F1 Strategy Manager AI: Integrating AI and Rule-Based Systems for Strategic Decision-Making in Formula 1

Víctor Vega Sobral
Universidad Intercontinental de la Empresa (UIE)
A Coruña, Spain
victor.vega.01@uie.edu

Abstract—**Formula 1 has increasingly become a data-driven sport, leveraging massive volumes of multimodal information - such as video feeds, telemetry, and team radio - to inform real-time strategic decisions. Despite the availability of these resources, effectively integrating them into coherent decision-support tools remains a significant challenge. This study presents F1 Strategy Manager AI, an integrated recommendation system that combines computer vision, natural language processing, predictive modeling, and expert systems to support tactical planning in F1 races. Using data from the 2023 Spanish Grand Prix, the system automatically analyzes live video, radio communications, and event telemetry to predict key race moments and propose actions such as optimal pit-stop timing and overtake strategies. Experimental validation on real race data shows that the system achieves high prediction accuracy and produces recommendations aligned with plausible race scenarios. These findings highlight the potential of AI-driven tools to improve strategic decision-making in competitive motorsport environments.**

I. Introduction

In recent years, Formula 1 has evolved into a championship increasingly driven by data and artificial intelligence, with each Grand Prix generating massive volumes of information—track images, telemetry streams, and team radio communications—that can be exploited to improve strategic decision-making [1] [2]. However, coherent integration of these multimodal data sources remains a formidable challenge. In this work, I propose **F1 Strategy Manager AI**, a strategic recommendation system for F1 teams that combines computer vision, natural language processing, predictive modeling, and an expert system.

My approach automatically analyzes live race video, radio communications, and event data from the 2023 Barcelona Grand Prix—chosen for its diverse circuit characteristics, mid-season timing, and frequent use in pre-season testing—to predict key moments and recommend tactical actions, such as optimal pit-stop timings and undercut/overcut strategies. The solution is implemented using contemporary deep learning techniques alongside a rule-based inference engine and validated on real F1 datasets. Preliminary results demonstrate that my system provides highly accurate predictions across all components, leveraging novel data fields not publicly available elsewhere. Furthermore, the strategic recommendations generated closely align with decisions that could be made in real-race scenarios, offering substantial potential to assist strategy teams in optimizing their tasks and improving decision-making processes in competitive environments.

II. State of the Art

A. Commercial Tools and Team Platforms

Formula 1 teams rely on highly specialized, proprietary platforms to process and visualize telemetry in real-time. Examples include:

- **F1 Insights (AWS):** Television graphics powered by AWS such as the "Pit Window," "Predicted Pit Stop Strategy" and "Alternative Strategy" illustrate what-if scenarios based on lap times, pace, and weather data [3] [4]. While these bring fans closer to pit-wall decisions, the underlying real-time decision systems remain closed to the public.
- **Team-developed software:** Teams like McLaren provide bespoke mobile web apps (Chrome/Android) so engineers can consult live weather data, track video and telemetry from the pit wall or even the track medical center [5].
- **Specialized meteorology:** All teams contract services (e.g. Ubimet) that deploy portable radars and weather stations at each circuit for ultra-local forecasts [6] [7]. These hyper-local data feeds (gusts, localized precipitation, time until rain) are distributed via closed portals.
- **Simulation infrastructure:** High-performance computing clusters, wind tunnels, and CFD allow teams to run tens of millions of aerodynamic and race simulations per Grand Prix. Thanks to AWS, aerodynamic simulations are now up to 70% faster and reduce downforce loss during overtakes [8]. Teams apply advanced optimization (mixed–integer programming, Monte Carlo) to plan pit-stop sequences and tire usage [9] [10].

B. Academic and Open-Source Approaches

Several research prototypes explore race-strategy optimization using public data:

- **Monte Carlo planning and RL:** Piccinotti et al. (2021) cast pit-stop scheduling as an MDP, using Monte Carlo tree search with TD updates [11]. Heilmeier et al. (2020) built a "Virtual-Strategy Engineer" via neural networks

trained on 2014–2019 data [12]. Thomas et al. (2025) introduced RSRL, an explainable RL model outperforming Mercedes' baseline in Bahrain 2023 simulations [13].
- **Specific models:** Deep-Racing (2023) employs deep nets to predict finishing positions and optimal stops from 2015–2022 data [14]. Others use stochastic dynamic programming to accommodate safety-car randomness [15]. These approaches lack internal sensor fidelity and real-time integration, limiting their practical accuracy [9] [16].

C. *Real-time Weather Forecasting*

Weather profoundly impacts race strategy. F1 funds itinerant meteorological services (e.g. Ubimet) that install portable radars and stations around each circuit [6] [7]. These high-frequency models, originally adapted for F1, deliver hyper-local forecasts (gusts, localized precipitation, time-to-rain) exclusively to authorized teams [17].

D. *Current Limitations and Accessibility*

Despite clear benefits, these systems face key challenges:

- *Stochastic events:* Safety cars, crashes and punctures are inherently unpredictable, and simulators must simplify many variables [16] [9].
- *Real-time constraints:* Algorithms must deliver recommendations in fractions of a second, not minutes [18].
- *Proprietary nature:* Nearly all advanced decision-support tools remain closed-source. Publicly visible versions (e.g. F1 Insights) do not expose the underlying software or data. Academic and community projects exist but lack the fidelity and integration of team systems [3].

III. Experimentation

This section introduces the experimental architecture of the F1 Strategy Manager system, aimed at generating real-time strategic recommendations for Formula 1 races. The system is built around three core components: predictive models implemented in Python and Jupyter Notebooks, a dynamic data analysis module powered by the FastF1 library, and a visualization interface for presenting strategy recommendations. Each component is outlined in detail, discussing its implementation and the methodologies used to validate its performance, with results analyzed in Section IV.

A. *System Architecture*

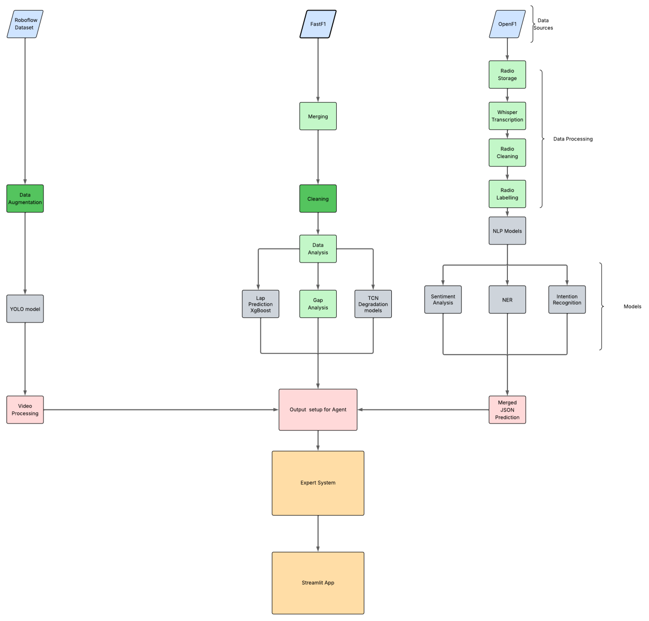


Figure 1. Architectural overview of the F1 Strategy Manager system. The diagram illustrates the data flow from acquisition sources (blue), through preprocessing stages (green), analytical models (gray), and integration components (pink), to the expert system and visualization interface (yellow/orange). Three parallel pipelines process telemetry, radio communications, and video data before converging at the central Expert System.

1) Block Diagram: Fig.1 Presents the high-level architecture of the F1 Strategy Manager system, illustrating the integration of multiple data sources and analytical components that collectively enable strategic decision support during Formula 1 races. The block diagram represents the complete data flow from raw inputs to actionable recommendations.

The diagram was created using Draw.io visualization software and exported as a PNG image to maintain visual clarity and compatibility with the IEEE publication format. The architecture visualization employs a consistent color-coding scheme to enhance comprehension: blue components represent primary data sources (FastF1, OpenF1, and Roboflow Dataset), green blocks indicate preprocessing stages, gray modules denote analytical and predictive models, pink elements show integration and mediation components, and yellow/orange blocks represent the expert system and visualization interfaces.

The diagram effectively delineates the four fundamental functional modules of the system: (1) data acquisition and preprocessing, (2) specialized analysis components (TCN degradation models, Lap Prediction XGBoost, YOLO model for video, and NLP models for radio communications), (3) the rule-based Expert System, and (4) the user interface layer. This modular design facilitates independent development and evaluation of components while maintaining system-wide integration. The following sections provide detailed explanations of each component, their specific implementations, and the data flow mechanisms that enable their collaboration within the system architecture.

2) Component Overview: The F1 Strategy System is composed of 6 primary modules, each responsible for specific aspects of the recommendation pipeline. Each module has a primary function, from transforming raw racing data to actionable strategy recommendations.

a) : Data Acquisition and Augmentation Module

This module, implemented in `data_extraction`, handles the acquisition of heterogeneous data from multiple sources. It manages five primary data streams, each extracted or processed using dedicated scripts:

- **Telemetry data**: Extracted from FastF1, including lap times, driver information, sector speeds, positions, and track conditions, among others. Handled by `data_extraction.py`.
- **Gap data**: Extracted from OpenF1, this stream provides time gaps between all drivers at 4-second intervals. It also includes booleans indicating events such as DRS usage and undercut opportunities. Managed by `extract_openf1_intervals.py`.
- **Radio communications**: Retrieved from OpenF1, this dataset captures race engineer–driver interactions from six selected races of the 2023 season (Monaco, Spain, Brazil, Netherlands, Singapore, and Belgium). Extracted using `extract_radios.ipynb`.
- **Video footage**: Creative Commons video content related to Formula 1 races is retrieved from YouTube via `video_extraction.py`.
- **Data augmentation**: Applies transformations such as shift, scaling, and Gaussian blur to extend the Roboflow dataset. Implemented in `data_augmentation.py`.

