Matthew Titmas

Modelling Formula One Qualifying using Machine Learning

Computer Science Tripos – Part II

Corpus Christi College

May 11, 2023

Declaration of originality

I, Matthew Titmas of Corpus Christi College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose. In preparation of this dissertation I did not use text from AI-assisted platforms generating natural language answers to user queries, including but not limited to ChatGPT. I am content for my dissertation to be made available to the students and staff of the University.

Signed Matthew Charles Titmas

Date May 11, 2023

Proforma

Candidate Number:	**2359A**
Project Title:	**Modelling Formula One Qualifying using Machine Learning**
Examination:	Computer Science Tripos – Part II, May 11, 2023
Word Count:	**11928**[1]
Code Line Count:	**2092**[2]
Project Originator:	Joseph McMillan
Supervisor:	Dr Neil Lawrence, Andrei Paleyes, Hanbo Li

Original Aims of the Project

Qualifying is an important aspect of the Formula One weekend, these sessions determine the order of the starting grid with the best positions being granted to those that make it into the final session. This project aims to solve the problem of predicting a probability of a given driver progressing into the next qualifying session without needing to set another lap. This dissertation involves a collaboration with Mercedes AMG Petronas Formula 1.

Work Completed

I have completed all work I set out to finish at the start of this project and was able to implement an extension as well. Furthermore, I have also packaged the project so that the client can easily use it themselves.

Special Difficulties

None

[1] Counted with overleaf (Using TexCount internally).

[2] This line count was computed using the statistics plugin for PyCharm (Excl. comments and whitespace).

Contents

List of Figures

List of Tables

Acronyms

TOD Time of day 16

F1 Formula One 1

P1 Practice Session One 15

P2 Practice Session Two 15

P3 Practice Session Three 15

Q1 Qualifying Session One 3

Q2 Qualifying Session Two 3

Q3 Qualifying Session Three 3

AI Artificial Intelligence 5

GP Gaussian Process 5

GPR Gaussian Process Regressor 5

GPC Gaussian Process Classifier 19

RBF Radial Basis Function 7

MSE Mean Square Error vii

Chapter 1: Introduction

1.1 Motivation

This project is inspired by the growing need for fast, accurate decision-making within every aspect of the Formula One (F1) weekend. It is assumed that every team will have software that allows them to predict many aspects of how the race weekend will unfold, including qualifying cut-off times, and as such it is important that every team is consistently upgrading these predictors in order to increase the likelihood that they can generate a correct prediction. While a new qualifying cut-off time predictor was made recently for Mercedes AMG Petronas F1 team (henceforth known as "the client") [1] this did not give the probability of a given driver passing into the next session, and only the cut-off time for that session. This project builds on [1] by focusing on the probabilistic aspect of models to allow for informed decisions while also including more data within the model.

Machine Learning is driven by data [2]. A typical F1 weekend can produce 60GB of data [3] with the majority of all data produced by the client being generated during the race and practice sessions, resulting in the qualifying sessions producing much less usable data comparatively. Even with this limitation, it is very important that the client does not fall behind the other teams in their ability to predict the wanted values and as such they are motivated to update the programs they use continually.

The model produced for this project is 'QualifAI'. Three separate models will be mentioned within this dissertation. The models and their properties are outlined in Table 1.1.

Model	Predicts cut-off times	Gives confidence intervals for cut-off times	Predicts if a driver should remain in the garage	Gives confidence intervals for if a driver should remain in the garage
QualifAI	✓	✓	✓	✓
Client Model	✓	✗	✓	✗
Deliver [1]	✓	✓	✓	✗

Table 1.1: Models mentioned within the dissertation and their salient properties.

1.2 Personal Motivation

Initially, I was drawn to this project due to my love for the sport and the unique setting in which the project was placed. However, the main motivation became ensuring that the client would be satisfied with the results I could produce. The most important aspect of this project is that it is educational; I wanted to further my understanding of various machine learning

models, especially in scenarios where data is not widely available and competition exists. This project allows me the opportunity to learn this through both literature and practice.

It also allows me to directly work with a company, which lets me to learn how to interact with clients and ensure their needs are met.

1.3 Project Goal

This project aims to build on the work completed by Draghici [1] in order to provide the probability of a given driver passing into the next qualifying session. To achieve this goal I first decide which data should be extracted and extract this via the supplied API. A pipeline is created that allows me to update the necessary data to be extracted quickly. I then establish the theory behind the model chosen and implement the model with optimised hyperparameters based on the given task. Finally, I evaluate the model accuracy on a held-out data set and compare it to the models already used and Draghici's model.

1.4 Related Work

This project builds on [1]. Furthermore, during my literature review, I found less than 10 publications used Machine Learning methods in the context of F1 races. These include literature on the car itself [4], the safety of F1 [5], or race analysis [6]. However, there is no public work on the use of Machine Learning in order to analyse and predict qualifying sessions due to this lack of academic research into the field of F1.

Chapter 2: Preparation

2.1 Formula One

F1 is a data-intensive sport, with teams collecting thousands of data points each second that needs to be analysed [3]. Because the amount of data collected is so large it is difficult to expect people to analyse this and produce realistic predictions with confidence intervals. This project aims to automate the process of predicting through the use of Machine Learning techniques.

2.1.1 Race Weekend

On a typical F1 weekend, also known as a Grand Prix, there are 3 practice sessions, a qualifying session, and the race [7, section 38.2, 39.2, 44]. While the practice sessions have no impact on the race, they are used by the client in order to produce a *BaseLapTime* which is a prediction of the time a given driver should be setting.

Qualifying Sessions

The qualifying session is split into 3 sub-sessions (Qualifying Session One (Q1), Qualifying Session Two (Q2), and Qualifying Session Three (Q3)) and is used to determine the starting lineup of the race. With this lineup significantly impacting the final race results [8] it is important that each team progresses through the sessions to maximise their probability of winning the race. At the end of a given sub-session, the drivers are ordered by their fastest lap time (time taken to complete a single lap of the circuit), thus the drivers will spend the sub-session attempting to finish a lap in as low a time as possible. Once the order has been confirmed all of the drivers will compete in the next sub-session except for the slowest 5. Of these slowest 5 drivers, their position in the order becomes their starting position in the race. Each qualifying session has a set time (18 minutes for Q1, 15 minutes for Q2, 12 minutes for Q3) in order to limit the number of laps that each driver can complete, putting significant importance on the laps they do complete in the given time.

The third qualifying session is used to determine the starting position of the remaining 10 drivers; any driver that competes in Q3 is guaranteed to start in at least 10^{th} position irrelevant of the lap time they set in the session (Assuming no penalties are applied). A higher starting position is beneficial as it reduces the number of cars that have to be overtaken during the race as well as reducing the distance required to reach the first corner compared to drivers behind. The drivers in Q3 are ordered based on the lap times they set in Q3 and are given grid positions equal to their position in the order.

Qualifying Laps

Not every lap will result in the driver crossing the starting line twice. There are 3 types of laps, an *out-lap*, an *in-lap*, and a *timed lap*. An out-lap is one where a driver is leaving the pit lane and entering the race track, an in-lap is one where a driver is entering the pit line and leaving the race track, and a timed lap is one where the driver remains on track. Only timed laps are used when considering a given driver's set of lap times.

Another distinction that should be made is the difference between a *flying lap* and a *slow lap*. Both of these are timed laps but a flying lap is an attempt at setting the fastest possible lap time a driver can while a slow lap is a lap in which the driver will ensure the car has the best possible conditions for a flying lap, or ensure the car is working as expected.

Tyre Compounds

Each car has a selection of 3 tyre compounds to use when setting a lap time. These are known as hard tyres, medium tyres, and soft tyres. Each tyre has advantages and disadvantages, with hard tyres giving the least amount of grip but remaining at higher grip levels for longer while soft tyres give the highest amount of grip but quickly losing these high levels of grip. Medium tyres sit between these two extremes.

Tyres also leave rubber on the track as laps are completed. The result of this is that the track itself gains more grip, resulting in more grip for the drivers as more rubber is left on the track.

Drivers also only have a limited number of each compound of tyre that they are allowed to use during a single race weekend. As tyres degrade with more use it is optimal for teams to limit the number of tyres they use before they race to ensure the maximal number of tyres with high amounts of grip for use within the race.

2.2 Problem Statement

The problem I am aiming to solve is that of a 2-player game with resource maximisation. Given all of the data collected by the client through previous races and all sessions that have occurred so far for the current race weekend, is it possible to produce a model that can provide informed decisions on how the client should act during the qualifying sessions? This project aims to answer this question through an example model that uses a subset of the data in order to aid the client in predicting the cut-off time for a qualifying session and classifying whether it is beneficial for a given driver to not attempt to set a faster lap such that other drivers do not cause the given driver to be within the slowest 5 drivers.

These are both necessary for the client to know as the former allows for knowledge of how other drivers are likely to act. The latter allows the client to keep the car in the garage. This saves a set of tyres for later use and increases the number of possible choices they have for race strategy. The supplied data is extensive, with information on each car, driver, race track corner, weather, etc.. The amount of data is too large to input into a model and must be filtered and augmented before it is used.

2.3 Machine Learning Theory

This section will define some fundamental terms and ideas in machine learning necessary for this project.

2.3.1 Machine Learning

Machine learning is a branch of Artificial Intelligence (AI) and Computer Science which focuses on imitating the way that humans learn, gradually improving its accuracy [9]. This branch of Computer Science is appropriate for the goals of this dissertation as it allows for accurate decisions to be made with minimal human intervention given there is enough data.

Given the structure of the dataset, the chosen approach is Supervised Machine Learning. Supervised Machine Learning requires the training data to contain both inputs and outputs in order to increase the accuracy of the model compared to the use of other methods such as unsupervised learning. The output of the model is the time a driver would take to perform another lap should they go out again. This is stored in the dataset, making supervised learning the obvious choice.

2.3.2 Gaussian Progress Regression

A Gaussian Process (GP) is defined as a collection of random variables, any finite number of which have a joint (multivariate) Gaussian distribution [10]. A GP is a valid choice for this type of project due to its support of uncertainty quantification. During a race, it is necessary to ensure confidence in a given prediction in order to ensure that the choice given is correct.

A Gaussian Process Regressor (GPR) is a model that attempts to complete one of three goals, each working on a function $f : X \to Y$

1. Modelling f, such that the GP is a mathematical representation of the relation between the inputs and outputs of f.

2. Exploring f, such that we can use the GP in order to explore the underlying function to learn about it quickly and accurately.

3. Exploration-Exploitation problems. The objective is to find inputs that produce the highest outputs in order to maximise the total reward accrued within a particular period of time.

For this project, we will be using a GPR that accomplishes the first goal.

The Weight Space View

Consider a standard approach to model functions, linear regression, which will be based on a Bayesian viewpoint. We assume that the outputs are a linear function of the inputs with some additional noise, ($\epsilon_\theta = \mathcal{N}(0, \sigma_\theta^2)$).

$$y_t = f(x_t) + \epsilon_\theta$$

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_\theta$$

Writing this as vectors allows us to adopt a Bayesian framework; we do so through the posterior distribution over the weights, $\boldsymbol{w}$

$$y_t = \boldsymbol{x_t}^T \boldsymbol{w} + \epsilon_\theta$$

where

$$\boldsymbol{x_t} = \begin{bmatrix} 1 \\ x_t \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

Using a Gaussian prior over the weights $\mathcal{P}(\boldsymbol{w}) = \mathcal{N}(0, \Sigma)$ and the Gaussian likelihood $\mathcal{P}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{X}_t^T\boldsymbol{w}, \sigma_\theta^2\boldsymbol{I})$ then the posterior distribution is: (X_t is the vector of all inputs.)

$$\mathcal{P}(\boldsymbol{w}|\boldsymbol{y}_t, \boldsymbol{X_t}) \propto \mathcal{P}(\boldsymbol{y}_t|\boldsymbol{X}_t, \boldsymbol{w})\mathcal{P}(\boldsymbol{w})$$

$$= \mathcal{N}(\frac{1}{\sigma_\theta^2}\boldsymbol{A}_t^{-1}\boldsymbol{X}_t\boldsymbol{y}_t, \boldsymbol{A}_t^{-1})$$

where $A_t = \Sigma^{-1} + \sigma_\theta^{-2}\boldsymbol{X}_t\boldsymbol{X}_t^T$.

An inference is then performed over the weights in order to find the best values for b_0 and b_1 given the data. An extension to linear regression is to map the inputs onto a feature space that allows for non-linear dependencies.

Linear models are unsuitable for predicting the shape of any function as they assume a linear shape, the non-linear extension allows for the accurate prediction of a function with any shape. However, this flexibility results in an infinite number of possible mappings in which we must choose one via a priori. An example function that a linear model could not predict the shape of is the lap time decrease due to rubbering in, which is a negative exponential.

GP Regression instead allows the data and the chosen kernel to decide the complexity of the function.

The Function Space View

In the weight space view, we focus on a distribution over the weights, which implies a distribution over functions. For GP regression, we instead focus directly on the distribution over functions.

Assume that the output of a function f, that we want to model, at input x can be written as (where $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$):

$$y = f(x) + \sigma_\epsilon \tag{2.1}$$

This is the same assumption as with linear regression. However, we assume that the signal term ($f(x)$) is also a random variable which follows a given distribution.

$$f(x) \propto \mathcal{GP}(m(x), k(x, x')) \tag{2.2}$$

where $m(x)$ is the mean function, and $k(x, x')$ is the covariance function. The mean function models the expected value of the function at input x, $m(x) = \mathbb{E}(f(x))$. The covariance function models the dependence between function values at two input points. This is known as the *kernel* of the Gaussian Process

A function is a valid kernel of a Gaussian Process if and only if it satisfies the following properties:

- A covariance function k must be symmetric. That is, $\forall x, x' \quad k(x, x') = k(x', x)$.

- The Gram matrix relating to a covariance function k must be positive semi-definite (all eigenvalues must be non-negative).

Two valid kernel functions can also be multiplied or added together in order to generate a new kernel function. The kernel function used within this project is the Radial Basis Function (RBF) kernel.[1]

$$k(x_i, x_j) = \exp -\frac{d(x_i, x_j)^2}{2l^2} \tag{2.3}$$

where $d(\cdot, \cdot)$ is the Euclidean distance between the two inputs. The value l is a hyperparameter of the kernel and determines the impact of two points being closer together.

An RBF kernel is able to accurately model complex functions such as the rubbering in effect and the effect of tyre degradation. While this kernel does have drawbacks, explained in §3.3.1.1, the ability to better model the complex functions involved in the lap time of an F1 car should help to produce a predictor that is more accurate than previous iterations, such as Deliver (which used linear kernels). An analysis of kernel choice and their respective MSE is presented in Appendix A.

