Mining the Podium

Predicting Formula 1 Race Results

Taylor Clark
Department of Computer and Information Science
Fordham University
Manhattan, New York, USA
tclark38@fordham.edu

***Abstract*—Formula 1, also known as F1, is a popular category of motorsports in which open-wheel, single-seater formula cars are raced. The races span internationally, and the sport generates billions of dollars annually. Many factors contribute to the outcome of a race, such as the composition of the tires, pitstop strategy, the car mechanics, and just plain chance. Each year, the F1 teams work tirelessly to engineer the most efficient car possible, and they can only hope their hard work will pay off once the race season starts, as modifications allowed during the season are extremely limited. Given how many factors can influence the outcome of a race, this study demonstrates that it is possible to predict with relative certainty if a driver will finish a given race on the coveted podium by implementing different data mining algorithms such as decision trees, random forest, and regression.**

Keywords—Formula 1, Decision Trees, Random Forest, Regression, Classification.

I. INTRODUCTION

Sports predictions are something that have captivated fans for decades. For example, the fantasy football industry generates over 18 billion dollars each year. Whether it be high-stakes horse betting in Las Vegas or a friendly challenge between friends, attempting to predict the outcome of a sporting event adds an extra level of excitement. Whereas humans, which cannot be modified, are the focus of most sports, Formula 1 focuses on the cars. Each year as technology and engineering advance, the boundaries of these machines are pushed more and more. While it would be interesting to analyze how car specifications such as engine type and tire composition impact the outcome of races, that data is kept highly confidential so as to not make years of research and development readily available to opposing teams. There is still much data to analyze though that is available to the public, which is what this paper accomplishes.

Being able to accurately predict sporting results is not trivial, as a large part of the excitement surrounding sports is due to chance. A race where you know who is going to win is not very exciting, but a race where an underdog comes out on top at the last second is. That is why sports betting is such a large industry as well; someone who has no knowledge of the sport at all still has a chance, no matter how small, against someone who has been watching and keeping track of stats for years. These unexpected moments are so exhilarating, however, because they do not occur most of the time. A majority of the time, stats and research do hold up, and these surprise outcomes are simply the occasional outlier. It is therefore worthwhile to attempt to learn a classification model which can predict Formula 1 race results, as it should be reasonably accurate for the majority of cases.

Data mining and data analysis are becoming increasingly important in the sports world, for both entertainment value and performance optimisation. For example, Amazon Web Services has a large department of *F1 Insights*, in which they analyze data in regards to race strategy, competitor analysis, and car performance. One insight, *Alternative Strategy*, will, "provide fans and broadcasters with an alternative view of how drivers and their teams' decisions impacted a race: by analyzing how their races may have panned out had they made different strategic decisions." [1]. This is interesting both to the F1 teams who may implement these alternative strategies in the future, and to the fans as they root for their favorite teams. In this example, race outcome is being predicted using past race data, which is similar to what this study accomplishes as well.

The goal of the study described in this paper is to determine the method which will most accurately predict the finishing position of a given driver in a given race. The dataset used includes Formula 1 race data from 1950, when F1 was first established, into 2023. Both classification algorithms, such as decision trees and random forest, as well as regression were applied, in addition to different data preprocessing methods such as binning and binarization.

II. BACKGROUND

Formula 1 spans worldwide. The number of races varies depending on the season, but the average is around 21. There are 10 Formula 1 constructors, which are the teams that compete, and each team is responsible for developing their own cars. Each team has two drivers that compete in each race. The day before each race, there are three qualifying rounds to determine the drivers' positions on the starting grid of the race. The number of laps in a race depends on the length of the circuit, but it ranges from about 40-80 laps. Drivers win points depending on which place they finish in, with only the top 10 places receiving points. Each race has a podium ceremony where the drivers finishing in first, second, and third place are awarded, as well as the first place constructor. At the end of the season, the driver with the most points wins the world championship, and the constructor with the most points wins the constructors championship.

III. EXPERIMENT METHODOLOGY

This section describes the dataset used, the steps for data preprocessing, the classification and regression algorithms implemented, and the evaluation metrics used to assess performance.

