



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85181>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

OpenTyreF1: A Data-Driven Mobile Platform for Formula 1 Telemetry Analysis, Tyre-Degradation Modelling and Counterfactual Race-Strategy Simulation

Atharva Joshi¹, Prachi Jain², Ankit Sawalakhe³, Kripa Patil⁴, Araju Yelekar⁵, Devshree Khodke⁶

Department of Computer Science & Engineering, G H Raisoni University, Amravati, Maharashtra

Abstract: Formula 1 publishes one of the richest open telemetry streams in professional sport, yet most public tools built on it are descriptive: they show what happened and stop there. This paper presents OpenTyreF1, a full-stack system that turns that data into decision support on a mobile phone. The system integrates three public sources (Jolpica-F1/Ergast, OpenF1 and FastF1) into one normalised 15-table relational corpus of 237,308 records covering 492 races from 2002 to 2026. On top of this corpus it builds a shared clean-lap filter, a gradient-boosted (XGBoost) lap-time model, a counterfactual stint-strategy simulator, a 1,500-run Monte Carlo outcome estimator, seven pit-wall decision modules and a grounded question-answering assistant, all served by a FastAPI backend to an Expo/React Native app. Evaluated on 12 completely unseen races, the model achieves a mean absolute error of 0.893 s, RMSE of 1.205 s and R^2 of 0.978, beating the strongest naive baseline by 0.270 s. An ablation shows that predicting a lap's offset from its race median, rather than the absolute lap time, cuts error four-fold, a far larger effect than adding features. The results show that, for motorsport data of this size, careful lap admission, target design and leakage-free evaluation matter more than the choice of learner.

Keywords: Formula 1, telemetry analytics, tyre degradation, XGBoost, race-strategy simulation, Monte Carlo methods, data leakage, mobile decision support, FastAPI, React Native.

I. INTRODUCTION

A modern Formula 1 car carries around three hundred sensors, and a meaningful part of that stream (lap times, tyre compounds and ages, pit-stop durations, race-control messages and trackside weather) is now published openly through community APIs. The raw material for quantitative race analysis is therefore no longer scarce. What is scarce is the analytical layer that turns it into decisions. Most fan and analyst tools plot what happened: a position chart, a stint diagram, a table of pit stops. The question a race strategist actually asks is counterfactual: "If this driver had pitted three laps later on the hard tyre, where would they have finished, and how sure can we be?" Answering it needs a calibrated lap-time model, an explicit rule for which laps represent true racing pace, a simulator that compares a hypothetical run against the field's real behaviour, and an honest statement of uncertainty.

OpenTyreF1 was built around exactly these requirements. Users can open any replayable race, build a tyre strategy, run it against what really happened, and receive an answer grounded in real lap data instead of generated text. The rest of this paper describes the architecture, the modules, the evaluation protocol and the measured results.

The key objectives of this research are:

To integrate heterogeneous public Formula 1 data sources into a single, reproducible relational corpus with idempotent, rate-limit-safe ingestion.

- 1) To define one shared clean-lap filter so that model training, published analytics and the mobile app always agree on what a representative lap is.
- 2) To train a lap-time prediction model and evaluate it with a leakage-free protocol that holds out whole races rather than random laps.
- 3) To build a counterfactual strategy simulator that converts predicted lap times into a finishing position against the real field.
- 4) To quantify uncertainty with Monte Carlo simulation and provide bounded pit-wall decision modules backed by historical base rates.
- 5) To deliver all of this through a fast, cross-platform mobile application with a grounded natural-language assistant.

II. RELATED WORK

Race-strategy simulation for Formula 1 was formalised by Bekker and Lotz [1], who modelled a race as a sequence of stochastic events and compared pit strategies by repeated discrete-event simulation. Heilmeyer et al. [2] developed a lap-wise race simulation that treats fuel mass, tyre degradation and pit-lane loss as separable additive terms, and later replaced hand-tuned degradation curves with a neural “virtual strategy engineer” trained on historical laps [3]. Liu and Fotouhi [4] applied Monte Carlo tree search to Formula-E strategy. These simulators are generative: they model every car in the field, which allows multi-agent what-if analysis but makes the output sensitive to every rival’s parameters. OpenTyreF1 is deliberately semi-parametric: it predicts only the subject driver’s laps and holds rivals at their observed lap times, anchoring each counterfactual to a real race.

For the learner, gradient-boosted decision trees [5] and their XGBoost implementation [6] remain the standard for heterogeneous tabular regression, and systematic comparisons continue to show tree ensembles matching or beating deep networks on tabular data of this scale [7]. The hazard of evaluating dependent samples with independent-sample methods is well documented: Kaufman et al. [8] catalogue leakage in data mining, and Roberts et al. [9] formalise blocked cross-validation for structured data. Our whole-race split is the motorsport analogue, with the race as the block.

On the data side, the Ergast API [10] and its successor Jolpica-F1 [11] supply historical results, OpenF1 [12] exposes live timing from 2023, and FastF1 [13] reconstructs laps from the official timing service back to 2018. To our knowledge, no published system reconciles all three into one normalised corpus and serves counterfactual strategy analysis on a phone. Existing tools either stop at visualisation or are proprietary team software.

Table 1 summarises the studies discussed above, with the method each used, its main finding and the gap it leaves for this work. Table 2 compares OpenTyreF1 with the three families of existing tools: descriptive fan tools, academic simulators and proprietary team software. Only OpenTyreF1 combines open availability, reconciled sources, a leakage-free evaluation, an uncertainty estimate, grounded answers and a mobile client.

III. SYSTEM ARCHITECTURE

OpenTyreF1 follows a modular client–server architecture designed for reproducibility, low running cost and fast response on a mobile network. At a high level it consists of three layers (Fig. 1):

- 1) User Interface Layer (Frontend): An Expo SDK 57 / React Native 0.86 application with four tabs (Live, Drivers, Strategy and Assistant) plus race, driver and strategy detail screens. Server state is held in a TanStack Query cache, so screens render instantly from cache and refresh in the background. Charts, track maps and telemetry traces are drawn with react-native-svg.
- 2) Application Layer (Backend): A FastAPI service exposing 36 REST endpoints. It hosts the shared clean-lap filter, the XGBoost predictor, the strategy simulator, the pit-wall modules and the answer engine. The phone never calls an upstream API directly; all third-party traffic is funnelled through the backend’s cache so that OpenF1’s free-tier limit of about three requests per second is never exceeded.
- 3) Data Layer (Database & External APIs): A 15-table relational store, portable between SQLite (shipped inside the repository, about 27 MB) and managed PostgreSQL through one SQLAlchemy layer. Car telemetry and track coordinates are fetched on demand through a TTL cache instead of being bulk-stored, because a single race is millions of rows.