Sampling from a Gaussian Process

Even though GPs are continuous, sampling is done by computing the value at a discrete set of input points. Suppose we have training data $\mathcal{D}_t = \{x_t, y_t\}$ and want to make new predictions for inputs $\boldsymbol{X}_*$ by drawing $\boldsymbol{f}_*$ from the posterior distribution $\mathcal{P}(f|\mathcal{D}_t)$. By definition of the Gaussian Process, we know that observations $\boldsymbol{y}_t$ and function values $\boldsymbol{f}_*$ follow a multivariate normal distribution.

$$\begin{bmatrix} \boldsymbol{y}_t \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{X}_t, \boldsymbol{X}_t) + \sigma_\epsilon^2 & K(\boldsymbol{X}_t, \boldsymbol{X}_*) \\ K(\boldsymbol{X}_*, \boldsymbol{X}_t) & K(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix}\right)$$

where $K(\boldsymbol{X}_t, \boldsymbol{X}_t)$ is the covariance matrix between training points, $K(\boldsymbol{X}_*, \boldsymbol{X}_*)$ is the covariance matrix between testing points, and $K(\boldsymbol{X}_t, \boldsymbol{X}_*) = K(\boldsymbol{X}_*, \boldsymbol{X}_t)$ is the covariance matrix between testing and training points. This means the conditional distribution is

$$\mathcal{P}(\boldsymbol{f}_*|\boldsymbol{X}_t, \boldsymbol{y}_t, \boldsymbol{X}_*) \sim \mathcal{GP}(m_t(\boldsymbol{x}), k_t(\boldsymbol{x}, \boldsymbol{x'}))$$

where

$$m_t(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{X}_t)\left[K(\boldsymbol{X}_t, \boldsymbol{X}_t) + \sigma_\epsilon^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y}_t$$

$$k_t(\boldsymbol{x}, \boldsymbol{x'}) = K(\boldsymbol{x}, \boldsymbol{x'}) - K(\boldsymbol{x}, \boldsymbol{X}_t)\left[K(\boldsymbol{X}_t, \boldsymbol{X}_t) + \sigma_\epsilon^2 \boldsymbol{I}\right]^{-1} K(\boldsymbol{X}_t, \boldsymbol{X}_*)$$

This means, to predict $\boldsymbol{f}_*$ we can use the mean function above, which can be calculated by calculating 4 covariance matrices.

Model output

The output for the model is chosen to be

$$\mathit{ActualLapTime} - \mathit{ExpectedLapTime} \tag{2.4}$$

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RBF.html`

where *ExpectedLapTime* is a number, predicted by the client, representing the expected time a driver should set on any given lap. By using this equation as the output to the model rather than the total time for a driver to complete a lap we are able to expect outputs from a smaller range of values, thus increasing model accuracy. However, one downside of this approach is that the error in *ExpectedLapTime* is included and compounded into the error of my model as well.

As predicted lap time can then be calculated with:

$$ExpectedLapTime + M(\vec{X}) \tag{2.5}$$

where $M \propto \mathcal{GP}(k)$ (k is the selected kernel) and $\vec{X}$ is the input feature to the model, we will henceforth assume the output of the model is instead the predicted lap time. Predicted lap time can always be easily calculated from the actual output (as *ExpectedLapTime* is always computed before the qualifying session).

Optimising Hyperparameters

Kernels usually contain unknown values called hyperparameters which need to be inferred from the data. As the posterior distribution over the hyperparameters is difficult to obtain, common practice is to obtain point estimates of hyperparameters by maximising the marginal log-likelihood [11].

$$\log \mathcal{P}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\Theta}) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}_y^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_y| - \frac{n}{2}\log 2\pi \tag{2.6}$$

To ensure hyperparameters are optimised, I will use random search as it is more effective than grid search [12] for finding optimal hyperparameters.

While SciPy has pre-built functions for performing a random search[1] the algorithm requires the model to implement "Estimator", which a Gaussian Process does not. Therefore a custom random search implementation was needed. This is shown in algorithm 1.

2.3.3 Monte Carlo Simulation

Monte Carlo simulation uses random sampling and statistical modelling to estimate mathematical functions and mimic the operations of complex systems [13]. Although we have a model that can give output the time a given driver will set (with an associated probability) we cannot yet give the probability that another driver will be knocked out. A Monte Carlo simulation allows us to generate possible outputs to a function and average these out to approximate the actual value of the function. Within this project, the function approximated is the probability that a given driver will be knocked out, should all drivers currently slower than them set another lap. The reason for needing a simulation is that this value is dependent upon the order in which the drivers go out again (the later in the session a driver goes out, the better lap time they are likely to set). Increasing the number of samples and changing the initial conditions each sample of the Monte-Carlo simulation will ensure that this simulation is an ergodic process [14].

The efficiency of a Monte Carlo simulation is dependent upon the convergence speed, and sampling efficiency [15]. To ensure that the algorithm is as efficient as possible it was important that as much is done via libraries that use native C to improve efficiency.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html`

Algorithm 1 Random Search - Hyperparameter optimisation.

Require: $N > 0$
  $R \leftarrow$ Initial ranges of hyperparameter values
  $M \leftarrow$ Gaussian Process Regression Model
  $PL \leftarrow \infty$
  for $n = 0, 1, \ldots, N-1$ **do**
    $\vec{v} \leftarrow$ Random sample of hyperparameter values from $r \in R$
    $\vec{M} \leftarrow$ Gaussian Process Regression Model with chosen hyperparameters
    $\vec{O} \leftarrow \vec{M}$(test set)
    $\vec{L} \leftarrow \text{Loss}(\vec{O})$
    if $L < PL$ **then**
      $PL \leftarrow L$
      $R \leftarrow$ Reduce the range, based on$\vec{L}$
    else
      Increase the range of R
    end if
  end for
  Hyperparameters $\leftarrow$ Randomly sampled values from R

2.4 Software Engineering Tools

2.4.1 Programming Language and Tools

I have chosen to use Python 3.8 as my programming language as the client uses this language for internal programs and the client-supplied API that extracts the necessary data from files can only be accessed in this language. For my IDE I have chosen PyCharm due to extensive debugging and VCS support. My VCS is git and I use Jupyter Notebooks for data analysis and experimentation.

2.4.2 Libraries

I chose SciPy[1] as my core ML framework due to having previous experience with a selection of their tools. As well as this, I am also extensively using Numpy[2], Matplotlib[3], and Jupyter[4]. A large portion of the project requires the manipulation of CSV files. I used pandas[5] as it is highly optimised for performance and has been successfully used in academic research prior. I also use the library provided by the client to extract information from the files

2.4.3 Licenses

All software dependencies of my project use permissive licenses that allow me to use their code without any restrictions. The data used is under an NDA and thus cannot be shared. In order

[1] `https://scipy.org/about/`
[2] `https://numpy.org/about/`
[3] `https://matplotlib.org/`
[4] `https://jupyter.org/about`
[5] `https://pandas.pydata.org/about/`

Algorithm 2 Monte Carlo simulation - Probability a driver will be knocked out.

Require: $S > 0$
  $S \leftarrow$ Number of simulations
  $M \leftarrow$ Gaussian Process Regression Model
  $P \leftarrow 0$
  for driver $\in$ Driver List **do**
    $\mu_d, \sigma_d \leftarrow M(\text{driver})$
  end for
  for $s = 0, 1, \ldots, S-1$ **do**
    for driver $\in$ Driver List **do**
    end for
    $P \leftarrow P + p$
  end for
  return $\frac{P}{S}$

to ensure I am conforming to every license I have published my project under an MIT license with added disclaimers as necessary with the data being withheld from the publication.

Software Dependency	License
SciPy Pandas Numpy Jupyter	3-Clause BSD
TQDM	MIT
Python Matplotlib	PSFL
Given Data	/

Table 2.1: Software licenses for all dependencies used within the project.

2.5 Requirement Analysis

2.5.1 The Original Success Criteria

The original success criteria created for the project proposal was as follows:

- *Extraction* - Produce a method to extract necessary data from the API, allowing for separation by timestamp so that I can simulate as if the data was being added to the database in real-time.
- *Implementation* - Implement a model that predicts the qualifying cut-off time and provides a probability of a given driver being faster than that time.
- *Evaluation* - Evaluate the model based on previously used models, and test the model on known results.

with the possible extension criterion:

- Increase accuracy by utilising more data from the API.
- Consider the likelihood of other drivers setting another lap and factor this into the probability of needing to send out your driver again.
- Wrap the trained models within a Python API such that they can be easily imported by Mercedes when needed.

2.5.2 Refining the Success Criteria

After having multiple meetings with the client, a few amendments were made to the original success criteria as well as flushing out the existing core criteria. The first criterion from the project proposal was split into more stages as it originally did not explain how this work should be done or any necessary requirements that the data extraction had. The third criterion was also updated to include how the evaluation would be completed.

The final extension is to package the project so it can be used by the client; it is important that the model can be used in real-time scenarios. Cut-off times calculated 3 minutes before the end of the session are used to influence decisions made in modifications to a team's strategy. This gives the team ample time to decide on a strategy and ensures the car can begin a flying lap before the session ends if necessary. Assuming all cars must be in the garage before this 3-minute cut-off, to allow for a new set of tyres to be put on, then it is safe to say that no laps will be completed between 4 minutes to the end of the session and 3 minutes to the end of the session (As this gives drivers the time required to complete the cool-down lap and return to the pit lane). This allows the model a 1-minute time frame at the minimum in order to produce a prediction to influence strategic decisions. If the project is to be packaged then it is important that it runs within this time frame.

From the meetings, another criterion was also generated. The client wanted to know whether a driver would pass into the next session and the associated confidence. It was suggested that this model could have an error rate (Resulting in the driver not being able to progress to the next session) of 1% as the loss associated with not progressing is much greater than the loss in strategic options due to having an extra tyre set. This was added to the success criteria as an extension. The first extension was also changed into a core criterion in order to ensure an improvement in accuracy over previous models.

2.5.3 Project Deliverables

Throughout the process of the dissertation I had regular meetings with the client; this resulted in the success criteria being constantly updated as we had more chats about progress and the direction the dissertation was moving.

2.6 Software Engineering Methodology

Due to the nature of the project having to change goals throughout the process (Due to meetings with clients and new ideas being brought up), the project fell towards requiring an agile development method [16] in order to efficiently meet any changing needs of the client. However,

Success Criteria	Type	Priority	Difficulty
Explore supplied API and files.	Core	High	Low
Decide which features to extract.	Core	High	Low
Produce a program for the extraction of features.	Core	High	High
Preprocess the relevant data.	Core	Medium	Medium
Implement the chosen Machine Learning model.	Core	High	Medium
Train the model.	Core	High	Low
Compute cut-off times with a confidence interval.	Core	High	High
Compute the probability a driver will progress.	Ext	Low	High
Evaluate the model against previous methods.	Core	Medium	Low
Package the project and deliver it to the client.	Ext	Med	Med

Table 2.2: Success Criteria for the project. The final 4 rows determine the deliverables while the former represents the steps required in order to deliver what is expected.

I switched to a waterfall development method in order to efficiently evaluate the project and ensure the needs of the client had been met.

2.7 Starting Point

I have used my personal laptop (Intel Core i5 1.00GHz, 8GB RAM, Windows 10) and data the client provided to complete this project. The data provided uses an API accessible via Python libraries to read through a large collection of files, once these files are downloaded onto my laptop an internet connection is not required to access the data.

I have previously used both Numpy and Scipy before, with Numpy being used extensively for personal projects. However, I have no knowledge of SciPy's implementations of Gaussian Processes. I also only have limited knowledge of Pandas.

While the part IA Scientific Computing course gave an introduction to Jupyter Notebooks, Numpy, MatPlotLib, and Python, and the part IB Artificial Intelligence course gave an introduction to ML, Gaussian Processes and Monte Carlo simulation were new to me as of the beginning of the dissertation.

It is important to note that while my project builds on previous work, I have not had access to any of the previous work and as such all code written is my own.

2.8 Verification

I verify my model by comparing the program's outputs to the qualifying sessions in the held-out test set where possible. Furthermore, I compare the model to the previous model [1] and Mercedes' predicted lap times.

Chapter 3: Implementation

This chapter introduces the work undertaken to implement the success criteria defined in Chapter 2. This is achieved via the following stages that are presented sequentially: feature selection (§3.1), data extraction and processing (§3.2), model training (§3.3), predicting qualifying times (§3.4), and extensions (§3.5). Lastly, a data flow diagram (§3.6) and repository overview is presented (§3.7).

3.1 Feature Selection

In the context of Machine Learning the dataset used is vitally important. The data supplied by the client is extensive, containing over 200 different data points (with some having a separate measurement per lap, per second, per driver, per session, etc.) that must be filtered in order to ensure the model trains fast enough to be usable within the time constraints of a qualifying session (§2.5.2). It must also ensure there is enough variation in the filtered data in order to avoid overfitting [17] of the model to the inputted data.

By filtering the data used to train the model we produce a subset of the original data supplied by the client; this subset of data will become the dataset I process in order to train and test the model. For experimentation purposes, the dataset is split into two parts, a training set and a testing set at a ratio of 80/20. This ratio is based on the Pareto Principle [18], which states that about 80% of the consequences are produced by 20% of the causes

Table 3.1 gives an overview of the features chosen for the model, as well as extra information pertaining to each feature. Categorical data refers to data that does not have a numeric representation, i.e. data that is stored as a string. Data that is not stored in the API must instead be inferred from other data that is stored in the API.

As the model chosen expects all inputs to be either numerical or lists of numerical values, the features that are categorical must be encoded in such a way that allows them to be interpreted as numerical values. The methods used are included in §3.2.

Feature Motivation

The driver's abilities vary greatly; for this reason it is important that *Driver Name* is included as a feature in the model input. Furthermore, while the lap times depend on the driver skill, the car used also impacts the lap times, thus it is necessary that *Driver Team* is included as well.

Another aspect of lap time is the tyre compound used (Explained in §2.1.1), as a tyre with more grip will reduce the time required to complete a lap. The age of the tyre directly impacts lap time as each tyre has a reduction in grip per lap completed using it. These reasons result in the inclusion of *Tyre Compound* and *Tyre Laps* in the input vector.

Feature	Type	Description	Example	In API
Driver Name	Categorical	Unique Identifier for each driver (full name).	Lewis Hamilton	Stored
Driver Team	Cateogorical	Unique Identifier for each team.	Mercedes	Stored
Tyre Compound	Categorial	Which tyre compound the lap used.	Soft	Stored
Tyre Laps	Numerical	How many laps have been completed on the current tyre set.	5	Calculated
Circuit	Categorical	Which circuit the lap was set on.	Barcelona	Stored
Year	Numerical	Which year the lap was set during.	2017	Stored
Session	Categorical	Which qualifying session the lap was set during.	Q1	Stored
Laps Completed by all drivers so far	Numerical	How many laps have previously been completed during the session.	73	Calculated

Table 3.1: Selected input features for the model.

Each circuit also has a different shape and uniquely defines an expected lap time. Circuits can also change between years (as well as the cars, and drivers) and as such both *Circuit* and *Year* are included.

As explained in §2.1.1, one effect of using Pirelli tyres designed for racing is that rubber is lost from the tyres and remains on the track. The result of this is that drivers set faster lap times as the number of laps completed by all drivers increases. This number grows very large and thus is reset to 0 as each session changes. This means both *Session* and *Laps Completed* are added to the input vector.

Circuit and Year

Previously used models only incorporate data from the current grand prix weekend, resulting in a minimal amount of data being used to train the model. While a GP can provide accurate predictions with lower amounts of training data compared to other Machine Learning models [19] an increase in the amount of training data can increase accuracy and confidence.

