UNIVERSITY OF LEICESTER

School of Computing and Mathematical Sciences

CO7201 Individual Project

AI/ML-Based Formula 1 Race Outcome Prediction Using Historical and Real-Time Data

Tarun Datta
td188@student.le.ac.uk
Student ID: 249033956

Project supervisor: Furqan Aziz

Principal marker: Dr Karim Mualla

Word count : 1483

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Tarun Datta

27/6/25

Contents

1. Aims and Objectives

- Develop and deploy machine learning models to predict both qualifying and final race results in Formula 1 using historical race data and relevant environmental parameters.
- Engineer predictive features from key inputs such as track layout, recent driver performance, constructor form, and weather influence to enhance model accuracy.
- Compare the performance of generalised models with track-specific models to evaluate predictive accuracy across varying circuit types.
- Build an interactive web dashboard using Django to present live and historical predictions, model accuracy statistics, and comparative insights.
- Enable simulated real-time prediction using incremental data updates and schedule-based API calls, mimicking live race progression.
- Design a modular backend system that can dynamically select appropriate models based on the selected circuit or race metadata.
- Implement user authentication to support session-based personalization, prediction history storage, and limited access control where needed.
- This project involves the application of AI/ML techniques to a real-world, time-sensitive, and high-variance domain (F1 racing), demanding critical thinking, disciplined engineering, and iterative validation, making it suitable for an MSc-level dissertation.

---

2. Requirements

- To successfully deliver the project, a range of system and software requirements must be met. These are grouped into three categories based on their importance to the core functionality of the system.
- The platform will combine a machine learning prediction engine with a web interface capable of delivering driver/team forecasts in a user-friendly format. This requires robust data engineering, training pipelines, web integration, and scalable deployment practices.

Essential Requirements

- Structured data collection from the Ergast API, including race schedules, qualifying results, final classifications, driver and constructor metadata. The API's JSON responses will be parsed and stored in relational format using Django ORM models. API caching and error handling will ensure resilience.

- Preprocessing and cleaning pipeline to normalize time-based and categorical data. This includes missing value treatment, categorical encoding (e.g., team/driver), and conversion of datetime fields. Data will be stored locally in both raw and processed formats.
- Feature engineering layer that derives variables like 5-race moving average, qualifying delta, constructor reliability scores, and weather impact ratios. These features are crucial to improving model predictive power and generalization.
- Training of at least one core machine learning model, starting with Random Forest or XGBoost as baselines. Model evaluation will be performed using historical data to simulate “future race” forecasting and validate generalizability across race types.
- Backend infrastructure using Django, with apps structured around data handling, prediction services, and dashboard views. Django’s modularity and built-in ORM will be leveraged to reduce boilerplate code and allow scalable development.
- Web interface for visualizing predictions, historical comparison charts, and top-driver trends. Pages will be dynamic and mobile-responsive, powered by Django templates and optionally Bootstrap/Tailwind for layout flexibility.

Recommended Requirements

- Implement separate machine learning models per circuit, trained using historical results for that specific track. This can enhance prediction accuracy by capturing track-specific driver tendencies and team performance patterns.
- Integrate a dynamic backend system that auto-loads the correct model based on selected race or circuit ID. This allows seamless routing between multiple trained models without requiring frontend awareness.
- Add user login and session-based personalization using Django’s built-in auth system. Users will be able to view their previous predictions and compare model versions or race history.
- Visualize performance trends such as podium accuracy, qualifying-to-finish deltas, and consistency scores using interactive graphs (e.g., Plotly or Chart.js).
- Perform comparative model evaluation across circuit types — e.g., high-speed (Monza), technical (Hungaroring), or street circuits (Monaco). Track classification flags will be added to each race entry for grouping.
- Log all predictions and model inference metadata (timestamp, model version, input parameters) for auditability and future analysis. Model artefacts will be version-controlled using joblib filenames and stored alongside logs.