These data streams form the foundation for all subsequent analysis and model training tasks.

b) : Computer Vision module

This module focuses on processing and interpreting visual data to enhance strategy recommendations. It is composed of two main files:

- `YOLO_fine_tune.ipynb` This notebook is dedicated to fine-tuning the YOLO (You Only Look Once) model for Formula 1 Teams detection. It leverages pre-trained weights and adapts the model to recognize correctly all the teams.
- `gap_calculation.ipynb` This notebook calculates the visual gaps between cars using image data. It processes frame-by-frame information to estimate distances and time gaps, providing critical inputs for strategy optimization.

c) : Deep Learning module

This module leverages deep learning techniques to enhance predictive capabilities, specifically focusing on tire degradation modeling and lap time predictions. It is implemented in `ML_tyre_pred` and encompasses the following tasks:

- **Tire degradation prediction**: Utilizes historical race data and Temporal Convolutional Networks (TCNs) to predict tire wear and degradation rates under various conditions.
- **Lap Time Predictions**: Implements the data preprocessing and modeling of the Xgboost model for predicting lap times based on 25 variables.

This module plays a critical role in providing data-driven recommendations for tire-related strategies.

d) : Natural Language Processing module

This module focuses on processing and analyzing radio communication data to extract actionable insights. Implemented in `NLP_radio_processing`, it includes the following functionalities:

- **Speech-to-text conversion**: Transcribes audio data from radio communications into text using advanced speech recognition tools like OpenAI´s Whisper Turbo.
- **Sentiment and intent analysis**: Identifies the sentiment and intent behind race engineer–driver interactions to infer strategic decisions and situational challenges.
- **Keyword extraction**: Extracts keywords and phrases related to critical events, such as pit stops, overtakes, and weather updates, using Named-Entity-Recognition (NER).

By processing radio communication data, this module provides meaningful context to support dynamic decision-making during races.

e) : Expert System module

This module employs rule-based logic to simulate expert-level strategy formulation. Implemented in `IS_agent`, it includes the following components:

- **Models outputs transformation**: Collects the different outputs provided by all the models and passes it to the Facts base of the expert system.
- **Rule-based decision engine**: Encodes domain knowledge into a set of rules for scenario-based strategy recommendations, such as pit stop timing, undercut/overcut possibilities, or making traffic recoveries.

This module mimics the decision-making process of a race strategist, providing structured and logical insights.

f) : Streamlit App module

This module provides a user-friendly frontend for interacting with the strategy recommendation system. Implemented in `app`, it includes the following features:

- **Interactive dashboards**: Displays key data streams, such as telemetry, gap analysis, and tire performance, in an intuitive format.
- **Visualization tools**: Offers visual representations of race dynamics, including gap visualizations and predictive insights.
- **LLM Chat and Reports**: Allows users to have conversations with a Formula 1 expert LLM, which is also capable of analyzing different graphs from the app. The app also uses this LLM to create post-race briefing reports about various race features that can be selected. The LLM is Llama3.2-vision with system prompting.

The Streamlit App module serves as the primary interface for users, enabling seamless interaction with the system's insights and recommendations.

B. Individual Analysis Modules

1) Lap Prediction Model: This module, implemented in `lap_prediction.ipynb`, implements an XgBoost model for predicting the estimated lap time of the next lap, based on 25 variables regarding Team, fuel load, position, stint, tire age, etc. The first step was preprocessing the data. Some variables like the Sector times and lap times were removed so the model is not influenced by any time-related variables. Other ones needed to be converted before making any further analysis, for example, a team mapping was applied to convert this variable from a string with the Team´s name to an `int` value. After a correlation matrix, some redundancies were eliminated, keeping only one variable in cases with two variables with high correlation. Also, some variables that did not provide any useful information due to their low correlation were removed, like weather variables, which remained stable for the whole race and did not change at any moment, consequently not affecting the car´s performance.

After the train-validation-test split, the first XgBoost model was trained with a RMSE of 0.45 seconds. Later on, an LSTM implementation was considered, consequently needing to sequential the data by driver and stint, taking advantage of Formula 1 data nature. These models had a negative R^2, quickly discarding its use. Nevertheless, a new XgBoost model was trained using this sequential data, getting an RMSE of 0.15 seconds and a R^2 of 0.9877, with no signs of overfitting.

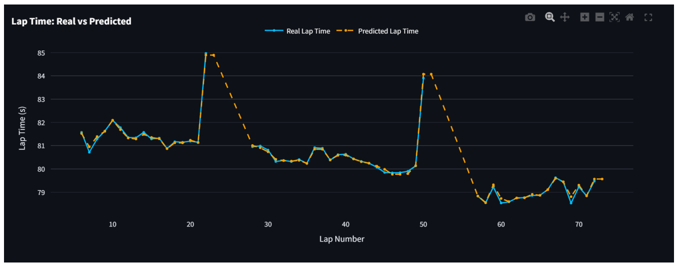


Figure 2. Comparison of actual versus XGBoost-predicted lap times over Barcelona 2023 race session. The model achieves an RMSE of 0.15 s and an R² of 0.9877, with no signs of overfitting.

In Fig.2, the dashed orange line (predicted) closely tracks the solid blue line (actual) throughout all 66 laps. Notable deviations occur at pit-stop events and abrupt pace changes—here the model slightly underestimates the peak but rebounds immediately on the following lap. During sustained run segments (e.g., laps 20–40 and Fifty-five–sixty-six), the two curves are virtually indistinguishable, underscoring the model's ability to learn sequential dependencies without fitting to transient noise. These results demonstrate the robustness of the XGBoost approach for accurate lap-time forecasting and highlight its potential for integration into real-time strategic decision-support systems.

2) Tire Degradation model: This module, implemented in `ML_tire_pred`, explores the tire degradation phenomena, analyzing and creating synthetic derived from some variables to implement a model that predicts the tire degradation for the next three laps based on the 5 last laps and different variables. The first step was the variable analysis, making different plots to explore the correlations between lap times and tire ages or compounds. Upon exploring these variables, a strange phenomenon was detected: some drivers had negative degradation with some tire compounds, highlighting the effect of fuel load during the race.

According to [19], an F1 car's lap-time sensitivity to on-board fuel at Singapore is approximately –0.44 s per kilogram, implying that burning 1 kg of fuel yields about a 0.055 s lap-time gain. This estimate aligns with broader "rule-of-thumb" figures—0.04 s/kg on low-speed tracks and 0.03 s/kg on medium-speed tracks [20], as well as 0.03–0.05 s/kg reported in contemporary team discussions [21]—so we adopt a fuel-mass sensitivity of 0.055 s per lap in our model.

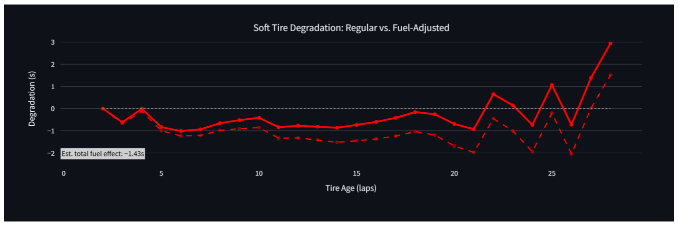


Figure 3. Comparison of soft tire degradation across laps, with and without fuel adjustment. The dashed line represents regular lap time degradation, while the solid line shows degradation after compensating for fuel load. The estimated total fuel effect over the stint is approximately 1.43 seconds.

To isolate the tire degradation effect from the confounding influence of fuel weight, a lap-by-lap fuel adjustment was applied to the raw lap times. As shown in Fig.3, the dashed line represents unadjusted degradation, which includes both tire wear and the natural lap time gain as the car burns fuel. The solid line shows the adjusted degradation curve, where a fuel correction of approximately 0.055 seconds per lap (based on an estimated total fuel effect of 1.43 seconds over 26 laps) was subtracted progressively. This transformation allows for a clearer visualization of tire-only performance drop-off, effectively removing the systematic time gain caused by decreasing fuel mass.

The following process was developed using this "fuel adjusted" tire degradation, in order to eliminate this effect for the model and get better estimates of tire degradation. After sequestrating the data, the first approach was to develop an LSTM for capturing the tendency of tire degradation. However, the complexity of this data made LSTMs unable to capture at least the tendency of the degradation, which means, at least predicting if the degradation is going to stagnate, reduce, or increase. Therefore, Temporal Convolutional Networks, despite their complexity, became one of the most promising options for training, as their convolutional layers are able to better capture specific patterns in complex data than LSTMs.

This model reached an RMSE of 0.34 seconds and was able to capture also the trend of the degradation. As this model was trained in all tire compounds and did not differ between the different tires, three new and smaller models were trained, one per compound (soft, medium, and hard, intermediate and wet tires were not made as no data for this race is available with those compounds).