Including the circuit and year within the input vector allows for the use of data from the previous grand prix, even though the cars and drivers will have been updated between the years. This increases the amount of training data we have for the model and thus, as long as we are careful to ensure overfitting does not occur, will improve the accuracy and confidence of the model.

3.2 Data Extraction and Processing

I developed a simple pipeline for pulling data from the client-supplied API and storing it in CSVs for easier usage and processing (§3.2.1). From here we could process the CSV file to process (§3.2.2) the data and convert it to a format that maximises the accuracy of the model when inputted.

3.2.1 Data Extraction

Data extraction is a slow process ($\approx$ 30 seconds per session) while using the client-supplied API. This is due to the API storing and opening all of the data that has been collected during the grand prix weekend. However, most of this data is not necessary for the model input. It was important that the necessary data be extracted from the API and stored in another file type to allow for quick access and manipulation for processing. One goal of this project is that the program should be able to give a quick answer as to whether a given driver should set another lap, by reducing the time spent in extracting the data we ensure that the model can spend more time on its training, allowing for more accurate results. Table 3.1 also shows that some data is not explicitly stored within the given API and must be calculated by extracting various other sources of data. An important note is that the feature *TyreLaps* is dependent upon not only qualifying sessions but also practice sessions as tyres are shared between all sessions, resulting in 4 sessions (Practice Session One (P1), Practice Session Two (P2), Practice Session Three (P3), Qualifying) that need to be loaded and have data extracted from before we are able to train the model.

For this reason, extracted data is stored in CSV files due to Python having many libraries allowing for the processing of data stored within these files. By storing data in quickly accessible CSV files we allow the client to extract all necessary data from previous races (including the P1, P2, and P3 data) before qualifying begins. This reduces time spent extracting data during the time-sensitive qualifying session and allows the client quicker access to the output of the model.

Data is stored in the client-supplied API using four formats: *string*, *real*, *bool*, and *DateTime*. During extraction, all are converted to their Python equivalent automatically via the use of NumPy[1] before storing it in the CSV file.

As well as the data required to generate an input vector satisfying Table 3.1 we also extract *ExpectedLapTime* and *ActualLapTime* in order to calculate a corresponding output following Equation 2.4.

3.2.2 Data Processing

Even though the data is now stored in a more usable format, it is still not ready to be used as input to the model. The amount of data extracted is 4 times larger than necessary (as it includes data for P1, P2, and P3) to calculate the extra features not stored in the API. Once these new features are calculated the data must be filtered to remove the extra (now unnecessary) information. Finally, the data must be processed into a numerical vector as that is the expected input to the model.

[1] `https://numpy.org/about/`

Augmenting Data

While the client API gives a very large amount of data, many useful features are not included within the data explicitly. As previously mentioned, two features that are not included but are needed for the feature vector are *Laps Completed* and *TyreLaps*. Both are important for the reasons given in §3.1 and thus must be inferred.

- *TyreLaps* references the number of laps a given set of tyres completed before the current lap. It is calculated by storing the *TyreID* of the tyre used for each lap stored in the CSV. Using this we can then order the laps by also storing the *Time of day (TOD)* and add a column that counts the number of times that TyreID has appeared before in the CSV for each row. As the *TyreID* is not unique per circuit, care must be taken to ensure that the value resets to 0 for each race weekend.

- *Laps Completed* references the number of laps completed in total for a given session; it is calculated by ordering the laps set (based on the circuit, session, and *TOD*) and adding a column that increments each row. This should be unique per circuit and session, resetting to 0 every time we change the circuit or session.

In order to increase efficiency we can calculate *Laps Completed* after filtering any laps from sessions that are not needed within our input vector. This will not result in a different column should we calculate *Laps Completed* before filtering due to *Laps Completed* resetting to 0 each time the session changes; this means that the number of laps set in any session we filter out during later stages would not impact the *Laps Completed* of the sessions we store within our input vector. However, we must calculate this new column before filtering out any in-laps and out-laps as these laps also contribute to the rubbering-in effect. *TyreLaps* must be computed before any filtering occurs as earlier sessions can impact the tyre life of the tyres used in qualifying.

Once we have these augmented columns added to the CSV, each row will specify the information required to produce a single instance of the input vector. The row is of the form shown in Table 3.2

Index	**Driver**	**Team**	**Compound**	**TyreLaps**	**Circuit**
0	Lewis Hamilton	Mercedes	44003	1	Austria
Year	**Session**	**LapsCompleted**	**IsTimedLap**	**Lap Time**	**Expected Time**
2018	Q1	6	True	65.342	67.234

Table 3.2: Example row from the augmented CSV with the extra features that were not directly stored within the API and had to be calculated.

Filtering Data

It is important that we remove any recorded laps from the dataset that provide inaccurate timings for how long a driver took to get around the circuit. This means that only flying laps are relevant to the model as they provide the most accurate timing. All other laps are considered outliers and would result in a decrease in the accuracy of the model. As well as this, it is not possible to assume that flying laps set in previous sessions of the same race weekend

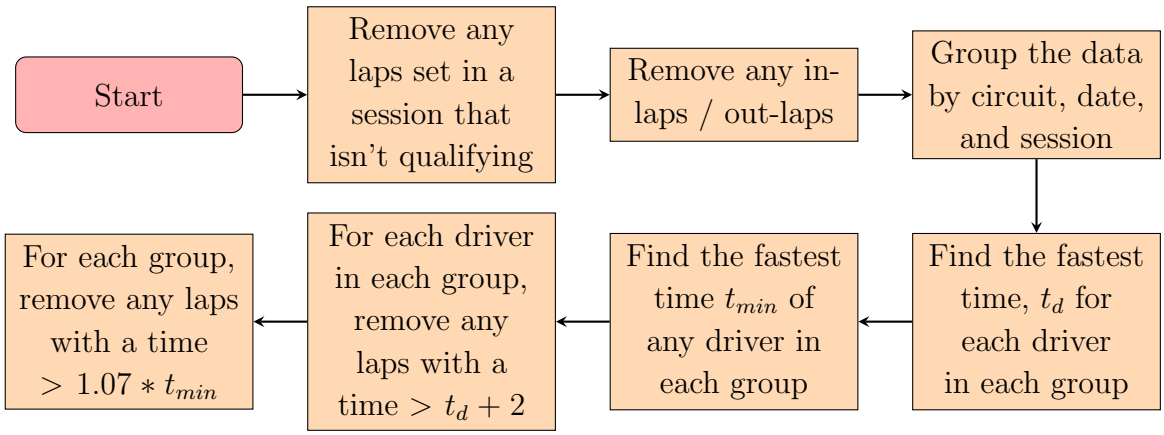

Figure 3.1: Flowchart specifying how laps should be filtered from the CSV produced by pulling all necessary data from the API. This flowchart removes any anomalous data points and leaves only healthy training data for use in later stages.

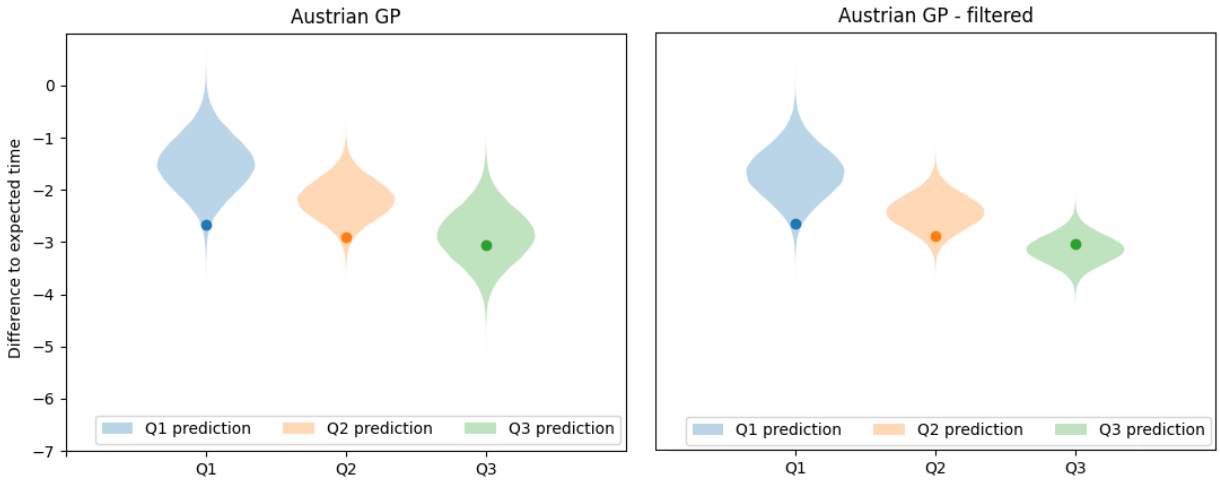

Figure 3.2: Violin plot showing the distribution of predictions for the 2020 Austrian GP for Lewis Hamilton if the filtering heuristics are not applied, and then if they are applied.

provide realistic timings of what a driver will set in qualifying and thus these must be removed as well.

Removing laps set in previous sessions is straightforward as pandas allows us to filter based on a column value and thus we can keep rows that have a value $\in \{Q1, Q2, Q3\}$ in the **Session** column. We can also easily remove any laps that aren't timed laps by filtering on the **IsTimedLap** column, which is generated when extracting information from the client API as the API stores whether a lap is an in-lap or an out-lap. (**IsTimedLap** $= \neg($**In-lap** $\vee$ **Out-lap**$)$)

Removing slow laps while ensuring we keep all of the flying laps is much more complicated as the client API has no field that distinguishes between the two, thus a heuristic must be used that minimises the number of flying laps filtered while maximising the number of slow laps that are filtered. The chosen heuristic for this was to, for each driver, find the fastest lap time set per session and remove any laps that take more than 2 seconds longer than this fastest lap as the assumption is that the driver was not attempting to drive as fast as they could. However, this heuristic alone is not perfect as a fixed 2-second increment does not take into account the

expected time taken to perform a lap. Furthermore, if the driver is to set only one fast lap then it will be accepted irrelevant of the time taken. This means we add another heuristic that takes into account other drivers' times as well. By finding the fastest time of any driver for a given weekend, t_{min} we remove any laps that are more than $1.07 * t_{min}$ seconds long. This value comes directly from the sporting regulations [7, section 42.2] and is the determining factor for whether a driver is allowed to participate in the race. Unlike the previous heuristic, this one is not calculated per driver and thus does not have the problem of a driver only setting one lap. Furthermore, it is also track-dependent as it is multiplicative and, as it comes directly from the sporting regulations, has been refined over the years according to FIA standards. The value 1.07 was used as it was deemed to be the correct value for ensuring a balance between the competitiveness of all competing cars and the safety of all drivers by removing any slow cars that could block faster cars and increase the risk of an accident occurring.

The effect of applying filtering is shown in Figure 3.2, with the violin plot for each qualifying session being compressed and the actual results moving closer to the mean of the violin plot after filtering is applied.

Processing Data

The final step after filtering is to process the data such that it can be input to the model. Four features within the input vector are held as categorical data and thus must be converted into numerical data [20]. Two methods are chosen for the categorical data encoding, one-hot encoding [21] and label encoding [22]. One-hot encoding generates a new binary feature for each possible unique value that the data can take while label encoding instead produces a single feature whose value depends on a function of the categorical data.

One-hot encoding is used to encode the *Driver Name*, *Driver Team*, and *Circuit*. An important note here is that as the model is designed to be trained on historical data as well as current data, it is possible that the number of possible values that a feature can take could grow very large. In order to combat this, the one-hot encoding function will also have an extra binary feature that is set to 1 should none of the other features fit (representing an unknown element). The one-hot encoding function is designed to take a `.txt` file input allowing the user to enumerate the binary features (based on how many inputs are in the file) as well as which values are valid for a given categorical data type.

Label encoding is used to encode the *Session* and *Tyre Compound*. Unlike one-hot encoding, label encoding does not increase the dimension of the feature vector and, as a consequence, produces an implicit relation between the labels generated. The motivation for using label encoding rather than one-hot encoding for these two features is that the average lap time decreases as the *Session* increases (in the order Q1, Q2, Q3) as the slower cars are removed. There is also an established order in how much grip, and thus the expected lap time, each tyre compound can give (explained in 2.1.1). If label encoding had been used for the other categorical data inputs then the model would learn an implicit relationship between the order of the labels given to the possible values which would not exist in reality.

Tables 3.3, 3.4, and 3.5 show a single example of an input to the model once filtering and categorical encoding has been applied. They are based on the row in Table 3.2

The last stage of data processing was normalisation. Of the 3 numerical values in the feature vectors, the possible ranges differ greatly. For example, *Tyre Compound* can range from 1 to 3 while *Laps Completed* can range from 0 to over 100 depending on the session. If feature scaling

Driver																			
Ham	Bot	Ver	Per	Ric	Sai	Alb	Lec	Nor	Gas	Str	Oco	Vet	Kvy	Hül	Räi	Gio	Rus	Gro	Mag	Other
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.3: First 21 elements of an input generated based on the first 5 races of the 2020 season for Lewis Hamilton on soft compound tyres.

Team									
Mercedes	Red Bull	McLaren	Racing Point	Renault	Ferrari	Alphatauri	Alfa Romeo	Hass	Willaims
1	0	0	0	0	0	0	0	0	0

Table 3.4: Elements 22-31 of an input generated based on the first 5 races of the 2020 season for Lewis Hamilton on soft compound tyres.

were not applied to these values then the model will give some features more importance than others when calculating an output [23]. The chosen normalisation method for this application was z-score normalisation, which normalises each element with Equation 3.1

$$x_{\text{normal}} = \frac{x - \mu}{\sigma} \tag{3.1}$$

where μ and σ are calculated as the mean and standard deviation of a given column.

This results in an updated column of features that have a mean of 0 and standard deviation of 1 which has been successfully applied to problems with similar ranges of data [24].

3.3 Model Training

There are two possible choices of models that could be used for the project based on the criteria, an GPR model or a Gaussian Process Classifier (GPC) model. Both models could be used to predict cut-off intervals (core criterion) and predict whether a driver should remain in the garage (extension criterion). While the latter model requires less extra work for the extension task than the GPR, it would require much more extra work for the core task than the GPR would. As it is more important to complete the core criterion, a GPR was selected. The model will receive an 8-dimensional input and return a 1-dimensional real value determining the predicted time given the input.

This model is suitable for the given task as, along with returning a mean output, it also returns standard deviations which allow for the calculation of confidence intervals that can be used to decide if a given output should be accepted by the client.

3.3.1 Gaussian Process Regression Model

As just said, this model is trained to predict lap times for a given driver based on a given input with 8 dimensions (before applying categorical encoding). The model makes certain assumptions, which were either given by the client or agreed upon by the client, such as assuming dry conditions and a mechanically stable car. These assumptions are necessary for this model as a race weekend becomes too uncertain to make accurate predictions for should either assumption not hold [25]. By not incorporating these assumptions we risk overfitting the data and reducing the accuracy of the model on the test dataset.

Circuit			Year	Session	Tyre Laps	Laps	Compound
Austria	Hungary	Great Britain					
1	0	0	2018	1	1	6	3

Table 3.5: Elements 32-39 of an input generated based on the first 5 races of the 2020 season for Lewis Hamilton on soft compound tyres.