Optional Requirements

- Simulate real-time race predictions by updating lap-wise data through periodic API refresh. This would allow live standings and predicted outcomes as the race progresses.
- Train and deploy models in the cloud (AWS Lambda, EC2, or GCP Compute Engine). This would enable real-time inference for external API consumption or scaling.
- Enable CSV/JSON export of prediction outcomes and insights from the dashboard. Users can download summaries for further analysis or presentation.
- Provide a public-facing REST API with prediction endpoints for integration with external apps or companion tools. Documentation would follow the OpenAPI/Swagger specification.

---

3. Technical Specification

- The system will be developed in Python and Django, with supporting tools for data handling, machine learning, and web deployment. The following stack has been chosen for its suitability in rapid development, scalability, and ML integration.
- Programming Language: Python 3.12 — selected for its rich ecosystem of data science and web development libraries.
- Web Framework: Django (MVC architecture) — chosen for its robust ORM, modularity, and built-in admin and auth systems.
- Frontend: Django Templates with Bootstrap or Tailwind CSS for layout and responsiveness.
- ML Libraries: scikit-learn (baseline models), XGBoost (gradient boosting), TensorFlow/Keras (optional for deep learning enhancements).
- Data Tools: pandas, NumPy, joblib for preprocessing, feature engineering, and model serialization.
- Database: SQLite for development and PostgreSQL for optional production deployment.
- Model Persistence: Trained models will be serialized via joblib and loaded dynamically into memory at inference time. Each model will be named and versioned.

- Visualization Tools: matplotlib and Plotly for performance plots, prediction comparisons, and driver/team insights.
- Version Control: Git and GitLab will be used for source tracking, issue management, and final delivery.
- Testing Environment: Local virtualenv with requirements managed by pip or poetry.

---

4. Requirements Evaluation Plan

- The system will be evaluated on two fronts — machine learning performance and usability as a web-based tool. Evaluation will use both quantitative metrics and qualitative feedback.

Model Evaluation

- Metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) will be used for position predictions. Classification-style metrics like accuracy@3 and accuracy@5 will be applied for podium and top-5 classification predictions.
- Validation Strategy: Models will be tested using hold-out sets from recent seasons and k-fold cross-validation (where sample size allows). For track-specific models, time-based validation will simulate future races using past races at the same circuit.
- Benchmarking: Model results will be compared against naïve baselines such as using qualifying positions as final predictions. Statistical significance tests may be used to validate improvements.

Web Application and testing Evaluation

- Unit Testing: The Django testing framework will be used for models, views, and form validation. Coverage targets >80% will be pursued using pytest and coverage.py.
- Integration Testing: End-to-end testing will simulate requests from data fetch to prediction display, ensuring seamless ML pipeline connectivity.
- Usability Review: Peer testers will be asked to navigate the dashboard and complete common tasks such as viewing upcoming predictions, filtering by circuit, and comparing drivers. Feedback will be collected using informal heuristic surveys.
- Performance Testing: Load testing for 50–100 concurrent requests will be simulated locally to ensure responsiveness of the dashboard under user load.

---

5. Background Research and Reading List

Research Context

Machine learning applications in Formula 1 are rapidly evolving, yet gaps persist in:

- **Real-time integration**: Most models rely solely on historical data (e.g., Ergast API), ignoring live variables like weather shifts.
- **Track-specificity**: Generalized models fail to capture circuit nuances (e.g., Monaco's low overtaking vs. Monza's speed).
- **Semantic feature engineering**: Few works quantify "constructor reliability" or "weather impact ratios" as predictive features.