Fig. 2 shows the workflow for the system’s central feature. The user selects a race, a driver and a lap; the backend loads that

TABLE 1 LITERATURE REVIEW

Ref.	Year	Methodology	Key finding	Limitation
[1]	2009	Stochastic discrete-event race simulation	Pit strategies compared by repeated simulation	Hand-set parameters; no learned lap model
[2]	2018	Lap-wise simulation with fuel, tyre and pit terms	Separable terms give interpretable lap times	Degradation curves tuned by hand
[3]	2020	Neural networks trained on historical laps	Learns strategy decisions from data	Needs team-grade data
[4]	2020	Neural networks with Monte Carlo tree search	Search finds energy-optimal strategies	Formula-E specific; heavy compute
[5]	2001	Gradient boosting of regression trees	Stage-wise trees give a strong tabular learner	Sensitive to tuning; prone to overfit

[6]	2016	Regularised gradient-boosted trees	State-of-the-art tabular accuracy, fast training	Feature importance hard to interpret
[7]	2022	Benchmark of trees vs. deep networks	Trees win on medium-sized tabular data	Excludes time-series structure
[8]	2011	Formal analysis of data leakage	Leakage inflates reported accuracy	General; not domain specific
[9]	2017	Blocked cross-validation	Blocking gives honest error estimates	Larger, more pessimistic error bars
[10]	n.d.	Public REST API for F1 results	Historical results, grids and standings	Discontinued; no lap-level timing
[11]	n.d.	Ergast-compatible open API	Continues Ergast data and schema	Results only; some calendar gaps
[12]	n.d.	Open live and historical timing API	Laps, stints, pits, weather, race control	Covers 2023 onwards; rate-limited
[13]	n.d.	Python parser of the official timing feed	Reconstructs laps and compounds from 2018	Slow bulk loads; heavy dependencies
[14]	2017	Shapley-value feature attribution (SHAP)	Consistent, additive explanations	Costly; correlated features share credit
[15]	1993	Bootstrap resampling	Distribution-free uncertainty estimates	Assumes independent samples
[16]	2011	Machine-learning library in Python	Uniform API for models and metrics	General-purpose; not domain specific

TABLE 2 COMPARISON OF EXISTING APPROACHES

Capability	Fan tools	Academic	Team	OpenTyreF1
Open / freely available	Yes	Partly	No	Yes
Multiple sources reconciled	Rarely	No	Yes	Yes (3)
Counterfactual simulation	No	Yes	Yes	Yes
Leakage-free evaluation	N/A	Varies	Unknown	Whole-race
Uncertainty estimate	No	Some	Yes	Monte Carlo
Grounded answers	No	No	No	Yes
Mobile application	Some	No	Internal	Android / iOS

race's laps, stints, pit stops and weather and passes them through the clean-lap filter. The user then builds a stint plan. The XGBoost model predicts every lap of the plan, laps run behind a safety car are clamped to the field's observed pace, and a pit loss measured at that circuit is charged for each stop. The resulting cumulative time is ranked against every rival's real cumulative time, a Monte Carlo pass estimates the spread of finishing positions, and the app shows the result with a fact-backed explanation.

The functional behaviour of the system is captured by the use case diagram in Fig. 3. The primary actor is the user of the mobile app, who can view the live feed, browse drivers and races, replay a race, compare telemetry, analyse tyre degradation, build and simulate a strategy, use the pit-wall modules and ask the assistant. Two secondary actors support the system: the external Formula 1 APIs that supply the data, and the background sync service that keeps it current.

IV. SYSTEM MODULES

The system is composed of the following modules, each of which maps to a service in the backend.

A. Data Ingestion Module

The three sources have complementary coverage and no shared identifier scheme (Table 3). Races are matched on a normalised (year, location) pair with a curated alias map. A session that matches no race is skipped with a logged reason rather than stored with

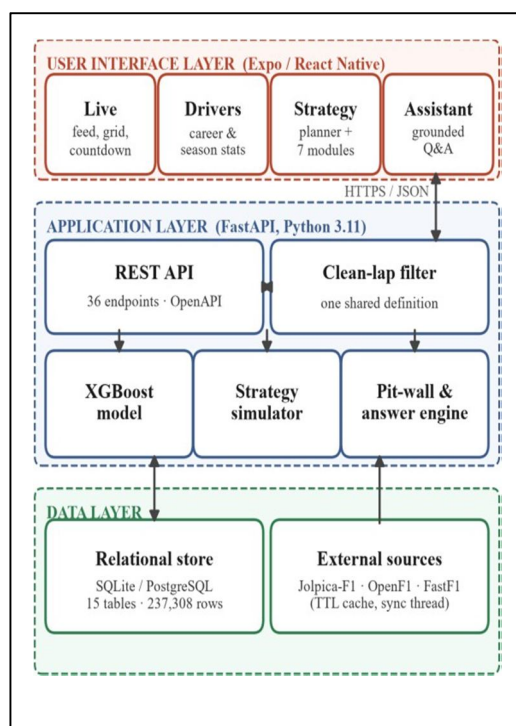


Fig. 1 Layered architecture of OpenTyreF1

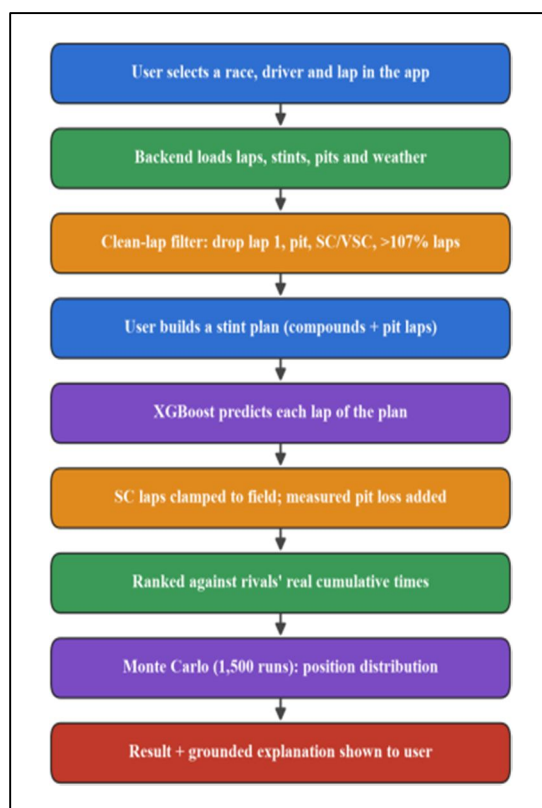


Fig. 2 Flowchart of a strategy-simulation request

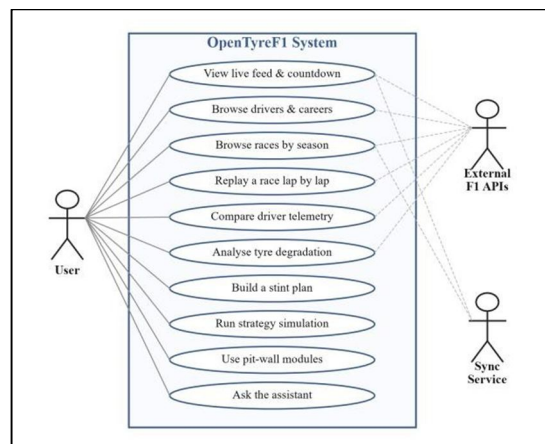


Fig. 3 Use case diagram of OpenTyreF1

a fabricated parent. This rule surfaced a real upstream gap: the Jolpica 2026 calendar omits two rounds that OpenF1 does contain. Every ingestion routine checks for existing data before making a network call, so it is idempotent and resumable under upstream

