Comparative Analysis of Machine Learning Methods for Predicting Formula 1 Sector Times

Samuel Timmins
Computer Science, B.S., M.S.
Lyle School of Engineering
stimmins@smu.edu
Dallas, TX United States

***Abstract*—Formula 1 is often referred to as the pinnacle of motorsport with fast, high-downforce, and extremely technical cars. In the world of motorsport, there is no class of car that is faster around a racetrack than a Formula 1 car where drivers will have to sustain up to 5 g under braking, cornering, and acceleration for over 300 km of race distance. Furthermore, with such extreme speed, strategy is incredibly important; the ability to predict both your own and your competitors' next lap times accurately can be crucial to ensure that you come out on top. With teams required to follow a strict set of development regulations imposed by the Fédération Internationale de l'Automobile (FIA), the lap times of competing cars are often incredibly close, making strategy decisions even more challenging. In 2022, the FIA introduced a new set of aerodynamic and engine regulations, which have drastically changed lap times. Using historical Formula 1 data from the FastF1 API going back to 2022, I compare four model architectures used for predicting the next sector time of any given driver using their previous three sector times. The four models are the RandomForestRegressor, XGBRegressor, a neural network, and a neural network with positional encodings for the sectors based on the lap number of the race.**

***Index Terms*—Neural Networks, Wide Neural Networks, Deep Neural Networks, RandomForest, XGBoost, Keras, Sci-kit Learn, Tensorflow, Formula 1, Motorsport, FastF1**

I. Introduction

Formula 1, known as the pinnacle of motorsport, features fast, high-downforce, and technically advanced cars that push the boundaries of speed and performance. These cars subject drivers to extreme forces, experiencing up to 5 G under intense braking, cornering, and acceleration throughout a race that spans over 300 kilometers. In such a high-speed and competitive environment, strategic decision-making plays a crucial role in determining success on the track. Accurate prediction of both a driver's and their competitors' next lap times becomes paramount to gain a competitive edge.

The sport's governing body, the Fédération Internationale de l'Automobile (FIA), imposes strict development regulations on teams, resulting in closely-matched lap times among competing cars and adding further complexity to strategizing effectively during races. Additionally, the performance of a car can change drastically from track to track, as one track may suit the strengths of the car whereas another punishes its weaknesses. For example, a car may have incredible performance through high-speed turns but struggle severely through low-speed turns; a track with lots of high-speed corners will allow this car to be faster compared to the competition, whereas a track with more low-speed corners would hurt the performance relative to the competition. Furthermore, because each team develops their own cars, this can lead to a wide range of performance with some teams being significantly faster throughout a whole lap as seen in figure 1.

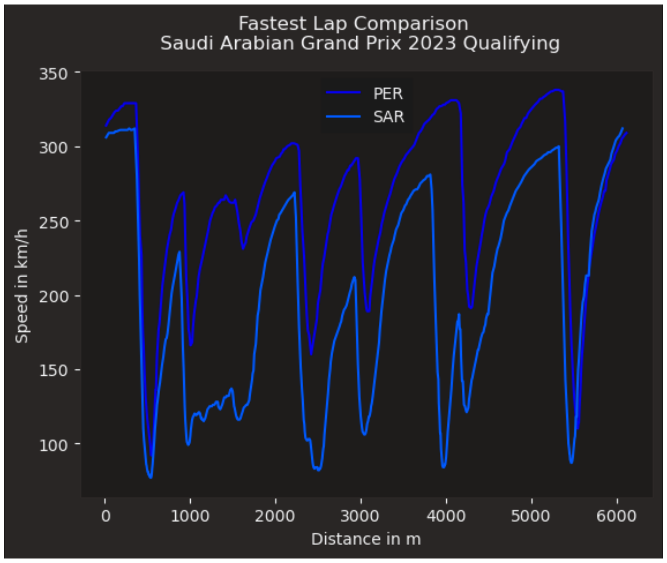

Fig. 1. Speed Comparison over Singular Lap of Fastest and Slowest Driver in Qualifying

Additionally, the type of tire and tire age play a major role in a car's pace through a lap. In Formula 1, there are six dry weather compounds available to a team throughout the season with three compounds being picked by the tire supplier for each race weekend [6]. These three compounds are then named "soft," "medium," and "hard" with the soft tire being the fastest, but shortest life and hard being the slowest yet longest life. Depending on the type of tire being used, the age of the tire, and the track temperature, the lap times will change drastically. This is important because in Formula 1, all drivers are required to use at least two different tire compounds in every race, and it is crucial for the strategy teams to ensure that the drivers are on the right compound at the right time.

