F1 Strat Manager

Revolutionising strategic decision-making in Formula 1 racing through advanced technology integration.

The Strategic Challenge

Main Problem

Optimizing Strategic Decissions during the race

Multiple Variables

Weather, tyres, fuel, track position affect strategy.

Complex Decisions

Teams must process vast data quickly.

Automated Solutions

Necessity to integrate advanced prediction technologies for intelligent solutions

Objective: create a strategy recommendator

Through available data, create predictive models for the most important variables during the race, used by a logical agent that provides strategic decissions to the team.

Project Foundation

Data Extraction: files to get the data

- FastF1.
  - Weather data.
  - Track data.
  - Race data.
- OpenF1.
  - Radios.
  - Gap intervals.
- Roboflow.
  - Team identification Dataset.

Processing, Cleaning, Labelling,

Robust data preparation pipelines

- Key for correct training.
- Created synthetic variables.
- Eliminated uncessary variables.
- Modified data depending of its nature.
- Manual radio labelling: sentiment, intention, NER,.

Technical Implementation

Jupyter Notebook and Python framework

- Each discipline with a directory: separated work.
- Pytorch implementation: all local, cuda drivers.
- Experta framework for Expert System.
- Pandas and other data managing libraries.

Data Integration & Augmentation

- Merge all FastF1 files into a dataframe.
- In depth data exploration.
- Augment the datasets: more radios, more images.

Model exploration

- YOLO for computer vision.
- Xgboost for lap time predictions.
- BERT-based models for NLP.
- Deep Learning models: degradation prediction.
- Expert System for strategy recommendations.

App development

- Framework: Streamlit.
- One section per model.
- User data exploration, graphs, explanations.
- LLM for chat, image analysis and report creations for post-race briefings.

Computer Vision: YOLO

1 Data Augmentation

- Only 500 images with unbalanced classes.
- Agumented 250 images each team. Then, 2500 images.
- Tenchniques:
  - Horizontal flip, 50% probability.
  - Shift, scale, rotate.
  - Gaussian Blur.

2 YOLO Model Training: First models

- Initial with 200 images per class instead of 250.
- Metrics:
  - mAP50: **0.772**, mAP50-95: **0.449**.
  - Precission: **0.733**, Recall: **0.615**.
- Classes with worst performance:
  - Williams (mAP50: 0.487).
  - Kick Sauber (Recall: 0.443)

3 New models: YOLO medium

- Hyperparameter adjustment and data augmentation.
- Improved metrics by 20%.
- Still bad performance in conflictive classes.

4 Final model: Fine-Tuned Yolo Medium with weight adjustment

- Model with best balance.
- Improved in problematic classes by 30%, others like Red Bull with 99% mAP50.
- Adjusted weights to give more importance to problematic classes.

5 Video implementation & Gap Calculation

- Pixel distance: calculated distances between the centroids of YOLO boxes.
- Scale: distance of pixels converted to meters using F1 car measurements (5.83 meters long).
- Time calculation: based on a 300km/h velocity, obtained distance between cars.

Computer Vision: video information extraction

Controls

- Adjustable detection threhold: the larger, more confidence needed by the model to catch the car.
- Added "F1 Car" common label if not enough confidence.
- Controls for skipping parts of the video

Constant extraction

- Adjustable time extraction .
- Every x seconds, through OpenCV is extracted:
  - Two nearest detected cars.
  - Gap in seconds between them.
  - Their teams.
  - Frame in which the data was taken.
- Important data for our gap analysis.

Advanced Machine Learning: Data preparation

1

Merging all the data

Need to merge all the data files into a single, huge Dataframe.

43 variables and 1312 rows regarding.

- Weather data.
- Track conditions.
- Gaps and Driver´s data.

2

Selecting Important Variables

- Need to select variables and make initial cleaning.
- All models will strart from this final CSV.
- If necessary, more data is added for a specific model

3

Sequentiating the data: taking advantage

Important for our models to perform well.

- F1 data: each driver, all laps, divided by stints.
- Stint: laps with a specific tire compound.
- Sequence by stint, by driver, and by 5 laps per stint.

Short Model´s background

XgBoost

- Advanced gradient boosting algorithm for structured data.
- Sequential ensemble of prediction trees that learn from previous errors.
- High accuracy, speed, scalability for tabular data problems.