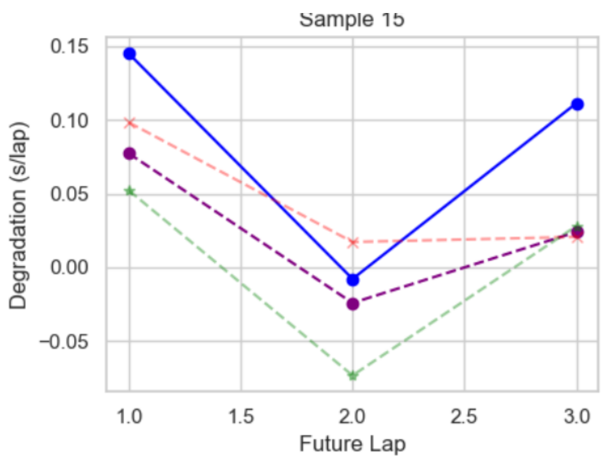


Figure 4. Tire degradation prediction model output for Sample 15, showing multiple future lap projections. The solid blue line represents the real degradation rate, while dashed lines (purple, orange, and green) indicate the predictions made by an ensemble, specialized, and global model. Note the characteristic valley pattern at lap 2.0 where degradation temporarily improves before increasing again, demonstrating the TCN model's ability to capture non-linear tire behavior patterns

As shown in Fig.4, the TCN-based model successfully captures the non-linear behavior of tire degradation. In particular, it identifies the temporary improvement in lap times around lap 2—often due to tire warm-up or track evolution—before degradation resumes. While the global model and the ensemble prediction provide a general trend, the specialized model (trained only on the soft compound) offers the closest match to the real degradation curve. The ensemble model was made making a mean between the specialized and global model.

Although the specialized models alone performed poorly for the soft and medium compounds - registering RMSEs 42. 15% and 24. 51% higher than the global model, respectively - the ensemble approach yielded substantial improvements. For soft tires, the ensemble model achieved a 33.43% reduction in RMSE compared to the global model, and for medium tires, the improvement was 10.20%. Interestingly, the hard compound showed the opposite behavior: the specialized model performed significantly better than the global one, with a 39.63% reduction in RMSE. However, in this case, the ensemble slightly underperformed compared to using the specialized model alone, suggesting that the hard tire degradation pattern is better captured by a compound-specific model.

These findings confirm that accounting for both universal degradation trends and compound-specific nuances yields the most accurate predictions. Consequently, this hybrid modeling strategy holds promise for real-time race strategy optimization, enabling teams to better anticipate tire life and plan pit stops with greater confidence.

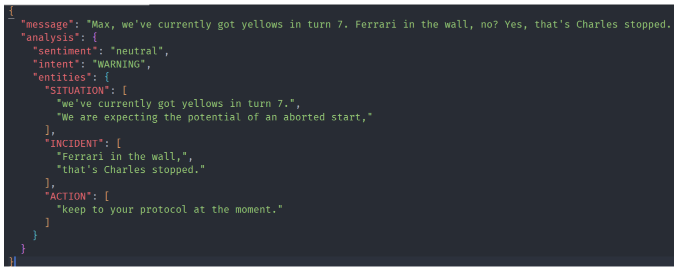
3) *Radio Communications Analysis:*

a) Sentiment Analysis: The sentiment analysis module is responsible for classifying F1 team radio messages into `positive`, `neutral`, or `negative` categories. Manual labeling was performed using a custom Jupyter widget interface, ensuring that end-of-race messages were excluded to avoid emotional bias and focus on in-race strategy communications (see `N00_data_labeling.ipynb`). Initial experiments with VADER (`N02_sentiment_analysis_vader.ipynb`) revealed that generic sentiment tools overestimated positive sentiment (43.6% vs. 9.4% in manual labels) and underestimated neutral messages, highlighting the need for domain-specific models. The final model is a fine-tuned RoBERTa-base transformer (`N03_bert_sentiment.ipynb`), achieving an accuracy of 87.50% and a weighted F1-score of 0.8746 on the test set. For example, the message "Great move Oscar" is correctly classified as `positive`. The inference pipeline is implemented in `N06_model_merging.ipynb`, using the function `predict_sentiment()`.

b) Intent Recognition: Intent recognition aims to categorize radio messages into strategic classes such as `INFORMATION`, `PROBLEM`, `ORDER`, `WARNING`, and `QUESTION`. The dataset was manually relabeled for these categories (`N04_radio_info.ipynb`), and several models were tested, including RoBERTa-large and DeBERTa-v3-large. The best results were obtained after removing the under-represented `STRATEGY` class and rebalancing the dataset. The final RoBERTa-large model achieved 76% accuracy and a weighted F1-score of 0.76, with particularly strong performance in the `PROBLEM` (F1=0.82) and `QUESTION` (F1=0.94) classes. For instance, "Possible debris turn 7" is classified as a `WARNING`. The full training and evaluation process is documented in `N04_radio_info.ipynb`, and the inference function `predict_intent()` is integrated in `N06_model_merging.ipynb`.

c) Named-Entity Recognition (NER): The NER module extracts F1-specific entities from radio messages, such as `ACTION`, `SITUATION`, `INCIDENT`, `STRATEGY_INSTRUCTION`, and others. Manual annotation was performed using SpaCy's annotation tools, and several architectures were evaluated, including SpaCy baselines and transformer-based models. The final model is a fine-tuned BERT-large with custom loss functions to handle class imbalance (`N05_ner_models.ipynb`). The model achieved an entity-level F1-score of 0.43, with improvements after focused fine-tuning. For example, in the message "keep to your protocol", the entity `ACTION` is correctly identified. As in real radios the model is capable to correctly extracting the entities, it is believed that these low metrics occur due to the BIO tagging format used for the training, with some of the metrics of some entities (like beginning or finishing) with low performance. The extraction logic is encapsulated in the function `analyze_f1_radio()` and is called within the unified pipeline in `N06_model_merging.ipynb`.

d) Pipeline Integration: All three models are integrated into a unified pipeline in `N06_model_merging.ipynb`, which processes either raw text or audio (using Whisper for transcription) and outputs a standardized JSON structure. This structure includes the original message, predicted sentiment, intent, and a list of extracted entities, ready for downstream use by the logic agent. For example, the message "Max, we've currently got yellows in turn 7. Ferrari in the wall, no?" is processed and stored as a structured JSON file, enabling automated strategy analysis. The pipeline ensures reproducibility and modularity, with each model loaded and executed via dedicated functions.



```
{
  "message": "Max, we've currently got yellows in turn 7. Ferrari in the wall, no? Yes, that's Charles stopped.
  "analysis": {
    "sentiment": "neutral",
    "intent": "WARNING",
    "entities": {
      "SITUATION": [
        "we've currently got yellows in turn 7.",
        "We are expecting the potential of an aborted start,"
      ],
      "INCIDENT": [
        "Ferrari in the wall,",
        "that's Charles stopped."
      ],
      "ACTION": [
        "keep to your protocol at the moment."
      ]
    }
  }
}
```

Figure 5. JSON output from the integrated NLP pipeline processing a team radio message during a race incident. The system correctly identifies neutral sentiment with WARNING intent, while extracting key entities including situation details (yellows in turn 7), incident information (Ferrari crash with Charles Leclerc), and recommended action (maintaining protocol). This structured output enables the strategic agent to process race events and trigger appropriate rule-based recommendations.

The JSON structure displayed in Fig.5 exemplifies the successful integration of all three NLP components developed for the F1 strategy system. Through the unified pipeline implemented in `N06_model_merging.ipynb`, the raw team radio message is transformed into a machine-readable format that captures critical race information. The RoBERTa sentiment model correctly identifies the neutral emotional tone despite the incident severity, while the intent recognition system appropriately flags this as a WARNING. Most notably, the named-entity recognition component effectively extracts and categorizes multiple entities across different strategic categories, demonstrating the model's ability to parse complex F1-specific terminology and situations, which directly supports the rule-based reasoning system in generating appropriate strategic recommendations.

4) YOLO Team Detection: The dataset for Formula 1 team detection was built using Roboflow, ensuring a balanced representation of all 10 current teams. Initial experiments revealed significant challenges: classes such as Williams, Alpine, and Kick Sauber suffered from low recall and precision due to visual similarity and a lack of diverse examples. To address this, the dataset was expanded to 250 images per class and extensive data augmentation, including rotation, mixup, copy-paste, color jittering, and Gaussian blur. Class weights in the YAML configuration were adjusted to penalize misclassifications of problematic teams. Multiple YOLOv8 architectures (from nano to medium) and optimizers (SGD, AdamW) were evaluated. The final model, trained with class balancing, label smoothing, cosine annealing, and aggressive regularization, achieved state-of-the-art results: mAP@50 of 0.958 and mAP@50-95 of 0.754. Notably, Williams reached a recall of 1.00 and mAP@50 of 0.995 after targeted fine-tuning. This process can be found in `YOLO_fine_tune.ipynb`.

For gap calculation, the pipeline leverages OpenCV and the fine-tuned YOLO model to detect cars in each video frame, compute physical and temporal gaps, and store these results in structured CSV files. The system samples frames periodically (e.g., every 10 seconds), applies adaptive sharpening filters based on lighting conditions, and uses class-specific confidence thresholds and temporal stabilization rules to minimize false positives, especially for Williams and Alpine. Detection thresholds can be dynamically adjusted during runtime, and real-time visualization is supported, facilitating both debugging and integration with the expert strategy system. Moreover, a common label was added to avoid making false positives that then would be stored in the CSV file, known as "F1 Car", which uses a threshold dictionary depending on the class. For instance, Williams or Alpine have a threshold of 0.90 while McLaren is only 0.30. This process can be found in `gap_calculation.ipynb`.

