier. F1 timing database, 2021. Accessed: October 2022.
- [9] Alexander Heilmeier, André Thomaser, Michael Graf, and Johannes Betz. Virtual strategy engineer: Using artificial neural networks for making race strategy decisions in circuit motorsport. *Applied Sciences*, 10(21), 2020.
- [10] Xuze Liu and Abbas Fotouhi. Formula-e race strategy development using artificial neural networks and monte carlo tree search. *Neural Computing and Applications*, 32(18):15191–15207, 2020.
- [11] Barry Pfitzner and Tracy D Rishel. Do reliable predictors exist for the outcomes of nascar races. *The Sport Journal*, 8(2), 2005.

- [12] Michael C Purucker. Neural network quarterbacking. *IEEE Potentials*, 15(3):9–15, 1996.
- [13] Eloy Stoppels. Predicting race results using artificial neural networks. Master’s thesis, University of Twente, 2017.
- [14] Theja Tulabandhula and Cynthia Rudin. Tire changes, fresh air, and yellow flags: challenges in predictive analytics for professional racing. *Big data*, 2(2):97–112, 2014.
- [15] UNAmelia. Srt sim racing telemetry, 2022. Accessed: October 2022.
- [16] Wikipedia. Formula One tyres — Wikipedia, the free encyclopedia, 2023. [Online; accessed 10-January-2023].

Appendices

APPENDIX A

Simulation data

Client data fields		
Field	Unit	Description
trackId	-	Unique track ID
lap_time	s	Lap time
pit_status	-	Pit status
race_position	-	Race position
fuel	kg	Fuel mass
tyre_compound_0	-	Tyre 0 compound
tyre_compound_1	-	Tyre 1 compound
tyre_compound_2	-	Tyre 2 compound
tyre_compound_3	-	Tyre 3 compound
wheel_slip_ratio_0	%	Slip ratio tyre 0
wheel_slip_ratio_1	%	Slip ratio tyre 1
wheel_slip_ratio_2	%	Slip ratio tyre 2
wheel_slip_ratio_3	%	Slip ratio tyre 3
track_temp	°C	Track temperature
SafetyCarStatus	-	Status of safety car
startGridPosition	-	Start grid position
tyre_temp_0	°C	Tyre 0 surface temperature
tyre_temp_1	°C	Tyre 1 surface temperature
tyre_temp_2	°C	Tyre 2 surface temperature
tyre_temp_3	°C	Tyre 3 surface temperature
tyre_tempInternal_0	°C	Tyre 0 internal temperature
tyre_tempInternal_1	°C	Tyre 1 internal temperature
tyre_tempInternal_2	°C	Tyre 2 internal temperature
tyre_tempInternal_3	°C	Tyre 3 internal temperature
tyres_age	laps	Age of tyres
tyre_wear_0	%	Tyre 0 wear

tyre_wear_1	%	Tyre 1 wear
tyre_wear_2	%	Tyre 2 wear
tyre_wear_3	%	Tyre 3 wear
tyre_damage_0	%	Tyre 0 damage
tyre_damage_1	%	Tyre 1 damage
tyre_damage_2	%	Tyre 2 damage
tyre_damage_3	%	Tyre 3 damage
front_left_wing_damage	%	Front left wing damage
front_right_wing_damage	%	Front right wing damage
carId	-	Car id
trackLength	m	Length of track
lapIndex	-	Lap index
lapFlag	-	Lap flag flag
binIndex	-	Bin index
validBin	-	Valid bin flag
lapNum	-	Lap number
lap_distance	m	Distance driven on current lap
lap_time_invalid	-	Invalid lap time flag
world_position_x	-	World position x-axis
world_position_y	-	World position y-axis
world_position_z	-	World position z-axis
world_forward_x	-	World forward x direction
world_forward_y	-	World forward y direction
world_forward_z	-	World forward z direction
velocity_x	m/s	Velocity x direction
velocity_y	m/s	Velocity y direction
velocity_z	m/s	Velocity z direction
gforce_x	m/s^2	G-force x-axis
gforce_y	m/s^2	G-force y-axis
gforce_z	m/s^2	G-force z-axis
flags_status	-	Status of flags
throttle	%	Amount of applied throttle
brake	%	Amount of applied brake
clutch	%	Amount of applied clutch
steering	-	Steering position
gear	-	Current gear
rpm	r/min	Engine RPM
rpm_perc	r/min	Idle RPM
yaw	rad	Yaw angle
pitch	rad	Pitch angle
roll	rad	Roll angle
susp_pos_0	-	Suspension 0 position
susp_pos_1	-	Suspension 1 position