There are two ways to combine kernels within a GP, through multiplication or addition. Multiplicative kernels can be thought of as an AND operation on the features while an additive kernel can be thought of as an OR operation on the features[1]. As the fastest lap is likely to be set when all of the features of the model are at their optimal value we can increase accuracy by using an additive model. Adding two kernels results in element-wise addition of their corresponding covariance matrices. The covariances of two added kernels will only have a low value if the covariances of the two kernels both had low values.

3.3.1.1 Kernels

Kernel choice was made using hyperparameter optimisation described in §2.3.2 as well as trial and error for picking the kernels. Various kernel choices and their respective MSE loss are presented in Appendix A. The method of trial-and-error was used to pick how kernels were combined while hyperparameter optimisation was used to ensure that all kernels were given a fair chance in maximising the accuracy of the model. While the model used was additive, multiplicative kernels were considered to ensure that an additive model was the correct choice. During this process, it was also found that there was no set of kernel hyperparameters that would reduce the error for all qualifying sessions. As such it was instead decided that hyperparameters should be trained on a per-model basis in order to ensure a minimisation of MSE. This was achieved by taking the laps set in the qualifying session we are attempting to predict the cut-off time for and splitting them into 2 equal-sized groups. One group was used to train the model while the other is used to find the optimum hyperparameters.

The final choice made was an `RBF() + RBF() + ConstantKernel()` which had an average MSE of `0.24979` across the training models. While an RBF kernel is not perfect, as it does not accurately simulate all of the expected results, it gives the minimal MSE and is, therefore, the kernel that best models the actual lap time. Figure 3.3 gives an example of how the kernel does not perfectly map to the expected function. The plots should decrease as the number of laps completed increases however at $\approx n = 45$ one plot instead begin to increase. This occurs as there is minimal training data for Q2 of which the "Laps Completed" input exceeds 50, resulting in unpredictable behaviour from the model when we try to predict beyond this value.

3.3.1.2 Hyperparameters

Hyperparameter choice is dependent upon factors such as the kernels chosen and how those kernels were combined. As stated earlier, all hyperparameters were optimised using a random search on a logarithmic scale.

$$\text{LogUniform} \propto \exp(\text{Uniform}(\log(a), \log(b))) \tag{3.2}$$

[1] `https://peterroelants.github.io/posts/gaussian-process-kernels/`

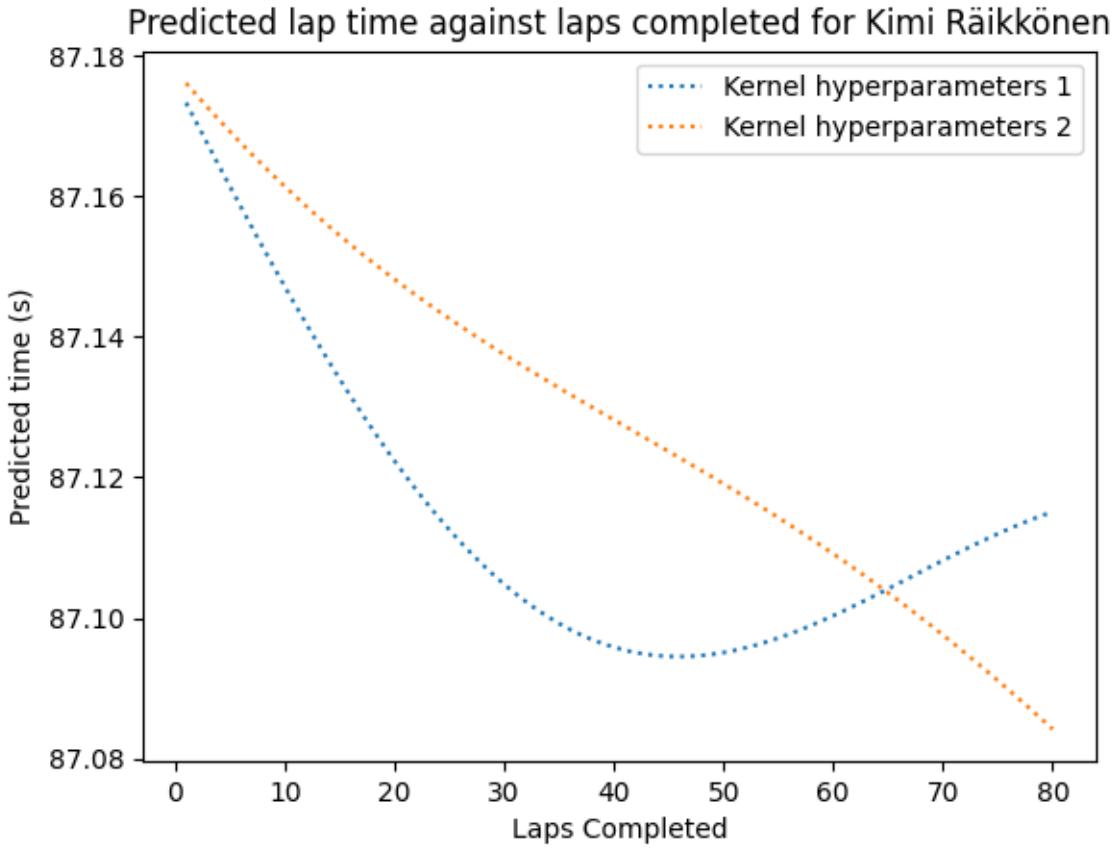

Figure 3.3: How model predictions change as the number of laps completed in the input increases. This plot is for Kimi Räikkönen at the 2020 Silverstone Grand Prix, during the second qualifying session (Q2). Two produced kernels are shown due to the random factor used in generating the kernel hyperparameters.

where a and b are the lowest and highest ranges. For hyperparameter tuning, these were initialised at $1e^{-10}$ and $1e^{1}0$. A logarithmic scale was chosen due to the large range of possible values that a hyperparameter can take [26]. Hyperparameters are calculated each time the model is trained; efficiency is important and using a uniform scale would decrease efficiency as a result of requiring more iterations to reach an optimum value.

As the hyperparameter values are distributed from a probability distribution, randomness is used to ensure a fair distribution. This results in a different set of hyperparameters each time the model is trained, leading to variation in the predicted cut-off times if the model is trained on the same data multiple times.

3.3.1.3 Model Evaluation Framework

To complete the final success criterion it is important that I can efficiently evaluate the model against the previous methods. This was produced by creating a super-class for each predictor (the one produced for this project, the previous model, and the current model.) that had functions that allowed for the retrieval of a variety of metrics that could be used to compare the models. Creation of this super-class (and the corresponding sub-classes) allowed for efficient loading of the data for the previous models, and loading of the model for the new implementa-

tion. This is shown in Figure 3.4.

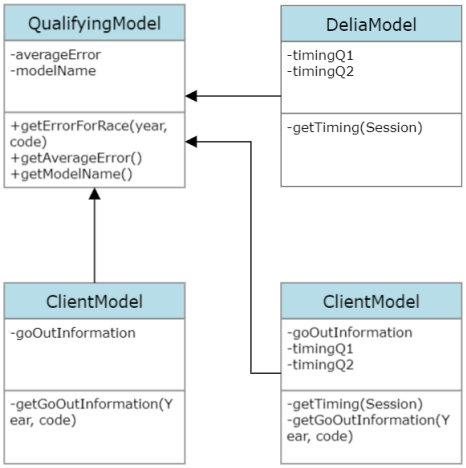

Figure 3.4: UML diagram of superclass designed to aid with the evaluation of the project.

3.4 Predicting Qualifying Times

As the output of the model is a mean and standard deviation for a lap time for a given driver, we can predict a cut-off time for Q1 and Q2. These values could be used directly in order to quickly calculate the mean cut-off time, the standard deviation is a lot more difficult to calculate numerically due to the requirement to sort the normal distributions (In order to find the $16^{\text{th}}/11^{\text{th}}$ value and thus the cut-off time). Thus, a solution that was easier to derive and encode was found and used instead.

This solution used a Monte-Carlo simulation assuming that every driver set one more lap time. The Monte-Carlo simulation would use the mean, μ_d, and standard deviation, σ_d, for each driver, d, and generate a large number of samples, N, for every driver which could then be organized to calculate a cut-off time. Unlike previous models, the simulation created also took into account that the order in which the drivers set the laps were not fixed. The original solution to this was calling the model 20 times for each driver, one call per possible position they set the lap, thus producing a total of 20 possible means and standard deviations per driver. The Monte-Carlo simulation then also randomized the order of the drivers during the simulation. While this would result in an increase in accuracy it was at the cost of a significant increase in time as it was no longer possible to use native-C libraries such as NumPy to perform the calculations. Without the native-C functions, the time required to sample the model was $\approx 50\times$ the time required while using native-C functions and another solution was found. This solution was to generate one new input to the model with all drivers setting one new lap time, this input would then be stacked multiple times and the column showing how many previous laps had occurred when a driver set their lap would be randomised such that each unstacked section would be a valid input with no jumps in laps. This input could then be fed to the model, which used native-C functions to quickly calculate the output, and the output could be analysed for a cut-off time. This method also results in an increase in accuracy that can be seen in §4.1.1. Algorithm 3 gives an implementation of this.

The output of this algorithm was originally a single real value that determined the cut-off time but was later updated such that it included confidence intervals as well. By only returning

Algorithm 3 Sampling directly from the model.

Require: $R > 0$ ▷ R is the repeat factor.

$M \leftarrow$ Trained Gaussian Process Regression Model
$E \leftarrow$ Driver expected times
$D \leftarrow$ Driver data
$L \leftarrow$ Lap to begin counting from
$D[\text{LapsCompleted}] \leftarrow L$
$T \leftarrow \text{len}(D) - 5$ ▷ Number of drivers that will pass through the qualifying session.
$\text{laps} \leftarrow \text{np.arange}(T + 5, \text{dtype} = \text{int})$ ▷ $[0, 1, \ldots, T + 5 - 1]$.
$\text{total\_laps} \leftarrow \text{np.tile}(laps, R).\text{reshape}(R, -1)$ ▷ Repeat the laps array and make it a matrix.
$\text{np.sort}(\text{total\_laps}, \text{axis} = 1)$ ▷ Sort each row of the matrix independently.
$\text{duplicate} \leftarrow D * R$ ▷ Repeat D R times.
$D[\text{LapsCompleted}] += \text{total\_laps.reshape}(-1)$ ▷ Convert to a list and add.
$P \leftarrow M(D)$ ▷ Apply the input to the model.
$O \leftarrow \text{np.reshape}(P, (T + 5, -1))$
$O \leftarrow \text{np.sort}(O, \text{axis=0})$
$\mu \leftarrow \text{np.mean}(O[T - 1])$
$\sigma \leftarrow \text{np.std}(O[T - 1])$

the real value we are unable to tell the standard deviations in the results produced by the model and cannot evaluate Equation 3.5, resulting in a lack of willingness to use the result from the client. Cut-off time is calculated via

$$\mu = \frac{\sum_i^N \text{orderedTimes}_i[n]}{N} \tag{3.3}$$

where $n \in \{16, 11\}$ and determines whether you are predicting the cut-off time for Q1 or Q2. The standard deviation is calculated via

$$\sigma = \sqrt{\frac{\sum_i^N (\text{orderedTimes}_i[n] - \mu)^2}{N}} \tag{3.4}$$

There is an inherent trade-off between accuracy (as increasing the number of simulations improves accuracy), time (as increasing the number of simulations increases time requirements), and memory (as increasing the number of simulations increases memory requirements). The confidence intervals are very important for real-time decisions and thus it is necessary that the standard deviations are calculated as well.

95% confidence intervals are calculated as:

$$\begin{aligned} \text{lower\_bound} &= \mu - 1.96\sigma \\ \text{upper\_bound} &= \mu + 1.96\sigma \end{aligned} \tag{3.5}$$

3.5 Extensions

There were two extensions defined when proposing the project, these were both completed and are outlined below.

3.5.1 Predicting if a driver needs to go out again

To predict if a driver needs to go out again or if it is worth remaining in the garage, in order to save tyres and other resources, we assume a worst-case scenario in that every driver that is currently slower than d sets one more lap. Unlike the core criterion, we will not extend the scenario to assume a differing lap order for all the drivers and instead use the same lap number for all drivers (next_lap = latest_lap_number + 1). We do this as, while accuracy is obviously important, it is more important to overestimate safety as the loss for being unable to move into the next qualifying session is much greater than the loss in strategy options due to using an extra set of tyres to set a better lap and move into the next session.

As this prediction is safe, it is important to give a confidence interval such that the client can still play an important role in deciding if the car should set another lap, even if the model suggests not.

Due to the safety assumptions there is only one mean, μ_d, and standard deviation, σ_d, for each driver. This allows the use of native-C normal distributions and, unlike the algorithm used to predict qualifying times, uses a Monte-Carlo simulation (algorithm 2) to predict whether a driver needs to go out again. Figures 3.5 and 3.6 show the probability that a driver will proceed into the next session if they remain in the garage that the model produces. Further evaluation of this extension is presented in §4.3

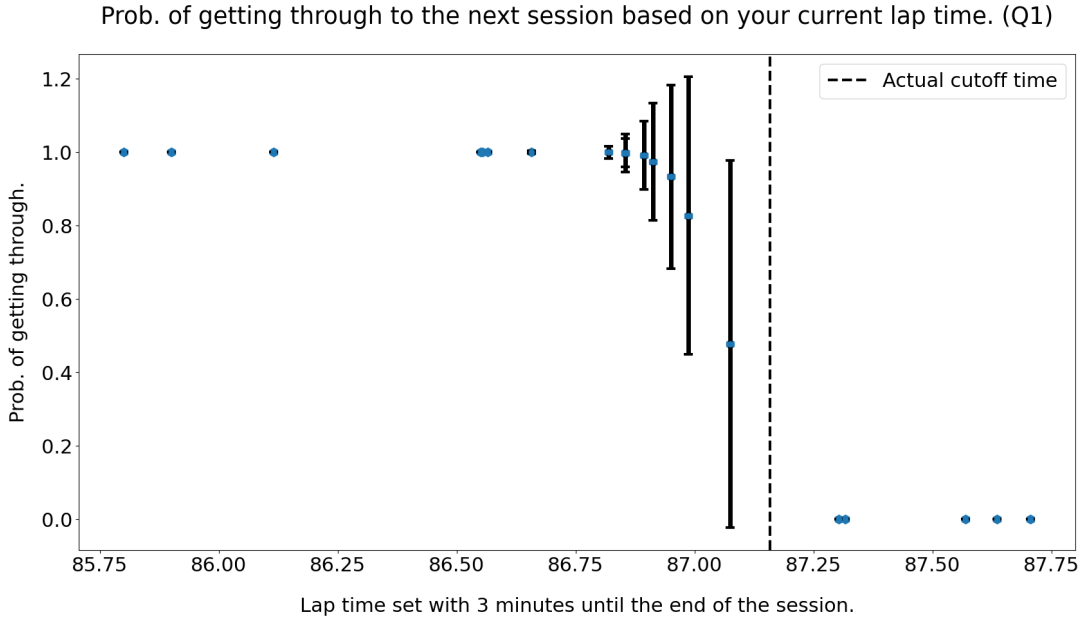

Figure 3.5: Probability of getting into Q2 given a current fastest lap time. Lap times are generated as the fastest lap each driver set with 3 minutes before the end of Q1. Error bars show confidence intervals for the probabilities.