Temporal Convolutional Network (TCN)

- Neural network specialized for sequential data.
- DIlated causal convolutions for temporal pattern recognitions.
- Outperfors RNNs/LSTMs

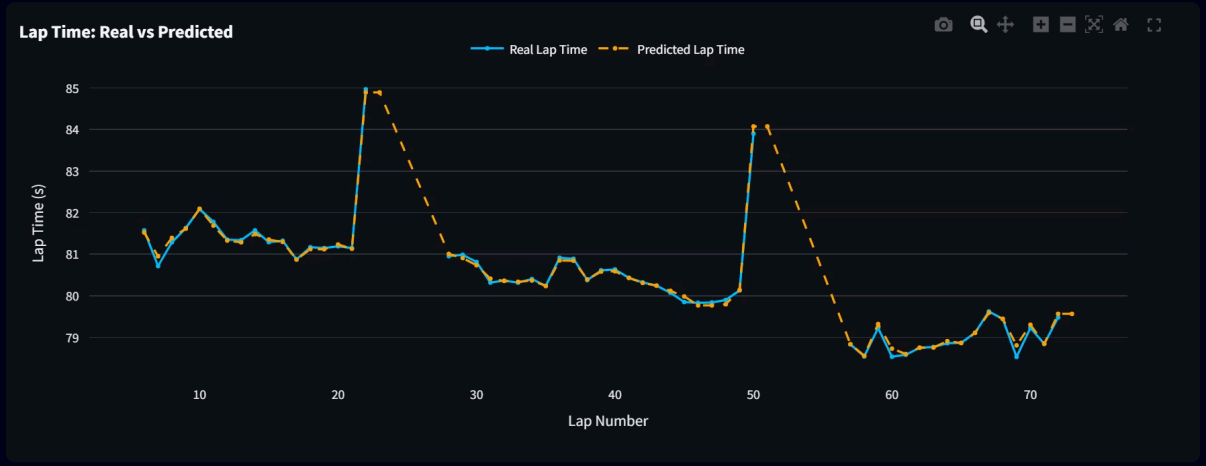
Advanced Machine Learning: Lap Prediction

1 Purpose

- Be able to predict the Driver lap times based on 25 variables with the least error possible.
- Barcelona 2023 Grand Prix.
- Tire age, position, team, DRS, stint, fuel, etc.

2 First XgBoost: not bad, but could be better.

- No sequential data.
- MAE: 0.3 seconds. RMSE: 0.45 seconds.
- R2 0.91
- 26 meters error in 4.67km track



3 LSTM: important notice

- Negative R2, worst than applying the mean
- However, sequentiating the data could improve the XgBoost

4 Second XgBoost

- 71% improvement.
- MAE: 0.09 seconds. RMSE: 0.15 seconds.
- 9.15 meters error in 4.67km.

Advanced Machine Learning: Tire Degradation

1

What is degradation? My purpose

Degradation occurs due to the friction between the tires and the circuit.

Its estimation is in seconds lost per lap, comparing to the first laps with that tire.

2

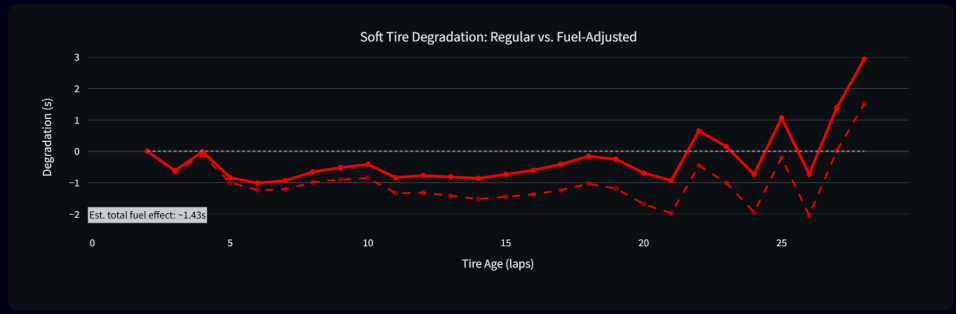
Tricky Data: created new variables.