susp_pos_2	-	Suspension 2 position
susp_pos_3	-	Suspension 3 position
susp_vel_0	-	Suspension 0 velocity
susp_vel_1	-	Suspension 1 velocity
susp_pos_2	-	Suspension 2 velocity
susp_pos_3	-	Suspension 3 velocity
susp_acc_0	-	Suspension 0 acceleration
susp_acc_1	-	Suspension 1 acceleration
susp_acc_2	-	Suspension 2 acceleration
susp_acc_3	-	Suspension 3 acceleration
wheel_slip_angle_0	deg	Angle of wheel 0 slip
wheel_slip_angle_1	deg	Angle of wheel 1 slip
wheel_slip_angle_2	deg	Angle of wheel 2 slip
wheel_slip_angle_3	deg	Angle of wheel 3 slip
angular_vel_x	m/s	Angular velocity x-axis
angular_vel_y	m/s	Angular velocity y-axis
angular_vel_z	m/s	Angular velocity z-axis
angular_acc_x	m/s	Angular velocity x-axis
angular_acc_y	m/s	Angular velocity y-axis
angular_acc_z	m/s	Angular velocity z-axis
weather	-	Weather condition
air_temp	°C	Air temperature
pitSpeedLimit	m/s	Speed limit pit lane
networkGame	-	Network status
pitIdealLap	-	Ideal pit lap
pitLatestLap	-	Latest pit lap
pitRejoinPos	-	Rejoin position after pit stop
steeringAssist	-	Steering assist mode
brakeAssist	-	Brake assist mode
gearAssist	-	Gear assist mode
pitAssist	-	Pit assist mode
pitReleaseAssist	-	Pit release assist mode
ersAssist	-	ERS assist mode
drsAssist	-	DRS assist mode
racingLineAssist	-	Racing line assist mode
racingLineTypeAssist	-	Racing line type assist flag
safetyCarDelta	s	Delta time safety car
penalties	s	Penalty time to be added
warnings	-	Amount of warnings
pitLaneTime	s	Time in pit lane
pitStopTime	s	Time in pit
wing_setup_0	-	Front wing aero setup
wing_setup_1	-	Rear wing aero setup

diff_onThrottle_setup	%	Differential adjustment on throttle
diff_offThrottle_setup	%	Differential adjustment off throttle
chamber_setup_0	deg	Camber 0 angle (suspension geometry)
chamber_setup_1	deg	Camber 1 angle (suspension geometry)
chamber_setup_2	deg	Camber 2 angle (suspension geometry)
chamber_setup_3	deg	Camber 3 angle (suspension geometry)
toe_setup_0	deg	Toe 0 angle (suspension geometry)
toe_setup_1	deg	Toe 1 angle (suspension geometry)
toe_setup_2	deg	Toe 2 angle (suspension geometry)
toe_setup_3	deg	Toe 3 angle (suspension geometry)
susp_spring_setup_0	-	Suspension 0 setup
susp_spring_setup_1	-	Suspension 1 setup
susp_spring_setup_2	-	Suspension 2 setup
susp_spring_setup_3	-	Suspension 3 setup
arb_setup_0	-	Front anti-roll bar setup
arb_setup_1	-	Rear anti-roll bar setup
susp_height_setup_0	-	Suspension 0 height setup
susp_height_setup_1	-	Suspension 1 height setup
susp_height_setup_2	-	Suspension 2 height setup
susp_height_setup_3	-	Suspension 3 height setup
brake_press_setup	%	Brake pressure
brake_bias_setup	%	Brake bias
tyre_press_setup_0	PSI	Tyre 0 pressure setup
tyre_press_setup_1	PSI	Tyre 1 pressure setup
tyre_press_setup_2	PSI	Tyre 2 pressure setup
tyre_press_setup_3	PSI	Tyre 3 pressure setup
ballast_setup	-	Ballast setup
fuel_setup	-	Fuel setup
drs	-	DRS on/off
brake_temp_0	°C	Brake 0 temperature
brake_temp_1	°C	Brake 1 temperature
brake_temp_2	°C	Brake 2 temperature
brake_temp_3	°C	Brake 3 temperature
tyre_press_0	PSI	Tyre 0 pressure
tyre_press_1	PSI	Tyre 1 pressure
tyre_press_2	PSI	Tyre 2 pressure
tyre_press_3	PSI	Tyre 3 pressure
terrain_0	-	Driving surface 0
terrain_1	-	Driving surface 1
terrain_2	-	Driving surface 2
terrain_3	-	Driving surface 3
engine_temp	°C	Engine temperature
suggested_gear	-	Suggested gear

traction_ctrl_setup	-	Traction control setup
abs_setup	-	ABS setup
fuel_type	-	Fuel mix
fuel_remainingLaps	-	Remaining fuel laps
front_brake_bias	%	Front brake bias
pit_limiter	-	Pit limiter status
drs_allowed	-	DRS allowance status
drs_distance	-	DRS activation distance
ers_store	joule	ERS energy store
ers_deployMode	-	ERS deployment mode
ers_harv_mguk	joule	ERS energy harvested MGU-K
ers_harv_mguh	joule	ERS energy harvested MGU-H
ers_deployed	-	ERS energy deployed this lap
brake_damage_0	%	Brake 0 damage
brake_damage_1	%	Brake 1 damage
brake_damage_2	%	Brake 2 damage
brake_damage_3	%	Brake 3 damage
rear_wing_damage	%	Rear wing damage
floor_damage	%	Floor damage
diffuser_damage	%	Diffuser damage
sidepod_damage	%	Sidepod damage
drs_damage	%	DRS damage
gear_box_damage	%	Gear box damage
engine_damage	%	Engine damage
mgih_damage	%	MGU-H damage
es_damage	%	Engine wear ES
ce_damage	%	Engine wear CE
ice_damage	%	Engine wear ICE
mguk_damage	%	MGU-K damage
tc_damage	%	Engine wear TC
wheel_speed_0	m/s	Wheel 0 speed
wheel_speed_1	m/s	Wheel 1 speed
wheel_speed_2	m/s	Wheel 2 speed
wheel_speed_3	m/s	Wheel 3 speed

Table 1.1: Data fields that is possible to recieve from simulation with the client UDP data stream.

Acronyms

F1 Formula 1. 1–5, 8, 9, 21, 38, 39

MIQP Mixed-Integer Quadratic Programming. 16, 37

ML machine-learning. 4, 5

NN neural network. 5, 6, 8–13, 15, 17–19, 28–35, 37

OBD on-board diagnostics. 21, 22

OLS ordinary least squares. 14, 27

RMSprop root mean square propagation. 19