3.5.2 Packaging the project

In order to send the project to the client I converted the project into a `.whl` file that could be installed by the client via `pip`. This method was chosen as the client supplied `.whl` files for accessing their data and as such they have knowledge of how to use them. This was also talked about in meetings with the clients and was the agreed method of packaging.

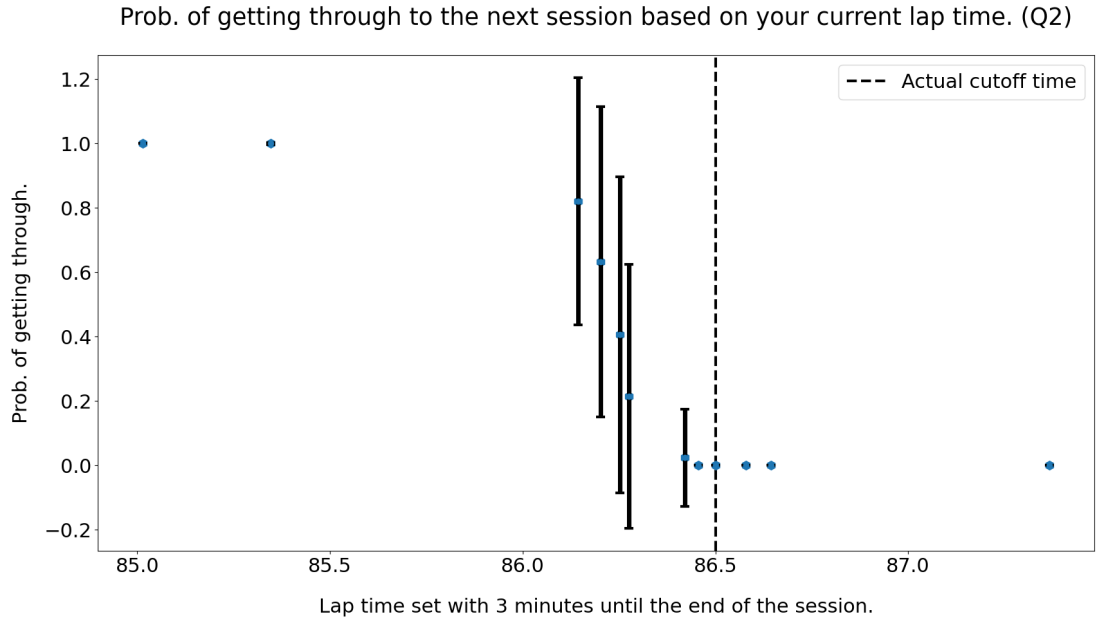

Figure 3.6: Probability of getting into Q3 given a current fastest lap time. Lap times are generated as the fastest lap each driver set with 3 minutes before the end of Q2. Error bars show confidence intervals for the probabilities.

3.6 Data Flow

Data is constantly moved between processes and data stores during the execution of various sections of the program. Figure 3.7 shows how this data is moved. It also shows the expected inputs from the user.

3.7 Repository Overview

Figure 3.7 gives an abstract view of the repository. Data files are omitted in order to comply with the NDA signed. The most important folders are `DataExtractionProcess` and `ModelUsage`. `DataExtractionProcess` contains all the required modules to convert the data to a CSV and then process the CSV while `ModelUsage` contains all the required modules to perform the predictions explained above. Together, these two folders constitute the backbone of the project. As well as these two folders, several scripts were created for the evaluation of this project. These are stored in `EvaluationFiles`.

The extension to package the project such that the client can easily download and install the modules themselves changes the folder structure significantly. Before this change, the `src/` and `tests/` folders were the roots with their children existing in the same relative position. However, completing this extension changed the folder structure to what is shown.

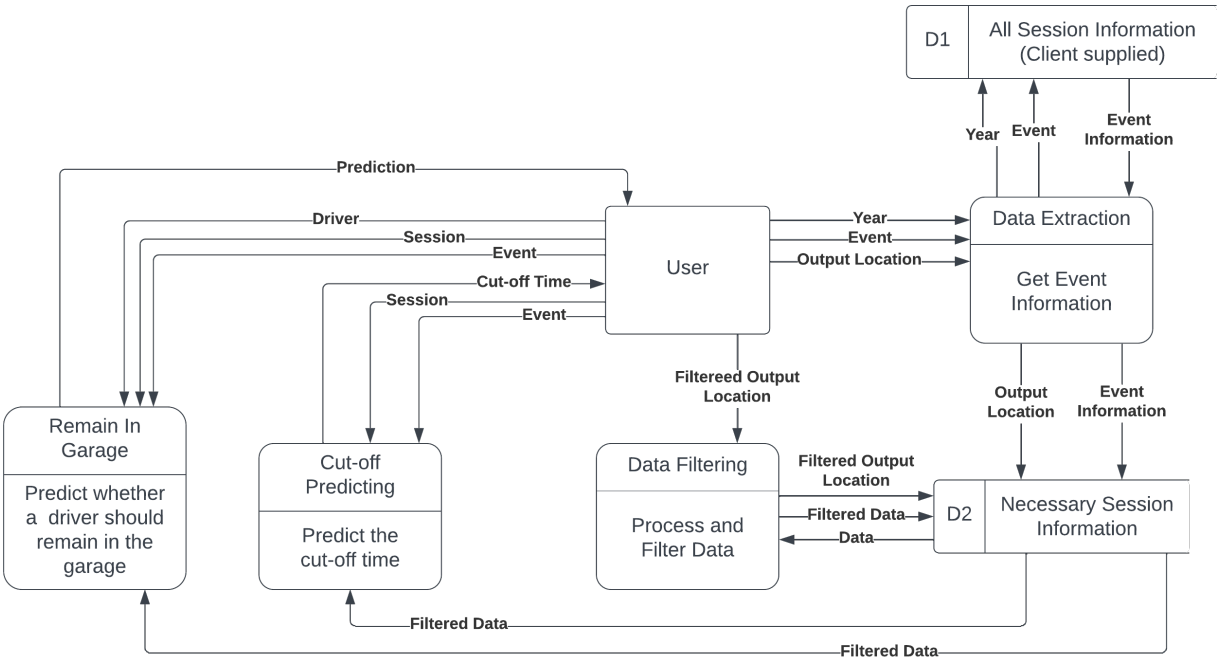

Figure 3.7: Data flow diagram using the Gane & Sarson notation [27]. The diagram shows data flow for all aspects of the project, including data extraction and processing, predicting cut-off times, and predicting if a driver should remain in the garage.

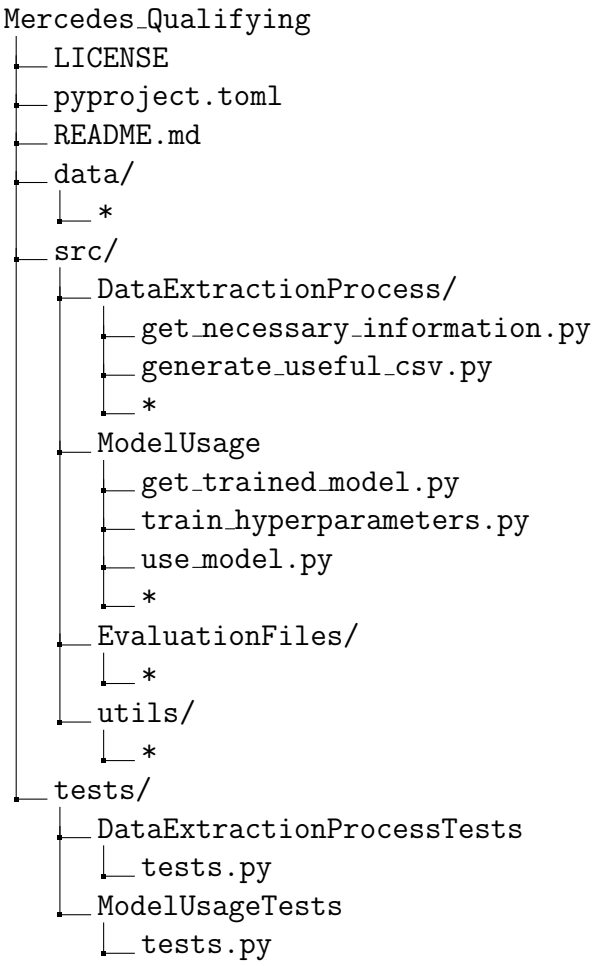

Figure 3.8: Overview of the folder structure for this project. * represents files that exist but are not shown (either due to NDA compliance or providing only helper functions.)

Chapter 4: Evaluation

The core success criterion of this project, to produce a model that accurately predicts qualifying cut-off times, has been completed. In the following section, I will provide an evaluation of the various techniques used to achieve this, as well as a critical evaluation of the model to show that it completes the core criteria. I will also give an evaluation of one of the extension criteria, classifying if a driver should remain in the garage.

4.1 Predicting Cutoff Time Accuracy

An evaluation of the accuracy and usability of QualifAI is given in this section. This includes an analysis of the sampling method (§4.1.1), accuracy compared to other models and an analysis of the QualifAI's accuracy (§4.1.2, §4.1.3), an analysis of the confidence intervals (§4.1.4), and an analysis of the time required for QualifAI to run (§4.1.5).

In order to evaluate the model's accuracy on unseen data, all training is completed on the 2016-2018 data and as such these data points are excluded from the test set. The test set will only contain data from the 2019-2020 races. Q1 and Q2 times are calculated for each race in the test set by using data available 3 minutes before the end of a session and the data of the previous 4 races as well. Cut-off times are calculated as the predicted time for the driver in 16^{th} position and 11^{th} position.

To quantify the comparisons between models, accuracy is measured in the MAE of the model over the test races between the predicted times and actual cut-off times. MAE is one of the most suitable methods for evaluating a regression task [28] and is calculated as:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i' - y_i| \tag{4.1}$$

where y_i' is the predicted value and y_i is the actual value.

4.1.1 Sampling Method

There are two ways to predict cutoff times. One is to use the same value for "LapsCompleted" for each driver when they set their next lap and the other is to use a randomised lap order. The clear disadvantage of the former is that it is likely to limit the accuracy of the model by providing unrealistic and overly safe conditions, resulting in the cut-off time prediction being slower than the actual result as many drivers would set faster times than predicted by the model. However, the latter also increases the complexity of the problem. This is demonstrated from the initial tests in Table 4.1.

From now on, I shall only present results that use a randomised lap order unless stated otherwise.

Qualifying Session	Same lap MAE (s)	Randomised lap order MAE (s)
Q1	0.3689	0.1427
Q2	0.1587	0.1230
Average	0.2638	0.1329

Table 4.1: Comparison between assuming every driver sets one more lap with the "LapsCompleted" property being constant for all drivers and assuming a different random order per sample.

4.1.2 Model Comparisons

The client already has a model that they use to predict a cut-off time for qualifying sessions in all races. Deliver also produces cut-off times. Deliver already produces a significant increase in accuracy over the client-predicted Q1 cut-off times with this effect being multiplied in the Q2 cut-off times. Both Deliver and QualifAI also produce confidence intervals, which will also be compared, while the client's model does not.

As QualifAI takes into account lap order and uses more data, it is expected that the accuracy should be higher than Deliver and the model the client currently uses. Results should show a reduction in MAE for QualifAI when compared to the MAE of both the client produced model and Deliver. These expected results are demonstrated in Table 4.2.

Qualifying Session	QualifAI MAE (s)	Client predicted cut-off times MAE (s)	Fixed-Hyperparameter Deliver MAE (s)
Q1	0.197	0.400	0.288
Q2	0.128	0.330	0.213

Table 4.2: Comparison between the MAE between three models used to predict cut-off times for the 2019-2020 races. Deliver is trained on the 2016-2017 races [1] and kernel choice for QualifAI is trained on 2016-2018 races.

QualifAI produces a significant increase in accuracy over the model the client currently uses, and an increase over Deliver. For Q1 QualifAI produces a MAE improvement of 0.203 seconds and an MAE improvement of 0.202 seconds for Q2 over the model currently used by the client. The prediction for Q2 will almost always be more accurate than the prediction for Q1. It has more training data available (≈ 35 more data points) with the data used for Q1 as a subset of the training data.

QualifAI produces the best cut-off predictions of the three models for both Q1 and Q2.

4.1.3 Cut-off Accuracy

In order to demonstrate the success criterion of producing a model that accurately predicts cut-off times, the error between the model's predictions and the actual cut-off times for the test sets will be used as an indicator. While a minimal MAE is obviously optimal, the client suggested that an average MAE less than 0.3 seconds provides useful information. The client's current model predicts a cut-off time less than 0.3 seconds off of the actual cut-off time approximately 45% of the time for Q1 and approximately 50% of the time for Q2. It is expected that QualifAI should improve on this.

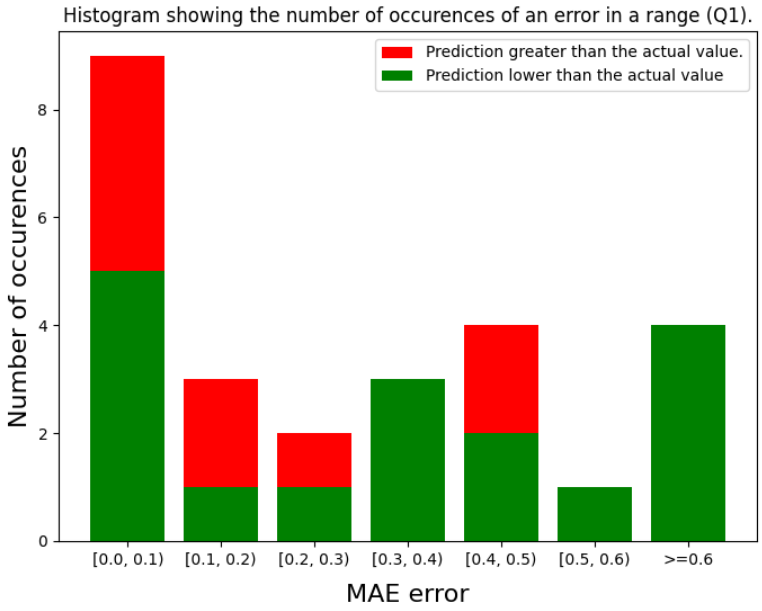

Figure 4.1: Histogram showing how many predictions fall within a certain distance from the actual cutoff time for Q1. Green bars represent predictions that are less than the actual cut-off time and red bars represent predictions that are greater than the actual cut-off time.

Table 4.1 shows that 30% of all predictions from QualifAI are within 0.1 seconds of the actual cut-off time for Q1 and 50% of all predictions are within 0.3 seconds for Q1. Figure 4.2 shows an improvement upon Figure 4.1, again providing evidence that the model performs better in Q2 than Q1, with 40% of all predictions being within 0.1 second of the actual cut-off time and 80% of all data being within 0.3 seconds.

These predictions are a direct improvement over the predictions supplied by Deliver and the model the client currently uses. Deliver provides predictions such that 45% of all predictions fall within 0.2 seconds of the cut-off time for Q1, which is matched by QualifAI. For Q2, Deliver has 56% of its predictions fall within 0.2 seconds, while QualifAI has 69% of its predictions fall in the same range.

Figures 4.3 and 4.4 show the difference in accuracy between qualifAI and the client's current model, with error bars added to the accuracy of the QualifAI as necessary.