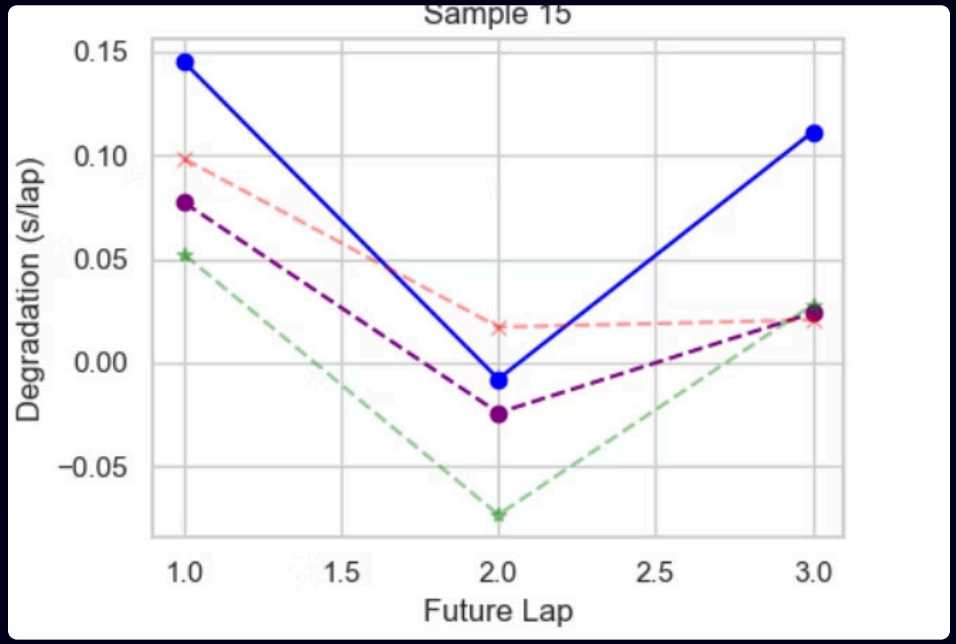
- Degradation is extremelly susceptible to data.
- Does not have recognizable patterns.
- Hard to predict, even for big models.

3

Model Creation

- Tried to predict the tendency.
- TCN: temportal convolutional network
- One global model, 3 others per tire compound.
- Is able to capture the tendency and also the time lost.





NLP: Radio Communication Analysis Pipeline

Audio Collection & Labelling

Capturing team-driver communications through Open F1

Strategic Insights

Saving analysis in JSON format for the Expert System.

Speech Processing

Converting audio to text through Whisper Turbo

NLP Analysis

Extracting key information:

- Sentiment.
- NER.
- Intention.

NLP: Audio extraction and Data labelling.

API Petition

- Almost 800 radios from 4 different races.
- Script for making 4 petitions at once and store each mp3 file for correspondant driver.

Data Processing & Labelling

- Identified end-of-race radios.
  - Normally summary of the race.
  - Most of them positive opinions and long duration (too much tokens to process)
  - Need to eliminate them for not having unbalanced classes.
- One script per task.
  - Sentiment labelling.
  - Intention labelling.
  - NER labbeling.

Speech to text Processing

- Call to Whisper Turbo.
- Based on encoder-decoder architecture.
- Uses log-mel spectogram to catch human voice patterns.

Models: BERT alternatives

- Transformer-based model utilizing bidirectional encoders to process text in both directions simultaneously.
- Pre-trained on massive text corpora using masked language modeling and next sentence prediction tasks.
- NLP revolution, enabled contextual understanding of language.

NLP: Sentiment

1 Objective

Implement a sentiment analisys for analyzing radio communications.

Tried NLTK Vader for initial exploration.

Using RoBERTa-base: key optimization changes and improved training (10x more training data).

- Positive.
- Negative.
- Neutral.

2 Problems

- Class imbalance.
- Majority of neutral messages.
- 530 labelled radios after eliminating end-of-race radios. Small dataset.
- Elevated computational cost for training from scratch.

3 Solution

- Fine-tuned a pre-trained RoBERTa-base to adapt to Formula 1 radios.
- Weight class implementation for managing class imbalance.
- 128 max tokens for clipping, 97th percentile.

4 Final metrics

- Mean precission of 87% on training set.
- Macro-avg F1 score: 80.44%.
  - Makes the mean of individuals F1 score per class.
- Only 6 epochs: significative computanional cost reduction.

NLP: Intention

1 Objective

Classify radio into specific intentions for enhancing the radio analysis.

