Interpretable Formula 1 Qualifying Predictions

reddit.com/u/TheHefi

1. INTRODUCTION

In this paper, we present four different models for predicting *Formula 1 (F1)* qualifying results based on historical data and data gathered during practice sessions. We compare their accuracy as well as their interpretability to the average F1 fan, based on how well broadcasters can incorporate the model's explanation in their live feed. We also draw comparisons to the model developed by *Amazon Web Services (AWS)* that is currently used to create qualifying pace prediction graphics shown during F1 broadcasts.

Data Science has gained an increasing importance in professional sports. Teams and athletes try to gather more and more data for the purpose of result prediction, performance assessments, and strategy evaluations [7, 3, 6]. Additionally, data is often used in order to make accurate predictions about future events. In motorsport, this allows teams to make informed strategic decisions when it comes to, for example, pit stop scheduling or energy management during races [9, 8, 4].

Predicting sports results is very popular among many fans nowadays and there is a lot of research focusing on the prediction of sport results [5]. Notable examples include the prediction of basketball [10] or football [11] match results. However, datascience in sports is not an entirely academic field and there are many practical applications for them, for example during sports broadcasts. Since the 2020 F1 season, AWS sponsor a variety of prediction graphics during F1 broadcasts [1]. Most notably a qualifying pace predictions during practice sessions, where for each driver, AWS give an estimate on the driver's delta to the ultimate pole sitters lap time. Fans often criticise these graphics for being not interpretable, as during broadcasts there are no explanations given on where the predictions are derived from. Additionally, predictions made by AWS are generally not considered to be particularly reliable, which further increases the frustration of the fans.

A normal Formula 1 weekends consists of three free practice sessions (FP1-FP3), a qualifying session and a race. FP1 is mostly used for tuning the car's setup and testing new components. FP2 and FP3 are more revealing regarding a driver's qualifying pace. During these sessions, teams usually conduct race and qualifying simulations in order to prepare for the respective sessions. Therefore, FP2 and FP3 are the most insightful sessions when it comes to predicting qualifying results. Nevertheless, predicting qualifying results is not trivial. Even ignoring unpredictable events like changeable weather conditions and driver errors, deriving qualifying pace from historical- and practice data is not easy for two main reasons. Firstly, during practice sessions, teams tend to hide their true pace in order to maintain a strategic advantage from their competitors. Teams use higher fuel levels or lower engine modes in order to artificially slow their cars down, even on simulation runs. This method is commonly referred to as *sandbagging*. Secondly, track conditions change throughout the weekend. The track improves as the cars run on it due to the cleaning effect and the rubber that is deposited on the surface, which increases the grip level.

1.1 AWS Model [13]

AWS use a supervised learning approach. The model learns from historical data in terms of how much each team improves between the practice sessions and the qualifying. This is achieved using two modelling methods. The first type of model considers each car's improvement based on historical data, as well as the team's fuel and power adjustments as a variance around this median value. The second type of model uses a Regression technique where the difference in lap times compared to the past is used and a prediction for the future is derived from the current result, using the teams, drivers, track and weather conditions as variables. There is no public record of how exactly the two types of models are implemented. Therefore, we cannot make any statement about the interpretability of their approach. However, even if their models were interpretable, during broadcasts the necessary explanations are not presented to the viewer. Thus, the frustration of the fans.

Therefore, the goal of this paper is to build a model that matches the accuracy of the AWS model, while being explainable to the average F1 fan. Possibly within a single, simple graphic which can be presented during broadcasts. Or to build a model which exceeds AWS's accuracy, without considering interpretability.

The remainder of this paper is organized as follows: Section 2 contains the methods of our study, including a description of the design, materials, and procedure. In Section 3 we present our results by comparing the accuracy and interpretability of four of our models. Finally, Section 4 has concluding remarks.