4.1.4 Confidence Intervals

A confidence interval is a range that is expected to contain the unknown but true value of some characteristic of a present or past population with a given probability [29]. In our scenario, the confidence interval should contain the actual cut-off time. A larger confidence interval suggests that QualifAI has more uncertainty in its predictions, while a smaller confidence interval suggests confidence within the prediction. Understanding the dependability of the confidence intervals is crucial for real-time decision-making and if the model provides inaccurate confidence intervals then these decisions will be made incorrectly. This section will provide an evaluation of the produced standard deviations and thus confidence intervals. Confidence intervals are calculated as given in Equation 3.5.

As Deliver uses less, but more up-to-date, data in the training process, it is expected that Deliver should have smaller standard deviations than QualifAI. The standard deviations produced for QualifAI should still be useful however and not significantly larger (> 0.1) than

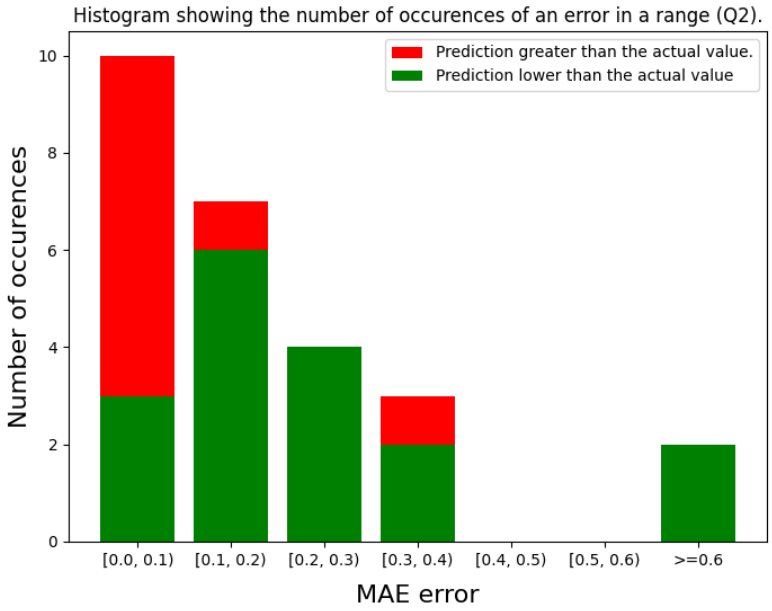

Figure 4.2: Histogram showing how many predictions fall within a certain distance from the actual cutoff time for Q2. Green bars represent predictions that are less than the actual cut-off time and red bars represent predictions that are greater than the actual cut-off time.

Deliver. Table 4.3 does not support this expectation fully.

Qualifying Session	Standard Deviation produced by Deliver	Standard Deviation produced by this project
Q1	0.323	0.657 ± 0.12
Q2	0.151	0.182 ± 0.08

Table 4.3: The mean standard deviations associated with each qualifying session for both the model used in this project and Deliver.

Table 4.3 shows that, for Q1, Deliver produced much tighter standard deviations. However, for Q2, QualifAI produces similar, but looser, standard deviations with a similar number of values that fall within 2 standard deviations of the actual value as Deliver. Confidence intervals for Deliver provide much more useful results but, as explained earlier, this is to be expected.

Figures 4.5 and 4.6 show that 85% of all cutoff predictions are within the produced confidence interval. This means that in the large majority of cases, the difference between the predicted value and actual value is correctly bracketed by the confidence interval. The confidence intervals are therefore reliable and can be used to aid in the decision-making process.

Figure 4.6 shows a decrease in the number of predictions that fall within 1 standard deviation of the actual value but also shows an increase in the number of predictions that fall within two standard deviations of the actual value. As the mean standard deviation of Q2 is much lower than that for Q1, this is unexpected and helps to reinforce the strength of the confidence intervals for the model.

4.1.5 Timing analysis

As explained in §2.5.2, for the project to be packaged and used by the client, it must adhere to strict timing constraints such that any information it produces can influence strategy decisions.

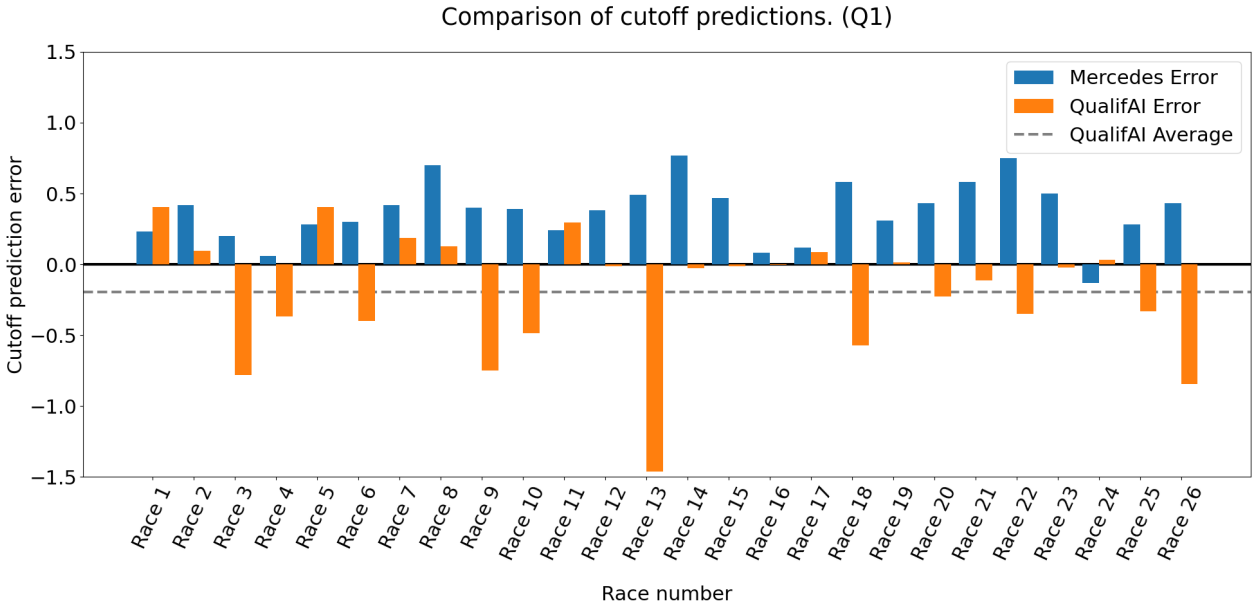

Figure 4.3: Comparison between the client's predicted cutoff times for races in the 2019 and 2020 seasons and the model's predicted cutoff times. Cutoff times are for Q1 only. Error bars are excluded for clarity. Races included follow the assumptions outlined in §3.3.1.

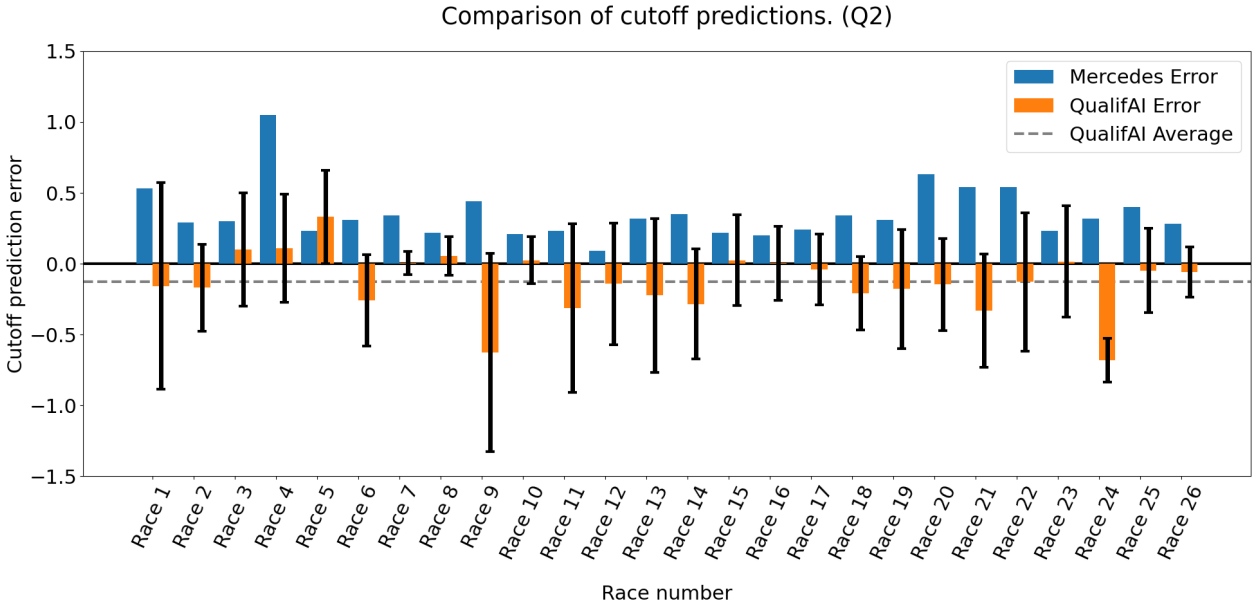

Figure 4.4: Comparison between the client's predicted cutoff times for races in the 2019 and 2020 seasons and the model's predicted cutoff times. Cutoff times are for Q2 only. Error bars are included to show the confidence intervals produced by the model. Races included follow the assumptions outlined in §3.3.1.

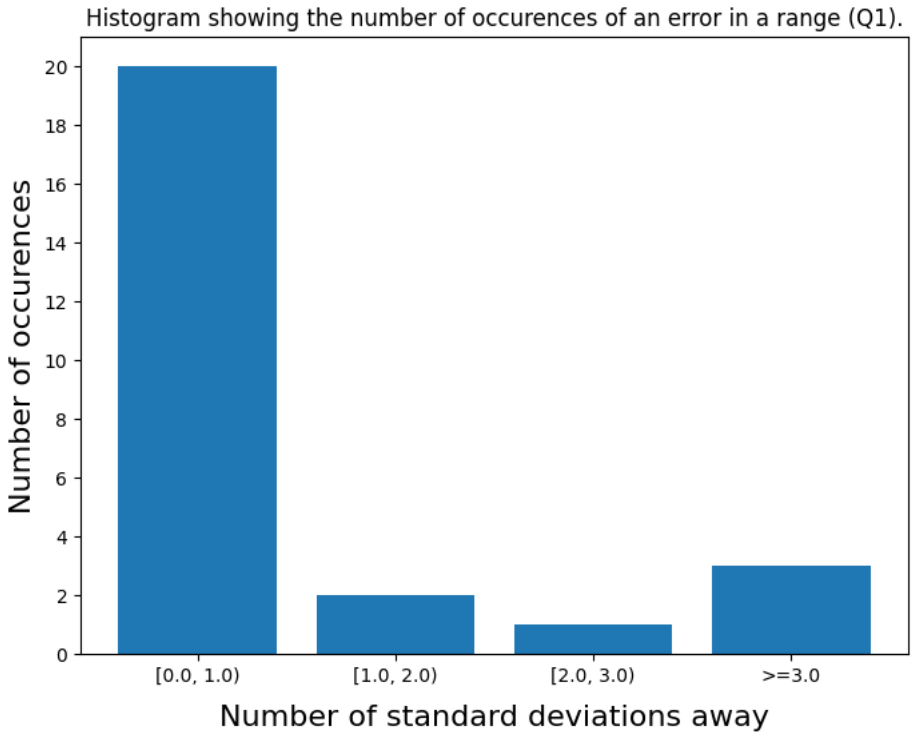

Figure 4.5: Histogram showing how many predictions fall within a given number of standard deviations from the actual value for Q1.

Table 4.1.5 shows the time required for the entire pipeline, from data extraction to prediction, to execute. Running the entire program takes a similar amount of time for both Q1 and Q2 with different sections of the pipeline being faster in different sessions.

Qualifying Session	Data Extraction (s)	Data Processing (s)	Model Training (s)	Model Sampling (s)	Total Time (s)
Q1	28.2	0.0810	10.2	1.75	12.1
Q2	28.2	0.0884	9.91	1.26	11.3

Table 4.4: Mean execution time in seconds for various pipelines within the project as well as the total execution time. These values were produced using algorithms 3 and 1 with fixed inputs ($N = 10, |\vec{v}| = 3, R = 1000$). Total time is a summation of a row without the Data Extraction. Data Extraction was not determined through my code but the access speed of the API, which couldn't be influenced.

The time difference between sessions during data extraction and data processing is insignificant and occurs due to an increase in the amount of available data (approximately 100 more laps are set in the time between extractions). This is the only section that is slower for Q2 than Q1. Model training has a time difference of approximately 0.3 seconds between Q1 and Q2 caused by the hyperparameter training requiring a model to be generated multiple times, This process takes longer in Q1 as more flying laps are set due to their being more drivers that can be on track. Model sampling is also slower during Q1 as, even though the same number of samples are tested, there are 20 drivers and thus 20000 samples compared to Q2's 15000. Altogether this produces a time difference of ≈ 0.8 seconds between the qualifying sessions.

While a 12s run time is a significant increase over previously used models, which had a run time of 1.59 seconds, it is still within the required 1-minute interval. This allows the program to be run multiple times and multiple cut-offs to be predicted and compared to other predictions.

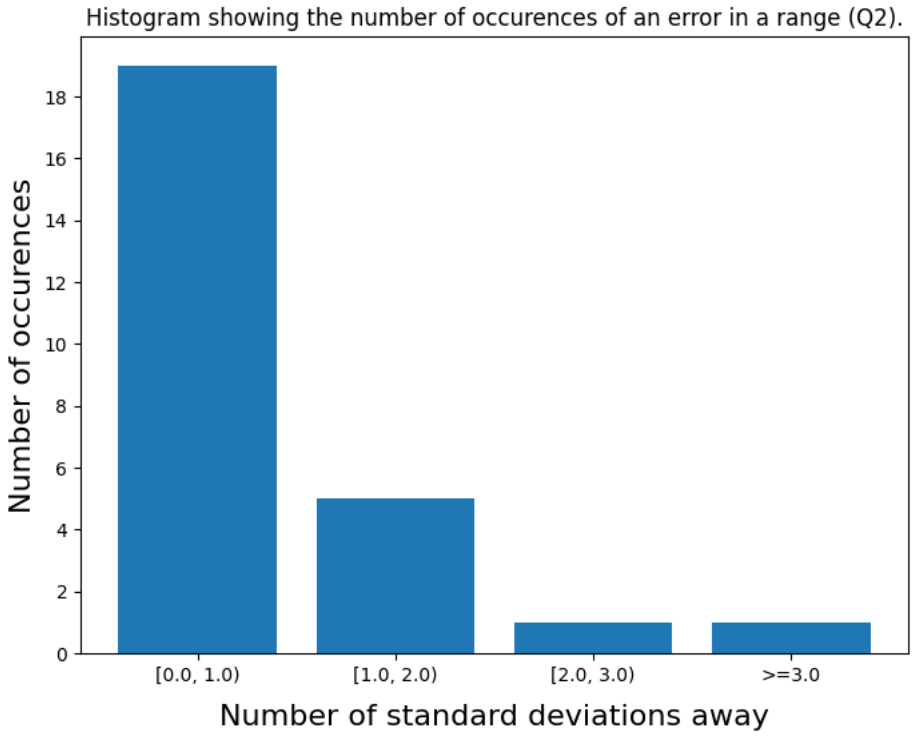

Figure 4.6: Histogram showing how many predictions fall within a given number of standard deviations from the actual value for Q2.