Labels:

- ORDER: direct instructions.
- INFORMATION: factual updates.
- QUESTION: questions to the driver.
- WARNING: alerts or warnings.
- PROBLEM: problems reported by the driver.
- STRATEGY (eliminated).

2 Problems

- Required manual labelling.
- Class imbalance: WARNING and STRATEGY with few examples.
- Two simillar models, need to choose which classes are most important for the analysis.
- STRATEGY had almost 0% precission. As it could be in other categories, like INFORMATION, it was finally deleted.

3 Solution

- Trained a RoBERTa-large with STRATEGY class deleted.
- Also trained a DeBERTa-v3-large.
- Class weights and label smoothing for managing imbalance.
- 12 epochs training and meticulous learning rate and weight decay adjustment.

4 Final metrics

Stick to RoBERTa: computationally faster, lighter and best individual class metrics.

- RoBERTa-large:
  - Improved in 19% weighted F1-score. 76%
  - Classes with 94%, WARNING still problematic with 40%.
- DeBERTa-v3-large:
  - Weighted F1-score same as RoBERTa.
  - Improved WARNING to 50%.
  - Decreased F1-score in PROBLEM, ORODER and QUESTION.

NLP: Named Entity Recognition (NER)

1 Objective

Make a NER system for extracting stuctured information from radios.

Labels:

- ACTION, SITUATION,, INCIDENT.
- STRATEGY_INSTRUCTION, POSITION_CHANGE.
- PIT_CALL, TRACK_CONDITION.
- TECHNICAL_ISSUE, WEATHER.

2 Problems

- Need to tag in BIO format: manual conversion.
- Class imbalance.
- Possible hardware limitations.
- Metrics appear low, but due to BIO tagging format.
- Need to test in real radios to diagnose the low BIO metrics.

3 Solution

- BIO preprocessing implementation. Beginning-Inside-Outside
- Tried DeBERTa, BERT-large adapted, fine-tuned bert.
- Model: fine-tuned BERT-large with aditional training on challenging classes.
- Regularization, loss function adjustments, class weights.

4 Final metrics

- 0.4298 F1-score.
- Balanced NER across all classes.
- Improved performance in challenging classes.
- Better generalization.
- Testing in radios, seems to have better performance than what metrics say.

NLP: merging into a single analysis

MP3 file → Whisper → Calling the models → Config the output → JSON merged for expert system


```
{
  "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."
      ]
    }
  }
}
```

Intelligent Strategy Pipeline

Strategy Development

Select Experta as the framework for building the agent.

Calling all the models and making the inference

Making predictions for data per lap and passing it to the agent.

Get a recommendations csv with all the strategy rules fired.

Agent Setup

One class per predictive task, with the data Facts to take.

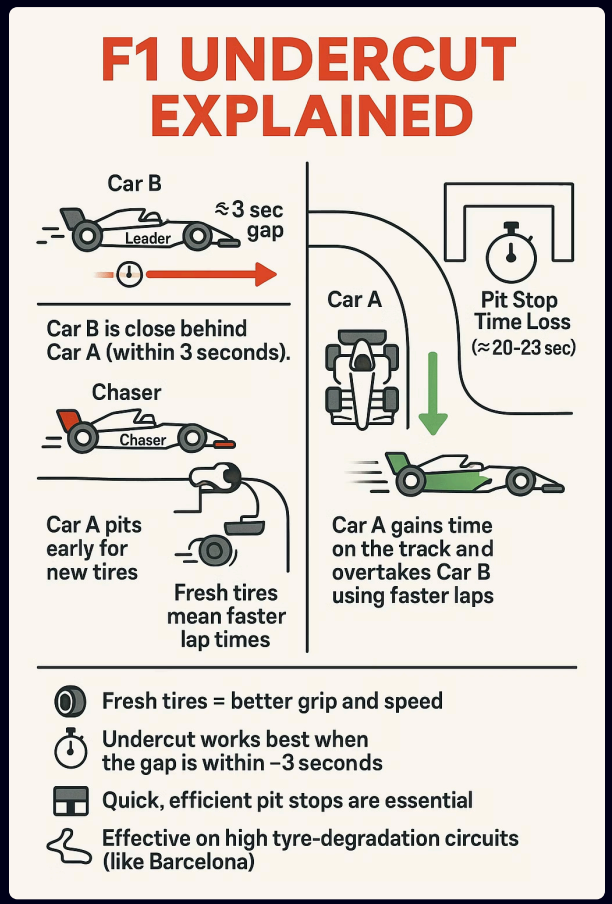
Rule Building

3 rules per task: NLP, Tire Degradation, Lap Prediction, Gap analysis.