2. METHODS

In order to gain access to the data of past Formula 1 seasons, we use the Python library *FastF1* [2], which is designed around *Pandas* and *Numpy*. FastF1 only provides data for the 2018-2021 seasons, which limits the amount of data available to us. After preprocessing, the data of the 2018-2020 seasons are used to train different models to predict the qualifying results of the 2021 season. For the training and the evaluation of the models, we only use the data of Grand Prixs for which all the practice sessions and the qualifying sessions were dry. We set these restrictions because wet sessions are extremely rare, so there is hardly any data to train models to adapt to them. For example, only 6 of the previous 81 qualifying sessions between 2018 and 2021 were not held under completely dry conditions. Additionally, rain is not always equally heavy. This means that cars that happen to be on the track during a period of less heavy rain have a distinct advantage over cars that complete their qualifying lap during the heaviest rain period. This makes wet qualifying sessions almost completely unpredictable. After removing all instances of wet sessions, the final dataset contains 1300 instances. After training, the models are then compared to each other and the AWS model. As there is no official compilations of all AWS qualifying predictions of the 2021 season, we manually tracked down screenshots of the broadcasts they were presented in. We validated their authenticity by cross-referencing the date they were posted on the internet with the F1 race calendar. This approach yielded qualifying predictions for 20 of the 22 Grand Prixs, which amounts to predictions for 398 instances in total. When we intersect the set of instances for which we have the predictions of the AWS model with the biggest set of instances one of our models is able to make predictions for, we are left with a total of 333 test

instances.

Some of our predictions are more informed than those of the AWS model. For example, for some Grand Prixs we were only able to find AWS predictions made directly after FP2. Which means that their prediction does not take FP3 results into account, while our models do. However, AWS clearly have the confidence that their model does not require data from both sessions in order to make accurate predictions. Otherwise, they would not have made them so early during the weekend. For our comparisons, we consider the absolute errors of the predictions. We use the following metrics: the R^2-Score, the median, the mean and the standard deviation. The R^2-Score or *coefficient of determination* measures how well the Regression predictions approximate the real data points and gives an indication of how accurately test instances not yet considered can be predicted. An R^2-Score of 1 indicates that the Regression predictions perfectly fit the data. The median is an interesting metric, as it is robust to outliers. Thus, it can help us identify models that hit most predictions perfectly and are only sometimes completely off the mark. Finally, we consider the mean and standard deviation, as they measure the expected accuracy of a prediction and the dispersion around it.

The Feature Sets we use to train the models contain 8 different kinds of variables.

- Team, Driver and Track
- FP2/FP3_time: fastest time posted by the driver in FP2/FP3
- FP2/FP3_tyre_life: number of laps completed on the set of tires with which FP2/FP3_time was set, before FP2/FP3_time was set with it
- T_fastest: theoretical fastest lap time of a driver across all practice sessions of the weekend. Every circuit is divided into 3 disjunctive sectors. For each of these sectors, T_fastest considers the quickest one a driver did across all his laps during free practice sessions.

As FP1 is mostly used for testing and tuning the car's setup, the fastest time of a driver during this session usually does not grant any insight into their qualifying pace. In contrast, the fastest lap time of a driver during FP2 or FP3 contain a lot more information about their true pace. Complementary to FP2/FP3_time, we can also include FP2/FP3_tyre_life into our models. Tyres offer less grip the more they are used. Thus, a quick lap time on an old set of tyres indicates that a driver can go even faster, when using a brand-new set of tyres during qualifying. T_fastest allows us to gain insights into the qualifying pace of a driver, which are not derivable by FP2/FP3_time. E.g., if during a lap, a driver is very quick in the first sector, but makes a mistake in the final sector, so that the overall lap time is slow, their fastest lap of the session does not reflect their potential pace in the first sector. Finally, we consider the team, the driver themselves and the track. Clearly, all three variables have a significant impact when it comes to qualifying pace predictions. From these features, we derive 12 different Feature Sets. Team, driver and track are encoded using *One-Hot Encoding*, as their values are completely independent of each other. One-Hot Encoding is a common way of preprocessing categorical features for machine learning models, where for each possible category, a new binary feature is created. Before testing and training, the data is scaled using the *scikit-learn StandardScaler*. From the set of all variables, we define 12 subsets, which are created based on intuitive sense. We refer to these subsets as *Feature Sets 1-12*.