Data extraction needs to be run during the qualifying session; by running the extraction of the three practice sessions before qualifying, the time required to extract data during qualifying is significantly reduced.

4.2 Hyperparameters

The hyperparameter random search is important to ensure that the model's accuracy is as high as possible. We know there is an inverse relationship between the marginal log-likelihood (Equation 2.6) and the loss of the model. As a result, we can calculate the MSE of the model for a given set of hyperparameters in order to optimise the model. SciPy GPRs optimise hyperparameters by default using the L-BFGS-B algorithm [30] however a custom optimizer using random search (Algorithm 1) achieves much better results in an acceptable time.

This custom algorithm also helps to reduce the MSE loss of the kernels used within the model, which directly leads to an increase in the accuracy of the model. This is quantified in Table 4.5.

Qualifying Session	Random Search MSE loss	L-BFGS-B MSE loss
Q1	0.40814	6.25062
Q2	0.31248	7.30801

Table 4.5: MSE loss comparison on the test sets of the custom random search produced for the project and the built-in L-BFGS-B algorithm given by SciPy.

[1] `https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html`

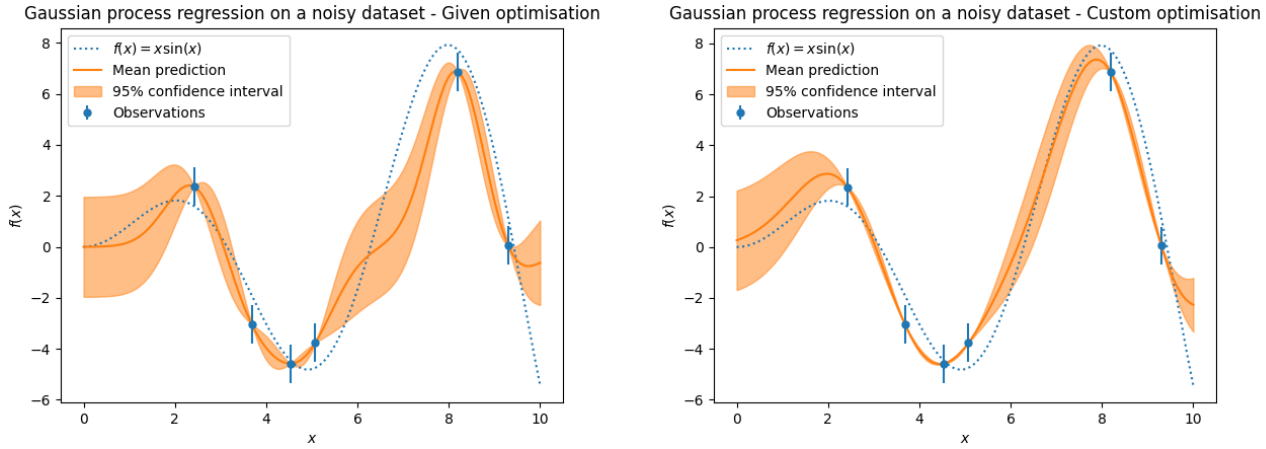

Figure 4.7: Effect of hyperparameter optimisation. The graph on the left uses SciPy's built-in optimiser while the graph on the right uses a custom hyperparameter optimiser using random search (algorithm 1). The hyperparameters on the left produce a loss of 1.461 and the hyperparameters on the right produce a loss of 0.396. This is based on the given example from SciPy's documentation[1].

Table 4.5 shows a significant decrease in loss for both Q1 and Q2 when compared to the default optimiser for Gaussian Processes in SciPy.

4.3 Accuracy of whether a driver should remain in the garage

An evaluation of the accuracy and usability of predicting whether a driver should remain in the garage rather than returning to the track to complete another lap. Results are calculated on 2019-2020 data.

4.3.1 Model Accuracy

F_1 is a widely used measure of a test that combines precision and recall [31]. In the given experiments, a positive prediction occurs when the model suggests the driver should remain in the garage. As stated before (§2.5.2), we also want the 'false positive' column of the table to be minimised, but not significantly above 1% in order to reduce loss.

The model returns a probability of proceeding into the next session given the driver does not return to track, by varying the required probability to accept the model's prediction we can alter the number of occasions in which the client sends out the driver again. It is expected that the number of false positives should always remain lower than false negatives, even with a lower limit (0.7). Furthermore, increasing the limit should always reduce the number of false positives and true positives as we produce stricter requirements to leave a driver in the garage. These expected results are presented and quantified in Table 4.6.

By using a limit of 0.99 we get an F_1-score of

$$F_1\text{-score} = 2 \times \frac{\text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} = 0.8091 \tag{4.2}$$

while this is not the maximal F_1-score, which occurs with a limit of 0.90 and has a value of 0.8497, it is the best score that also keeps the proportion of false positives not significantly

Limit	True Positive	True Negative	False Positive	False Negative
0.70	415	342	46	69
0.80	403	348	40	81
0.90	379	359	29	105
0.95	368	366	22	116
0.99	337	376	12	147
1.00	229	381	7	255

Table 4.6: Results of predicting whether a driver should remain in the garage given their current lap time. All numbers are produced using lap times from three minutes before the end of the session for the races in the 2019 and 2020 seasons.

above 0.01. This F_1-score suggests the model has high precision and recall, and is a good predictor of whether the car should remain in the garage.

A comparison between the predictions that the model produces and the choices that the client made shows that the model produces a direct improvement in choices made. This is quantified in Figure 4.8 which shows that the number of correct predictions for the model is greater than the number of correct predictions used by the client for the 2019-2020 races.

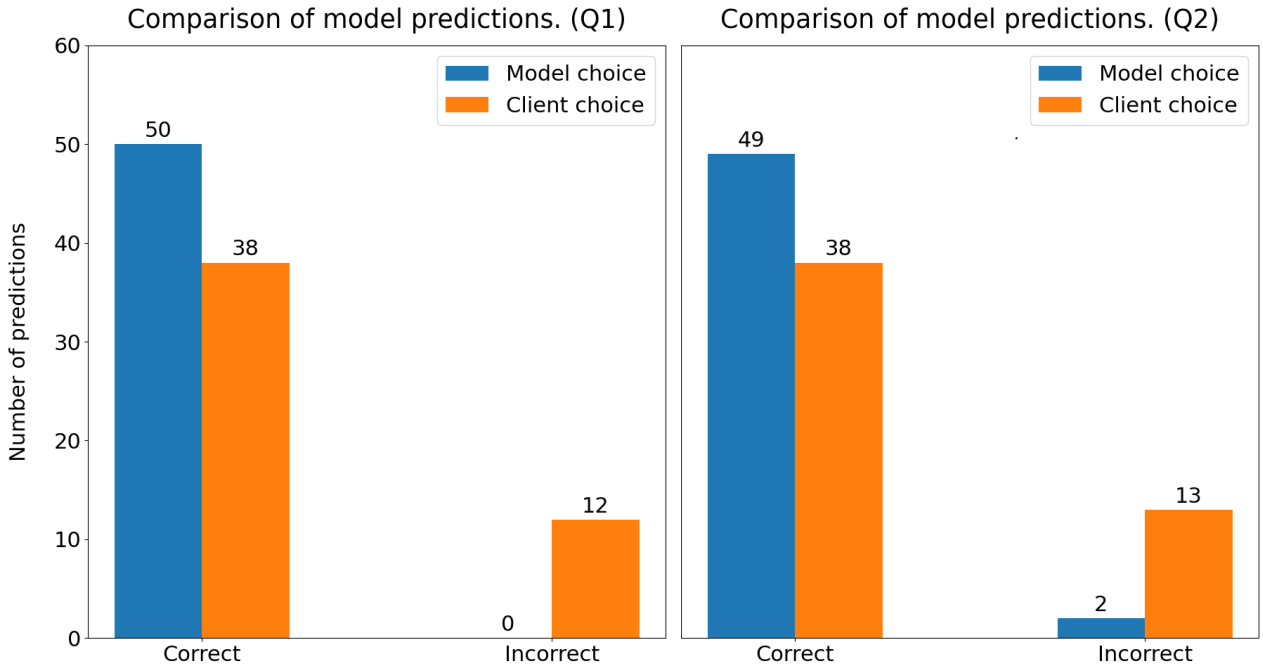

Figure 4.8: Comparison between the choices the client made and the choices that the model predicts for Q1 and Q2. Decisions are made for *Lewis Hamilton* and *Valtteri Bottas* as these are the drivers that the client employed in 2019 & 2020.

4.3.2 Execution Time

Predicting if a driver should remain in the garage has a faster execution time than predicting the cutoff time (table 4.1.5). While both are important parameters, this gives more information to the client and as such it is more important that this can be run many times.

Qualifying Session	Data Extraction (s)	Data Processing (s)	Model Training (s)	Model Sampling (s)	Total Time (s)
Q1	28.2	0.0810	10.2	0.903	11.2
Q2	28.2	0.0884	9.91	0.557	10.6

Table 4.7: Mean execution time in seconds for various pipelines within the project as well as the total execution time.

4.4 Overview of success

The project deliverables outlined in Chapter 2 (§2.5.3) were vital for guiding the progress of the dissertation and were all satsisfied by the end of the project. Moreover, all of the project's success criteria have been achieved.

Deliverable	Core or Extension	Status
Compute cutoff times with a confidence interval.	Core	Successful
Compute the probability a driver will progress.	Extension	Successful
Evaluate the model against previous methods.	Core	Successful
Package the project and deliver it to the client.	Extension	Successful

Table 4.8: Overview of which deliverables were core criterion or extensions and whether or not they were completed.

Chapter 5: Conclusions

5.1 Project Summary

The project goal was to produce a machine learning model to aid the client in producing qualifying cut-off predictions that could be used in real-time scenarios in order to aid their strategy-making and save resources. The main function of the project is to estimate the maximum time a driver could set and still progress to the next qualifying session. As well as this functionality, it can also recommend whether a driver should attempt to set another lap, or if it is safe for them to remain in the garage.

I successfully researched the techniques necessary to implement the pipeline for this project in Chapter 2 and performed the necessary evaluation in chapter 4. Furthermore, I implemented all of the necessary stages of the pipeline as explained in Chapter 3, including data extraction and processing (§3.2), model training (§3.3), and model predicting (§3.4) such that a cut-off prediction can be produced when using a curated digest of performance from previous races.

This project was a success; I managed to complete all of the core criteria and implement multiple extensions as well.

5.2 Conclusion

The main conclusion of this project is that, while it is possible to predict cut-off times, due to the time restrictions and small amounts of training data, producing a large improvement over currently used models is very difficult. However, due to the competitive nature of the sport, a small improvement is still valuable. More sophisticated methods, such as incremental training, may produce better results but I did not have time to research and implement these.

5.3 Lessons Learnt

The organisation and planning of the code base and experiments was an important lesson. Due to the nature of the degree, I was forced to take many breaks from my dissertation to focus on other work, resulting in having to relearn how the functions connected, in order to return to work. By organising my code more clearly and adding further comments I was able to reduce the time to relearn how the project worked. Furthermore, as many functions required many hours to be run, if there was an error that I would not notice until the end of the function, my plan would be delayed by a day. This made careful planning more important and taught me the importance of using smaller test sets and proper tests to ensure functions are working as expected.

I also learnt many of the differences between producing software in an industrial setting versus producing software in a personal setting due to my collaboration with Mercedes AMG Petronas F1, as well as learning the importance of understanding client needs and setting expectations. My view of software design also changed throughout the process of this dissertation; originally the pipeline was written for my specific needs and as such would be difficult for someone without prior knowledge of the program to use. This meant a lot of time was spent refactoring the code such that the client could use it as well; had I done this from the beginning more time could have been spent programming new functionality.

Bibliography

[1] Delia Draghici. Machine Learning for modelling Formula 1 races. Technical report, University of Cambridge, 2021.

[2] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, 1142:012012, 2018.

[3] Amazon Web Services (AMAZON). Formula 1 Case Study, 2018. Publication Title: Amazon Web Services, Inc.

[4] Giacomo Perantoni and David J.N. Limebeer. Optimal control for a Formula One car with variable parameters. *Vehicle System Dynamics*, 52(5):653–678, May 2014. Publisher: Taylor & Francis.

[5] S. Dhanvanth, Rohith Rajesh, S. S. Samyukth, and G. Jeyakumar. Machine Learning-Based Analytical and Predictive Study on Formula 1 and Its Safety. In Anuradha Tomar, Hasmat Malik, Pramod Kumar, and Atif Iqbal, editors, *Proceedings of 3rd International Conference on Machine Learning, Advances in Computing, Renewable Energy and Communication*, pages 257–266, Singapore, 2022. Springer Nature Singapore.

[6] M. Keertish Kumar and N. Preethi. Formula One Race Analysis Using Machine Learning. In Vinit Kumar Gunjan and Jacek M. Zurada, editors, *Proceedings of 3rd International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*, pages 533–540, Singapore, 2023. Springer Nature Singapore.

[7] Fédération Internationale de l'Automobile (FIA). 2023 FORMULA ONE SPORTING REGULATIONS, 2023.

[8] Thomas Muehlbauer. Relationship between Starting and Finishing Position in Formula One Car Races, 2017.

[9] International Business Machines (IBM). What is Machine Learning? | IBM.

[10] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[11] Manuel Blum and Martin A Riedmiller. Optimization of Gaussian process hyperparameters using Rprop. In *ESANN*, pages 339–344. Citeseer, 2013.

[12] James Bergstra and Yoshua Bengia. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[13] Robert L. Harrison. Introduction to Monte Carlo Simulation. *AIP Conference Proceedings*, 1204(1):17–21, 2010. _eprint: https://aip.scitation.org/doi/pdf/10.1063/1.3295638.

[14] J. P. Neirotti, David L. Freeman, and J. D. Doll. Approach to ergodicity in Monte Carlo simulations. *Physical Review E*, 62(5):7445–7461, November 2000. Publisher: American Physical Society (APS).

[15] Hans Janssen. Monte-Carlo based uncertainty analysis: Sampling efficiency and sampling convergence. *Reliability Engineering & System Safety*, 109:123–132, 2013.

[16] M Steven Palmquist, Mary Ann Lapham, Suzanne Miller, Timothy Chick, and Ipek Ozkaya. Parallel Worlds: Agile and Waterfall Differences and Similarities. Technical report, Software Engineering Institute, 2013.

[17] Xue Ying. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168:022022, 2019.

[18] R Dunford. *The Pareto Principle*. PhD thesis, Plymouth, 2014.

[19] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.

[20] John T. Hancock and Taghi M. Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):28, April 2020.

[21] Eduardo C. Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian Optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.

[22] Deval Shah, Zi Yu Xue, and Tor M. Aamodt. Label Encoding for Regression Networks, 2022. _eprint: 2212.01927.

[23] Xing Wan. Influence of feature scaling on convergence of gradient iterative algorithm. *Journal of Physics: Conference Series*, 1213(3):032021, June 2019. Publisher: IOP Publishing.

