ApexMetry — Project Report

Author:

1. Summary

Apexmetry is a web app that is used to predict the podium of an upcoming Formula 1 race. It uses historical race data from the year 2018 onwards, as well as the qualifying data of the upcoming race. The data is collected from the FastF1 library. This data is then used to train a machine learning model. The predictions are served through the FastAPI backend to a Next.js frontend.

2. Motivation

As a Formula 1 fan and an AI undergraduate student, I wanted to build something combining both of my interests. Before a race, I would always try to predict who the winner would be. So rather than intuitively trying to guess the podium, I decided to build a ML model to predict the podium based on historical race data.

3. How it works

3.1 Data

- **Source:** FastF1 — official F1 timing, results, and weather data
- **Seasons:** [2018 – 2025] — more recent data

3.2 Features

Feature	What it captures
Grid position	Where the driver starts, from qualifying
gap to pole in qualifying	How far off the fastest qualifying lap they were
driver form (last 5 races)	Average finish over the last 5 races
team form (last 5)	The team's average finish over the last 5 races
Track history	The driver's average past finish at this circuit
Team and track	Which constructor and which circuit

[Edit to match your final features. Optional: one sentence on which features mattered most.]

3.3 Model

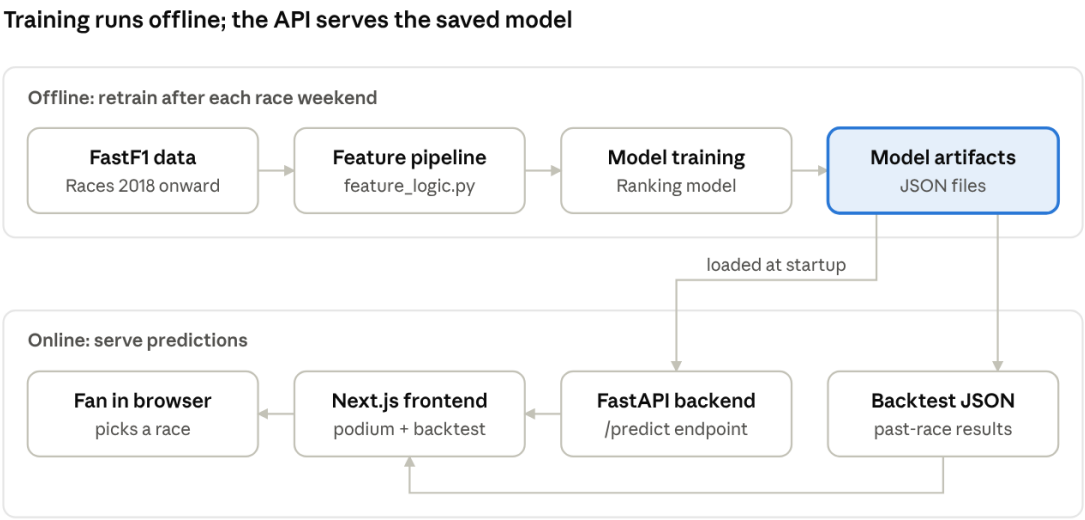
The model ranks each driver based on the predicted finishing position. It uses a gradient-boosted ranking model, XGBRanker. The model is trained on data from 2018 onwards.

3.4 Avoiding data leakage

[One short paragraph. Keep this section — it shows you understand a common ML pitfall.]

Suggested: Every feature is built only from information available before the race. Form features use only earlier races, the data is split by season rather than randomly, and a rain feature that relied on actual race-day weather was [removed / handled by ___] because it wouldn't be known in advance.

3.5 Architecture



ApexMetry architecture · offline training and online serving

Training runs offline after each race weekend and saves the model as JSON files. The FastAPI backend loads them once at startup and builds features for upcoming races from the latest qualifying data, using the same feature code as training. The frontend calls the backend for live predictions and reads precomputed backtest results for past races.

[Edit the diagram labels to match your final build, and add where each part is hosted.]

4. Results

Tested on the [2025] season ([24] races), with no data from that season used in training.

Races	Count	Winner (P1) correct	P2 correct	P3 correct	Avg NDCG@3
Dry	[18]	[ ]%	[ ]%	[ ]%	[ ]
Wet	[6]	[ ]%	[ ]%	[ ]%	[ ]
All	[24]	[ ]%	[ ]%	[ ]%	[ ]

NDCG@3 scores the predicted top three from 0 to 1, giving partial credit when a driver is placed close to their real position.

[Optional but strong: a baseline row, e.g. "assume the top 3 qualifiers finish in grid order", to show the model adds value.]

[2–3 sentences on what the results mean: where it does well, and where it struggles.]

Suggested: ApexMetry is strongest at picking the race winner on dry weekends. Third place is the hardest to call, and wet races score noticeably lower, because rain is rare in the training data and reshuffles results.

5. Limitations and next steps

Limitations

- Predictions are only available after qualifying
- Wet races are predicted less accurately, due to less amount of data.
- Crashes, penalties and strategy calls can't be anticipated

Next steps

- [Predictions before qualifying, using practice data]
- [Better wet-race handling, e.g. weather forecasts as input]
- [Automated retraining after every race weekend]

6. Tech stack and links

- **Data and ML:** Python, FastF1, pandas, Scikit-learn, XGBoost (XGBRanker)
- **Backend:** FastAPI, [hosting: Cloudflare Containers]
- **Frontend:** Next.js, [hosting: Cloudflare Workers]

ApexMetry is an independent personal project, not affiliated with or endorsed by Formula 1, the FIA, or any team. Data is sourced through the open-source FastF1 library.