This research aims to leverage historical Formula 1 data obtained from the FastF1 API, specifically focusing on the

2022 and 2023 seasons, when the FIA introduced a new set of aerodynamic and engine regulations that significantly impacted lap times. The data consists of all official laps driven by each driver in a race session– excluding laps completed under caution, a virtual safety car, or safety car– and is made up of driver and team information, basic telemetry data, weather data, tire data, and, crucially, the previous three sector times completed.

In Formula 1, all tracks are split into three distinct sectors that are used to measure lap times. This project aims to predict any given driver's next sector time utilizing the previous three to enable Formula 1 teams to make informed decisions, even with a limited amount of data. The study compares four distinct model architectures employed for predicting the next sector time of any given driver, using information from the previous three sector times. By analyzing and contrasting the performance of these models, this research aims to shed light on the effectiveness of traditional machine learning techniques, using RandomForestRegressor and XGBRegressor, in comparison to the capabilities of neural network architectures. Furthermore, the inclusion of positional encodings [5] and the use of cosine decay with warmup [4] as a learning rate scheduler in the neural network model introduces an innovative approach to exploit additional information for lap time predictions. Insights derived from this study have the potential to inform Formula 1 teams' strategies, enabling them to make data-driven decisions that optimize race performance and ultimately secure victory in this fiercely competitive motorsport class.

II. Methodology

A. Data Collection

The data collection process for this project was made incredibly straightforward thanks to the Python FastF1 API [1]. FastF1 is an easy-to-use Python API that holds a wide range of historical Formula 1 data, from driver information to weather information to telemetry data for individual laps. Because Formula 1 underwent a massive aerodynamic regulation overhaul for the 2022 season onward, I decided to only use data from the 2022 season and the current season to ensure the data was representative for the current generation. FastF1's API provides several methods for collecting the wide range of data in several distributed custom dataframes. A lot of the data available through the service was not required for this project and was not used. The final dataframe was created through the combination of the race session data and weather data while removing any laps that were under some form of caution flag, as predicting these cautions is out of scope for this project. Then, the previous three sectors and the next sector are captured to create a new row representing the data over the previous three sectors. The final dataframe consists of 21 feature columns and just over 33,000 entries.

B. Data Pre-Processing

First, all data for the race session is loaded using the FastF1 API. This data includes all lap information for each driver and team in one dataframe, as well as the weather data over the course of the session in a separate dataframe. All pit-in and pit-out laps are removed from the lap data, as well as any rows that contain any null information. Additionally, all laps that were completed under a yellow flag, virtual safety car, or safety car are removed using the track status data provided by Formula 1 to ensure that all laps were completed on a clear track. Then, because the sector and lap times are recorded in a datetime format representing the minutes and seconds, these are converted into seconds with three decimal points of precision, as Formula 1 lap times are recorded to the thousandth of a second. After this, the lap dataframe and the weather dataframe are merged using the session time feature data such that each lap has the weather data for that specific time in the session. After the two dataframes are merged, all unnecessary features are removed, leaving only the driver, team, lap time, basic telemetry, and weather data in the dataframe. Finally, all categorical data (i.e. tire compounds, driver names, teams, etc.) are label encoded to integer values for later use in the model.

With the final laps created, the dataframe was then updated such that, rather than each row being an entry, each row represented the three previous sector times of the driver. In other words, each entry would represent the previous three sectors completed by the driver in order, whether that order be sector 1 followed by 2 followed by 3 or sector 3 followed by sector 1 and 2 of the next lap. An additional column was added to hold the next sector time for the model's target. Then, depending on the model, all continuous features were normalized; however, for the RandomForestRegressor and XGBRegressor, all continuous features except for the sector times were normalized, as this provided a significant performance boost.