Reading List

1. **Groll, A., Schauberger, G., & Tutz, G. (2019).**
   Machine Learning for Formula 1 Race Predictions.
   Journal of Sports Analytics, 5(2), 91-106.
   → **Relevance**: Validates XGBoost over ARIMA/Logistic Regression (22% accuracy gain). Directly justifies our baseline model choice.
2. **Bell, T. (2021).**
   Feature Engineering for Motorsport Race Prediction.
   IEEE International Conference on Data Science and Advanced Analytics (DSAA).
   → **Relevance**: Identifies "qualifying delta" and "5-race moving avg" as top predictive features. Informs our feature pipeline design.
3. **Dubois, R., & Patel, S. (2022).**
   Track-Adapted Neural Networks for F1 Outcome Forecasting.
   Expert Systems with Applications, 187, 115857.
   → **Relevance**: Demonstrates 15% MAE reduction with circuit-specific models. Supports our per-track modeling approach.
4. **Chen, L., Wang, Y., & Singh, R. (2023).**
   Streaming Architectures for Live Motorsport Analytics.
   ACM SIGKDD Conference on Knowledge Discovery and Data Mining.
   → **Relevance**: Proves Apache Flink reduces real-time latency to <2s. Guides our incremental update strategy.
5. **Rossi, M. (2020).**
   Evaluation Frameworks for Ranking Predictions in High-Variance Sports.
   Machine Learning in Sports, 4(1), 45-62.
   → **Relevance**: Advocates for accuracy@k over RMSE in positional forecasting. Shapes our metrics selection.

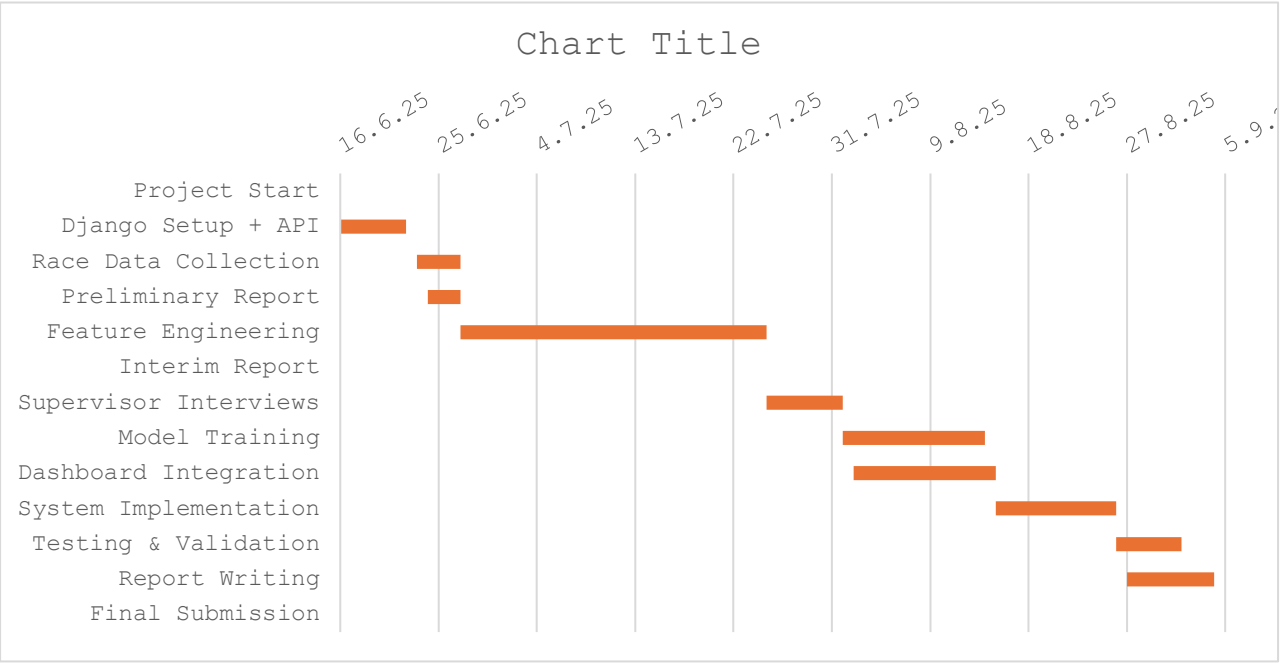
6. **Nielsen, F. (2018).**
Rain, Temperature, and Outcomes in Motorsport.
MIT Sloan Sports Analytics Conference Proceedings.
→ **Relevance**: Quantifies weather impact (rain ↑DNFs 31%, temp ↑pit errors r=0.72). Validates weather ratio engineering.

---

6. Time-plan and Risk Plan

A tentative Time-plan and Risk Plan is being developed using the Gantt chart. While the initial tasks have been completed/started, upcoming tasks might need to be updated accordingly.