TABLE 3 DATA SOURCES AND COVERAGE

Source	Seasons	Contributes
Jolpica-F1 / Ergast	2002–2026	Results, grids, standings, calendar
OpenF1	2023–2026	Laps, stints, pits, weather, race control
FastF1	2018–2022	Lap backfill, tyre compounds

rate limits. A background daemon thread keeps the dataset fresh. It re-checks every 15 minutes near a session boundary and every 6 hours -level coverage of the corpus by season is shown in Fig. 4. OpenF1 supplies the 2023 to 2026 seasons, while FastF1 backfills 2018 to 2022, giving between roughly 15,000 and 29,000 lap records per season.

The historical side of the schema is centred on races, drivers and constructors, linked through results and standings. The timing side is centred on sessions: every session belongs to a race and owns its laps, stints, pit stops, weather readings, race-control messages, results, overtakes and the list of drivers who took part. Table 4 summarises the purpose of each table.

B. Clean-Lap Filter Module

This filter decides which laps count as representative racing pace. It is the single highest-leverage component, because it defines the training data, every published pace statistic and every degradation estimate. It is one function imported by the trainer, the analytics services and the API, so they can never disagree. Laps are admitted by four successive exclusions:

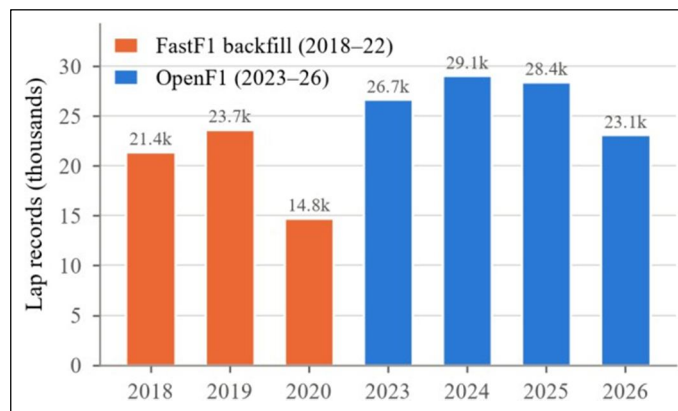


Fig. 4 Lap records by season and source TABLE 4 DATABASE TABLES

Table	Purpose
seasons	One row per championship year
races	Round, name, circuit, date, location
drivers / constructors	Driver and team master data
results / standings	Classification and championship standings
sessions	Race and sprint sessions linked to their race
session_laps	Lap time, sectors, compound, tyre life, position
session_stints	Compound and lap range of each stint
pit_stops	Pit lap and stationary / lane duration
session_weather	Air and track temperature, humidity, rain, wind
race_control_messages	Flags, safety-car and official messages
session_results	Session classification incl. DNF / DNS / DSQ
overtakes	Reconstructed position changes between cars
session_drivers	Number, acronym and team colour per session

- 1) Opening lap: a standing start is not a measure of racing pace.
- 2) Pit-affected laps: out-laps are flagged in the source; in-laps are recovered from the pit-stop table.
- 3) Neutralised laps: safety-car and virtual-safety-car periods are reconstructed from free-text race-control messages with a single-pass state machine.
- 4) Per-driver 107% rule: laps slower than $1.07 \times$ the driver's own clean median are removed. A per-driver threshold avoids over-pruning slower cars.

Neutralised periods are not given as a column in the source data; they must be reconstructed from the free-text race-control messages. The state machine used for this is shown in Fig. 5. Every lap that starts or ends while the track is in the SC, VSC or red-flag state is marked as neutralised.

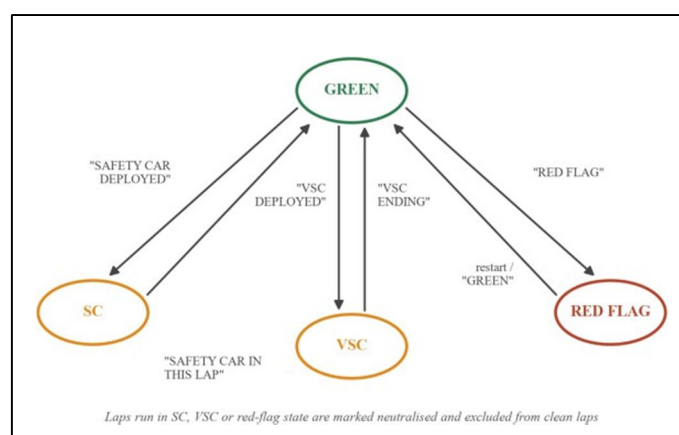


Fig. 5 Track-status state machine

Over 93 ingested race sessions, 58,848 of 93,282 timed laps survive, a retention of 63.1% (Fig. 6).

Fig. 7 shows the effect on a single race: the long tail of slow laps caused by pit stops and safety cars disappears, leaving a compact distribution that reflects true racing pace.

C. Lap-Time Prediction Module

The model predicts the time of a lap from six features: tyre compound, circuit and driver (categorical), and tyre age, lap number and track temperature (numeric). Weather is joined *as of* each lap's start: the lap takes the most recent reading at or before it begins,

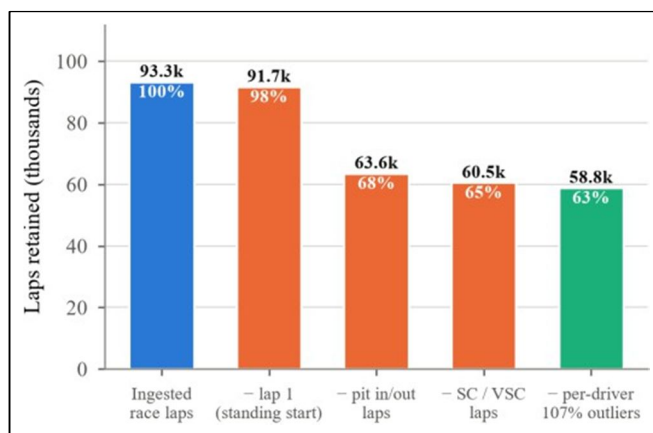


Fig. 6 Clean-lap admission cascade over 93 race sessions

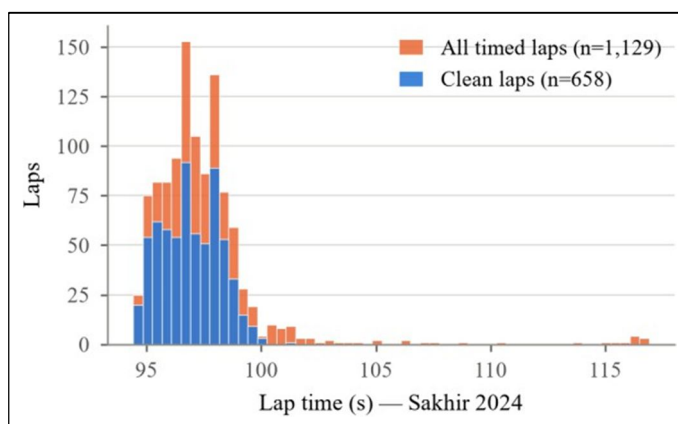


Fig. 7 Lap-time distribution before and after filtering

found by binary search. A lap is never described by a measurement taken after it finished, and the same code path is used in training and simulation.