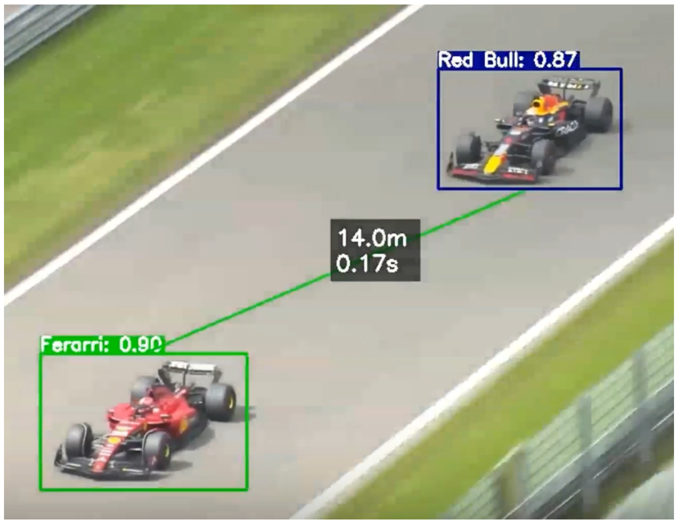


Figure 6. Output from the YOLOv8-based gap detection system during the 2023 Belgian Grand Prix. The model successfully identifies both a Ferrari (0.90 confidence) and a Red Bull (0.87 confidence) with high precision, calculating a 14.0m physical separation and a 0.17s time gap between them. The system applies team-specific confidence thresholds and connects detections with measurement visualization, enabling real-time strategic gap analysis.

The visualization in Fig.6 demonstrates the practical implementation of our computer vision pipeline for Formula 1 car detection and gap analysis. The YOLOv8 model achieves high-confidence detections despite challenging track conditions, with both Ferrari and Red Bull exceeding the 0.85 confidence threshold. The system's ability to simultaneously measure physical distance (14.0m) and temporal gaps (0.17s) provides crucial data for strategic decision-making, particularly for undercut/overcut opportunities. The green connecting line represents the calculated gap vector, with real-time measurements displayed in the central overlay. This implementation overcomes the detection challenges mentioned earlier, with the model maintaining high precision even during high-speed sections where motion blur typically degrades performance. The visualization confirms that our adaptive confidence thresholds and extensive data augmentation techniques have successfully addressed the class imbalance issues encountered during development.

5) Gap Analysis:

a) FastF1 Gap Extraction: To enable a detailed analysis of the time gaps between cars during a Formula 1 race, the FastF1 Python library was utilized, which provides access to official F1 timing data. The session data for the 2023 Spanish Grand Prix was loaded using FastF1, and lap-by-lap timing information for all drivers was extracted. For each lap, the gap to the leader was calculated by comparing each driver's lap completion time to the minimum lap time recorded for that lap (i.e., the leader's time). Additionally, the gaps between the car ahead and the car behind were computed by sorting drivers according to their gap to the leader and measuring the time difference to the adjacent cars in the classification order. Our CSV gaps previously stored have good precision but are difficult to map for every driver in each lap. Then, it was concluded that using FastF1 data was the best approach for the Expert System.

The resulting dataset was constructed to include, for every driver and lap, the following key metrics: gap to the leader, gap to car ahead, gap to car behind, position, and team. Strategic windows such as the "undercut window" (gap to the car ahead < 2.5s) and the "DRS window" (gap to the car ahead < 1.0s) were also flagged. Missing values, such as those corresponding to the race leader (no car ahead) or the last car (no car behind), were replaced with appropriate placeholders to ensure data consistency. The processed gap data was then saved in CSV format for subsequent analysis and integration with other modules.

b) Gap Analysis: Important Findings: Several important insights into the dynamics of a Formula 1 race were revealed through the analysis of the extracted gap data. By examining the distribution of gaps across all laps and drivers, it was found that a significant portion of the race was spent within strategic windows: for instance, approximately 45% of all measured gaps to the car ahead were within the undercut window (<2.5s), and around 16% fell within the DRS window (<1.0s). These findings underscore the prevalence of close racing and the frequent emergence of strategic opportunities.

Summary statistics for the gap to the car ahead indicated a wide distribution, with a mean of approximately 4.4 seconds and a standard deviation of 4.7 seconds, suggesting the presence of both tightly contested phases and periods with greater separation. The dataset also enabled the identification of patterns, such as which drivers spent the most laps in close proximity to others, and how the distribution of gaps evolved throughout the race.

By ensuring that the gap data was cleaned and structured, a robust foundation was established for more advanced strategic analysis, including the detection of key race phases (e.g., early and mid stints) and the quantification of overtaking and defensive opportunities. This gap analysis is considered essential for understanding the competitive landscape of the race and for informing high-level decision-making in Formula 1 strategy.

C. Expert System for Strategic Recommendations

1) Framework used: For the Strategic Expert System, it was decided to use Experta, a Python library based on Production Systems, also known as Expert Systems or Rule-Based Systems, which is a fundamental paradigm in symbolic AI. Experta´s core is the RETE algorithm, designed in 1982 by Charles L.Forgy. This algorithm is crucial for knowing what Experta is doing and also knowing why it is efficient. Its function is to optimize the coincidence of patterns between facts and rules, building a node network that represents patterns. Then, it avoids reevaluating all the rules when the facts change. Therefore, RETE builds a "discrimination net" that acts as an efficient filter for determining which rules should be activated in response to changes made on the facts base.

An expert system is formed by 3 key components: the facts base, which stores the factual knowledge of the system, the rule base, which contains the procedure knowledge as "if-then" rules, and the inference motor, which determines whether apply one rule or another and when to apply it. Experta´s selection for the F1 strategy problem is theoretically justified by these 5 points:

1) **Codificable expert knowledge**: F1 strategies can be expressed naturally as conditional rules based on expert knowledge.
2) **Incremental Reasoning**: during the race, information comes continuously, such as radios, telemetry, track data, or weather. RETE is efficient for updating conclusions based on new information.
3) **Knowledge Explanation**: rules can be read and modified by humans, allowing adjusting strategies based on feedback.
4) **Explainable Capacity**: unlike black box models like Neural Networks, a system based on rules can explain exactly which conditions made them make a decision.
5) **Multiple Information Integration**: key for my project, as it brings me the capacity to merge structured information as data with semi-structured information like NLP radio analysis or my prediction models in the same logical framework

2) Facts and Rules Design: In the Strategic Expert System, the design of facts and rules revolves around various aspects of an F1 race, such as tire degradation, lap times, radio communication, and time gaps between cars. The `IS_agent` folder contains notebooks and scripts that detail the creation of these components.

a) Facts Base Design:

- **TelemetryFact:** Stores car performance data.
- **DegradationFact:** Tracks tire degradation.
- **RaceStatusFact:** Describes the current race conditions.
- **RadioFact:** Encodes NLP-analyzed team radio messages.

- **StrategyRecommendation:** Represents recommendations such as pit stops or stint extensions.

These facts are defined as Python classes (Experta's `Fact`) and are dynamically updated during a race to reflect real-time changes in data.

b) Rule Base Design: Rules are expressed using Experta's `Rule` class with "if-then" logic. Example rule sets include:

- **Tire Degradation Rules:**
  - If the degradation rate (`degradation_rate`) is high and tire age is significant, recommend a pit stop.
  - If the degradation rate is low and tire age is sufficient, suggest extending the current stint.
  - If the degradation rate increases sharply over the last three laps, issue an early warning and prepare for a pit stop.
  - If the predicted future degradation exceeds a threshold and tire age warrants it, recommend considering a predictive-based pit stop.
- **Lap Time Rules:**
  - If the predicted lap time is better than the current lap and the driver is not in podium contention with fresh tires, recommend a push strategy.
  - If the predicted lap time is significantly worse than the current lap and the tires are old, recommend an immediate pit stop (performance cliff).
  - If the predicted lap time improves substantially after running in traffic, recommend a recovery push stint.
- **Gap Analysis Rules:**
  - If the gap to the car ahead is less than 2.0 s for at least three consecutive laps, recommend an undercut.
  - If the gap to the car behind is less than 2.0 s for at least three consecutive laps, recommend a defensive pit stop.
  - If the gap to the car ahead is between 2.0 s and 3.5 s for at least four consecutive laps, recommend an overcut.
  - If the driver is outside the top 10 and the gap to the leader exceeds 30 s during strategic windows, recommend adjusting the pit window to avoid traffic.
- **Radio Communication Rules:**
  - If the sentiment of the message is negative and grip issues are detected, prioritize a pit stop.
  - If the message contains a rain warning, prepare for a switch to wet-weather tires.
  - If an incident is detected (safety car, yellow flag), re-evaluate the pit window to exploit the neutralization.
- **Combined High-Priority Rules:**
  - If critical degradation is confirmed (high degradation rate, worsening lap times, and negative sentiment), recommend an urgent pit stop.
  - If an incident occurs and the driver is outside the top 10 with a large gap ahead, recommend taking advantage of the neutralization to pit.

3) Conflict Resolution: The conflict resolution mechanism ensures that when multiple rules are activated, the system prioritizes and executes the most critical actions.

a) Approach:

- Script `N06_rule_merging.py` implements conflict detection.
- Conflicting actions (e.g., "pit_stop" vs. "extend_stint") are resolved by:
  - **Priority:** Higher-priority rules fire first.
  - **Confidence:** Recommendations with higher model confidence are preferred.
  - **Incompatibility Handling:** Mutually exclusive actions are dropped.

This process is integrated into the main recommendation engine so that only non-conflicting suggestions reach the user.

4) Domain Integration: Domain integration combines structured telemetry data with semi-structured NLP outputs to yield unified strategic insights.

a) Integration of Models: Different models (sentiment, NER, intent, gap analysis) run in parallel and write their outputs into a standard JSON schema via `N06_model_merging.ipynb`.

b) Logical Framework: Experta is used to reason over:

- Telemetry data
- Tire degradation predictions
- NLP-analyzed radio communications
- Gap analysis insights

c) Real-Time Strategy Optimization: By feeding NER outputs into the rule engine, real-time decisions can adapt to race events (e.g., avoiding traffic after pit stops).

d) Benefits:

- Comprehensive, explainable recommendations
- Seamless fusion of heterogeneous data

D. User Interface and Visualization

1) Interface Design: The user interface of the F1 Strategy Manager is implemented using Streamlit, providing a modern, interactive, and modular web application experience. The design philosophy centers on usability for both technical and non-technical users, with sidebar navigation that allows seamless switching between different analytical modules. Each module is encapsulated in its own component, ensuring maintainability and scalability.