On each Feature Set, we train models based on four different kinds of Regressions: *Linear Regression (LR)*, *Quadratic Linear Regression (QR)*, *Cubic Linear Regression (CR)* and *Multilayer Perceptron Regression (MLPR)*. Linear Regression is the simplest of the models and assumes a linear relationship between the input variables and the output variable. However, Linear Regression cannot account for polynomial correlations of higher degree or connections between different variables. For that, we need Quadratic- or Cubic Linear Regression, which allow for this up to a degree of 2 or 3 respectively. We generate these polynomial and interaction features using the *PolynomialFeatures* function offered by scikit-learn. For example, for Quadratic Linear Regression, this function turns the Feature Set $[a, b]$ into $[1, a, b, a^2, ab, b^2]$. Note that with growing degree, polynomial Regression is more keen to overfitting and models get increasingly difficult to interpret. Finally, we use Multilayer Perceptron Regression, which is the least interpretable of the models. Generally, MLPRs can learn any function approximator, which makes them a powerful tool. However, the final model works as a black box and cannot simply be interpreted. Additionally, finding a good MLPR model can be difficult, as there are many *hyperparameters* that can be tuned. As training MLPRs with every possible combinations of hyperparameters is too computationally expensive, we utilize a *randomized search*. We define a set of different values for the size and number of the *hidden layers*, the *alpha value*, the *maximum number of iterations* and the *tolerance for the optimization*. For all of our MLPR models, we use the *solver* "lbfgs" for *weight optimization*, as our dataset is relatively small. Additionally, for the *activation function for the hidden layers*, we use "identity" as it proved to be most effective during our initial tests. For each Feature Set, we train 5 MLPRs with randomly assigned hyperparameters from our previously defined search space. We use *2-fold cross-validation* in order to avoid overfitting. To refit estimators with the best found parameters on the whole dataset, we use the R^2-score. More details to the mentioned methods can be found in the documentation of scikit-learn [12].

Of the 48 trained models, we want to discuss four in more detail:

- The Linear Regression model, that is based on Feature Set 1, which only contains T_fastest. We refer to this model as *LR1*.
- The Quadratic Linear Regression model, that is based on Feature Set 8, which contains T_fastest, FP2- and FP3_time, FP2- and FP3_tyre_life and the team. We refer to this model as *QR8*.
- The Multilayer Perceptron Regression model, which is based on Feature Set 12, which contains T_fastest, FP2- and FP3_time, FP2- and FP3_tyre_life and the track. We refer to this model as *MLPR12*.
- The Linear Regression model, that is based on Feature Set 12. We refer to this model as *LR12*.

Depending on what Feature Set a model uses, some instances from our dataset cannot be used to train or test them. Clearly, this makes models requiring viewer variables not only more understandable, but also more practical. In order to get a fair comparison between models, we only compare accuracies based on predictions for the same instances. This ensures that models do not gain an advantage by only predicting easy to predict instances. Although Feature Sets 8 and 12 contain different features, the set of instances available for the training and testing of models is identical. As QR8, MLPR12 and LR12 require data from FP2 and FP3, instances of our dataset where there is no data from these sessions cannot be used for training or testing. Among these instances, there are 60 instances from the 2021 British, Italian and Brazilian Grand Prix. These three Grand Prixs are so-called *Sprint Weekends*, where the qualifying takes place directly after FP1. Therefore, we cannot take the results of FP2 and FP3 into account for our predictions. In order to get fair comparisons between models, we define two different instance sets.

- *Instance Set A*: Contains all instances of our dataset, which are 966 training and 333 test instances.
- *Instance Set B*: Only includes instances which contain the required data for QR8, MLPR12 and LR12 to make predictions. Among them are 906 training and 246 test instances.

