Predicting Race outcomes in Formula-1

Submitted in partial fulfillment of the requirement for STAT 4F90 Honours Project

Bachelor of Science in Mathematics and Statistics

Department of Mathematics and Statistics

Faculty of Mathematics and Science Brock University

St. Catharines, Ontario

Ravi Solanki

[rs19ck@brocku.ca](mailto:rs19ck@brocku.ca)

Prof. William Marshall

May 12, 2024

**Contents**

**1 Introduction 1**

1.1Formula-1 1

**2 Data 2**

2.1 Telemetry Data 3

2.1.1 Lapwise Telemetry Analysis 4

2.1.2 Race-level Telemetry Aggregation 4

2.1.3 Seasonal and Driver Performance Review 5

2.1.4 Points Data 6

**3 Cluster Analysis 7**

3.1 K- Means 8

3.2 Silhouette Analysis 9

3.3 Results 12

**4 Machine Learning Algorithms 12**

4.1 Standardization 13

4.2 Model Evaluation 14

4.3 Machine Learning Models 15

4.3.1 Linear Regression 15

4.3.2 Random Forest Regression 17

4.4 Result 19

**5 Conclusion 21**

5.1 Reflection on Model Performance 21

5.2 Strategies for Improvement 21

5.2.1 Incorporating Additional Features 22

5.2.2 Preview 23

5.2.3. Additional Analysis 24

1 Introduction

In the high-speed, high-stakes world of Formula 1, the difference between victory and defeat often lies in the fractions of a second. This realm of roaring engines and lightning-fast reflexes may seem an unlikely place for data science to make its mark, yet this is exactly where it thrives. Just like a well-oiled engine, the gears of this cutting-edge discipline hum in harmony with the rhythm of F1. From race strategy to car design and driver performance, data science is the invisible co-pilot, charting the path to the podium. In this project, we delve into how data science applications are revolutionizing the world of Formula 1, making it a sleek, digitally driven sport.

“Being second is to be the first of the ones who lose.”

* Ayrton Senna
  1. Formula-1

In Formula 1, drivers control high-speed vehicles to be the first to complete a set number of laps, thus securing victory. However, it's not merely about racing; in Formula 1 teams of drivers, mechanics, and engineers work together to integrate technological innovation, strategic planning, and driving skills. Each season, teams participate in a sequence of races, known as Grands Prix, held on various circuits worldwide. These races occur over a weekend, commencing with two practice sessions on Friday, an additional one on Saturday, and a qualifying round. The final race, the true test of endurance and speed, takes place on Sunday.

During Friday's practice session, drivers familiarize themselves with their vehicles to mitigate issues such as understeering or unstable handling. Their performance during Saturday's qualifying rounds is crucial as it determines their starting position for the Grand Prix. A superior performance leads to an advantageous starting position. A lower starting position, on the other hand, imposes a significant challenge, compelling the driver to overtake up to 19 other drivers, each traveling at speeds exceeding 200mph! Qualifying consists of three segments: Q1, Q2, and Q3. In Q1, all 20 drivers have 18 minutes to record their fastest lap. The slowest five are eliminated, securing the final five slots on the grid (starting formation of the cars before the race begins). Q2 follows, lasting 15 minutes, with the five slowest drivers once again eliminated, determining grid positions 11 to 15. Similarly, with Q3 the final round with which the starting order for the Grand Prix is set. Qualifying is a tactical play that influences the race. The top ten qualifiers must commence the race on the tires used to clock their fastest lap in Q2. This rule forces teams to strike a balance between a quicker qualifying time and an optimal race strategy. Meanwhile, those who didn't advance to Q3 have the liberty to choose their tires, occasionally allowing them to deploy innovative strategies to compensate for their grid position deficit.

The top 10 finishers in each race earn points. The driver securing the first position receives 25 points, the second-place finisher 18 points, the third-place finisher 15 points, and so on, down to the tenth-place driver who is awarded 1 point. Points are accumulated by drivers and teams throughout the season, and the one amassing the most points claims the championship.

|  |  |
| --- | --- |
| Position | Points Awarded |
| 1st | 25 |
| 2nd | 18 |
| 3rd | 15 |
| 4th | 12 |
| 5th | 10 |
| 6th | 8 |
| 7th | 6 |
| 8th | 4 |
| 9th | 2 |
| 10th | 1 |

1. Data

In this chapter, we will explore the telemetry data obtained from the FastF1 API, a comprehensive tool for accessing Formula 1 timing and telemetry data. Telemetry data offers intricate details about each lap driven by the Formula 1 cars, providing insights into the performance and strategies adopted by teams and drivers. FastF1 is a Python library that allows users to access Formula 1 timing and telemetry data. It provides functionalities to fetch event schedules, race results, lap times, and detailed telemetry data for each driver.