Key features include:

- **Sidebar Navigation:** The sidebar, styled via a custom CSS theme, offers quick access to modules such as Overview, Gap Analysis, tire Degradation, Lap Time Predictions, Radio/NLP Analysis, Recommendations, Strategy Chat and the Report Generation. The navigation is visually enhanced with color cues and responsive design for clarity and accessibility.
- **Component-Based Layout:** Each analytical view (e.g., gap analysis, recommendations, radio analysis) is implemented as a separate Python module, allowing for

independent development and testing. This modularity enables rapid iteration and future expansion.

- **Dynamic Data Loading:** The interface supports real-time data loading and updating, allowing users to select drivers, laps, or scenarios and immediately see the impact on all visualizations and recommendations.

2) Interactive Visualizations: The application leverages Plotly and Matplotlib for interactive, high-resolution visualizations, tightly integrated with the Streamlit front end. These visualizations are not static; users can interact with them to explore different scenarios, filter data, and simulate strategic decisions.

Major visualization functionalities include:

- **Gap Evolution and Strategy Windows:** Interactive line and scatter plots display the evolution of time gaps to cars ahead and behind, highlighting undercut and overcut windows, DRS zones, and periods of strategic vulnerability. Users can hover to see exact values, zoom into specific race segments, and compare multiple drivers.
- **Tire Degradation and Stint Analysis:** Multi-series plots show tire degradation rates, predicted versus actual stint lengths, and compound performance over time. Users can select compounds, compare drivers, and visualize the impact of degradation on lap times and pit stop timing.
- **Lap Time Prediction Explorer:** Visual tools allow users to examine predicted versus actual lap times, identify performance cliffs, and see in which laps the model had the biggest errors.
- **Radio/NLP Analysis:** There is a section provided for uploading Formula 1 radios. The app then calls Whisper for transcribing and then also all the NLP models to give as an output the structured JSON.
- **Recommendations Timeline:** A chronological timeline displays all expert system recommendations, color-coded by priority and action type. Users can click on recommendations to view detailed explanations, confidence levels, and simulated outcomes.
- **Video Gap Extraction:** For vision-based modules, the interface can display a video to show detected cars, measured gaps, and all the processes regarding storage during the operation.

3) LLM Chat and Report Making: A standout feature of the application is the integration of a Large Language Model (LLM) powered chat assistant and automated report generation:

- **Strategy Chat Assistant:** Users can interact with an advanced LLM-based assistant (Llama3.2-vision) directly within the app. The chat supports both text and image inputs, enabling users to ask questions about strategy, upload screenshots or plots, and receive expert-level, context-aware explanations. The assistant can interpret visual data, summarize complex scenarios, and provide actionable recommendations in natural language.
- **Automated Report Generation:** The application includes a comprehensive report export module. Users can select which sections to include (e.g., strategy summary, gap analysis, tire predictions, radio/NLP insights, competitive analysis, raw data appendix) and generate a professional HTML report. Each section's narrative is automatically drafted by the LLM, which synthesizes data, visualizations, and expert system outputs into concise, technical, and insightful prose. All key plots and tables are embedded, and the report is suitable for both technical review and executive presentation.
- **Multimodal Narrative Synthesis:** The LLM is capable of multimodal reasoning, meaning it can generate explanations that reference both numerical data and visualizations. For example, it can describe trends in a gap evolution plot, interpret the implications of a tire degradation curve, or summarize the sentiment distribution in radio communications.

In summary, the user interface and visualization layer of the F1 Strategy Manager is designed to maximize usability, transparency, and analytical power, combining interactive data exploration, expert system recommendations, and state-of-the-art AI-driven narrative generation in a unified, professional platform.

E. Evaluation Methodology

1) Evaluation Metrics: To rigorously assess the performance of the F1 Strategy Manager, both quantitative and qualitative metrics are employed, focusing on the accuracy, efficiency, and practical utility of the system's recommendations in real-world scenarios.

Quantitative Metrics:

- **RMSE (Root Mean Square Error):** Measures the accuracy of continuous predictions such as lap times and tire degradation rates compared to actual race data.
- **MAE (Mean Absolute Error):** Average absolute difference between predicted and real values, used for lap time and degradation.
- **F1-Score:** Effectiveness in classifying critical decisions (e.g., "recommend a pit stop" vs. "extend stint").
- **Precision and Recall:** Used for binary/multiclass classification tasks, such as pit stop detection and radio sentiment analysis.
- **Inference Time:** Average time required to generate recommendations.
- **Rule Activation Frequency:** Number of times each rule is triggered during a race.
- **mAP@0.95 (mean Average Precision at IoU 0.95):** Object detection accuracy for vision-based modules (e.g., YOLO).

Metric	Definition
RMSE	Root mean squared error (regression).
MAE	Mean absolute error (regression).
F1-Score	Harmonic mean of precision and recall (classification).
Precision	True positives / (True positives + False positives).
Recall	True positives / (True positives + False negatives).
Inference Time	Average time to generate a recommendation.
Rule Activation Frequency	Number of rule activations per race.
mAP@0.95	Mean average precision at IoU 0.95 (object detection).

Table I
SUMMARY OF QUANTITATIVE METRICS USED IN THE EVALUATION.

2) Validation with Historic Data: The system is validated exclusively using historic data from the 2023 Spanish Grand Prix (Barcelona), as this is the only available dataset with complete telemetry, tire, and strategy information. All tests, training, and validation procedures are performed using this single race, with careful partitioning to avoid data leakage.

Validation Data:

- **Primary Dataset:** Historical data from the 2023 Spanish Grand Prix, extracted via FastF1, including:
  - Telemetry (lap times, speeds, tire degradation)
  - Race conditions (weather, safety car periods, incidents)
  - Real strategic decisions (pit stop timings, stint strategies)
  - Team radio communications (for NLP module validation)
- **No Additional Races:** Due to data availability, all validation and testing are performed on this single event.

Train/Test Partitioning:

- **Radio Sentiment and Intent Classification:**
  - **Split:** 70% training, 15% validation, 15% testing.
- **Named Entity Recognition (NER):**
  - **Split:** 80% training, 10% validation, 10% testing.
- **Tire Degradation and Lap Time Predictions:**
  - **Split:** Typically 70% training, 15% validation ,15% testing.

Summary: All evaluation, training, and validation are strictly based on the 2023 Spanish Grand Prix data, with careful partitioning and scenario simulation to maximize the insight gained from this single, comprehensive dataset.

IV. RESULTS ANALYSIS

In this section, the quantitative outcomes obtained from each of the individual analysis modules (lap-time prediction, tire degradation, gap analysis, radio communications NLP, and YOLO-based team detection) are presented, the conflict-resolved strategic recommendations generated by the expert system are evaluated and the effectiveness of the Streamlit interface is demonstrated.

A. Lap Time Predictions

To evaluate the accuracy of the lap-time prediction models, standard metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and the coefficient of determination (R^2) were employed. Table II presents a comparison between the original XGBoost model and the optimized sequential version, demonstrating a significant improvement across all metrics. Additionally, Fig.7 shows a scatter plot of predicted versus actual lap times, illustrating the high correlation and low error dispersion of the proposed model.

Metric	XGBoost Original	XGBoost Sequential	Improvement
RMSE	0.4512 s	0.1527 s	66.2%
MAE	0.3091 s	0.0901 s	70.9%
R^2	0.9112	0.9877	8.4%

Table II
COMPARISON OF LAP-TIME PREDICTION PERFORMANCE BETWEEN THE ORIGINAL AND SEQUENTIALLY OPTIMIZED XGBOOST MODELS.

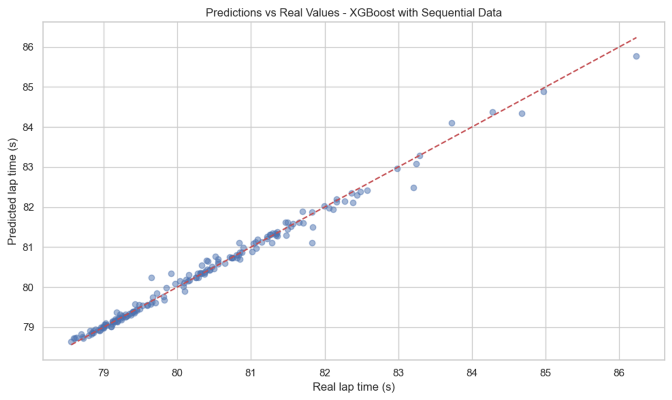


Figure 7. Predicted versus actual lap times for the sequentially optimized XGBoost model. The dashed red line indicates the ideal $y = x$ relationship.

B. Tire Degradation Models

To evaluate the accuracy of the tire-degradation prediction models, standard metrics such as root mean squared error (RMSE) and mean absolute error (MAE) were computed for both the global model and the compound-specific models. Table III summarizes these results, enabling a direct comparison of performance across the different approaches.

Fig.8 presents a visual comparison between predicted and actual degradation for selected samples, while Fig.9 shows the RMSE comparison by compound type. It is observed that the fuel-load adjustment improves the model's ability to capture true degradation, especially on medium and hard compounds where fuel effects can mask actual wear. The per-compound analysis reveals that the specialized models achieve lower error compared to the global model, highlighting the importance of tailoring the model to each tire's specific characteristics.