Rather than regressing the absolute lap time y , the model predicts the offset from the race's median clean lap m_r , which is added back at scoring time:

$$\hat{y}_i = m_r + f(\mathbf{x}_i), \quad f \text{ trained on } y_i - m_r$$

This is legitimate for the simulator, which always operates on a completed historical race whose median is known at prediction time, exactly as in training. It also changes which baseline is fair, as discussed in Section 5. Fig. 8 summarises the data and training pipeline.

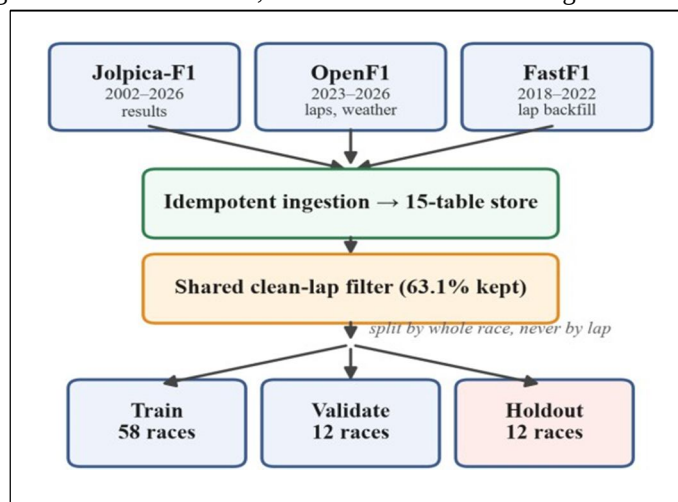


Fig. 8 Data and model-training pipeline

D. Strategy Simulation Module

Given a real race, a real driver and a hypothetical stint plan, the simulator estimates where that driver would have finished. It performs five steps:

- 1) Expand the plan into a per-lap feature sequence over the real race distance.
 - 2) Predict each lap with the trained model.
 - 3) Clamp neutralised laps: if the whole field ran more than 10% slower than reference pace on a lap, the simulated car follows the field's observed time.
 - 4) Charge pit losses measured at that circuit: observed stop durations are clamped to 15–45 s and the median is taken. With fewer than five samples, a declared 22 s default is used and reported.
 - 5) Rank the counterfactual cumulative time against every rival's real cumulative time, lap by lap, to produce a finishing position.
- The interaction between the components during a simulation request is shown in the sequence diagram of Fig. 9. The app sends one request; the backend resolves the session and weather, loads laps, stints and pit samples, builds the per-lap feature matrix, obtains predictions from the model, applies the safety-car clamp and pit losses, and returns frames, deltas and the final position.

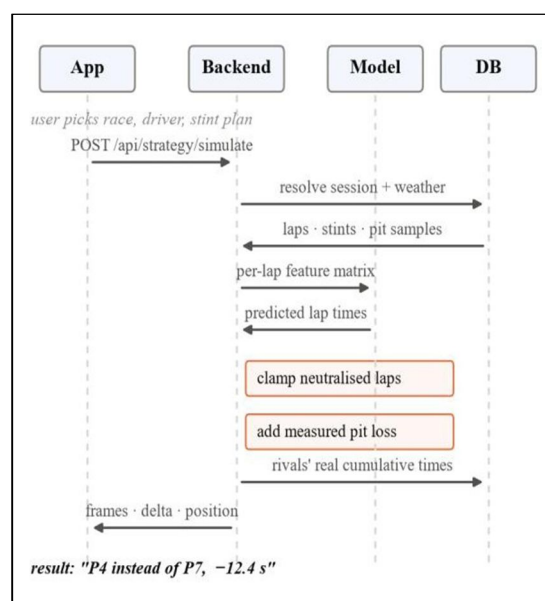


Fig. 9 Sequence diagram of a strategy simulation

If no trained model is available, the simulator falls back to an empirical linear degradation fit with declared per-compound rates (0.085, 0.050 and 0.030 s/lap for soft, medium and hard). Every response carries a *modelSource* field stating which path produced the numbers.

E. Monte Carlo Outcome Module

A single simulated position conveys no uncertainty, so the outcome module runs 1,500 bootstrap [15] replications from a chosen lap. Laps already completed are held fixed. Only the remainder of the race is resampled from the driver's clean laps, perturbed by the model's published error, charged once for each remaining pit stop and exposed to historical safety-car and retirement rates per lap. The output is a probability distribution over finishing positions.

F. Pit-Wall Decision Modules

Seven modules answer bounded in-race questions from historical base rates (Table 6). All share one response format: a verdict, a confidence, an expected gain, the reasoning, the supporting facts and the sample size behind them. A recommendation derived from 13 observations is therefore never presented with the same authority as one backed by thousands.

TABLE 5 PIT-WALL STRATEGY MODULES

Module	Basis
Safety Car	Pit vs. stay-out outcomes under SC/VSC
Red/Yellow Flag	Reconstructed flag periods and outcomes
Weather	Inter/wet crossover base rates
Overtake	Pass rate by gap bucket (<0.5 s ... >1.5 s)
Defence	The same swap data, from the defender's side
Position Prediction	Monte Carlo position distribution
Risk Analysis	Downside summary of the same runs

G. Grounded Assistant Module

The assistant is a retrieval-and-template engine, not a generative language model. A question is scored against keyword sets for each intent, entities such as drivers and races are resolved from the text or the app's context, and a per-intent renderer turns database rows into prose together with the list of facts used. Because it can only state what retrieval returned, it cannot produce a fluent claim that no data supports. An unanswerable question returns an explicit statement and the list of questions it can answer.

H. REST Interface

The backend exposes 36 REST endpoints grouped into routers for health, drivers, races, sessions, telemetry, strategy, live data and the assistant. Request bodies are validated with Pydantic models, so an unknown tyre compound is rejected with HTTP 400 or 422 and an unknown driver with HTTP 404 rather than guessed. Table 5 lists the main endpoints.