[24] Chris Cheadle, Yoon S Cho-Chung, Kevin G Becker, and Marquis P Vawter. Application of z-score transformation to Affymetrix data. *Applied bioinformatics*, 2(4):209–217, 2003.

[25] Andrew Bell, James Smith, Clive Sabel, and Kelvyn Jones. Formula for success: Multilevel modelling of Formula One Driver and Constructor performance, 1950-2014. *Journal of Quantitative Analysis in Sports*, 12:99–112, June 2016.

[26] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges*, pages 15–33. Springer Publishing Company, Incorporated, 1st edition, 2019.

[27] Qing Li and Yu-Liu Chen. Data Flow Diagram. In *Modeling and Analysis of Enterprise and Information Systems: From Requirements to Realization*, pages 85–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[28] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(1):79–82, 2005.

[29] Jonathan V Roth. Prediction interval analysis is underutilized and can be more helpful than just confidence interval analysis. *Journal of clinical monitoring and computing*, 23(3):181, 2009. Publisher: Springer Nature BV.

[30] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, December 1997.

[31] Cyril Goutte and Eric Gaussier. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In David E. Losada and Juan M. Fernández-Luna, editors, *Advances in Information Retrieval*, pages 345–359, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

Appendices

Chapter A: Kernels

The choice of kernel is very important in order to correctly model the function the GPR is trying to predict. As explained in §3.3.1.1 this was completed by choosing a selection of possible kernels and comparing their performance. Table A.1 shows the metrics that were compared and determines the best kernel choice for the model.

Kernel choice	Q1 Accuracy MAE	Q2 Accuracy MAE	Total Accuracy MAE
RBF() + RBF() + ConstantKernel()	0.14274	0.16277	0.15279
ConstantKernel() * RBF() + RBF()	0.23423	0.07225	0.23931
ConstantKernel() + RBF()	0.18607	0.16243	0.17425
ConstantKernel() * RBF()	0.22195	0.17762	0.19979
RBF() + RBF()	0.53144	0.30311	0.41727
RBF() * RBF()	0.58484	0.29323	0.43903
ConstantKernel() + ConstantKernel()	0.92017	0.89508	0.90762
ConstantKernel() * ConstantKernel()	1.10819	1.30240	1.20530
RBF()	0.54323	0.32855	0.43589
ConstantKernel()	1.25407	1.49697	1.37552

Table A.1: Kernel choices and the respective MAE for Q1, Q2, and both combined for a variety of possible kernel choices within the project. Green text shows the minimal entries for a given column, which present the best score achieved by all kernels.

Project Proposal

Modelling Formula One Qualifying using Machine Learning

Part II Project Proposal

Project Supervisors: Dr Neil Lawrence, Andrei Paleyes, Han-Bo Li

1 Introduction and Description

Formula One (F1) is the "world's most prestigious motor racing competition, as well as the world's most popular annual sporting series" [1]. The goal of the F1 is to produce the fastest car, that follows the FIA's [1] regulations, to get more points than your competitors (points are given based on your final race classification), the team with the most points at the end of a season gets the largest share of the prize money. F1 has 20+ events per year, with each event being split into 3 sections: Practice, Qualifying, and the race.

F1 has high amounts of constantly updating data. Every strategic decision a team makes must use this data efficiently, with a high degree of accuracy, to ensure that their drivers are at an advantage compared to the other 18. This scenario is clear during every qualifying session that drivers partake in.

Each race has a starting grid that is determined by the qualifying sessions. There are 3 qualifying sessions and in each session drivers are ranked by the fastest lap they are able to set in the given time frame (The length of the session.) The slowest 5 drivers are eliminated and unable to compete in the next session. It is important to predict the time required to not fall in the slowest 5 (Henceforth called "Qualifying cut-off times") so teams can decide whether it is worth having their driver do another lap. While it is always possible for a driver to do another lap, given they have enough time, this will also use more resources. A competitive lap completed during qualifying requires an unused set of tyres, as tyres that have been previously used will have less grip and thus produce a slower lap. The number of tyres a team has is limited and reducing the tyres used in qualifying gives greater strategy options in the race or latter qualifying sessions.

This can be seen as a symmetric 2-player game theory problem as all 10 teams have access to the same data and are able to make moves at the same time. The aim of this game is to maximise the probability of being faster than the qualifying cut-off time while minimising resource usage. The aim of this project is to predict these qualifying cut-off times and provide the likelihood of passing into the next qualifying session (being faster than the predicted qualifying cut-off time) without performing another lap for a given driver. As there are 3 qualifying sessions there are 2 separate cut-off times to predict in order to pass into the '2nd and 3rd session respectively.

My focus is on real-time data analysis such that Mercedes AMG Petronas F1 Team can model live qualifying sessions. The software produced must run fast enough to be used in real-time and be accurate enough to produce information that the team can confidently use. I am likely to use a Gaussian process as these are robust enough to work with limited amounts of data. Each driver will have a different model and each model will take a subset of possible data points as input, and return a likely time for that driver to set if they complete another lap.

I will have semi-regular meetings with 2 Mercedes team members for tech details, feedback, and project updates during development to ensure I remain on track and my software will be useful.

2 Starting Point

2.1 Data

Mercedes AMG Petronas F1 Team will supply their data for use within this project, it is a collection of large CSV files with a C# API. Once I have the data on my local device I will not require an internet access to reach the data.

There is a large amount of data contained within these CSV files, such as the timing of previous laps set by each driver, GPS location of each driver, weather conditions of the current session, and tyre usage. This

[1] The governing body of Formula 1.

data also has timestamps allocated to each entry, allowing us to simulate the data as if it were live when testing.

If I am unable to access the Mercedes data, I will use publicly accessible data. This is outlined in the Additional Resources.

2.2 Previous Work

This project will build on the work of a previous dissertation [2] but have a stronger focus on validation and metrics. There have been significant changes in the way qualifying is held since the previous dissertation and, as such, I will be using a different model as well as different data in order to improve the accuracy.

2.3 Tools

To produce and test models for the project I will use TensorFlow or PyTorch. Most of the work is likely to be completed in Jupyter notebooks that will be stored on GitHub for version control, with the work being ported to a .py files at the end for easy deployment such that Mercedes can use the model themselves.

The project will be written in LaTeX, and I will Zotero as a literature management tool. As the project should work on live data I will be using my personal laptop to develop, run, and test the model. Should my laptop fail I have funds put aside for another or will use department devices.

3 Additional Resources

Should I be unable to access the supplied API, there is also a large amount of public data online that I can use. While public data is likely to not be as accurate as the supplied data, it will provide a basis I can use to train and test the model. Some examples of public APIs include SportsMonks[2], Ergast[3], or FastF1[4]. While none of the alternatives will be as accurate as the Mercedes API, using them will allow me to progress with my project and produce a model that can predict qualifying cut-off times.

While using public data will allow me to progress with my project, it does so at a cost of accuracy to the supplied data and thus a cost to the final accuracy of my produced model.

4 Project Content

4.1 Content

The idea of this project is to give the reader a broad overview of the F1 Qualifying Format and how the data collected by the teams can be used to create strategic decisions that benefit them not only in that session, but future qualifying sessions and the race.

Firstly I will introduce the reader to the qualifying format, explaining the details of tyre degradation and other factors that result in a range of lap times from a single driver. This should show the reader the importance of both collecting the data (Which is performed by the Mercedes team) and using the data effectively.

Then I will explain the drawbacks of the previous system in the current era of F1 as well as some drawbacks caused by the previous choice of model. This section will explain the need for extending the previous project as well as give a brief overview of how this will be done. This section will be succint as the project should focus on my work, not previous work.

The next stage will include going through the API and selecting the data that will be used to train the model, this stage will also include writing any program required for managing that data and changing it to a more usable format. Afterwards, I will write about models found in literature that could be used within the project. I will focus on both model-specific and model-agnostic tools. The former will have a focus on how model behavior changes based on architecture, loss function etc. while the latter will focus on hyperparameters. At this stage I will also implement the chosen model with the chosen hyperparameters.

Then I will verify the model based on the testing data from the API and review the accuracy and, if necessary, update the model to improve the accuracy or use more data from the API. In order to evaluate

[2] https://www.sportmonks.com/formula-one-api/
[3] http://ergast.com/mrd/
[4] https://theoehrly.github.io/Fast-F1/index.html

the model I will consider the accuracy of the model and compare the model to the model currently used at Mercedes, as well as the model used for the previous dissertation, in order to evaluate if there has been an improvement through the new model.

At the end of the project I would like to summarize my findings, discussing the chosen model and why it was the best choice as well as giving possible extensions to the project.

4.2 Potential Structure

1. Introduction.
   - Project Motivation
   - Introduction to F1 and the Qualifying format
   - Related work
2. Evaluation of previous model.
   - Explain choices made by previous dissertation [2] .
   - Explain what has changed and how this impacts accuracy.
3. Data Selection and management.
   - Decide on which data should be used from the API.
   - Manage data so that the data used to train / test is data that the team would have at the time of qualifying.
   - Split data into a training and test set, separate into files so I can't accidentally use training data.
   - **Possible risk**: Lack of testing data.
4. Model selection and implementation.
   - Research various models and decide upon the model that will give the best accuracy.
   - Implement this model.
   - It is possible I may have to implement multiple models to compare them, this will link closely to section (5).
5. Model Verification and handover.
   - Test the model and compare to the previous dissertation [2] to ensure that accuracy has improved while remaining fast.
   - Refactor the code such that it can be packaged and given to the Mercedes Team.
6. Summary.

5 Success Criteria

5.1 Core Criteria

- *Extraction* - Produce a method to extract necessary data from the API, allowing for the separation by timestamp so that I can simulate as if the data was being added to the database in real-time.
- *Implementation* - Implement a model that predicts the qualifying cut-off time and provides a probability of a given driver being faster than that time.
- *Evaluation* - Evaluate the model based on previously used models, test the model on known results.

5.2 Possible Extensions

- Increase accuracy by implementing more data from the API.
- Consider the likelihood of other drivers setting another lap, and factor this into the probability of needing to send out your driver again.
- Wrap the trained models within a python API such that it can be easily imported by Mercedes when needed.

6 Suggested Timetable

Interval 0: Monday, 10th October – Sunday, 16th October

The goal of this interval is to ensure I have the required information, such as the API, as well as a valid project proposal that can get the

- Get the API from Mercedes AMG Petronas Formula One Team.
- Complete the final version of the Project Proposal.

Milestones:

- Project Proposal complete and submitted.

Deadlines:

- Final Proposal Submission – Friday 14th October 2022 (5pm).

Interval 1: Monday, 17th October – Sunday, 30th October

The goal of this interval is to setup my work environment and ensure I am at a position that I can efficiently start my project in the following weeks.

- Setup the work environment (Create a virtual environment, setup GitHub repository etc.)
- Decide on a project template.
- Familiarise myself with the supplied API, library, and how the data is used.

Interval 2: Monday, 31st October – Sunday, 13th November

The goal of this interval is to familiarise myself with the API, after this interval I should be comfortable with retrieving any necessary data.

- Explore the API and decide on initial data to be used in the model.
- Begin writing programs that can be used to clean the data and emulate live data collection via the associated timestamps.

Interval 3: Monday, 14th November – Sunday, 27th November

The goal of interval is to learn different Machine Learning models and decide upon the most accurate model as well as the data that will be used to train the model.

- Complete programs that can assist in data collection / cleaning.
- Rewatch relevant Machine Learning lectures.
- Explore various Machine Learning models and select the most appropriate model.

Interval 4: Monday, 28th November – Sunday, 11th December

The goal of this interval is to focus on training the Machine Learning Model.

- Focus on completing training.

Interval 5: Monday, 12th December – Sunday, 8th January

This is a longer interval with the goal of testing the Machine Learning Model, and giving extra time to amend the model if necessary.

- Test the machine learning model using a subset of the testing data.
- If necessary, update the model / the parameters of the model.
- Begin data visualisation.

Milestones:

- Model programmed and trained.

Interval 6: Monday, 9th January – Sunday, 22nd January

At this point I should be happy with the model and with the accuracy given on previous tests, this interval will involve testing on the most recent data (This years data) as well as cleaning up code.

- Test on up-to-date data, recording and visualising the results.
- Perform code refactoring and any necessary software testing.
- Begin a draft for the progress report and presentation.

Milestones:

- Model tested and results visualized / put into perspective of existing systems.

Interval 7: Monday, 23rd January – Sunday, 5th February

This interval focuses on the progress report and presentation.

- Submit the progress report.
- Finish the presentation.
- Start work on the Introduction and Preparation sections of the dissertation.

Milestones:

- Core success criteria completed.

Deadlines:

- Progress Report deadline – Friday 3rd February 2023 (12 noon).

Interval 8: Monday, 6th February – Sunday, 19th February

At this point, the progress report should be completed. The presentation will need to be presented as well.

- Present.
- Finish working on the Introduction and Preparation sections of the dissertation.
- Begin working on the Implementation section of the dissertation.
- Work on extensions and / or outstanding sections of Implementation and Evaluation.

Milestones:

- Introduction and Preparation section of dissertation complete.

Interval 9: Monday, 20th February – Sunday, 5th March

The goal of this interval is to complete any extensions or last minute changes to implementation. After this interval no more time will be given to coding.

- Complete the extension / outstanding section of Implementation and Evaluation.
- Complete the Implementation section of the dissertation.

Milestones:

- Implementation section of dissertation complete.

Interval 10: Monday, 6th March – Sunday, 19th March

The goal of this interval is to complete the first draft of the dissertation.

- Begin and complete the Evaluation and Conclusion sections of the dissertation.
- Send the first draft of the dissertation to my project supervisors and DoS for review.

Milestones:

- Evaluation and Conclusion section of the dissertation complete.
- First draft of the dissertation complete and sent for review.

Interval 11: Monday, 20th March – Sunday, 2nd April

The goal of this interval is to receive feedback and improve the dissertation based on the feedback.

- Update dissertation based on feedback received.
- Send the updated dissertation to my project supervisors and DoS for review.

Milestones:

- Updated dissertation completed and sent for review.

Interval 12: Monday, 3rd April – Sunday, 16th April

The goal of this interval is to receive feedback and improve the dissertation based on the feedback.

- Update dissertation based on feedback received.
- Send the updated dissertation to my project supervisors and DoS for review.

Milestones:

- Updated dissertation completed and sent for review.

Interval 13: Monday, 17th April – Sunday, 30th April

The goal of this interval is to receive feedback and improve the dissertation based on the feedback.

- Update dissertation based on feedback received.
- Send the updated dissertation to my project supervisors and DoS for review.

Milestones:

- Updated dissertation completed and sent for review.

Interval 14: Monday, 1st May – Friday, 12th May (Deadline)

By this point, the project should be completed. This interval serves a time buffer given I am unable to complete a previous interval in time and fall behind

Milestones:

- Dissertation completed!

Deadlines:

- Dissertation Deadline (Electronic) – Friday 12th May 2023 (12 noon).
- Source Code Deadline (electronic copies) – Friday 12th May 2023 (5pm).

.

References

[1] Formula One Corporation. *About F1.* URL: https://corp.formula1.com/about-f1/.

[2] Delia Draghici. *Machine Learning for modelling Formula 1 races.* URL: https://www.cl.cam.ac.uk/teaching/projects/archive/2021/dd525-dissertation.pdf.