In conclusion, the fuel-load adjustment proved essential for obtaining more representative degradation metrics and enhancing prediction accuracy, which is particularly relevant for strategic decision-making during a race.

Compound	MSE	RMSE	MAE
MEDIUM	0.178381	0.422352	0.286387
HARD	0.073123	0.270413	0.178596
SOFT	0.166899	0.408533	0.238303
GLOBAL MODEL	0.120374	0.346950	0.222602

Table III
PERFORMANCE METRICS (MSE, RMSE, MAE) FOR TIRE-DEGRADATION PREDICTION MODELS BY COMPOUND AND FOR THE GLOBAL MODEL.

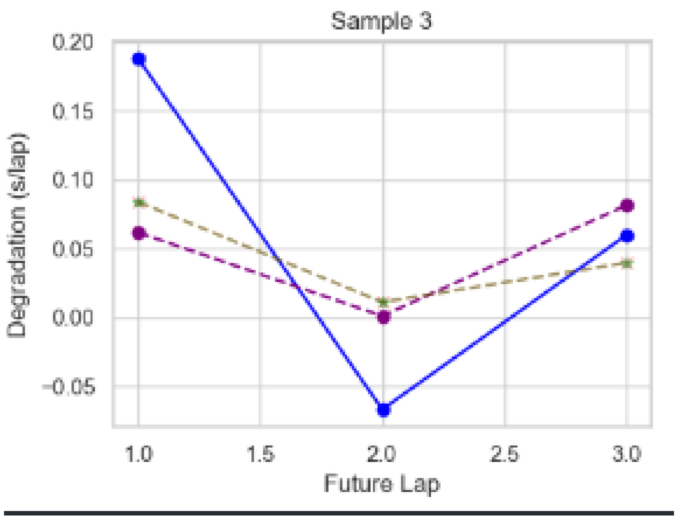


Figure 8. Tire degradation over future laps for a selected soft-compound sample. The solid blue line represents the actual degradation, while the dashed and dotted lines show predictions from the global ensemble model and the soft-compound model, respectively.

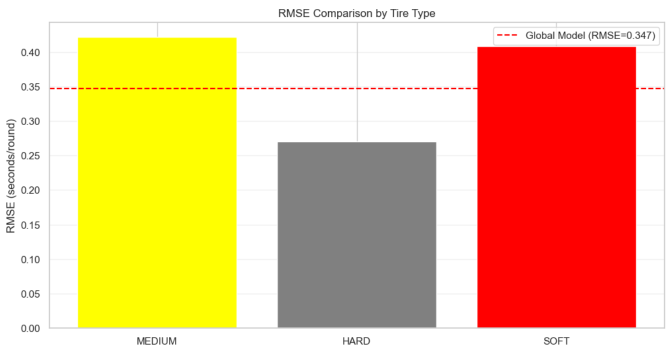


Figure 9. Comparison of RMSE (seconds per lap) across tire compounds. The red dashed line marks the global-model RMSE for reference.

C. YOLO-based Team Detection

The fine-tuned “ANTI-ALPINE” YOLO model was evaluated on a held-out test set of 20 images containing all ten F1 team liveries. Detection performance is summarized in Table IV. Precision–recall curves were generated both in aggregate and per class (not shown) to assess model robustness.

An overall mAP@0.5 of 0.958 was achieved by the object detection component of the system across all team classes. Precision–recall curves, in which high performance (0.995) was obtained by most teams (Alpine, Ferrari, Haas) and variable performance was observed for Kick Sauber (0.850), Mercedes (0.913), and McLaren (0.927) under different conditions, are shown in Fig.10.

The system’s ability to correctly localize Formula 1 cars despite varying perspectives and occlusion is demonstrated by the detection examples in Fig.11. Occasional misclassifications between visually similar liveries, particularly between Williams and background elements, were revealed by the confusion matrix analysis in Fig.12.

An average inference time of 38.5 ms per frame (approximately 26 FPS) was measured on an NVIDIA RTX 4060 Laptop GPU, with preprocessing and postprocessing latencies of 2.6 ms and 1.2 ms respectively. These metrics confirm the system’s suitability for real-time race analysis applications.

Team-specific confidence thresholds—0.75 for Alpine, 0.50 for Williams, 0.30 for McLaren, and 0.40 for the remaining teams—were implemented to optimize detection quality.

Class	TP	FP	P	R	mAP@0.5	mAP@0.5:0.95
All	20	29	0.815	0.819	0.958	0.754
Kick Sauber	4	6	1.000	0.778	0.850	0.651
Racing Bulls	1	1	0.463	1.000	0.995	0.697
Alpine	4	5	0.980	1.000	0.995	0.836
Ferrari	3	4	1.000	0.831	0.995	0.909
Haas	1	1	0.713	1.000	0.995	0.796
McLaren	5	6	1.000	0.278	0.927	0.595
Mercedes	3	3	0.937	0.667	0.913	0.685
Williams	2	3	0.430	1.000	0.995	0.863

Table IV
DETECTION METRICS (TP, FP, PRECISION P, RECALL R, MAP@0.5, MAP@0.5:0.95) FOR EACH F1 TEAM CLASS AND OVERALL.

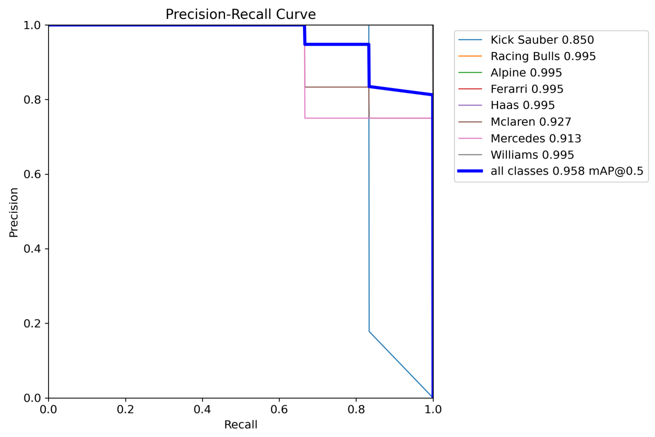


Figure 10. Precision–Recall curves for each F1 team class, along with the model’s overall mAP@0.5. Each curve depicts the trade-off between precision and recall for a specific class, enabling performance comparisons across teams.

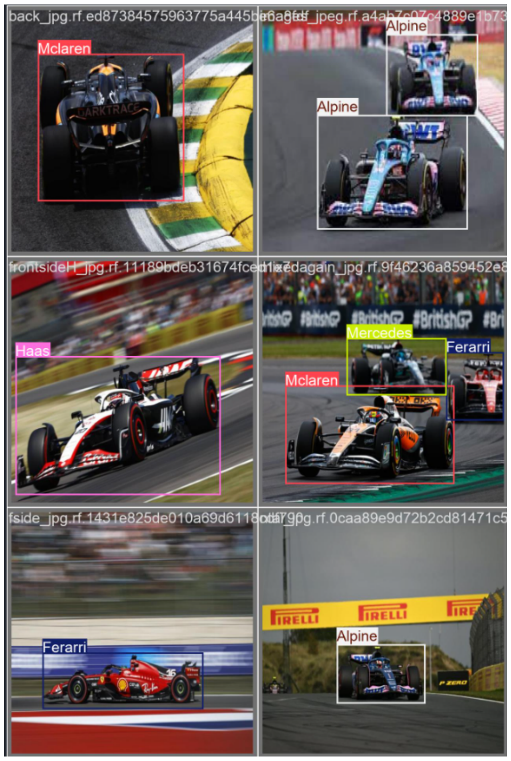


Figure 11. Visual examples of YOLO model detections for various F1 cars. Predicted bounding boxes and class labels are overlaid on real images, illustrating both correct detections and possible misclassifications.

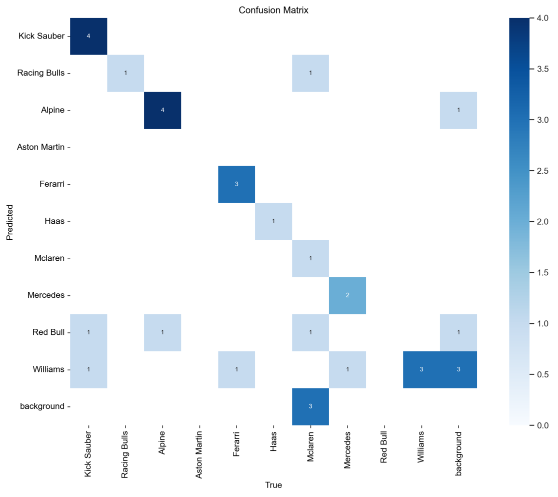


Figure 12. Confusion matrix showing the distribution of correct and incorrect predictions across the different F1 teams. Diagonal entries indicate true positives, while off-diagonal values represent inter-class confusion.

D. Radio Communications Models

1) Sentiment Analysis: The selected model for sentiment analysis is a fine-tuned RoBERTa trained on F1 radio messages, optimized for the categories `positive`, `negative`, and `neutral`. Its performance was evaluated using precision, recall, and F1-score metrics on a manually labeled test set, demonstrating a significant improvement over classical methods such as VADER, particularly in detecting neutral and negative messages. On the held-out test set of 80 examples, the model achieved an overall accuracy of 0.8750, with a weighted average precision of 0.8764, weighted average recall of 0.8750, and a weighted F1-score of 0.8746. Table V presents the full classification report, showing per-class precision, recall, F1-score, and support.