TABLE 6 MAIN REST API ENDPOINTS

Method	Endpoint	Purpose
GET	/api/live/feed	Latest result or next race
GET	/api/drivers/{id}	Driver career statistics
GET	/api/races/{id}/replay	Lap-by-lap positions
GET	/api/races/{id}/analysis	Pace, degradation, gaps
GET	/api/telemetry/compare	Two-driver telemetry
GET	/api/strategy/options/{id}	Drivers, distance, pit loss
POST	/api/strategy/simulate	Run a stint plan
POST	/api/ai/query	Grounded assistant

I. Key Features

- Live feed with next-race countdown, starting grid and latest results
- Circuit dossier traced from real telemetry coordinates
- Driver career and season-by-season statistics
- Race replay, pace charts and fuel-corrected tyre degradation
- Two-driver speed, throttle, brake and DRS telemetry comparison
- Stint planner with counterfactual simulation and seven pit-wall modules
- Grounded assistant whose every answer cites its data

V. METHODOLOGY

The project followed a structured, iterative methodology:

- Requirement analysis: identify the questions a strategist asks and the public data able to answer them.
- Data engineering: build ingestion, schema and identifier reconciliation; verify row counts against each source.
- Model development: define the clean-lap filter, the features, the target parameterisation and the evaluation protocol.
- System implementation: develop the FastAPI services and the React Native client against a typed API contract.
- Testing: verify endpoints, sync behaviour and database migration (all 15 tables copied to PostgreSQL with zero row-count mismatches).
- Deployment: containerise the backend with the dataset baked into the image and build the app with Expo EAS.

Testing covered unit, integration, data-validation, model-evaluation, performance and usability levels. Functional testing used an end-to-end verification script that drives the real API against the real ingested database, because the failures that matter, such as a race with no pit data or a driver who retired, only appear with real data. Representative cases are listed in Table 8.

TABLE 7 REPRESENTATIVE TEST CASES

ID	Test case	Expected result
TC-01	GET /api/health	HTTP 200 with database status
TC-02	Model metadata	Beats baseline; whole-race splits
TC-03	Simulate the real plan	Close to the real race time
TC-04	Simulation frames	Positions 1..n, leader gap zero
TC-05	Cumulative time	Increases monotonically
TC-06	Unknown compound	Rejected with HTTP 400 / 422
TC-07	Unknown driver	Rejected with HTTP 404
TC-08	Ungrounded question	Refused, with answerable list
TC-09	Answer sources	Every answer cites its sources

A. Evaluation Protocol

Laps from the same race share a car, a fuel load, a track surface and a weather trace, so a random lap-level split leaks most of the relevant structure and overstates accuracy. Whole races are therefore assigned to train, validation and holdout sets in a 70/15/15 seeded split: 58, 12 and 12 races, covering 40,012 clean laps from 2023–2026. Early stopping monitors the validation races only, and the holdout is scored exactly once. Sprint sessions are excluded because their lighter fuel load would teach a fuel effect that describes neither format.

Two naive baselines are scored on the same holdout. **B1** predicts each driver’s median lap at that circuit. **B2** predicts the race’s own median clean lap. B2 is the matched baseline for the relative target, and beating it is the only evidence that the model has learned anything about tyres, fuel or conditions. Accuracy is computed with scikit-learn [16] as:

$$MAE = \frac{1}{n} \sum |y_i - \hat{y}_i|, \quad RMSE = \sqrt{\frac{1}{n} \sum (y_i - \hat{y}_i)^2}$$

TABLE 8 XGBOOST CONFIGURATION

Parameter	Value	Parameter	Value
n_estimators	1200	subsample	0.85
learning_rate	0.05	colsample_bytree	0.85
max_depth	6	reg_lambda	1.5
min_child_weight	4	early stopping	50

VI. RESULTS AND DISCUSSION

A. Prediction Accuracy

Table 9 and Fig. 10 report holdout performance on the 12 unseen races. The model attains 0.893 s MAE and $R^2 = 0.978$, improving on the matched baseline B2 by 0.270 s MAE. B1, which knows circuit and driver identity but nothing about within-race evolution, still reaches $R^2 = 0.545$. This confirms that circuit identity alone explains much of the pooled variance, which is why a meaningful baseline must already encode it.

TABLE 9 HOLDOUT PERFORMANCE (12 UNSEEN RACES)

Predictor	MAE (s)	RMSE (s)	R^2
XGBoost (proposed)	0.893	1.205	0.978
B2: race median lap	1.163	1.494	0.965
B1: driver × circuit median	4.096	5.420	0.545

B. Target Design versus Feature Richness

Fig. 11 crosses three feature sets with both targets. With the feature set fixed, switching from the absolute to the relative target cuts MAE from 3.586 s to 0.893 s, a factor of four. With the target fixed, almost doubling the feature count from 6 to 11 improves MAE by only 0.033 s. That is inside our declared parsimony tolerance of 0.05 s, so the simpler six-feature model is the one shipped. Early stopping chose iteration 16 of a permitted 1,200, which suggests that once the target is well parameterised the problem is close to linear in these features.

The practical consequence is that further accuracy is more likely to come from better data than from more features: correcting the pit-lap rule, as outlined in Section 7, would recover about 22,000 usable laps, whereas the five extra features moved MAE by only 0.033 s. Keeping the model at six features also keeps the simulator simple, because every feature must be known for every hypothetical lap of a stint plan. Gain importance is led by lap number (0.422), which acts largely as a fuel-mass proxy, followed by driver (0.185), circuit (0.122), track temperature (0.102), compound (0.091) and tyre age (0.078). Tyre age ranks low because it advances in lockstep with

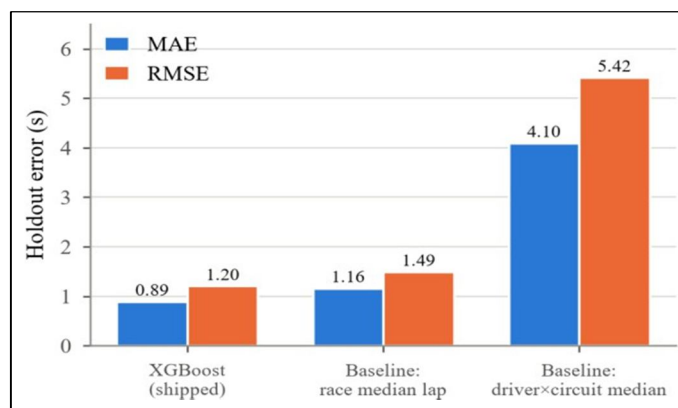


Fig. 10 Holdout error of the model against both baselines

lap number within a stint, and gain attribution splits credit between correlated features arbitrarily. It should not be read as tyres being unimportant, as the next result shows.