Constant testing

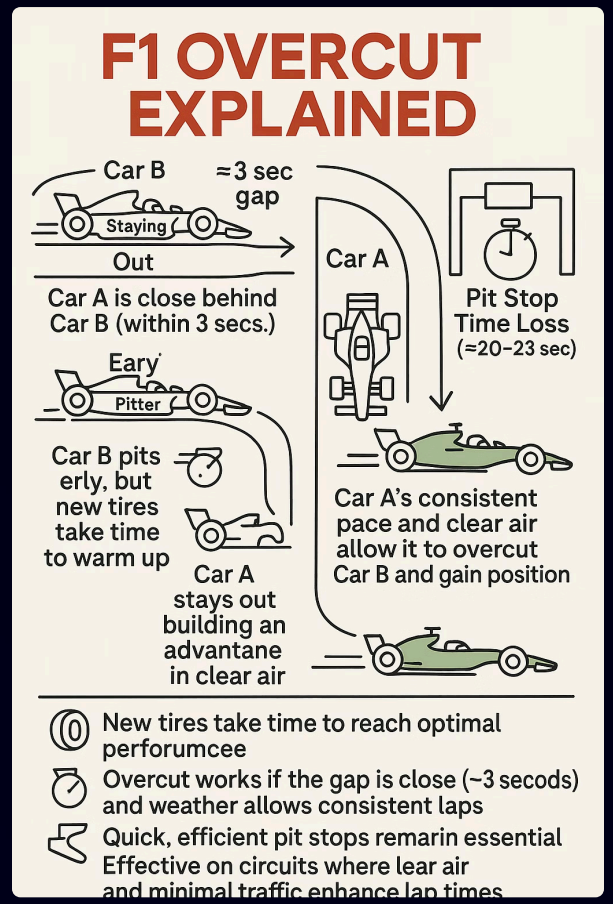
Identifying highest-probability winning strategies

Key Concepts to know:



Undercut

- Early pit strategy - Car A pits for fresh tires while rival continues on worn tires
- Speed advantage - Fresh tires enable faster laps to overcome pit stop time loss
- Best conditions - Gap under 3 seconds, high tire degradation circuits



Overcut

- Delayed pit strategy - Car A stays out while rival pits first
- Clear air advantage - Car A maximizes pace while rival's new tires warm up
- Best conditions - Consistent weather, minimal traffic, tracks where clean air matters

The rules.

1 Radio Communications.

- **Grip Problems Response Rule**: Negative sentiment + mentions of grip problems or related technical entities (85% confidence).
- **Weather Information Adjustment Rule**: Entities related to rain or wet track detected in messages (90% confidence).
- **Incident Reaction Rule**: "Safety car" or "yellow flag" mentioned in messages (85% confidence).

2 Lap Time Predictions

- **Optimal Performance Window Rule**: Predicted lap time < current, position > 3 and tires with less than 8 laps (75% confidence).
- **Critical Performance Detection Rule**: Predicted time > current + 0.7s and tires with more than 15 laps (85% confidence).
- **Post-Traffic Recovery Rule**: Predicted time < current - 0.5s and position worsened in the last lap (70% confidence).

3 Tire Degradation-Based Rules

- **High Degradation Rule**: Degradation rate > 0.15 and tires with more than 10 laps (85% confidence).
- **Stint Extension Rule**: Degradation rate < 0.08, tires with more than 12 laps and position < 5 (75% confidence).
- **Early Degradation Warning Rule**: Increase in degradation rate > 0.03 for 3 consecutive laps (70% confidence).

4 Gap Analysis-Based Rules:

- **Undercut Opportunity Rule**: Gap with car ahead < 2.0s, consistent for 3 consecutive laps (85% confidence).
- **Defensive Stop Rule**: Gap with car behind < 2.0s, consistent for 3 consecutive laps (80% confidence).
- **Strategic Overcut Rule**: Gap with car ahead between 2.0s and 3.5s, consistent for 4 consecutive laps (80% confidence).
- **Traffic Management Rule**: Position > 10, gap with leader > 30.0s (70% confidence).

How the agent works

1 What is Experta? Declarative vs Imperative Programming

Based on Expert Systems.