3. RESULTS

We begin by considering the simplest of all trained models: LR1, which was trained and tested using Instance Set A. Since this model

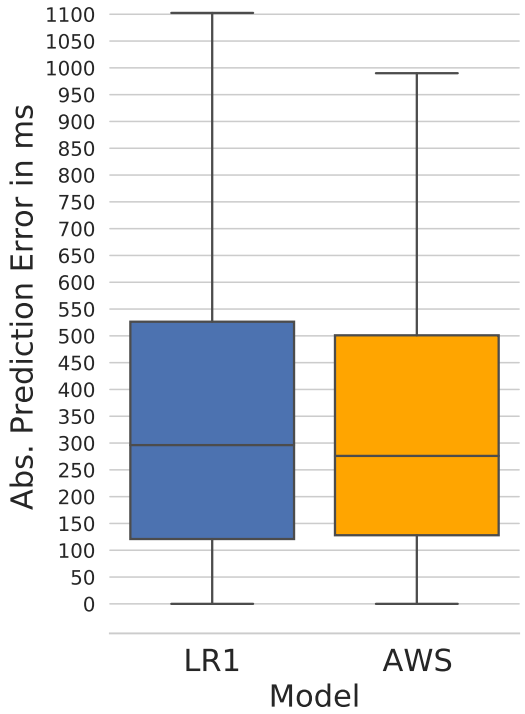

Figure 1: Boxplots visualising Q_1, Q_2 and Q_3 of the absolute prediction errors of LR1 and the AWS model on the test instances of Instance Set A. The whiskers extend to the lowest (highest) data point still within 1.5 IQR of the lower (upper) quartile. For clarity, outliers have been omitted from the figure.

only requires a single feature, which can be determined for every instance in the dataset, this is the model with the most training instances available to it. After training on 966 instances from the 2018-2020 seasons, LR1 predicted 333 test instances of the 2021 season. The trained model achieves an R^2-Score of 0.6336, a median absolute error of 296.16ms and a mean absolute error of 387.27ms with a standard deviation of 392.68ms. In comparison, the AWS model achieves an R^2-Score of 0.7031, a median absolute error of 276ms and a mean absolute error of 355.31ms with a standard deviation of 346.69ms, on the same test instances.

Boxplots comparing the two models can be seen in Figure 1. Table 1 contains a summary of the most important key figures. Clearly, LR1 is not as accurate as the AWS model. However, the differences are only marginal, with an R^2-Score only 0.0695 below that of the AWS model. Additionally, the median absolute error only is 20.16ms higher, while the mean absolute error and standard deviation only lie 31.96ms and 27.99ms above the respective values of the AWS model. This is surprising, as LR1 only takes the theoretical fastest lap time and multiplies it by ca. 0.9871. Nevertheless, to achieve such competitive accuracy suggests an extremely strong correlation between the theoretical fastest lap and a driver's actual qualifying pace. This interpretation is underlined by the relatively high R^2-Score of LR1. The simplicity of LR1 makes it very easy to explain to the average F1 fan, and therefore would make it possible for broadcasters to teach the interpretation of the model to their viewers within the bounds of a single graphic. In its current state, the AWS model is not significantly more accurate. Therefore, if the AWS model were to be replaced by LR1, many viewers would not notice the lower accuracy, but would most likely welcome the interpretability of the model.

Next, we discuss the model with the highest R^2-Score and lowest mean absolute error of our tests: QR8. The idea behind QR8 is that Quadratic Linear Regression allows us to exploit the connection

	AWS	LR1	QR8	MLPR12	LR12
Instance Set A					
R^2	0.7031	0.6336	-	-	-
Q_1	128.0	120.94	-	-	-
$Q_2=\tilde{x}$	276.0	296.16	-	-	-
Q_3	501.0	526.4	-	-	-
$\bar{x}$	355.31	387.27	-	-	-
σ	346.69	392.68	-	-	-
Instance Set B					
R^2	0.7027	0.6833	0.7287	0.7139	0.7139
Q_1	131.0	144.23	136.08	123.78	123.94
$Q_2=\tilde{x}$	274.0	310.53	293.83	274.61	274.75
Q_3	515.75	518.69	492.5	532.88	532.69
$\bar{x}$	364.9	380.49	355.92	363.21	363.25
σ	319.78	336.13	296.9	307.56	307.59