C. Fuel-Corrected Tyre Degradation

Fig. 13 shows how lap records are spread across tyre compounds. Medium and hard tyres dominate race running, soft tyres account for most of the rest, and the 2018 compound names appear only in the backfilled seasons. Wet-weather compounds are rare, which limits what can be learned about them. Raw lap-time-versus-tyre-age slopes mix tyre wear with fuel burn. We therefore subtract a per-session fuel slope, fitted by least squares on lap number, from each driver-stint's tyre-age slope. Aggregated over the corpus (Fig. 14), soft tyres degrade fastest at a median 0.057 s per lap, against 0.037 s for hard and 0.034 s for medium. Medium and hard are statistically indistinguishable here. Intermediates show -0.114 s per lap: a drying track gets faster faster than the tyre wears, which marks where the linear estimator stops being valid.

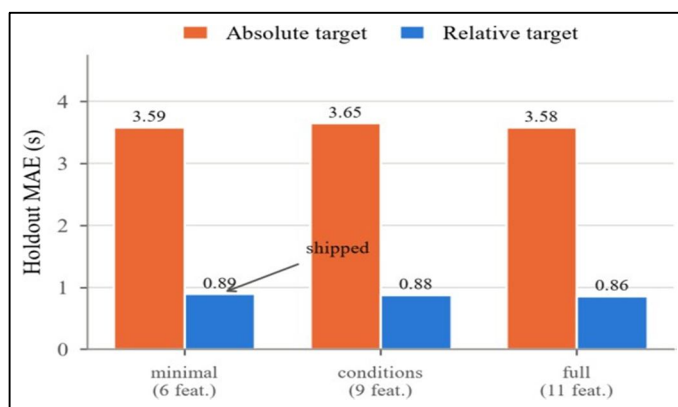


Fig. 11 Ablation: target parameterisation vs. feature set

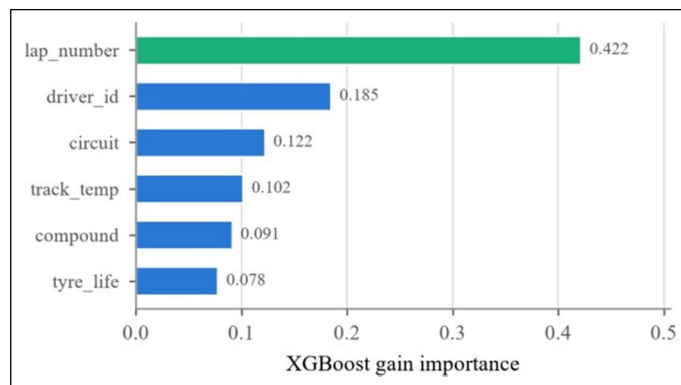


Fig. 12 XGBoost gain importance of the six features

D. System Performance

The production container uses about 204 MB of resident memory with the model loaded, which fits a 512 MB free-tier host. The serving image omits FastF1 and its scientific stack, so no ingestion code can run on the request path. A full background sync pass completes in seconds, is idempotent across repeated runs and caused no rate-limit hits after the catalogue and harvest steps

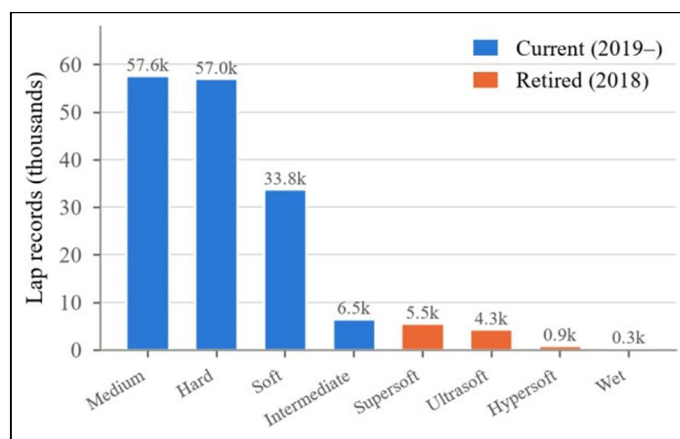


Fig. 13 Lap records by tyre compound

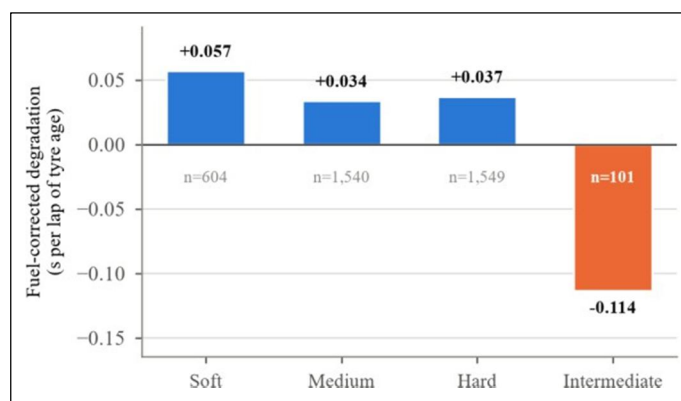


Fig. 14 Median fuel-corrected degradation by compound

were separated. The app renders screens from its query cache immediately and refreshes in the background.

E. Application Screenshots

Fig. 15 shows four screens of the mobile application: the live feed, the driver list, the tyre-degradation analysis and the strategy planner.