* 1. Telemetry Data
* Telemetry data in Formula 1 refers to the detailed information collected from various sensors installed on a race car during a race or practice sessions. This data includes metrics like speed, throttle and brake application, gear selection, engine RPM, and tire temperature. It provides real-time insights into the car's performance and the driver's behavior on the track. Teams analyze this data to optimize car setup, improve race strategies, and enhance driver performance, making it an indispensable tool in the highly competitive world of Formula 1 racing. The telemetry data includes several key metrics:
* Speed: The car's speed at various points on the track.
* Throttle: The degree of throttle application by the driver.
* Brake: Indicates if and how hard the driver is braking.
* Gear: Current gear in which the car is.
* RPM: Engine revolutions per minute.
* DRS: Status of the Drag Reduction System (DRS)

For our analysis, the focus will be narrowed to key telemetry variables: Speed, Throttle, RPM, and Brake. These specific metrics provide a comprehensive view of a car's performance dynamics and driver behavior. We will aggregate this data across various scales, encompassing individual races and spanning the entire season.

2.1.1 Lapwise Telemetry Analysis

The first dataset, "Seasonal Lapwise Telemetry," provides a lap-by-lap breakdown of telemetry data, highlighting the average speed, RPM, throttle, and brake application for each lap. By examining this data, we can identify how a driver's performance fluctuates throughout a race, how they adapt to different track conditions, and how their strategy evolves lap by lap. To create the "2018 Lapwise Telemetry" dataset, we first enabled caching in FastF1 to speed up data access. This was crucial for efficient processing given the large volume of data. We then defined a function, summarize\_lap\_data(), to extract and average the telemetry data for each lap. This function processed each lap of a driver’s race, calculating the average speed, RPM, throttle, and brake usage.

We started with the summarize\_driver\_laps() function, which calculates the average speed, RPM, throttle, and brake data for a driver in a single race. This function iterates through all laps completed by a driver, summing the telemetry metrics and then dividing by the total number of laps to obtain average values. This lap-by-lap analysis is foundational for understanding driver performance in each race.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Lap** | **Average Speed** | **Average RPM** | **Average Throttle** | **Average Brake** | **Driver** | **Year** | **Race** |
| **0** | 180.07 | 9953.02 | 57.68 | 0.20 | RIC | 2018 | Shanghai |
| **1** | 194.31 | 9869.69 | 60.54 | 0.20 | RIC | 2018 | Shanghai |
| **2** | 194.30 | 9880.17 | 61.022 | 0.18 | RIC | 2018 | Shanghai |

Lapwise Telemetry Data

A group of graphs showing different types of data

Description automatically generated with medium confidence

Average Telemetry Performance for the first 20 laps for Driver ‘Ricardo’

2.1.2 Race-Level Telemetry Aggregation

In the second dataset, "Season Racewise Dataset," we aggregate the telemetry data for each race. Instead of lap-by-lap details, this dataset focuses on average values for each race, giving us an overall picture of a driver’s performance in a specific event. By comparing these averages across different races, we can determine how various tracks and racing conditions impact car performance and driver choices. This dataset is useful for understanding the broader trends and patterns in a driver's performance over the season.

For constructing the racewise telemetry dataset, we employed the FastF1 library in Python, a specialized tool for extracting and processing Formula 1 telemetry data. We defined a function, summarize\_race\_data(), to process individual races for a given driver. This function took a session object (representing a specific race) and a driver's code as inputs. It then iterated through all the laps driven by the driver in that race, collecting telemetry data such as speed, RPM, throttle, and brake usage for each lap. By concatenating the telemetry data from all laps and calculating their average values, we obtained a summarized view of the driver's performance for that specific race.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Driver** | **Year** | **Race** | **Average Speed** | **Average RPM** | **Average Throttle** | **Average Brake** |
| **RIC** | 2018 | Shanghai | 190.08 | 9797.41 | 58.99 | 0.18 |
| **BOT** | 2018 | Shanghai | 189.67 | 9766.84 | 60.97 | 0.17 |

Race-Level Telemetry Aggregated data

2.1.3 Seasonal and Driver Performance Overview

The final dataset, "Seasons Driver Year Summary," compiles telemetry data for the entire season, summarized at the driver level. It includes average speed, RPM, throttle, and brake data for each driver throughout the season. This high-level overview is essential for evaluating a driver’s overall performance, comparing different drivers, and observing performance trends over the course of the season. It helps us identify which drivers consistently perform well and which aspects of performance vary significantly from one driver to another.