Table 1: Comparison of the accuracy of the AWS model, LR1, QR8, MLPR12 and LR12 for Instance Sets A and B. Metrics are the R^2-Score, the lower percentile Q_1, the median $\tilde{x}$, the upper percentile Q_3, the mean $\bar{x}$ and the standard deviation σ of the absolute error of the predictions in ms.

between the FP2- and FP3_time variables and the respective tyre_life variables, as well as to discover non-linear correlations. Adding the team, as a feature, is supposed to approximate the procedure of the AWS model by allowing QR8 to learn how much specific teams sandbag during practice sessions. QR8 was trained and tested using Feature Set B, containing 906 training- and 246 test instances. The trained model achieves an R^2-Score of 0.7287, a median absolute error of 293.83ms and a mean absolute error of 355.93ms with a standard deviation of 296.9ms. In comparison, the AWS model achieves an R^2-Score of 0.7027, a median absolute error of 274ms and a mean absolute error of 364.90ms with a standard deviation of 319.78ms, on the same test instances.

On these test instances, QR8 achieves a slightly higher R^2-Score, by 0.026 and a lower mean absolute error and standard deviation by 9.98ms and 22.88ms respectively. Only when it comes to the median absolute error, the AWS model achieves the slightly better result of the two models by 19.83ms. Overall, the two models can be considered to be equally accurate. These figures suggest that QR8 has fewer and less severe outliers than the AWS model. This assumption can be substantiated by considering Figure 2. The AWS model produces one very strong outlier, which likely is the reason for the higher mean and standard deviation of the model's absolute prediction errors.

In comparison to LR1, QR8 is a lot more complex and requires a lot more explanation in order to make it interpretable to the average F1 fan. However, QR8 could be made interpretable to F1 fans by giving a brief overview of the *feature importances*, as seen in Figure 3. Even people with little or no knowledge in machine learning can get an intuitive understanding of which metrics affect predictions and how much. Surprisingly, QR8 does not seem to profit from the connection between the FP2- and FP3_time variables and the respective tyre_life variables, as their absolute feature importances are relatively low. Considering the decent accuracy of LR1, a model which solely considers T_fastest, it is not surprising, that the three most important features for QR8 all contain T_fastest. Interestingly, the T_fastest·Ferrari feature is among the 10 features with the biggest absolute importance. This suggests that the theoretical fastest lap time is a much stronger indicator of the final qualifying pace for Ferrari than for other teams.

Finally, we consider the model with the lowest median absolute error: MLPR12. The idea behind MLPR12 is to find a model that is

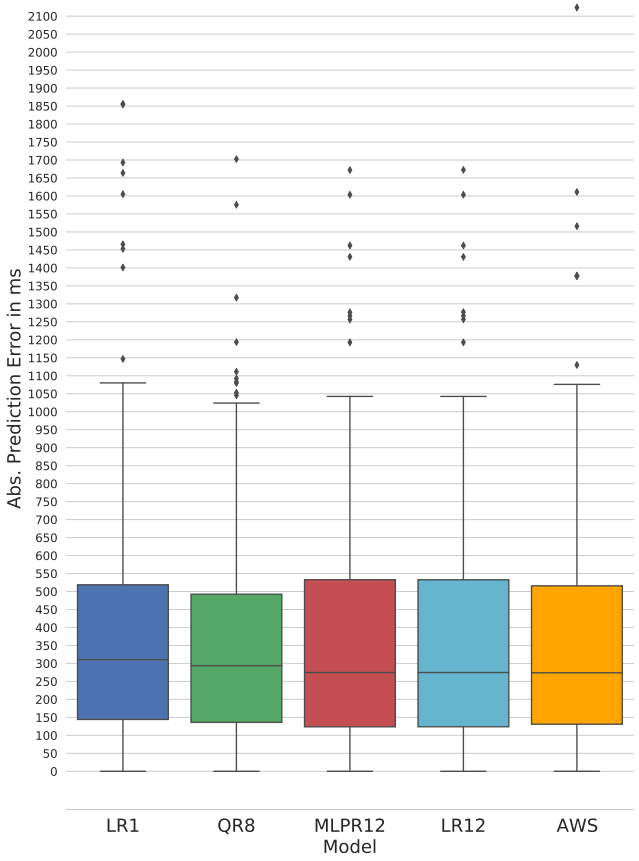

Figure 2: Boxplots visualising Q_1, Q_2 and Q_3 of the absolute prediction errors of LR1, QR8, MLPR12, LR12 and the AWS model on the test instances of Instance Set B. The whiskers extend to the lowest (highest) data point still within 1.5 IQR of the lower (upper) quartile.