A. Data Collection and Preprocessing

The dataset used in this study, *Formula 1 World Championship (1950-2023)* [2] was downloaded from Kaggle, and contains data from Formula 1 races spanning from 1950 to

2023. Data from two CSV files, results.csv and qualifying,csv, were needed, so the files were first read in, and then were cleaned before merging. The 'results' file originally included the features shown below in figure 1-a, and the features in figure 1-b are the features that were selected. The features 'resultId', 'number', 'positionText', 'positionOrder', and 'statusId' were removed because they were irrelevant for predicting race outcome. The features 'points', 'time', 'milliseconds', 'fastestLap', 'fastestLapTime', and 'fastestLapSpeed' were removed because they biased the dataset; the values of these features would not be known until a race is over, so therefore they could not be used to predict the outcome of a race.

	Original Features
0	resultId
1	raceId
2	driverId
3	constructorId
4	number
5	grid
6	position
7	positionText
8	positionOrder
9	points
10	laps
11	time
12	milliseconds
13	fastestLap
14	rank
15	fastestLapTime
16	fastestLapSpeed
17	statusId

Figure 1-a

	Selected Features
0	raceId
1	driverId
2	constructorId
3	grid
4	position
5	laps
6	rank

Figure 1-b

The 'qualifying' file was processed similarly. It originally contained the features shown in figure 2-a, and the features in figure 2-b are the features that were selected. The features 'qualifyId' and 'number' were removed because they were irrelevant features, and the features 'q1', 'q2', and 'q3', which represent the qualifying race times, were removed because their string representation of 'hours:minutes.seconds' was not compatible for analysis or for being simply transformed to an integer or float data type. Then, the 'position' feature was renamed to 'qualiPosition' to clarify that it represents the final position in the qualifying race (also the starting position of the main race), not the final race position, which is the feature that will be predicted.

	Original Features
0	qualifyId
1	raceId
2	driverId
3	constructorId
4	number
5	position
6	q1
7	q2
8	q3

Figure 2-a

	Selected Features
0	raceId
1	driverId
2	constructorId
3	qualiPosition

Figure 2-b

After cleaning the data, the two data frames were merged together on the features 'raceId', 'driverId', and 'constructorId'. The 'position' feature was renamed 'position_race' to clarify that it represents the final race position, and that column was moved to the furthest right for clarity since that is the feature which will be predicted. In Formula 1, it is possible that a driver may not finish a race, such as if there are mechanical difficulties with the car. The dataset represented a DNF (did not finish) with '\N', so to make this value comparable with the rest of the integers in the position_race class, all the values of '\N' were changed to be -1. The dataset also included some position_race values of 21-24, but since F1 races typically have only 20 drivers, those instances were removed. Lastly, for consistency and simplification, all the values in the dataframe were converted to integer values. The resulting data frame is shown below in figure 3.

	raceId	driverId	constructorId	grid	laps	rank	qualiPosition	position_race
0	18	1	1	1	58	2	1	1
1	18	2	2	5	58	3	5	2
2	18	3	3	7	58	5	7	3
3	18	4	4	11	58	7	12	4
4	18	5	1	3	58	1	3	5
...	...	...	...	...	...	...	...	...
9800	1110	817	213	19	44	15	19	16
9801	1110	858	3	18	44	9	18	17
9802	1110	807	210	0	44	4	20	18
9803	1110	832	6	4	23	19	5	-1
9804	1110	857	1	5	0	0	6	-1

9754 rows × 8 columns

Figure 3 - Finalized Data Frame

B. Building the Prediction Models

- Model 1: Decision Tree

The data was first split into X, the attributes in the first seven columns, and y, the class to be predicted, which was position_race. The test_train_split method was imported from Scikit Learn [3] and was used to split X and y into a training set comprising 80% of the data and a test set comprising 20% of the data. The DecisionTreeClassifier method was also imported from Scikit Learn and a decision tree was fitted to the training set. Then the test set of X was used with the resulting tree to make a prediction, which was compared to the test set of y for evaluation.