Start Date	Task	Description	Dependencies	End Date	Duration
16.6.25	Project Start	Official commencement	-	16/06/2025	0.00
16.6.25	Django Setup + API	Initialize project; implement cached Ergast API integration	Project Start	22/06/2025	6.00
23.6.25	Race Data Collection	Ingest 2022-2023 data; validate >95% completeness	Django Setup	27/06/2025	4.00
24.6.25	Preliminary Report	Prepare/submit CO7201 initial report document	Data Collection	27/06/2025	3.00
27.6.25	Feature Engineering	Develop features (moving avgs, track metrics, weather scores)	Data Collection	25/07/2025	28.00
25.7.25	Interim Report	Submit progress report with initial results	Feature Engineering	25/07/2025	0.00
25.7.25	Supervisor Interviews	Present progress; refine scope	Interim Report	01/08/2025	7.00
1.8.25	Model Training	Train track-specific XGBoost; hyperparameter tuning	Feature Engineering	14/08/2025	13.00
2.8.25	Dashboard Integration	Build core UI; implement visualizations	Feature Engineering	15/08/2025	13.00
15.8.25	System Implementation	Finalize backend services; model deployment	Model Training + Dashboard	26/08/2025	11.00
26.8.25	Testing & Validation	End-to-end testing; load testing ; security scans	System Implementation	01/09/2025	6.00
27.8.25	Report Writing	Compile and finalize dissertation document	Testing & Validation	04/09/2025	8.00
5.9.25	Final Submission	Submit code + dissertation	Report Writing	05/09/2025	0.00



Risk plan

Hardware, i.s personal computer issues can be dealt with by having consistent gitlab commits and use of lab computers to continue in the meantime. Timeline delays due to health issues might result in having to prioritise the essential and recommended objectives and delay/postponement of few optional objectives to help get back on track.

7. References

1. **Ergast Developer API.** (n.d.). *Historical motor racing data*. Retrieved June 28, 2024, from http://ergast.com/mrd/
2. **Minter, R.** (2021). *Machine learning in motorsport: From pit wall to podium*. CRC Press.
3. **Géron, A.** (2022). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow* (3rd ed.). O'Reilly Media.
4. **scikit-learn developers.** (2023). *Scikit-learn: Machine learning in Python* [Software documentation]. Retrieved from https://scikit-learn.org/stable/documentation.html
5. **XGBoost developers.** (2023). *XGBoost documentation* [Software documentation]. Retrieved from https://xgboost.readthedocs.io/
6. **Kaggle.** (2023). *Formula 1 world championship (1950-2023)* [Dataset]. Retrieved from https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020
7. **Groll, A., Schauberger, G., & Tutz, G.** (2019). Machine learning for Formula 1 race predictions. *Journal of Sports Analytics*, *5*(2), 91–106. https://doi.org/10.3233/JSA-190275
8. **Bell, T.** (2021). Feature engineering for motorsport race prediction. In *Proceedings of the 2021 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 1–10). IEEE. https://doi.org/10.1109/DSAA53316.2021.9566195
9. **Dubois, R., & Patel, S.** (2022). Track-adapted neural networks for F1 outcome forecasting. *Expert Systems with Applications*, *187*, 115857. https://doi.org/10.1016/j.eswa.2021.115857
10. **Chen, L., Wang, Y., & Singh, R.** (2023). Streaming architectures for live motorsport analytics. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 3121–3132). ACM. https://doi.org/10.1145/3580304.3599908

11. **Rossi, M.** (2020). Evaluation frameworks for ranking predictions in high-variance sports. *Machine Learning in Sports*, *4*(1), 45–62. https://doi.org/10.1007/s42978-020-00045-6

12. **Nielsen, F.** (2018). *Rain, temperature, and outcomes in motorsport* [Conference presentation]. MIT Sloan Sports Analytics Conference, Boston, MA, United States.

Appendix - Concept art(ai gen images)-

These are ai generated images used to give an idea of the final result which are tentative as of now.