not necessarily interpretable, but as accurate as possible. MLPR12 was trained and tested using Instance Set B, the same as QR8. Therefore, the performances of the two models are comparable. MLPR12 achieves an R^2-Score of 0.7139, a median absolute error of 274.61ms and a mean absolute error of 363.21ms with a standard deviation of 307.56ms. Although we do not require interpretability, MLPR12 cannot exceed QR8's accuracy. MLPR12's R^2-Score is 0.0148 below that of QR8. Additionally, the mean absolute error and standard deviation lie 7.31ms and 13.71ms over that of QR8. Only the median absolute error of MLPR12 is 19.08ms lower than that of QR8. This is surprising, as one would think that not being restricted to quadratic functions would allow for a much more accurate model. One reason for that might be, that we did not sufficiently search the hyperparameter space. Another reason might become clear, when considering the accuracy of the linear Regression model based on Feature Set 12. LR12 achieves an R^2-Score of 0.7139, a median absolute error of 274.75ms and a mean absolute error of 363.25ms with a standard deviation of 307.59ms. As LR12 almost matches the key figures of all considered metrics of MLPR12, it is likely, that MLPR12 simply approximates a near linear function and might only achieve the better scores based on rounding errors. This assumption can be underlined by considering the boxplots of the respective models in Figure 2, as they are almost identical. MLPR12 and LR12 are the only models capable of consistently matching the accuracy of the AWS model across all of our metrics. MLPR12's R^2-Score is 0.0112 above that of the AWS model. Additionally, the mean absolute error and standard deviation lie 1.673ms and 12.24ms below that of the AWS model. Only the median absolute error of MLPR12 is 0.745ms higher than that of the AWS model. Similar to QR8, LR12 could also

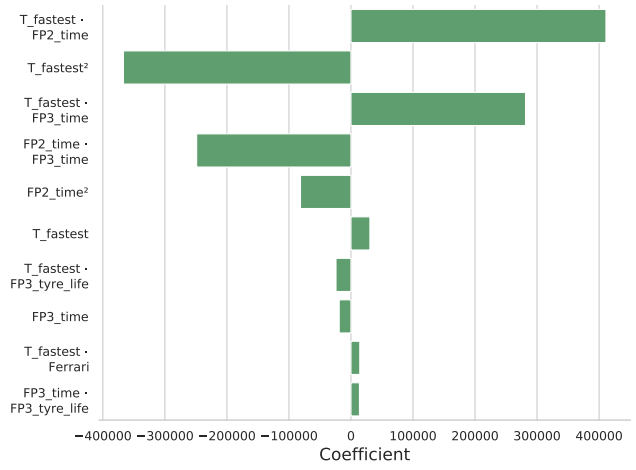

Figure 3: Coefficients of the 10 most important features for QR8.

be explained to to viewers during broadcasts by presenting a graphic giving an overview of the feature importances of the model. As there are a lot fewer features in LR12 that there are in QR8, they are even easier to interpret for LR12.