C. Modeling

The following models used the same input data, albeit modified slightly as described below to reach the best performance for each model. Each model was trained with mean squared error as its loss function.

1) RandomForestRegressor and XGBRegressor: Using scikit-learn's RandomForestRegressor, I trained an initial model with some base hyperparameters using the raw input data from the process established above. I then normalized the data in varying ways to explore how to extract the most performance from the base model before completing a grid search of hyperparameters to find the optimal model. I repeated these same steps using the XGBoost XGBRegressor to find the best-performing model.

2) Neural Network: Using Keras, I generated a neural network with the architecture shown in figure 2. This model consists of three inputs that allow for the use of categorical data, lap data, and weather data. The categorical data is first passed through embedding layers being concatenated with the output of the weather branch. This concatenated output is then concatenated again with the output of the lap data branch before passing through two final dense layers to produce the predicted sector time. This initial model was trained using

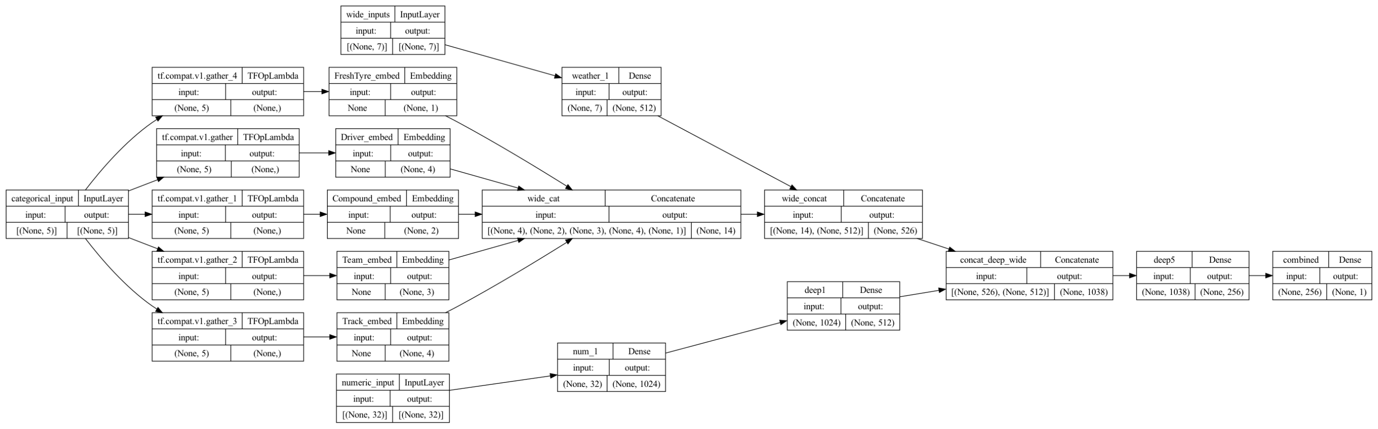
wide_inputs | InputLayer | input: | output: | [(None, 7)] | [(None, 7)]
tf.compat.v1.gather_4 | TFOpLambda | input: | output: | (None, 5) | (None,)
FreshTyre_embed | Embedding | input: | output: | None | (None, 1)
weather_1 | Dense | input: | output: | (None, 7) | (None, 512)
tf.compat.v1.gather | TFOpLambda | input: | output: | (None, 5) | (None,)
Driver_embed | Embedding | input: | output: | None | (None, 4)
categorical_input | InputLayer | input: | output: | [(None, 5)] | [(None, 5)]
tf.compat.v1.gather_1 | TFOpLambda | input: | output: | (None, 5) | (None,)
Compound_embed | Embedding | input: | output: | None | (None, 2)
wide_cat | Concatenate | input: | output: | [(None, 4), (None, 2), (None, 3), (None, 4), (None, 1)] | (None, 14)
wide_concat | Concatenate | input: | output: | [(None, 14), (None, 512)] | (None, 526)
concat_deep_wide | Concatenate | input: | output: | [(None, 526), (None, 512)] | (None, 1038)
deep5 | Dense | input: | output: | (None, 1038) | (None, 256)
combined | Dense | input: | output: | (None, 256) | (None, 1)
tf.compat.v1.gather_2 | TFOpLambda | input: | output: | (None, 5) | (None,)
Team_embed | Embedding | input: | output: | None | (None, 3)
deep1 | Dense | input: | output: | (None, 1024) | (None, 512)
tf.compat.v1.gather_3 | TFOpLambda | input: | output: | (None, 5) | (None,)
Track_embed | Embedding | input: | output: | None | (None, 4)
num_1 | Dense | input: | output: | (None, 32) | (None, 1024)
numeric_input | InputLayer | input: | output: | [(None, 32)] | [(None, 32)]