- Model 2: Random Forest

The RandomForestClassifier method was first imported from Scikit Learn, then its parameters were set. The first parameter is the criterion on which to evaluate the split point, which in this case was set to be gini. Gini is a measure of impurity, therefore the split point with the lowest gini index will be chosen by the model to split on. The second parameter is the max depth of each tree, which was set to eight. This means that each decision tree in this random forest model has a maximum of eight levels. The third parameter is the maximum number of samples required for a split, which was set to 10, so if a node had less than 10 instances, it could no longer be split. Both max_depth and min_samples_split help avoid model

overfitting. The last parameter is random_state, which was set to 5, and is used as a seed for the random number generator that is used to sample the data and features. The same X and y training sets from the decision tree model were used to fit the random forest classifier, and then the test set of X was used to test the resulting model, which was compared to the test set of y for evaluation.

- Model 3: Regression

For Regression, the Python library numpy [4] had to be imported, as well as the pyplot method from the matplotlib library [5] and the LinearRegression method from the linear_model method of the Scikit Learn library. X was set to be all the values in all the features except the position_race feature, and y was set to be all the values in the position_race feature. test_train_split was used to split X and y into a training set comprising 70% of the data, and a test set comprising 30% of the data, and the random_state parameter was set to zero. A linear regression model was fit to the training sets of X and y, and the test set of X was used to test the resulting model. These predictions were float values, so they had to be converted to integers using numpy.rint() to match the integer class values.

- Model 4: Decision Tree with Binning

Since the original decision tree was trying to predict a class value out of 21 possible values, it was unlikely that the accuracy would be very high. For example, the probability of randomly guessing the class value correctly is only about 5%, whereas the probability of randomly guessing correctly given only two possible class values is 50%. Therefore, for multiclass problems it can be beneficial to implement a strategy such as binning. Binning, also known as discretization or bucketing, is a data preprocessing strategy in which the original data values are divided into small intervals referred to as bins, and then the class value for each instance is replaced by a general number calculated for that bin. Since this problem had 21 possible class values, they were divided into 5 bins; the first bin comprised values 1-5, the second bin values 6-10, the third bin values 11-15, the fourth bin values 16-21, and the fifth bin the value -1, which signified a DNF. DNF was placed in its own bin because there can be multiple DNFs in one race, whereas each position value can only be used once per race, so there are more instances with a position_race of -1 than of the other values. After this preprocessing, the decision tree model was learned as before. The data was split into X and y, then X and y were split into a training set and a test set, the decision tree was fit to the training sets of X and y, and the model predictions were made using the test set of X.

- Model 5: Decision Tree with Binarization

Binarization is a preprocessing technique similar to binning, but instead of having multiple bins, now there are only two. This turns the classification problem into a binary classification problem, which is somewhat easier to predict. Since in Formula 1 it is ideal to finish on the podium, the position_race class was binarized to represent whether a given driver finished on the podium or not. Therefore, instances with a race position between 1 and 3 were given a position_race value of 1, and instances with a position_race value between 4 and 20, or of -1, were given a value of 0. Then the decision tree model was learned as before.

C. Evaluation Metrics

To evaluate the decision trees, a confusion matrix was generated, which shows the predicted class values compared to the actual class values, and the number of predictions for each. A classification report was also generated, which includes the model accuracy, precision, recall, and f1-score (not to be confused with Formula 1!). These metrics all had similar values, but their meanings are different. Accuracy represents the closeness of measurements to the true value being measured, precision is the closeness of repeated measurements of the same quantity to one another, recall is the fraction of instances in each class that the model classified correctly out of all the instances in that class, and f1-score is a combination of the precision and recall scores. A classification report was also generated to evaluate the random forest model. For the regression model, the R2 score and accuracy score were used. The R2 score is a metric that compares the variance of the predicted values to the variance of the actual values. An R2 score of 1 indicates that the model is a perfect fit, while a score of 0 indicates that there is no relationship between the predicted values and the true values at all.

IV. RESULTS

This section presents the results of the methods described in section 3 in the same order as previously described.