Another interesting comparison can be made to LR1. When tested on the test instances of Instance Set B, LR1 improves it's R^2-Score to 0.6833. The mean absolute error and the standard deviation drop to 380.49ms and 336.13ms respectively, while surprisingly the median absolute error increases to 310.53ms. The increasing median and decreasing mean could be due to the fact that the difference between the test instances in Instance Set A and B contains many instances that LR1 predicted very accurately and only relatively few instances where the error of the model is very large. This overall improvement in accuracy indicates that predicting the test instances from Instance Set B could generally be slightly easier than predicting those of Set A. This makes sense, when considering that Set A contains the 60 Sprint Weekend instances, while B does not. In these Sprint Weekend instances, the only data one can use to predict qualifying results stem from FP1. On these weekends, teams do not have as many free practice sessions before qualifying as on a usual race weekend. Therefore, they might not do any qualifying simulation runs in order to have more time to tune their car's setup before going into qualifying. Interestingly, the AWS model does not perform as well on the test instances of Instance Set B as on those of Instance set A. It's R^2-Score drops by 0.0004 to 0.7027, and it's mean absolute error increases by 9.59ms to 364.9ms. Only its median absolute error and the standard deviation drop by 2ms and 26.91ms, respectively. This might be due to the fact that the AWS model is a lot more generalized than our models and does not have specific weaknesses. With regard to the 60 Sprint Weekend instances, our guess is that the AWS model can better handle a lack of data for a particular instance because it has seen a larger amount of training data overall.

Generally, Cubic Linear Regression models performed rather poorly. The Cubic Linear Regression model with the highest R^2-Score is the one based on Feature Set 2 (CR2), which only contains FP2- and FP3_time. With a score of 0.6407, it only slightly outperforms LR1. One reason for this might be that the dimension of Cubic Regression models is too big compared to our available dataset. This phenomenon is also known as the *curse of dimensionality*. A higher number of features theoretically allow more information to be incorporated in the model, but practically it rarely helps due to the higher possibility of noise and redundancy in the real-world data. Other models worth mentioning are the Linear Regression models based on Feature Sets 7, 8, 9 and 10 (LR7-LR10). Feature Set 7 is almost identical to Feature set 8, with the only difference being that it includes the driver instead of the team. Feature Set 9 corresponds to Feature Set 7 and additionally contains the track, while Feature Set 10 corresponds to Feature Set 8 and also additionally contains the

	LR7	LR8	LR9	LR10	QR2
Instance Set B					
R^2	0.7188	0.7062	<-2.7e+25	0.7029	0.6375
$Q_2=\tilde{x}$	283.83	293.98	> 5.4e+15	303.3	284.52
$\bar{x}$	362.409	371.77	> 4.1e+15	373.12	395.21
σ	302.19	307.31	> 2.3e+15	309.82	402.46

Table 2: Comparison of the accuracy of LR7-LR10 and QR2. Metrics are the R^2-Score, the lower percentile Q_1, the median $\tilde{x}$, the upper percentile Q_3, the mean $\bar{x}$ and the standard deviation σ of the absolute error of the predictions in ms.

track. Both LR7 and LR8 pretty much match the accuracy of LR12. This is interesting because the Feature Sets only differ in whether they contain the driver, the team or the track. Thus, the three features all seem to have a similar importance when it comes to predicting qualifying pace. Interestingly, LR10 also has a pretty high accuracy, while LR9 is the model with by far the worst accuracy of all models trained within the context of this paper. This also might relate to the curse of dimensionality. While there are only 10 different teams, there are 32 different drivers and 31 different tracks in our dataset. That means that after One-Hot Encoding the features, LR9 has to deal with 68 features. While, LR10 only has to handle 46. A compilation of the key figures of the aforementioned models can be seen in Table 2.

4. CONCLUSION

In this paper, we tried to find an easy-to-understand, interpretable model capable of accurately predicting Formula 1 qualifying results. We presented four notable models and evaluated their accuracy and interpretability to the average F1 fan, based on how well broadcasters can incorporate the model's explanation into their live feed. Furthermore, we compared the models with each other and the current model used to present qualifying predictions during the broadcast of F1 practice sessions by AWS.

With LR1, we proposed an extremely simple model whose interpretation can be easily communicated to viewers. The model's accuracy is not very far off that of the current AWS model and therefore, could be a suitable replacement if interpretability to the fans was the priority of broadcasters. After LR1, the most interpretable models we presented were LR12 and QR8. Both of the models offer a compromise between interpretability and accuracy. When tested on the 246 test instances of Instance Set B, both models match the accuracy of the AWS model. For LR12 and QR8, it is possible to give the viewer a rough idea of how the predictions of the two models can be interpreted by showing a brief overview of their respective feature importances during broadcasts. Viewers might not be able to get an in depth understanding of the models, but it should be sufficient to give them a basic understanding of why the models make the predictions they do. The least interpretable model, we presented in this paper, is MLPR12. The idea behind this Multilayer Perceptron Regression model was to ignore interpretability and exclusively focus on accuracy. However, this did not work, as MLPR12 endet up approximating the same linear function LR12 describes.