Fig. 2. Neural Network Model Architecture

the data as described above with all continuous features normalized using the Adam optimizer. Additionally, the model was trained again with positional encoding added to the input data and using a mix of the Adam optimizer as well as cosine decay with warmup as a learning rate scheduler with the positional encoding calculated per driver using the lap number of the race. The Adam optimizer was used for the first 100 epochs of training, and then the remaining epochs used cosine decay with warmup as a learning rate scheduler until the loss function converged.

The same positional encoding calculation from above was also used with the RandomForestRegressor and XGBRegressor to assess how the models performed with the additional information.

III. Results and Analysis

A. *RandomForestRegressor*

To begin modeling and set a baseline performance, I initially trained a RandomForestRegressor model using the scikit-learn package. This model used the default parameters with a max depth of 20 and was able to achieve an early mean squared error (MSE) of 0.115. While this was a great start for the initial model, I knew there was room for improvement and performed a grid search across several of the model's hyperparameters, with the best model found achieving an MSE of 0.149. This was using the data that had been normalized across every feature except for the previous three sector times and proved to be the best-performing model until the inclusion of positional encoding.

After seeing the success of positional encoding with the neural network, I added this information to the input data for the best-performing model and trained it again. The inclusion of the positional encoding data further improved the performance of the RandomForestRegressor model to an MSE of 0.124.

B. *XGBRegressor*

Using the same methodology as the RandomForestRegressor, I trained an XGBRegressor model using the XGBoost library. The initial model using the default hyperparameters with 1000 estimators and a learning rate of 0.05 resulted in an MSE of 0.109. I then performed a similar grid search across several hyperparameters of the model to tune the performance. This resulted in an MSE of 0.159, providing only a slight improvement over the initial model.

As with the RandomForestRegressor, I then retrained the best model with the inclusion of the positional encoding data and was able to achieve an MSE of 0.042, the best-performing combination.

C. *Neural Network*

When developing the architecture for the neural network, I initially started with a simple multilayer perceptron (MLP) network with a single input, two hidden layers, and a single output. This model performed poorly in comparison to the RandomForestRegressor and XGBRegressor models with an MSE of 0.212. Through some trial and error, I was able to achieve an improved MSE of 0.347 with the architecture shown in figure 2. The performance for this model was tuned with a grid search to be trained with Adam as the optimizer with a learning rate of 0.001 and using the ReduceLROnPlateu learning rate scheduler for 100 epochs. However, this model was still drastically worse than the initial RandomForestRegressor and XGBRegressor models that had been established as the baseline models.

D. *Neural Network with Positional Encoding and Cosine Decay with Warmup Learning Rate Scheduler*

I was incredibly surprised to see the neural network performing poorly in comparison to the more traditional machine learning techniques, so I decided to expand the data further to provide additional context to the neural network. I provided this context by calculating the positional encoding of each lap per driver in the dataframe and storing this encoding with the input data. I then treated this positional encoding data as continuous numerical data in terms of the model architecture seen in figure 2. The inclusion of positional encoding provided a performance boost to achieve an MSE of 0.117.

To try and further improve the performance of the model, I used the cosine decay with warmup learning rate scheduler from [4] to fine-tune the model's learning. Using this scheduler alone provided lackluster results compared to the previous model with an MSE of 0.299. However, the best neural network results were achieved when using the standard Adam optimizer with a fixed learning rate for the first 100 epochs, then completing the training with the cosine decay with warmup scheduler until the model converged. This combination resulted in an MSE of 0.117 on the test data.