- Model 1: Decision Tree

The first decision tree model attempted to predict the final race position of a given driver in a given race out of 21 possible position values. This resulted in the large confusion matrix shown in figure 4. The accuracy of this model was 0.32, the precision 0.33, the recall 0.32, and the f1-score 0.32. These low scores were to be expected, as the probability of guessing the class correctly by random guessing is only 0.05.

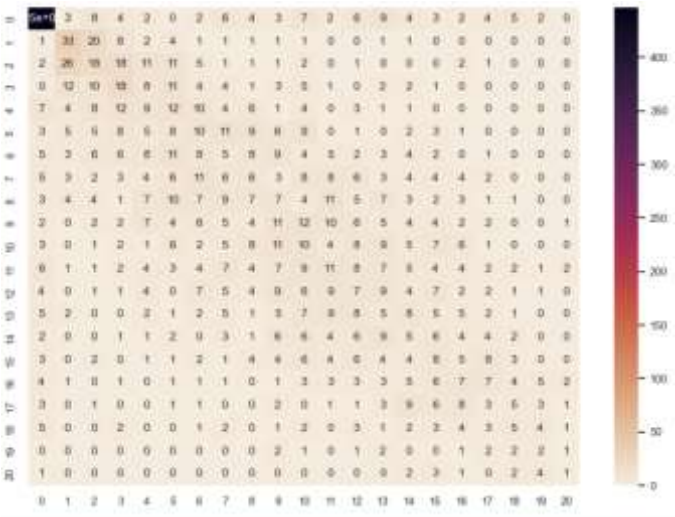

Figure 4 - Decision Tree Confusion Matrix

- Model 2: Random Forest

As random forest is an ensemble of decision trees, it is expected that a random forest model would be better fit than a singular decision tree model, which is what this study found. Given by the classification report, the accuracy of the random forest model was 0.36, the precision 0.32, the recall 0.36, and

the f1-score 0.33. Even though the difference is small, these scores are slightly higher than just the singular decision tree.

- Model 3: Regression

Regression is specifically for predicting numbers, so since the class value being predicted in this problem is an integer, it made sense to try it. The regression model was not the best fit to the data, with an R2 score of 0.50, and it was less accurate than the decision tree model and the random forest model, with an accuracy score of 0.11. The plot in figure 5 shows the actual class values of the data versus the values predicted by the regression model, and it can be observed how they almost cluster in the shape of a line, which is what a better fit model would be, but they are still quite spread out.

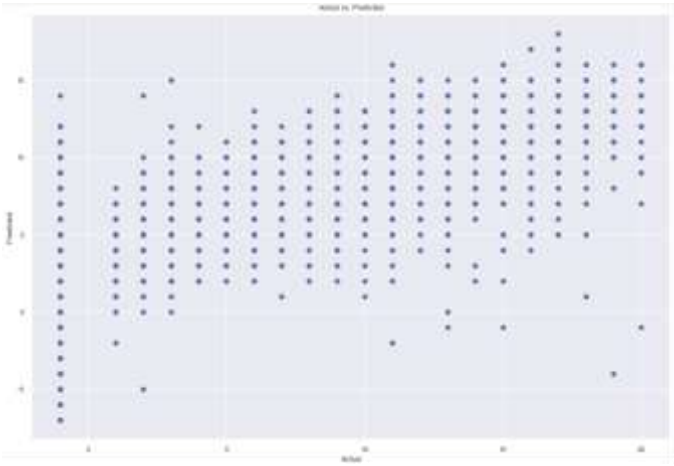

Figure 5 - Actual vs. Predicted Values

- Model 4: Decision Tree with Binning

As is to be expected, the decision tree model with binning implemented in the preprocessing stage performed even better than the previous models. The confusion matrix for this model is shown in figure 6. The accuracy of this model was 0.60, the precision 0.61, the recall 0.60, and the f1-score 0.60. These scores are almost double those of the decision tree model without binning, demonstrating that binning is a very effective strategy for improving model performance for this problem. This is because multiclass problems are very complex, so the less possible class values there are, the more likely it is the decision tree model will be able to make accurate predictions.