Overall, our tests show that it is possible to make accurate predictions about F1 qualifying results, while retaining a certain level of interpretability. With LR12 and QR8, we trained two interpretable models, which match the overall accuracy of the currently used AWS model. However, there are a few aspects to consider when evaluating the performances of the models we trained. For training and testing of our models, we made rather restrictive assumptions about the weather conditions during practice and qualifying sessions. This was necessary, as we do not have access to sufficiently large datasets to properly train models to cope with wet sessions. Although we only made comparisons between the models accuracies based on predictions for the same instances, it is clear that the AWS model was at a disadvantage as it could not assume, that every practice and qualifying session is dry. Furthermore, it is clear that the models we trained in the scope of this paper are very likely to perform significantly worse, when making predictions about wet qualifying sessions. Finally, it is worth to consider that training models to predict events that already took place is generally easier than doing it beforehand.

For future work, it would be interesting to train and test similarly interpretable models on larger datasets, while not making restrictive assumptions about weather conditions. This might require to augment the models presented in this paper with additional features which allow them to account for the strength of rain, etc.

As a final thought, we would like to remark, that it is very likely that AWS is not interested in providing information about the interpretation of their qualifying prediction model at all. For many viewers, machine learning is a mysterious tool that can magically predict the future and is invincible in chess. AWS ultimately try to promote and sell a product. And explaining to the consumer that this highly complex machine learning model is really based on a simple Regression that many people could train themselves would be counterproductive.

5. REFERENCES

[1] F1 Insights powered by AWS. https://aws.amazon.com/de/f1/. Accessed: 2022-02-26.

[2] Fast f1. https://theoehrly.github.io/Fast-F1/index.html. Accessed: 2022-02-26.

[3] BANERJEE, A., ET AL. Motorsport data acquisition system and live telemetry using FPGA based CAN controller. In *Journal of Physics: Conference Series* (2022), vol. 2161, IOP Publishing, p. 012041.

[4] BEKKER, J., AND LOTZ, W. Planning formula one race strategies using discrete-event simulation. *Journal of the Operational Research Society 60*, 7 (2009), 952–961.

[5] BUNKER, R. P., AND THABTAH, F. A machine learning framework for sport result prediction. *Applied computing and informatics 15*, 1 (2019), 27–33.

[6] FONTANELLA, A., ET AL. High speed wireless optical system for motorsport data loggers. *Electronics 8*, 8 (2019), 873.

[7] HAGHIGHAT, M., ET AL. A review of data mining techniques for result prediction in sports. *Advances in Computer Science: an International Journal 2*, 5 (2013), 7–12.

[8] HEILMEIER, A., THOMASER, A., GRAF, M., AND BETZ, J. Virtual strategy engineer: Using artificial neural networks for making race strategy decisions in circuit motorsport. *Applied Sciences 10*, 21 (2020), 7805.

[9] LIU, X., AND FOTOUHI, A. Formula-e race strategy development using artificial neural networks and monte carlo tree search. *Neural Computing and Applications 32*, 18 (2020), 15191–15207.

[10] MILJKOVIĆ, D., ET AL. The use of data mining for basketball matches outcomes prediction. In *IEEE 8th international symposium on intelligent systems and informatics* (2010), IEEE, pp. 309–312.

[11] MIN, B., KIM, J., CHOE, C., AND IAN, R. A compound framework for sports prediction: The case study of football. *Knowledge-Based Systems 21*, 7 (2008), 551–562.

[12] PEDREGOSA, F., ET AL. Scikit-learn: Machine learning in Python. 2825–2830. https://scikit-learn.org/stable/, Accessed: 2022-02-26.

[13] SMEDLEY, R. Qualifying Pace. https://aws.amazon.com/f1/qualifying-pace/. Accessed: 2022-02-26.