RETE algorithm to optimize pattern coincidence between facts and rules.

- Declarative (Experta).
  - **What** needs to be made.
- Imperative.
  - **How** it is done.

2 Key Components

- Knowledge Base.
  - Stores factual knowledge (actual data).
- Rules base:
  - Stores procedimental knowledge.
- Inference motor:
  - What rules to trigger and when.

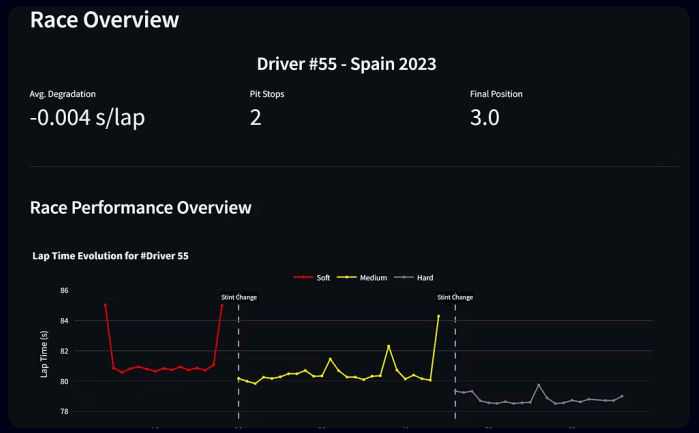
3 Execution Cycle

$$Match \rightarrow Conflict\ resolution \rightarrow Act \rightarrow Repeat$$

4 RETE

- **Optimizes** pattern matching between facts and rules.
- Builds a network of nodes to represent patterns.
- **Avoids reevaluating all rules** when facts change.
- Acts as an efficient filter to determine which rules should be activated.

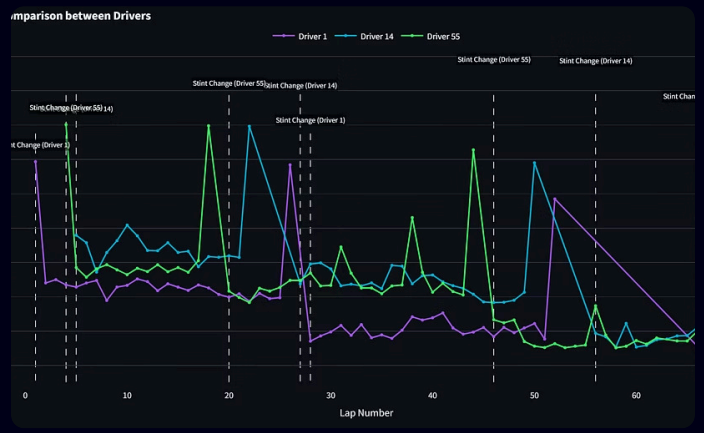
Streamlit Application Interface: Demo



Main Dashboard

Interactive race overview with key metrics, final position, recommendations, strategy simulators, etc.

Choose a driver to see their data

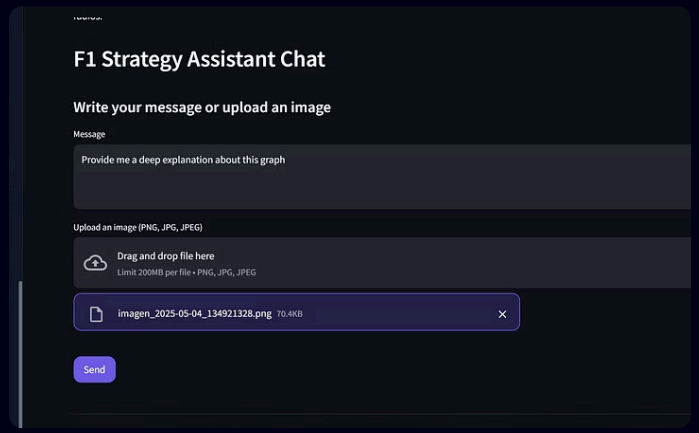


Graphs, comparisons, prediction tools.

Visual representation of compound performance.

Compare position changes, lap times and other metrics by driver.

Different graphs for data analysis.



Ollama 3.2-vision chat and Report Export

Chat for analyzing strategies, graphs, etc.

Tunable parameters like temperature.

Can make reports for post-race briefings with explanations, graphs, etc.