For the seasonal summaries, we used Python and the FastF1 library to process telemetry data across multiple Formula 1 seasons. The summarize\_driver\_season() function expands this analysis to an entire season. For each race in each season, we retrieved the session data and applied summarize\_driver\_laps() to accumulate the averages of the telemetry data. This function handles any data anomalies or missing data, ensuring a robust dataset. The result is a comprehensive summary of a driver's average performance metrics across all races in a season. To broaden the scope, we processed data for multiple seasons and drivers. We iterated through a list of years and drivers, applying summarize\_driver\_season() to each combination. This approach provided a rich dataset, capturing the nuances of driver performance over different seasons and under varying conditions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Driver** | **Year** | **Average Speed** | **Average RPM** | **Average Throttle** | **Average Brake** |
| **RIC** | 2018 | 197.36 | 9970.51 | 61.86 | 0.18 |
| **BOT** | 2018 | 197.59 | 9929.16 | 62.34 | 0.19 |

Seasonal Driver Performance

2.1.4 Points Data

We decided to make our dataset more useful by adding Formula 1 drivers' points, which are the points a driver earns in each race, to our existing data. We got these points from the Formula 1 website and combined them with our current data on race-wise telemetry and overall seasonal telemetry. By doing this, we can now look for any patterns between how drivers perform in races and the points they earn, making it easier to see what might lead to better results.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Driver** | **Year** | **Race** | **Average Speed** | **Average RPM** | **Average Throttle** | **Average Brake** | **Points** |
| **RIC** | 2018 | Shanghai | 190.084 | 9797.41 | 58.99 | 0.18 | 25 |
| **BOT** | 2018 | Shanghai | 189.67 | 9766.84 | 60.97 | 0.17 | 18 |
| **RAI** | 2018 | Shanghai | 189.78 | 9871.06 | 60.52 | 0.20 | 15 |

This is Race-wise data but with points

A graph of numbers and lines

Description automatically generated

Comparing driver performance over the years

3 Clustering Analysis

Clustering analysis is fundamentally about grouping a set of objects in such a way that objects in the same group (or cluster) are more similar to each other than to those in other groups. Its essence lies in its ability to impose structure on a collection of unlabeled data points. This method is crucial in revealing hidden patterns and structures within datasets, where the inherent similarities among data points are not immediately obvious.

Unlike supervised learning, where the model is trained on a labeled dataset, clustering analysis is a form of unsupervised learning that doesn't rely on pre-defined labels or categories. Instead, it identifies patterns and intrinsic structures in the data based on a similarity metric. One of the primary challenges in clustering analysis is the selection of an appropriate number of clusters and the choice of a suitable algorithm. The choice of algorithm and parameters largely depends on the nature of the data and the specific objectives of the analysis. For our dataset we will be primarily focusing on K-Means and Silhouette analysis.

3.1 K-Means Clustering

K-means clustering is a method used to partition a dataset into K distinct, non-overlapping subsets (or clusters) based on the distance between data points and the centroid of clusters. The goal of K-means is to minimize the variance within each cluster, essentially grouping data points into clusters such that the sum of squares from points to the assigned cluster centroids is minimized. This method begins with an initial set of randomly selected centroids, which are points representing the center of a cluster. The algorithm then iterates through two key steps: assignment and update. In the assignment step, each data point is assigned to the closest centroid, based on the Euclidean distance or another distance metric. In the update step, the centroids are recalculated as the mean of all data points assigned to that cluster. These steps repeat until the centroids stabilize and no longer change significantly, indicating that the clusters have been formed.

However, it requires the number of clusters to be specified in advance and can be sensitive to the initial choice of centroids. The algorithm performs best on datasets where the clusters are convex and isotropic, but it may struggle with complex cluster shapes or varying cluster sizes.

Let xidenote the ith data point and cj denote the centroid of the jth cluster. The algorithm iteratively updates the centroids to minimize the total within-cluster variance, given by:

where aijis 1 if xi is assigned to the jth cluster and 0 otherwise. The centroids are updated by computing the mean of all points assigned to each cluster:

This ensures that the algorithm converges to a set of centroids that minimize the within-cluster variance, forming distinct, non-overlapping groups of data points with minimized intra-cluster distances and maximized inter-cluster distances.

The elbow method is used in determining the number of clusters in a dataset, commonly applied to the K-means clustering algorithm. The elbow method involves plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use. More formally, it plots the sum of the squared distances of samples to their closest cluster center for various numbers of clusters (k). This metric is often referred to as the Within-Cluster Sum of Squares (WCSS).

As you increase the number of clusters, the WCSS decreases because the samples will be closer to the centroids they are assigned to. The rate of decrease slows down significantly at a point, creating an "elbow" in the graph. The idea is to choose the number of clusters so that adding another cluster doesn't give much better modeling of the data. The "elbow point" is considered to be an indicator of the appropriate number of clusters because it represents the point at which adding more clusters results in diminishing returns in terms of the explained variation.