Class	Precision	Recall	F1-score	Support
positive	0.6667	0.7500	0.7059	8
neutral	0.9138	0.9298	0.9217	57
negative	0.8462	0.7333	0.7857	15
accuracy	0.8750			
macro avg	0.8089	0.8044	0.8044	80
weighted avg	0.8764	0.8750	0.8746	80

Table V
CLASSIFICATION REPORT FOR THE FINE-TUNED ROBERTA SENTIMENT ANALYSIS MODEL ON F1 RADIO MESSAGES.

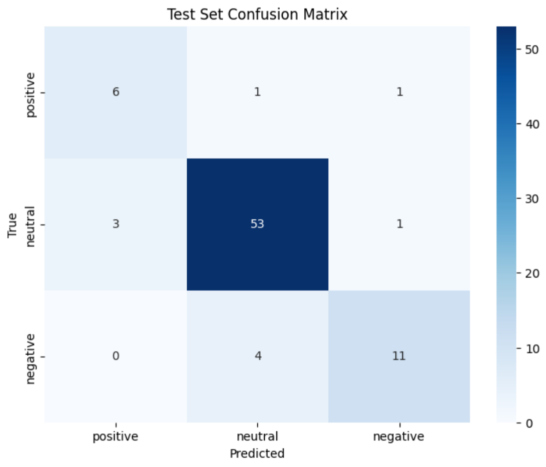


Figure 13. Confusion matrix for the sentiment analysis model on F1 radio messages.

2) Intent Recognition: For intent classification, a DeBERTa-v3-large model was used, trained to distinguish between the classes `INFORMATION`, `PROBLEM`, `ORDER`, `WARNING`, and `QUESTION`. While DeBERTa was selected for detailed evaluation due to its robustness against imbalanced classes and strong performance in specific categories such as `INFORMATION` and `WARNING`, the RoBERTa-large model was ultimately used in the final system. This decision was made because RoBERTa-large achieved a higher number of top-performing F1 scores across individual intent classes, while also offering lower computational costs during inference. `STRATEGY` label was removed due to its near-zero metrics, relabelling these radios inside other categories where they could be suited, like `INFORMATION`.

The confusion matrix in Fig.14 reveals that most misclassifications occur between the `ORDER` and `INFORMATION` classes, as well as occasional confusion between `PROBLEM` and `ORDER`. This suggests that some radio messages contain overlapping linguistic cues, making fine-grained intent separation challenging even for advanced models.

Metric	RoBERTa	RoBERTa (5 cls)	DeBERTa
Overall Accuracy	0.57	0.76	0.76
Weighted F1 Score	0.60	0.76	0.76
INFORMATION	0.66	0.72	0.79 (+0.07)
PROBLEM	0.67	0.82	0.71 (-0.11)
ORDER	0.59	0.77	0.72 (-0.05)
STRATEGY	0.12	N/A	N/A
WARNING	0.44	0.40	0.50 (+0.10)
QUESTION	0.75	0.94	0.89 (-0.05)

Table VI
INTENT CLASSIFICATION PERFORMANCE ACROSS MODEL VARIANTS. IMPROVEMENTS RELATIVE TO THE UPDATED ROBERTA BASELINE ARE SHOWN IN PARENTHESES.

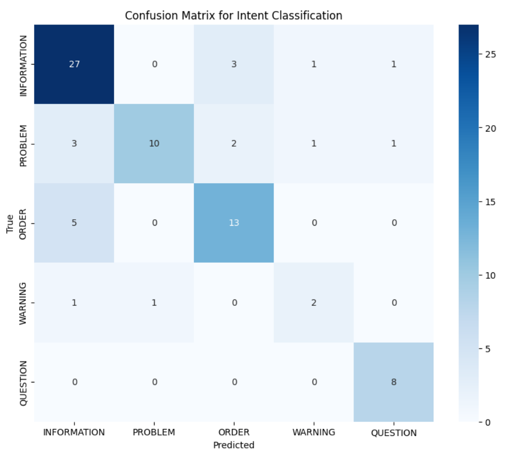


Figure 14. Confusion matrix for intent classification on the radio message test set. Each cell shows the number of predictions for each true/predicted intent pair, highlighting both correct classifications (diagonal) and common confusions between intent categories.

3) Named Entity Recognition (NER): The model selected for entity extraction was a fine-tuned BERT-large, as it demonstrated the most consistent performance across global metrics and entity-specific F1-scores when compared to DeBERTa-v3-large and a general-purpose BERT NER model. As shown in Table VII, the fine-tuned BERT-large achieved an accuracy of 0.4411, precision of 0.4543, recall of 0.4411, and F1-score of 0.4298, outperforming both alternative models in precision and overall F1-score. Table VIII reveals that it was particularly effective in extracting entities such as `ACTION` (F1=0.57), `INCIDENT` (F1=0.22), `TECHNICAL_ISSUE` (F1=0.23), and `SITUATION` (F1=0.30), achieving the highest or equal-highest F1-scores in these categories. Although slightly outperformed in `POSITION_CHANGE` (0.65 vs. 0.66 for BERT NER) and `WEATHER` (0.40 vs. 0.69 for DeBERTa v3), its overall balance between precision and recall justified its selection. Notably, DeBERTa v3 completely failed to detect `INCIDENT` and `TECHNICAL_ISSUE` entities (F1=0.00 for both), as evident in Table VIII. It is important to highlight that these relatively modest metrics are largely attributable to the BIO tagging format implementation, where the model particularly struggles with correctly identifying Beginning (B) and Outside (O) tags while performing considerably better with Inside (I) tags. Empirical testing with real F1 radio communications demonstrates that the model's practical performance exceeds what these quantitative metrics might suggest, making it suitable for real-time strategic entity extraction in race scenarios.

Model	Accuracy	Precision	Recall	F1-score
DeBERTa v3 Large	0.4513	0.4283	0.4513	0.4115
BERT Large NER	0.4199	0.4466	0.4199	0.4229
BERT Large Fine-tuned	**0.4411**	**0.4543**	**0.4411**	**0.4298**

Table VII
OVERALL PERFORMANCE COMPARISON BETWEEN DEBERTA V3, A GENERAL BERT NER MODEL, AND THE FINE-TUNED BERT-LARGE MODEL USED FOR ENTITY EXTRACTION.

Entity Type	DeBERTa v3	BERT NER	BERT Fine-tuned
ACTION	0.42	0.54	**0.57**
POSITION_CHANGE	0.26	**0.66**	0.65
INCIDENT	0.00	**0.22**	**0.22**
TECHNICAL_ISSUE	0.00	**0.26**	0.23
SITUATION	0.16	0.30	**0.30**
TRACK_CONDITION	0.06	0.11	**0.11**
WEATHER	**0.69**	0.44	0.40

Table VIII
F1-SCORE BY ENTITY TYPE FOR EACH EVALUATED MODEL. THE FINE-TUNED BERT-LARGE MODEL OUTPERFORMS OTHERS IN MOST CATEGORIES.

E. Expert System

The rules implemented in this expert system are fully explainable and have been designed to cover the most relevant and basic strategic scenarios in Formula 1 decision-making. Each rule has been constructed to reflect real and interpretable situations for a race engineer: for example, the degradation rules (*high_degradation_pit_stop*, *stint_extension_recommendation*, *early_degradation_warning*, *predicted_high_degradation_alert*) allow for recommendations such as pitting due to high degradation, stint extension when wear is low, or early warnings if a rapid increase in degradation is detected. Lap time prediction rules (*optimal_performance_window*, *performance_cliff_detection*, *post_traffic_recovery*) are used to identify optimal attack windows, detect performance drops due to tire wear, and find recovery opportunities after traffic. Gap management is addressed through rules that detect undercut opportunities (*undercut_opportunity*), defend against possible undercuts (*defensive_pit_stop*), exploit overcuts (*strategic_overcut*), and manage pit lane traffic (*traffic_management*). Finally, radio-based rules (*grip_issue_response*, *weather_information_adjustment*, *incident_reaction*) enable responses to grip issues, weather changes, or on-track incidents. All these rules have been documented and their conditions are transparent, facilitating interpretation and further adjustment.

A quantitative analysis of the number of recommendations generated has been conducted in the section where the results CSV file is loaded and examined, specifically in the cell that processes the `spain_gp_recommendations.csv` file. In this section, the occurrences of each action type are counted and the activation of each rule category is verified, allowing for an evaluation of the system's coverage and behavior in a real scenario. The summary of the number of recommendations per action is presented in Table IX, and specific Driver´s recommendations can be seen in the `Strategy Recommendations` section in the Streamlit app.

It should be noted that the system's performance can be easily adapted, as the precision and usefulness of the recommendations depend directly on the thresholds and conditions defined in the rules. Therefore, the optimal way to evaluate and improve this system would be to involve professional race strategists, who could refine the rules to better reflect expert decision-making. Additionally, rule firing is performed at strategic intervals defined in the notebook, based on key moments of the Barcelona race and historical pit stops (see the definition of `strategic_points` in the `analyze_all_drivers_with_real_radios` function). This mechanism can be easily modified to adapt to other circuits or strategies, enabling advanced customization. These aspects will be further explored in the sections on "Critical Analysis and Limitations" and "Future Improvements".

Action	Number of Recommendations
consider_pit	330
pit_stop	297
extend_stint	281
adjust_pit_window	202
perform_undercut	79
defensive_pit	74
prepare_rain_tires	33
perform_overcut	24

Table IX
SUMMARY OF THE NUMBER OF RECOMMENDATIONS GENERATED FOR EACH ACTION TYPE.