E. Comparison of Methods

As shown in table I, the best performing model was the XGBRegressor using the positional encoding data with an MSE of 0.042 and a mean absolute error (MAE) of 0.048. This is a fantastic result with the average sector time prediction being within 3 hundredths of a second of the actual completed time. The next best model was the neural network using positional encoding with an MSE and MAE of 0.117 and 0.137 respectively. I found it very surprising that the neural network was not the highest-performing model. I also found it surprising that the addition of positional encoding to the input data resulted in such a large performance increase compared to other methods, specifically the RandomForestRegressor. With the RandomForestRegressor being the best-performing model without positional encoding, I would have thought the inclusion would improve all models equally. However, both the other methods improved significantly more than the RandomForestRegressor leaving it as the worst model tested.

Overall, it is clear that the usage of positional encoding is crucial to high performance in this field, as it provides important context to the model on the progress through the race. Perhaps this is because refueling is banned in Formula 1, and as fuel burns off, the pace of cars changes drastically as they become lighter; providing the model context of the drivers' progress through the race can help it better estimate the small changes in pace between sectors.

F. Future Work

I would like to continue expanding this project to further explore how precise the predictions can become. I believe the inclusion of granular telemetry data over the previous three sectors provided through the FastF1 API could add further context to the model about the car's current condition.

Another extension of this project would be to extend beyond the next sector but to try and predict the next several sectors and laps to provide broader information to Formula 1 team strategists.

Finally, this project was completed using publicly available telemetry and race data, which is incredibly limited. Modern Formula 1 cars have hundreds of sensors on them, and I believe that a more precise prediction model could be generated using all the data available. This data, however, is kept incredibly close to the Formula 1 teams as it could easily be used by other teams to gain a competitive advantage.

TABLE I
MODEL COMPARISON

Model	MSE	MAE
RandomForestRegressor	0.149	0.215
RandomForestRegressor with Positional Encoding	0.124	0.120
XGBRegressor	0.159	0.221
XGBRegressor with Positional Encoding	**0.042**	**0.048**
Neural Network	0.347	0.325
Neural Network with Positional Encoding	0.397	0.468
Neural Network with Positional Encoding and Cosine Decay with Warmup Learning Rate Scheduler	0.117	0.137

IV. CONCLUSION

This research focused on the prediction of Formula 1 drivers' next sector times using historical race data and machine learning techniques. The study aimed to enable teams to make data-driven decisions and optimize race performance in the highly competitive motorsport class. The analysis involved comparing the performance of traditional machine learning methods, including RandomForestRegressor and XGBRegressor, with the capabilities of a neural network architecture.

The results demonstrated the effectiveness of machine learning models, particularly when combined with positional encoding, in predicting sector times. The XGBRegressor with positional encoding emerged as the best-performing model, achieving an impressive mean squared error (MSE) of 0.042 and a mean absolute error (MAE) of 0.048. The neural network with positional encoding and a cosine decay with warmup learning rate scheduler also performed well, with an MSE of 0.117 and an MAE of 0.137. These models' predictions were highly accurate, providing valuable insights into drivers' performance during a race.

Looking ahead, future work could focus on incorporating more granular telemetry data from modern Formula 1 cars, which could further enhance the precision of the predictions. Additionally, extending the models to predict multiple future sectors and laps could provide broader information for race strategists to make informed decisions.

REFERENCES

[1] P. Schaefer, "FastF1." https://github.com/theOehrly/Fast-F1.

[2] F. Pedregosa et al., "Scikit-learn: machine learning in Python," Oct. 2011, doi: 10.5555/1953048.2078195.

[3] T. Chen and C. Guestrin, "XGBoost," Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[4] D. Landup, "Learning Rate Warmup with Cosine Decay in Keras/TensorFlow," 2022, [Online]. Available: https://stackabuse.com/learning-rate-warmup-with-cosine-decay-in-keras-and-tensorflow/.

[5] M. Saeed, "A Gentle Introduction to Positional Encoding in Transformer Models, Part 1," 2022. https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/.

[6] M. Seymour, "F1 tyres explained: The beginner's guide to Formula 1 tyres," Apr. 09, 2023.