3.2 Silhouette Analysis

Silhouette analysis is a technique used to evaluate the quality of clusters formed by any clustering algorithm, such as K-means. It provides a succinct graphical representation of how well each object has been classified. The silhouette coefficient for each data point is a measure of how similar the data point is to its own cluster compared to other clusters. The coefficient values range from -1 to 1, where a high value indicates that the data point is well matched to its own cluster and distinct from other clusters. A value near 0 suggests that the data point is on the border of two clusters, while a negative value indicates that the data point might have been assigned to the wrong cluster.

To calculate the silhouette coefficient, two distances are computed for each data point: (a): The average distance of a data point to all other points in the same cluster. If the point is x and its cluster is C, with n points in C including xitself, then a is calculated as:

where d(x,xi) is the distance between point x and another point xi​ in the same cluster C. And (b) The smallest average distance of the data point to all points in any other cluster, minimized across all other clusters. If D represents a different cluster from that of x, then b for point x is calculated as:

where ∣D∣ is the number of points in cluster D and d(x,xj) is the distance between point x and a point xj in cluster D.

The silhouette coefficient for a point is then given by the formula,

Silhouette analysis is particularly useful for determining the optimal number of clusters in a dataset. By calculating the average silhouette coefficient across all data points for different numbers of clusters, one can identify the number of clusters that best fits the data.

However, silhouette analysis requires at least two clusters to be meaningful because the calculation of the silhouette coefficient for each point involves comparing its distance to points within its own cluster a(intra-cluster distance) with its distance to the nearest cluster to which it does not belong b(inter-cluster distance). If there's only one cluster in the dataset, then the calculation of b, the smallest average distance of the data point to all points in any other cluster, doesn't make sense because there are no "other clusters" to compare against.

This limitation means that silhouette analysis is not suitable for determining if there should be only one cluster. It assumes the existence of more than one cluster and is used to assess the quality of the clustering. Thus, when applying silhouette analysis, it's important to already have or to consider multiple clustering configurations (with 2 or more clusters) to evaluate the clustering performance and the appropriateness of the number of clusters chosen for the data.

## A graph with a line Description automatically generated

Silhouette analysis for Lap-Wise Driver Telemetry data

A graph with a line

Description automatically generated

Race\_wise\_yearly\_data\_for\_each driver\_2018\_2022

A graph with a line

Description automatically generated

Seasonal\_aggregeted\_driver\_data

3.3 Results

In our project, we used two methods, K-means and silhouette analysis, to analyze clusters in our Formula-1 datasets. These datasets were "Seasons Driver Year Summary", “Season Racewise Dataset” "Seasonal Lapwise Telemetry."

With the K-means method, initially guided by the elbow method, we leaned towards identifying two main clusters. However, this finding wasn't very clear-cut because the elbow method isn't always precise in determining the number of clusters.

Moving on to silhouette analysis, it confirmed two clusters for the "Seasons Driver Year Summary" and “Season Racewise Dataset” dataset. Given the simpler nature of this data, two clusters seemed like a reasonable result. However, for the "Seasonal Lapwise Telemetry" dataset, which is more complex, we were expecting to find a more detailed clustering, ideally more than two clusters. Again, silhouette analysis suggested two clusters, which made us question whether this was due to the method's limitations, especially since it can't indicate if there's just one cluster or none.

A diagram of a number of dots

Description automatically generated

K-means Cluster for Lap-level Dataset with value k set to 2 by using Silhouette

Given these outcomes and the complex nature of the "Seasonal Lapwise Telemetry" data, which suggested a continuous rather than a distinct set of clusters, we decided it was better to use machine learning models to explore the data further. This approach aims to understand the relationships between telemetry variables and driver performance(points) better, moving beyond the initial cluster analysis.

4 Machine Learning Algorithms

4.1 Standardization

Before deploying machine learning models such as Linear Regression, Random Forest, KNN, and SVM, it is crucial to standardize telemetry data to optimize model performance. Standardization adjusts different variables to a uniform scale by calculating the Z-Score, which normalizes the features to have a mean of zero and a standard deviation of one. This process is performed by subtracting the mean and dividing by the standard deviation for each feature value. By eliminating disparities (differences or inconsistencies in the scale or range of feature values within our dataset) in scale among features, standardization prevents models from being biased towards variables with larger ranges, thereby enhancing the interpretability and effectiveness of the machine learning algorithms employed.

The formula for the Z-Score is:

Here, *m* is the observed metric (e.g., 'Average Speed', 'Average RPM', etc.) for the driver in a specific race, is the sample mean of that metric across the dataset you're analyzing, and represents the sample standard deviation of the metric, indicating the variation of metric values from the sample mean.

When it comes to splitting our data into training and testing sets, or when doing cross-validation, we standardize based on the training set. This means we calculate the mean and standard deviation for standardization from the training set only, then apply that to the test set. For cross-validation, this standardization is done separately in each fold. This is to make sure that the testing data remains 'unseen' and doesn't influence the training process, giving us a true picture of how the model will perform on new data.