Figure 6 - Decision Tree with Binning Confusion Matrix

- Model 5: Decision Tree with Binarization

The last method implemented in this study was the decision tree with binarization applied in the preprocessing stage. As was stated for the previous model, the less possible class values there are, the more likely a decision tree model will make accurate predictions, so it is expected that with only two possible class values, this model would perform the best out of all the previous models, and that is what this study observed. The confusion matrix for this model is displayed in figure 7. The accuracy of this model is 0.88, the precision 0.87, the recall 0.88, and the f1-score 0.88.

Figure 7 - Decision Tree with Binarization Confusion Matrix

V. Related Work

Past studies with similar goals are described in this section and compared to this study.

One study by Sobrie [6] also applied tree-based models to perform different analyses of Formula 1 race data. Once such analysis was a study on the factors that lead to a top-three finish result, which is similar to what this study accomplished with binarization and a decision tree model. Sobrie's study found that starting position in the race was the most important feature in the top-three performance analysis. Observing the feature importances for the model in this study (shown in figure 8) reveals that the ending position in the qualifying round, which is also the starting position of the main race, was also the most important feature for predicting if a driver would finish in the top-three or not. This makes sense because a driver starting in pole position is more likely to finish the race first than a driver at the back of the grid.

	Feature Importance
raceId	0.158036
driverId	0.080202
constructorId	0.057897
grid	0.066842
laps	0.174047
rank	0.097421
qualiPosition	0.365555

Figure 8 - Feature Importance for Decision Tree with Binarization

Another study done by Patil, Jain, Agrahari, Hossari, Orlandi, and Dev [7] also implements linear regression to

predict which driver will win the season. They found that the number of races completed by a driver and the type of tires used were the biggest factors in predicting a driver's total points for the season. The total points for the season corresponds directly to which driver will win the season because the driver with the most points at the end of the season is the winner. This study is another example of how linear regression can be applied to predict Formula 1 race outcomes.

Finally, a study done by Dhanvanth, Rajesh, Samyukth, and Jeyakumar [8] presents a machine learning model to predict the winner of the next Formula 1 race based on previous race results. One interesting feature included was weather conditions. While that data was not used in this study, that would be an interesting feature to include and analyze in the future.

VI. Conclusion & Future Work

This study presents a comprehensive analysis of different methods of data preprocessing as well as different prediction algorithms to predict the outcome of Formula 1 races. The best fit model of all the models in this study was the decision tree model with which the data had been binarized in the preprocessing stage. This model had an accuracy of 0.88. While this model does not predict a driver's exact finishing position in the race, it is still interesting and relevant to predict if a driver will finish on the podium. Even with the 21-class problem, however, the random forest model had an accuracy of 0.36. Given that the probability of randomly guessing the final race position of a driver is 0.05, the random forest model performs seven times better than random guessing, which is significant. The outcome of any sporting event is in some part due to chance, which is part of what makes sports so exciting, so being able to apply data mining methods to improve the accuracy of race result prediction is meaningful.

In future work, it would be interesting to implement oversampling in the binarization example to balance the two classes and observe if this has any impact on the accuracy of the decision tree model. It would also be interesting to learn a random forest model and a linear regression model on the binarized data to compare their accuracies to the decision tree model with binarized data. These methods should likely increase model accuracy even further.

References

[1] Amazon Web Services, "F1 Insights Powered by AWS | Alternative Strategy," Amazon Inc. 2023.

[2] "Formula 1 World Championship (1950-2023)," Retrieved October 1, 2023 from https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020?select=races.csv.

[3] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[4] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

[5] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[6] L. Sobrie, "Sifting Through the Noise in Formula One: Predictive Performance of Tree-Based Models," Universiteit Gent, 2019-2020.

[7] A. Patil, N. Jain, R. Agrahari, M. Hossari, F. Orlandi, S. Dev, "A Data-Driven Analysis of Formula 1 Car Races Outcome," Communications in Computer and Information Science, volume 1662, February 23, 2023.

[8] S. Dhanvanth, R. Rajesh, S. S. Samyukth, G. Jeyakumar, "Machine Learning-Based Analytical and Predictive Study on Formula 1 and Its Safety," Lecture Notes in Electrical Engineering, volume 915, September 18, 2023.