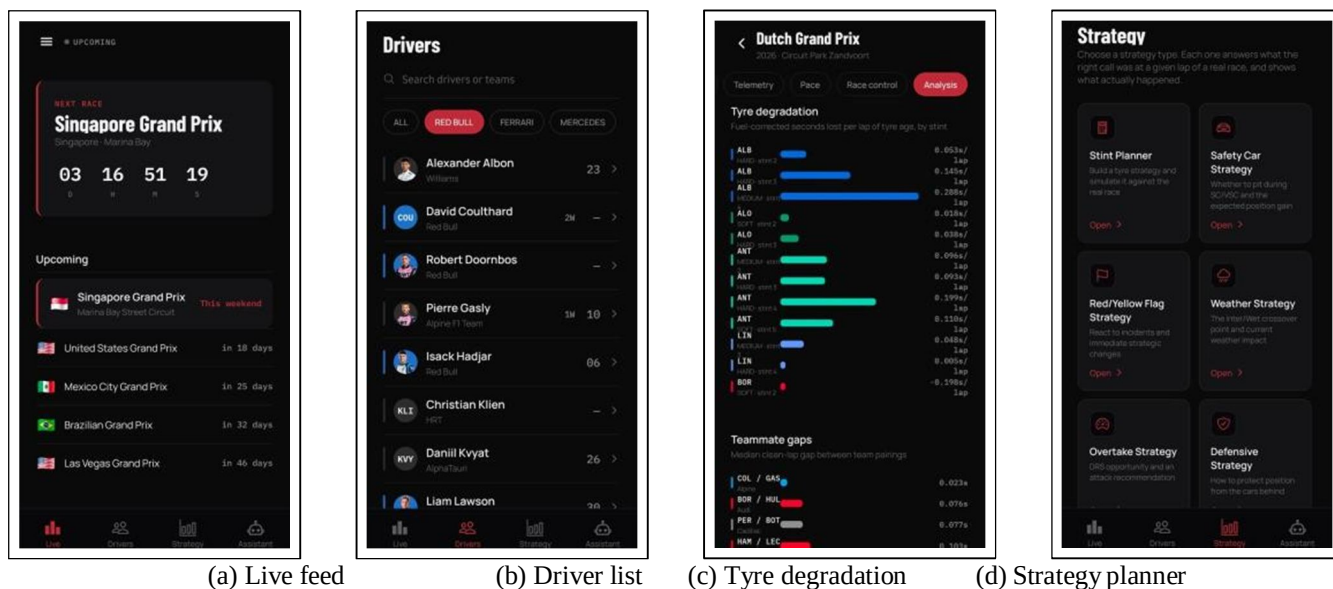


Fig. 15 Screenshots of the OpenTyreF1 mobile application

F. Discussion

A random lap-level split would have reported a much better, and misleading, score; the whole-race holdout gives an estimate that users can rely on. The system also shows that grounding and uncertainty can be built into a consumer product: every simulation states which model produced it, every pit-wall module reports its sample size, and the assistant cites its facts. Table 10 maps each research objective to the outcome achieved.

TABLE 10 OBJECTIVES AND ACHIEVED OUTCOMES

Objective	Outcome
Integrate public F1 sources	237,308 records, 492 races, 15 tables
One shared clean-lap filter	Used by trainer, analytics and API; 63.1% retained
Leakage-free lap-time model	Whole-race split; 0.893 s MAE, R^2 0.978
Counterfactual simulator	Ranked against the real field, declared fallbacks
Uncertainty and decisions	1,500-run Monte Carlo, seven modules
Mobile app with assistant	Four tabs, assistant citing its facts

VII. CONCLUSION AND FUTURE WORK

This paper presented OpenTyreF1, an end-to-end system that integrates three heterogeneous public Formula 1 data sources into a single 237,308-record corpus and builds on it a lap-time model, a counterfactual strategy simulator, a Monte Carlo outcome estimator, seven pit-wall decision modules and a grounded assistant, delivered through a cross-platform mobile app.

Our central finding is methodological. On data of this size and shape, the decisive choices are not about the learner but about which laps are admitted, how the target is parameterised and how evaluation is partitioned. Reparameterising the target improved MAE four-fold, while nearly doubling the feature count moved it by only 0.033 s. Under a strict whole-race holdout, the model reaches 0.893 s MAE and beats the strongest naive baseline by 0.270 s.

Beyond motorsport, the same discipline applies to any product that turns open, noisy event data into recommendations: one shared data definition, a grouped holdout, a matched baseline, declared fallbacks and answers that cite their evidence. OpenTyreF1 shows that these properties can be delivered on free-tier infrastructure, with a 204 MB container and a database small enough to ship inside the repository, so the whole system can be reproduced and audited by anyone with access to the public data.

Future work includes correcting the pit-lap rule to work per (driver, lap) and retraining, which would recover about 38% more training data; using Shapley-value attribution [14] to separate tyre-age effects from the fuel-mass proxy; and completing the 2021–2022 backfill so the model can be evaluated across a regulation change. We also plan to estimate the race median online so that the accurate relative model can support live races, and to add a multi-agent field simulation and an energy-deployment (ERS) estimate.

In summary, the main contributions of this work are:

- A reproducible, idempotent ingestion pipeline that reconciles Jolpica-F1, OpenF1 and FastF1 into one normalised 15-table corpus that can run on either SQLite or PostgreSQL.
- A single shared clean-lap filter, imported by the trainer, the analytics services and the API, so that every published pace figure and every model input follows the same definition of a representative lap.
- A leakage-free evaluation protocol based on whole-race splits and a matched baseline, which gives an honest estimate of how the model performs on races it has never seen.
- A semi-parametric counterfactual simulator that anchors every hypothetical strategy to a real race and to the real lap times of every rival.
- Uncertainty-aware decision support through Monte Carlo outcome estimation and seven pit-wall modules that always report the sample size behind each recommendation.
- A grounded assistant and a cross-platform mobile client that make this analysis available to fans, students and analysts without proprietary team data.

Taken together, these results suggest that useful race-strategy analysis no longer depends on access to private team telemetry. With careful data handling and transparent evaluation, openly published Formula 1 data is sufficient to answer counterfactual strategy questions with measurable accuracy and a clearly stated level of confidence.

The same architecture also opens several practical applications. For students and educators, OpenTyreF1 provides a complete, reproducible case study in applied machine learning, in which data ingestion, feature design, leakage-free evaluation and deployment can all be inspected on real public data. For motorsport journalists and commentators, the simulator offers a quick way to test strategy claims made during a broadcast against what the data actually supports. For fans, the mobile client turns post-race discussion from opinion into evidence, since every answer is tied to the laps, stints and pit stops that produced it. For smaller racing teams in junior categories, which rarely have dedicated strategy staff, the same workflow of a shared clean-lap definition, a matched baseline and a declared uncertainty can be adopted with modest engineering effort.

Several extensions are planned beyond those listed above. The first is live prediction: the relative target currently needs the race median, so an online estimate built from the opening clean laps of a race and refined as the race progresses would allow the more accurate relative model to be used during live sessions. The second is a multi-agent simulation in which rivals respond to a changed strategy, for example by covering an early pit stop, so that counterfactual positions account for the reactions of the field rather than holding every rival fixed. The third is broader coverage: once the backfilled seasons are fully usable, the model can be retrained across regulation changes and evaluated on how well it transfers from one technical era to the next. Finally, a structured user study with fans and student analysts is planned to measure how well the explanations, sample sizes and uncertainty ranges shown in the app are understood, and whether they improve the quality of strategy judgements made with the tool.

Building the system also produced a number of practical lessons that are likely to carry over to similar projects. The most important is that data definitions should be written once and imported everywhere: when the trainer, the analytics services and the API each kept their own idea of a clean lap, their numbers drifted apart, and a single shared filter removed that whole class of inconsistency. The second is that a baseline is only meaningful when it is matched to the target, because a model that predicts offsets from the race median must be compared against the race median itself, not against a weaker global average that it will beat easily. The third is that every fallback should be declared in the output rather than hidden, so that a user can always tell whether a number came from the trained model, from an empirical fit or from a default value. The fourth is that rate limits and data embargoes are design constraints, not afterthoughts, and are best handled by an idempotent ingestion layer that can be stopped and resumed at any point without duplicating or losing records.

Finally, OpenTyreF1 is intended to remain open and reproducible. The relational store is small enough to ship with the source code, the serving container runs on free-tier infrastructure, and every figure reported in this paper can be regenerated from the public data sources using the same ingestion, filtering and evaluation code. We hope this makes the system useful not only as a fan and analyst tool but also as a transparent reference implementation for others who wish to build, test and improve data-driven race-strategy methods on openly available Formula 1 data.