4.2 Model Evaluation

To determine the most effective algorithm for our predictions, it's crucial to evaluate each model comprehensively. In our analysis, we primarily focus on two metrics: the R-squared value and the CV R2, with a particular emphasis on using R-squared as the parameter for CV R2.

R-squared, a key performance metric for regression models, quantifies the percentage of the variance in the dependent variable that is predictable from the independent variables. This measure provides insight into the strength and effectiveness of the model's ability to capture and explain the variability in the data. An R-squared value closer to 1 indicates a model that accurately reflects the dataset, offering high predictive power.

To further validate our model's performance and ensure its robustness across different subsets of the data, we employ n-fold cross-validation. This technique meticulously divides the entire dataset into n equal parts, or 'folds'. The process of cross-validation involves sequentially using each fold as a test set, while the remaining n-1 folds are combined and used as the training set. Specifically, 'training' refers to the process of fitting the model on a subset of the dataset to learn the patterns and relationships between the independent variables and the dependent variable. 'Testing', on the other hand, involves applying the trained model to a separate subset of data (the test set) that was not used during the training phase. This step is critical for evaluating the model's performance and its ability to generalize to new, unseen data.

By cycling through all n folds in this manner, each part of the dataset is used exactly once as the test set, while the model is trained n-1 times with different portions of the data. The cross-validation approach, combined with R-squared scoring for each fold, allows us to comprehensively assess the model's consistency and reliability across multiple splits of the data. The CV R2 value, calculated as the average of the R-squared values obtained from each testing phase across all folds, serves as a robust indicator of the model's overall predictive accuracy and its generalization ability to unseen data.

4.3 Machine Learning Models

Machine learning models are algorithms that are trained on data to make predictions or decisions, without being explicitly programmed to perform the task. These models can range from simple linear regression models to complex deep neural networks. The choice of model depends on the problem at hand, the type of data available, and the complexity required to make accurate predictions or decisions.

4.3.1 Linear Regression

Linear Regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It aims to predict the outcome variable based on the input variables. The method involves fitting a linear equation to observed data, where the equation represents the predicted relationship between the variables. In this equation, each independent variable's coefficient reflects its impact on the dependent variable. Linear Regression is vital for predicting outcomes and understanding the variables' relationships in various fields. The formula at the heart of our Linear Regression model for analyzing Formula 1 driving behaviors—such as speed, RPM, throttle usage, and brake engagement—is

Here, y represents the points a driver earns, xi1, xi2, ..., xinare the independent variables (our chosen performance metrics), β0 is the intercept, β1, β2, ..., βn are the coefficients demonstrating each variable's impact, and ϵ is the error term. These parameters are deduced using the least squares method, minimizing the discrepancies between observed and predicted values. The β values are typically calculated using the matrix equation:

