Predictive Race Strategy Optimization for Petit Le Mans using XGBoost

Overview:

* This model is an XGBoost model, which is a type of gradient-boosted decision tree, to predict lap times based on features like fuel load, stint length, and track conditions. Mathematically, it builds many small decision trees one after another, each trying to correct the errors made by the previous ones by following the gradient of the loss function (the difference between predicted and actual lap times). The final prediction for a lap time is a weighted sum of all those trees, which lets the model capture both simple and subtle non-linear relationships between variables.

Job: The models job is to predict lap time (in seconds) based on:

Lap Number: how far the driver is into the stint.

Top Speed: top speed per lap.

Sector 1, 2, 3 Times: how long it takes to get to certain points on the track from the start line.

What is it Learning?

How these values correlate with total lap time, and how they evolve together. It is also

learning the importance of each feature.

What is XGBoost?

XGBoost stands for eXtreme Gradient Boosting. It’s one of the fastest and most accurate algorithms for regression and classification. It works by building decision trees one after another, and each tree tries to fix the mistakes of the last one. Ex: First tree makes a prediction. It’s wrong. The second tree sees the error, and tries to fix it. The third tree learns from both of them and improves even more. This process repeats until performance stops improving.

What is a Decision Tree?

A decision tree splits the data into branches based on thresholds. For example:

If top speed < 245; the tree goes left

If sector 3 > 30; the tree goes right

Else; the tree goes left

Else; the tree goes right

This continues until it hits the max\_depth that we set.

What is boosting?

Boosting is a technique that combines many weak learners, trees that are only slightly better at guessing than others, into one strong learner. For example in XGBoost specifically:

The first tree tries to predict y (an actual lap time)

The model then measures the error

The next tree is trained to predict the error (this is called the residual)

These trees are added together to correct each other

Let’s take one example data point, mathematically:

* True lap time: 80 sec
* Prediction from tree 1: 78 sec
* Error = +2 sec

Tree 2 will try to predict that missing +2 sec

Tree 3 might correct another small part of the error, and so on.

What is it learning from?

XGBoost is learning non-linear patterns between our input features and lap time:

* How later laps tend to get slower (due to tire wear)
* How later laps may get faster (due to lower fuel weight)
* Whether higher top speed = faster lap (sometimes it isn’t)
* Whether sector times are consistent predictors
* Interactions between sectors like:
  + If S1 is fast but S3 is slow, total lap time might still be slow

XGBoost is great at learning interactions between variables without someone having to code them manually, which works hand-in-hand with racing.

Finding 1: Feature Importance Chart

* Sector 2 time dominates: It contributes over 81% of the model’s decision-making. That makes sense since S2 is the longest sector.
* Sectors 1 and 3 help but not as much.
* Lap Number is a very small factor- suggesting minimal lap-to-lap variation considering consistency and tire degradation aren’t a huge factor here.

Finding 2: Observations of the Model Itself

* The model is great at predicting normal laps (where lap time is ~80 sec)
* When a lap is unusually long, (i.e. pit stop, spin, traffic), the model struggles, these are the tall blue spikes where the actual time was much longer than the predicted.
* It learns incredibly well from consistent driving patterns

Finding 3: What is the Model Good For?

* Right now it’s a baseline lap time estimator. It predicts what a “clean lap” would be given sector data
* Could be used for real-time prediction, like estimating final lap time based on early sector performance accurately (useful for qualifying)
* Can highlight outliers, when the gap between actual vs. predicted is much higher than expected, we can reasonably assume some type of racing incident has happened. (spin, traffic, collision, outlap)

Things to Work On

* Predicting Strategy like when to pit or how tire wear affect lap time – no degradation/tire/fuel features
* Adapting to driver behavior- doesn’t learn from inputs like throttle/brake or steering telemetry
* Race conditions- weather, traffic, cautions, etc