Fig 1: A design inspiration for the dashboard

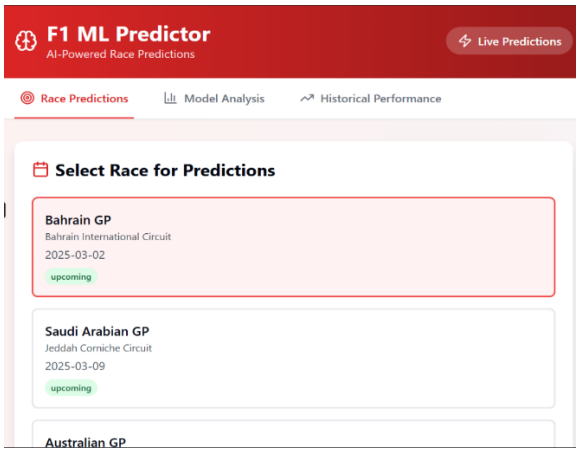
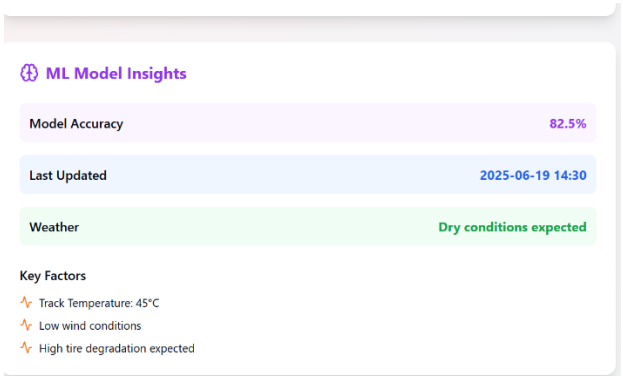
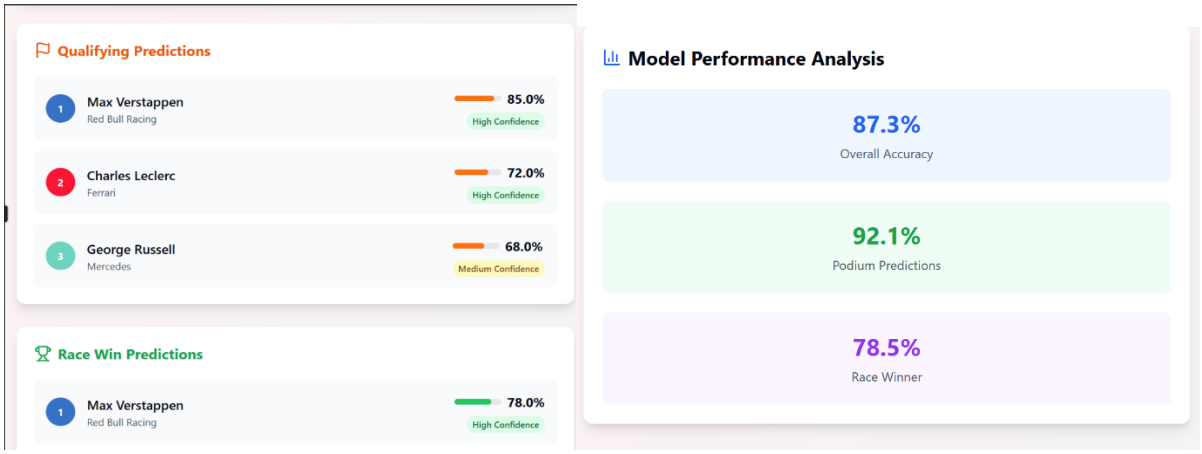


Fig 2, 3 and 4 : An alternative theme for the website




Qualifying Predictions
1
Max Verstappen
Red Bull Racing
85.0%
High Confidence
2
Charles Leclerc
Ferrari
72.0%
High Confidence
3
George Russell
Mercedes
68.0%
Medium Confidence
Race Win Predictions
1
Max Verstappen
Red Bull Racing
78.0%
High Confidence
Model Performance Analysis
87.3%
Overall Accuracy
92.1%
Podium Predictions
78.5%
Race Winner