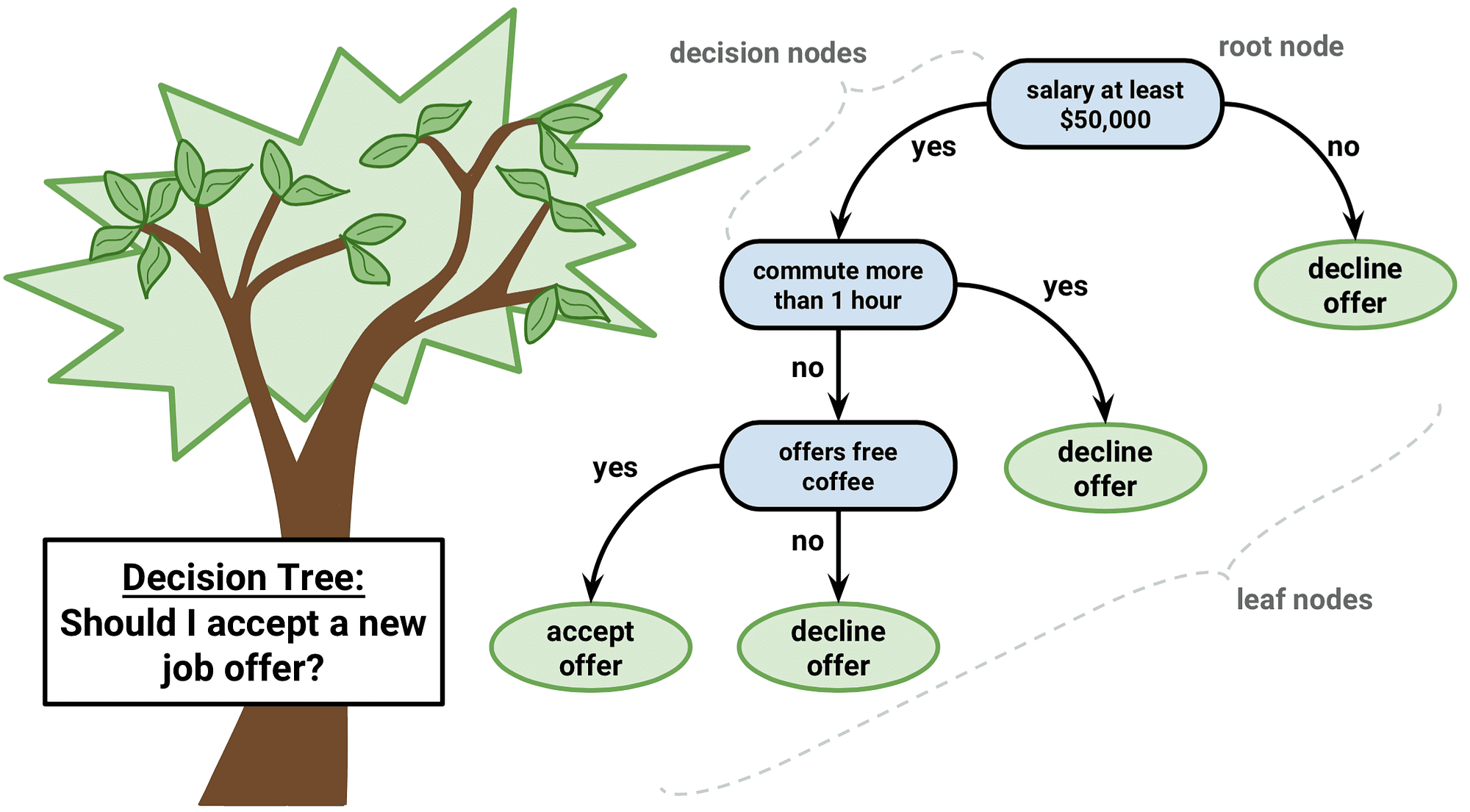
In this equation:

* X is a matrix where each row represents a driver's race performance metrics (with a column of ones for the intercept).
* y is a vector of the observed points earned by each driver.
* is the transpose of X.
* ​ is the vector of estimated coefficients that minimize the sum of squared residuals.

By solving this matrix equation, we can estimate the values of β0, β1, …, and βn that provide the best fit of our linear regression model to the Formula 1 data.

We are using Linear Regression to see if the driving data like speed, RPM, throttle, and brake usage can tell us something about how drivers score points in Formula 1 races. We selected these specific performance metrics as they offer a comprehensive overview of a driver's behavior throughout a race. Our aim is to uncover a definitive correlation between these driving behaviors and the points drivers accumulate at the race's conclusion. Linear Regression might work well for this because it’d help us spot trends and make predictions. We’re focusing on the ‘Points’ each driver earns as what we want to predict, using their average speed, RPM, throttle, and brake data as clues to build our prediction model.

4.3.2 Random Forest Regression



Random Forest Regression is a statistical method that operates by constructing multiple decision trees from randomly selected subsets of the training dataset. A decision tree is a simple model that splits data into branches at decision points, leading to possible outcomes based on input features.

For each continuous variable (like variables in our dataset), the decision tree looks for the best value to split the data on. The process of choosing the "best value" to split a continuous variable in a decision tree involves identifying the point that most effectively separates the data into groups with distinct outcomes. This is usually done by measuring how much each possible split will reduce the impurity (refers to how mixed up the different outcomes are within a group) or increase the purity (indicates how similar the outcomes within a group are) of the nodes—essentially, how well-separated the groups are in terms of the target variable after the split. The goal is to make the groups as homogeneous as possible with respect to the target variable. The decision tree algorithm will look at each feature (Average Speed, Average RPM, Average Throttle, and Average Brake) and try different split points within these features to see where it can most effectively divide the drivers into groups with different average points. For each potential split, the decision tree will calculate how much that split will reduce impurity. After evaluating the splits across all features, the decision tree will choose the feature and split point that resulted in the greatest reduction in variance. This becomes the decision at the root of the tree.

Building on this, Random Forest Regression creates a 'forest' by generating multiple decision trees, each constructed from a randomly selected subset of the training dataset. This randomness ensures that each tree in the forest is unique, varying in the data points and features it analyzes. The method then 'aggregates' the predictions from all these individual trees to produce a final result. This aggregation is typically done by averaging the predictions in the case of regression, which smooths out the errors of individual trees. As a result, the combined prediction of the forest is generally more accurate and reliable than that of any single tree, benefiting from the diverse perspectives of multiple trees while mitigating their individual biases and variances.