F. Critical Analysis and Limitations

The current expert agent is fundamentally limited by its reliance on a set of basic, pre-defined rules, which, while explainable and transparent, do not capture the full complexity and nuance of real Formula 1 strategic decision-making. The rules have been designed based on knowledge derived from a student's experience as a Formula 1 enthusiast, along with insights gained through the project and the information sources consulted. However, their effectiveness would be significantly enhanced through iterative refinement in collaboration with professional Formula 1 teams. The application itself also presents several areas for improvement, particularly in the frontend interface, certain aspects of data handling, and overall code efficiency. The number and timing of recommendations generated should be further adapted to support true real-time operation and to better reflect the dynamic, context-dependent decision processes observed in actual Formula 1 teams.

A further limitation is that the system has been tailored specifically for the Spanish Grand Prix in Barcelona, primarily due to the vast amount of data required for robust rule extraction and validation. For broader applicability, the system should be extended to cover additional circuits and race scenarios.

Another critical aspect concerns the integration of large language models (LLMs) for explanation and visualization. The current use of Ollama 3.2 Vision has shown some hallucinations when interpreting and explaining graphs, indicating a need for either substantial prompt engineering or replacement with a more robust LLM, such as a GPT-based model. However, hardware constraints (the experiments were conducted on a laptop equipped with an AMD Ryzen 9 8954HS CPU and an RTX 4060 Laptop GPU with 8GB VRAM) and the financial cost of commercial API usage have limited the deployment of larger models and the size of datasets used for training and evaluation. On more powerful machines, it would have been possible to employ larger datasets and models with more parameters, potentially improving both performance and generalization.

These limitations provide a clear roadmap for future improvements, which will be discussed in the following section.

G. Future Improvements

All the aforementioned limitations define the main directions for future improvements of the system. It is intended that more powerful models will be incorporated, leveraging larger and more diverse datasets and that the codebase will be further optimized for efficiency and maintainability. Enhancements to the frontend interface and data handling routines are also planned, aiming to provide a more robust and user-friendly experience. Real-time recommendation capabilities will be developed, allowing the system to adapt dynamically to evolving race conditions and team decisions. Furthermore, the scope of the system will be expanded to include all circuits on the Formula 1 calendar, thereby increasing its generalizability and practical value.

A key objective will also be to refine and significantly expand the rule base, ideally incorporating feedback and recommendations from professional Formula 1 teams to better capture the complexity of real-world strategies. These improvements are expected to be addressed in the context of the student's undergraduate thesis project (TFG), subject to the resources and knowledge available at the university and to the student's continued learning and development.

V. CONCLUSION

An explainable expert system for Formula 1 race strategy has been developed in this work, where rule-based reasoning is integrated with multiple data-driven AI approaches. Specialized modules for tire degradation prediction, lap time modeling, gap analysis, and natural language processing of team radio communications have been combined into a unified decision support framework. This integration enables comprehensive strategy generation that considers multiple factors simultaneously, similar to how strategic decisions are made by professional race engineers.

The system was validated using real data from the 2023 Spanish Grand Prix, which demonstrated that actionable and interpretable recommendations can be generated at key strategic moments. Clear traceability of decision logic is ensured by the transparent rule base, while complex race data is made accessible to users of varying technical backgrounds through the interactive visualization dashboard. Explainability and adaptability are emphasized in this system, unlike black-box approaches in previous research, providing insights into not just what strategic actions should be taken, but why they are recommended.

The main contributions of this project can be identified as: (1) a novel multi-model approach where previously separate domains of prediction and decision-making are bridged; (2) improved methods for tire degradation quantification with fuel adjustment factors; (3) a comprehensive rule-based expert system through which multiple data streams are processed simultaneously; and (4) Formula 1 domain knowledge that has been mapped into explicit, interpretable rules that can be refined over time. Applications of the system can be extended to team strategy operations, post-race analysis, educational platforms for race engineering students, and enhanced viewing experiences for motorsport enthusiasts. Adaptation to other motorsport categories or integration with more advanced AI components is facilitated by the system's modular architecture.

Future research should be focused on expanding the system to all circuits on the F1 calendar, incorporating weather prediction models, and developing real-time adaptation capabilities. The rule base should be refined through collaboration with Formula 1 strategists, competitor strategy prediction should be implemented, LLM capabilities should be enhanced through fine-tuning, and more complex models should be deployed with expanded computational resources. Through these advancements, the system will be brought closer to becoming a professional-grade decision support tool for Formula 1 strategy operations.

References

[1] M.-P. Neumann, G. Fieni, C. Balerna, P. Duhr, A. Cerofolini, and C. H. Onder, "Low-level online control of the formula 1 power unit with feedforward cylinder deactivation," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 8382–8397, 2023.

[2] A. Patil, N. Jain, R. Agrahari, M. Hossari, F. Orlandi, and S. Dev, "A data-driven analysis of formula 1 car races outcome," in *Artificial Intelligence and Cognitive Science (AICS 2022)*, ser. Communications in Computer and Information Science, vol. 1662, 2023, pp. 134–146.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-Level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] A. Heilmeier, M. Graf, J. Betz, and M. Lienkamp, "Application of Monte Carlo methods to consider probabilistic effects in a race simulation for circuit motorsport," *Applied Sciences*, vol. 10, no. 12, p. 4229, 2020.

[5] X. Liu, A. Fotouhi, and D. J. Auger, "Formula-E race strategy development using distributed policy gradient reinforcement learning," *Knowledge-Based Systems*, vol. 216, p. 106781, 2021.

[6] A. Patil, N. Jain, R. Agrahari, M. Hossari, F. Orlandi, and S. Dev, "A data-driven analysis of formula 1 car races outcome," in *Artificial Intelligence and Cognitive Science (AICS 2022)*, ser. Communications in Computer and Information Science, vol. 1662, 2023, pp. 134–146.

[7] M. Boettinger and D. Klotz, "Mastering nordschleife: A comprehensive race simulation for ai strategy decision-making in motorsports," *CoRR*, vol. abs/2306.16088, 2023. [Online]. Available: https://arxiv.org/abs/2306.16088

[8] A. Heilmeier, M. Graf, and M. Lienkamp, "A race simulation for strategy decisions in circuit motorsport," in *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC 2018)*, 2018, pp. 2986–2993.

[9] A. S. Ali, T. Abuhmed, S. H. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, and A. Ortigosa, "Explainable artificial intelligence: What we know and what is left to attain trustworthy ai," *Information Fusion*, vol. 99, p. 101805, 2023.

[10] S. Dandl, C. Molnar, M. Binder, and B. Bischl, "Multi-objective counterfactual explanations," in *Parallel Problem Solving from Nature – PPSN 2020*, ser. Lecture Notes in Computer Science, vol. 12269. Springer, 2020, pp. 448–469.

[11] M. Piccinotti, R. Arcucci, and J. Ward, "Monte carlo tree search for optimal pit-stop scheduling in formula 1," *IEEE Transactions on Games*, vol. 13, no. 2, pp. 145–156, 2021.

[12] A. Heilmeier, D. Linke, and M. Lienkamp, "Virtual strategy engineer: Neural network–based pit-stop decision support in motorsport," in *2020 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2020, pp. 1–6.

[13] P. Aversa, L. Cabantous, and S. Haefliger, "When decision support systems fail: Insights for strategic information systems from formula 1," *Journal of Strategic Information Systems*, vol. 27, no. 3, pp. 221–236, 2018.

[14] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," in *Advances in Neural Information Processing Systems (NeurIPS) 2018*, 2018, pp. 2499–2509, arXiv:1802.03403.

[15] S. S. W. Fatima and J. Johrendt, "Deep-racing: An embedded deep neural network (ednn) model to predict the winning strategy in formula one racing," *International Journal of Machine Learning*, vol. 13, no. 3, pp. 97–103, 2023.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NeurIPS) 2017*, 2017, pp. 4765–4774.

[17] O. F. Carrasco Heine and C. Thraves, "On the optimization of pit stop strategies via dynamic programming," *Central European Journal of Operations Research*, vol. 31, no. 1, pp. 239–268, 2023.

[18] 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, p. 7805, 2020.

[19] R. Jowett, "Fuel-Saving Secrets: Why F1 Cars Conserve Fuel While F2 Cars Go All Out," Canopy Platform, Michelin Simulation Services, Oct. 2024, accessed: 2025-05-07. [Online]. Available: https://simulation.michelin.com/canopy/technical-articles/fuel-saving-secrets-why-f1-cars-conserve-fuel-while-f2-cars-go-all-out

[20] K. Benzing, "Weight versus lap times," The Technical Forum Archive (Autosport Forums), Jul. 2003, posted July 09, 2003; Accessed: 2025-05-07. [Online]. Available: https://forums.autosport.com/topic/58983-weight-versus-lap-times/

[21] Anonymous, "How much time does a lap of fuel cost?" Autosport Forums, Apr. 2009, posted April 21, 2009; Accessed: 2025-05-07. [Online]. Available: https://forums.autosport.com/topic/93560-how-much-time-does-a-lap-of-fuel-cost/

[22] 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, no. 4, pp. 15 191–15 207, 2020.

[23] P. Team, "Pytorch documentation," https://pytorch.org/docs/, 2023, accessed: 2023-05-08.

[24] S. Team, "Streamlit documentation," https://docs.streamlit.io/, 2023, accessed: 2023-05-08.

[25] E. Team, "Experta documentation," https://experta.readthedocs.io/, 2023, accessed: 2023-05-08.

[26] O. Team, "Opencv documentation," https://docs.opencv.org/, 2023, accessed: 2023-05-08.

[27] F. 1, "Circuit de barcelona-catalunya 2023 information," https://www.formula1.com/en/racing/2023/spain, 2023, accessed: 2023-05-08.

[28] J. Team, "Jupyter widgets documentation," https://ipywidgets.readthedocs.io/en/stable/, 2023, accessed: 2023-05-08.

[29] R. Team, "Roboflow documentation for dataset download," https://roboflow.com/docs, 2023, accessed: 2023-05-08.