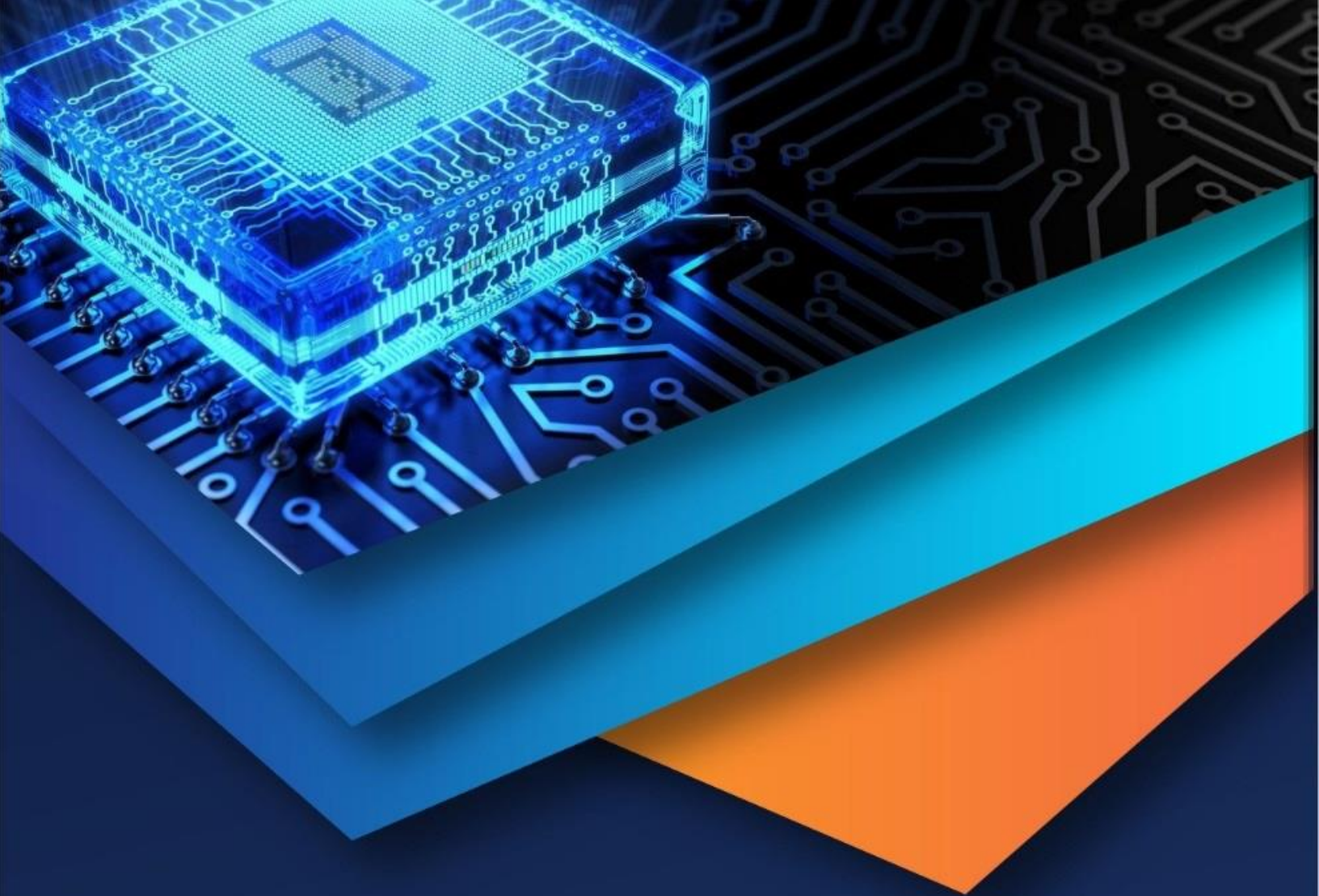
VIII. ACKNOWLEDGMENT

The authors thank the Department of Computer Science & Engineering, G H Raisoni University, Amravati, for its guidance and support throughout this project. The authors also thank the maintainers of the Jolpica-F1, OpenF1 and FastF1 open-source projects, whose freely available data and tools made this work possible. We are grateful to the faculty members and reviewers whose feedback during project evaluations helped refine the system design, the clean-lap definition and the evaluation protocol. We further acknowledge the developers of XGBoost, scikit-learn, FastAPI, SQLAlchemy, Expo and React Native, on whose open-source tools the entire system is built, and the wider Formula 1 data community, whose documentation, discussions and shared analyses helped us understand the structure and limitations of the public timing data.

Finally, the authors thank their families and friends for their patience, encouragement and constant support throughout the development of this project. Their understanding during long hours of data collection, model training and testing made it possible to complete this work. We also thank our classmates, who tried early versions of the mobile application and shared honest feedback on its screens, explanations and features, which helped make the final system clearer, more reliable and easier to use.

REFERENCES

- [1] J. Bekker and W. Lotz, "Planning Formula One race strategies using discrete-event simulation," *Journal of the Operational Research Society*, vol. 60, no. 7, pp. 952–961, 2009.
- [2] A. Heilmeier, M. Graf and M. Lienkamp, "A race simulation for strategy decisions in circuit motorsports," in *Proc. 21st IEEE Int. Conf. Intelligent Transportation Systems (ITSC)*, 2018, pp. 2986–2993.
- [3] A. Heilmeier, A. Thomaser, M. Graf and J. Betz, "Virtual strategy engineer: Using artificial neural networks for making race strategy decisions in circuit motorsport," *Applied Sciences*, vol. 10, no. 21, art. 7805, 2020.
- [4] X. Liu and A. Fotouhi, "Formula-E race strategy development using artificial neural networks and Monte Carlo tree search," *Neural Computing and Applications*, vol. 32, pp. 15191–15207, 2020.
- [5] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [7] L. Grinsztajn, E. Oyallon and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Advances in Neural Information Processing Systems*, vol. 35, 2022.
- [8] S. Kaufman, S. Rosset and C. Perlich, "Leakage in data mining: Formulation, detection, and avoidance," in *Proc. 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2011, pp. 556–563.
- [9] D. R. Roberts et al., "Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure," *Ecography*, vol. 40, no. 8, pp. 913–929, 2017.
- [10] C. Newell, "Ergast Developer API: A web service for Formula One motor racing data." Available: <http://ergast.com/mrd/>
- [11] Jolpica-F1, "An Ergast-compatible API for Formula 1 data." Available: <https://github.com/jolpica/jolpica-f1>
- [12] OpenF1, "Open-source API for real-time and historical Formula 1 data." Available: <https://openf1.org/>
- [13] P. Oehrly, "FastF1: A Python package for accessing and analysing Formula 1 results, schedules, timing data and telemetry." Available: <https://github.com/theOehrly/Fast-F1>
- [14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 4765–4774.
- [15] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. New York, NY, USA: Chapman & Hall, 1993.
- [16] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)