Let’s assume that we have a set of training data (xi, yi), i = 1, ..., N, where xirepresents the feature vectors and yi the responses, a Random Forest regression model builds Bregression trees. Each tree bprovides a prediction yb(x) for an input *x*. The Random Forest regression prediction (x) is the average over all trees:

For our dataset that tracks various metrics like 'Average Speed', 'Average RPM', 'Average Throttle', and 'Average Brake', Random Forest Regression might be advantageous due to its ability to handle a high number of features and to model complex, non-linear relationships. It is also less sensitive to outliers, which are uncommon or extreme values, than many other regression methods, making it well-suited for the intricate dynamics of Formula 1 telemetry data. By applying Random Forest Regression, we aim to yield a nuanced understanding of how different telemetry factors correlate with the points scored by drivers.

4.4 Results

For our Linear Regression model applied to the Race-level Formula-1 dataset, the Mean Squared Error (MSE) on the testing set was 49.27, which suggests a moderate level of error in the predictions. The R2 score was notably low at 0.0162, indicating that the model explains only about 1.6% of the variability in the race points.

The cross-validation results further reflect the model's limited effectiveness, with a Mean Cross-Validation R2 value of 0.0106 across five folds. Some individual fold scores were slightly negative, which highlights specific instances where the model performed poorly.

The results suggest that the linear regression model, with the features chosen, is not very effective in predicting the points scored by drivers in Formula 1 races. The low R2 and cross-validation R2 value indicate that the model fails to capture the complexities and variabilities associated with race outcomes, which could be influenced by many unaccounted factors such as race conditions, driver skill, and team strategies.

For the Linear Regression model applied to the season-level Formula-1 dataset, the performance metrics indicate significant challenges in model efficacy. The R2 score is -0.119, suggesting that the model does not fit the data well and fails to account for the variability in the season points; in fact, it performs worse than a simple horizontal line model.

The cross-validation results reinforce this observation, with highly varied scores across the folds. Scores range from a high of 0.08 to a low of -0.91, indicating inconsistent and generally poor performance across different subsets of the data. The Mean Cross-Validation R2 value is -0.38, further highlighting the model's inability to provide reliable or accurate predictions for this dataset.

These results suggest that the linear regression model, as currently configured with the existing features, is inadequate for modeling outcomes at the season level in Formula-1. This could prompt a review of both the model assumptions and feature selection to better capture the dynamics influencing season-long performances in the sport.

For the Random Forest model applied to the race-level Formula-1 dataset, we set the depth of the trees to 8. This decision was made after comparing the R² values for different tree depths, where a depth of 8 provided the best balance of model complexity and performance. The R² score for the Random Forest model at this tree depth is 0.0328, indicating that it explains approximately 3.28% of the variance in race points. Although this suggests a modest level of predictive accuracy, it still demonstrates limited effectiveness.

The cross-validation results further support this observation, showing some variability in model performance across different data subsets, with individual R² scores ranging from 0.0006 to 0.0901. The Mean Cross-Validation R² value is 0.0452, reinforcing the model's moderate capability in capturing variability in race outcomes.

These findings suggest that while the Random Forest model with a tree depth of 8 offers some improvement over simpler models, such as linear regression, its overall predictive power remains modest. To enhance performance, further adjustments in model parameters or the incorporation of additional or different features might be necessary to better address the complexities of Formula-1 race data.

For the Random Forest model applied to the season-level Formula-1 dataset, the performance metrics indicate significant challenges. We set the depth of the trees to 8, which was determined to be the optimal depth after comparing the R² values for different tree depths. Despite this optimization, the R² score for the model is -0.0560, indicating that the model does not adequately fit the data and performs worse than a simple horizontal line model.

The cross-validation results highlight substantial variability and generally poor performance across different subsets of the data. Individual R² scores vary widely, ranging from -1.1966 to 0.5461. This extreme variation suggests that the model struggles significantly with certain segments of the data. The Mean Cross-Validation R² value stands at -0.3273, further underscoring the model's inability to consistently predict season-level outcomes in Formula-1 racing.

These results suggest that the Random Forest model, even with a depth of 8 and within the context of the chosen features, fails to capture the complex dynamics that influence season-long performances in Formula-1. This may necessitate a re-evaluation of both the model structure and the features used, or perhaps a shift towards more sophisticated modeling approaches that can handle the high variability and complex interactions typical of season-level sports data.

5 Conclusion

Throughout our analysis of Formula-1 race and season-level datasets, we explored several statistical and machine learning models, including Clustering, Linear Regression, and Random Forest. Despite the sophistication and diversity of these approaches, none of the models delivered satisfactory predictive power. This consistent underperformance across different methodologies prompts a deeper consideration of both the complexity of the dataset and the suitability of the models applied.

5.1 Reflection on Model Performance

The poor performance observed suggests that the current features may not capture all the relevant variables affecting Formula-1 outcomes, or the relationships among these variables are too complex for standard linear and tree-based models. This conclusion highlights the need for a more nuanced approach to feature selection and model architecture, especially given the dynamic and multi-faceted nature of Formula-1 racing.

5.2 Strategies for Improvement

The To improve on our model’s performance we could try adding more features since it is clear that telemetry variables alone don’t significantly affect the points earned. Since we had different time stamps in our raw data there are additional analysis techniques we could try such as Time Series Analysis.

5.2.1 Incorporating Additional Features

As we continue to refine and enhance our analysis of Formula 1 telemetry data, numerous opportunities arise for expanding our methodologies and improving the precision of our predictions. The complex nature of Formula 1 racing, where every minor factor can significantly impact performance, calls for a detailed and sophisticated approach to data analysis. Here are some future applications and recommendations for advancing our project:

* **Climate Conditions:** Adding weather-related data like temperature, humidity, and precipitation could enhance model accuracy, as these factors significantly influence race conditions.
* **Starting Position:** Including data on drivers' qualifying positions could provide more insight into the capabilities of both the cars and drivers under race-like conditions.
* **Safety Car:** Accounting for races where the safety car was deployed could help adjust predictions since such events can alter race outcomes dramatically.
* **Tire Type:** Different tire compounds play a crucial role in strategy and performance, particularly under varying weather conditions.

5.2.2 Preview of what these changes could look like

Why should we include Tire type? In Formula-1 stint refers to the portion of a race between pit stops. Drivers divide each race into several “stints,” over which the performance of a car can vary due to tire condition, fuel load, and other factors.

The type of tire compound used in the stint (e.g., SOFT, MEDIUM, HARD).

Soft = Red, Medium = Yellow, Hard = White

We use violin plot and swarm plot; Violin plot will show the distribution of lap times for each driver, displaying the density of different time values. It is called a violin plot because of its symmetrical shape. This allows us to see not only common lap times (like median) but also how the lap times spread. With this we can also compare the performance of different drivers, we can see who was more consistent, who had outliers, etc. The width of the plot at different values indicates the density pf the data at that point. A wider section means more lap times around that value, and a narrow section means fewer lap times (if a driver’s plot is consistently narrow, it may indicate very consistent lap times. Also if there are any usual or unexpected lap times they may show up as narrow “spikes” in the plot).

A graph of different colored dots

Description automatically generated

The accompanying violin and swarm plot visualizes the distribution and individual data points of lap times for drivers in the Monza 2018 race, segmented by tire compound used. Each violin plot displays the probability density of the lap times, reflecting common and diverse performance metrics across different compounds. The additional layer of swarm plot points delineates individual lap times, offering insights into the consistency of each driver and the effectiveness of tire choices under race conditions. By correlating these findings with additional factors such as weather conditions, more nuanced strategies can be developed. The plot reveals not only the raw pace but also the strategic decisions influencing race outcomes, emphasizing the critical role of tire selection in Formula 1.

5.2.3 Additional Analysis

Once we have incorporated all the additional variables, we can try a few more analysis such as;

* **Series Analysis:** Utilizing time series methods might reveal underlying trends and performance cycles that affect outcomes over different seasons or series of races.
* **Interaction Effects:** Exploring interactions between variables, such as how weather conditions and tire types combine to affect performance, could uncover complex dynamics that simpler models overlook.
* **Survival Analysis:** This technique could be particularly useful for predicting the likelihood of a driver finishing a race, influenced by their start position and other operational conditions.

By embracing these advanced analytical approaches and technologies, we can significantly enhance our understanding of Formula 1 dynamics and contribute to the evolution of the sport. These recommendations aim to not only improve predictive accuracy but also to deepen our understanding of the complex interplay of factors that define the pinnacle of motorsport competition.

6. References

Brownlee, Jason. A Gentle Introduction to k-fold Cross-Validation. 2019. url: <https://machinelearningmast.com/k-fold-cross-validation/>

Mahmoud Parsian. Data Algorithms**.** 2015. url: <https://www.oreilly.com/library/view/data-algorithms/9781491906170/>

Bertrand Clarke, Ernest Fokoue, Hao Helen Zhang. Principles and Theory for Data Mining and Machine Learning. 2009. URL: <https://link.springer.com/book/10.1007/978-0-387-98135-2>

Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning Data Mining, Inference, and Prediction. 2008. URL: <https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf>

What is F1? URL: <https://www.formula1.com/en/page.what-is-f1.html>

The beginner’s guide to the F1 weekend, URL: <https://www.formula1.com/en/latest/article/the-beginners-guide-to-the-formula-1-weekend.5RFZzGXNhEi9AEuMXwo987>

FastF1 API. URL: <https://docs.fastf1.dev>

Formula-1 Driver Standings. URL: <https://www.formula1.com/en/results/driver-standings.html>

Leo Breiman, Adele Cutler. Random Forests. URL: <https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm>