Predizione dei risultati di una gara di Formula1 con AI

Report progetto AI Lab

A.A. 2024/2025

Autori:

Cinieri Giovanni
2054772

Gautieri Alessandro
2041850

Abstract

L'analisi predittiva nel motorsport, e in particolare nella Formula 1, rappresenta una sfida complessa data la natura dinamica e multifattoriale delle competizioni. Questo progetto si propone di sviluppare un sistema innovativo per la predizione della posizione finale di ciascun pilota in un Gran Premio di Formula 1. Sfruttando dati telemetrici, cronometrici e contestuali (come i rating Elo di piloti e team) relativi ai primi giri di gara, è stato implementato un modello predittivo principale (`Modello 1`) basato su un ensemble di algoritmi di **Gradient Boosting** (**LightGBM** e **CatBoost**). Parallelamente, è stato sviluppato un secondo modello (`Modello 2`) per la predizione pre-gara, utilizzando informazioni quali la griglia di partenza, i rating Elo e le performance storiche dei piloti e dei team su specifici circuiti. Entrambi i sistemi sono stati integrati in un'interfaccia utente interattiva (GUI) realizzata con Gradio, permettendo l'esplorazione delle predizioni per gare concluse e future. Il progetto mira a dimostrare l'accuratezza di tali approcci e a fornire uno strumento pratico per l'analisi delle competizioni.

INTRODUZIONE

L'ambito delle competizioni motoristiche, con la Formula 1 come sua massima espressione, è un terreno fertile per l'applicazione di tecniche avanzate di analisi dati e apprendimento automatico. La capacità di predire l'esito di una gara, o di comprendere le dinamiche che portano a un determinato risultato, non solo riveste un grande interesse per appassionati e addetti ai lavori, ma può anche fornire spunti strategici di rilievo.

La Formula 1 genera una mole immensa di dati ad alta frequenza, che spaziano dalla telemetria delle vetture ai tempi sul giro, dalle condizioni ambientali alle strategie di gara. Tradizionalmente, le analisi si sono concentrate su statistiche post-gara o su modelli qualitativi. Tuttavia, con l'avanzamento delle capacità computazionali e la disponibilità di librerie specializzate, è oggi possibile approcciare il problema in modo più quantitativo e dinamico. Lavori precedenti in contesti sportivi simili hanno esplorato l'uso di serie storiche, reti neurali e modelli basati su alberi decisionali per la predizione di performance [(Wiechno et al., 2023), (Wang et al., 2022), (Yang et al., 2021)].

Una sfida comune risiede nell'integrare efficacemente la grande eterogeneità dei dati e nel catturare la natura evolutiva delle performance di piloti e team. Sistemi di rating dinamici come l'Elo, originariamente sviluppato per gli scacchi, sono stati adattati con successo ad altri sport per quantificare la forza relativa dei competitori nel tempo[(Bunker et al., 2023), (Hvattum & Arntzen, 2010)].

Il presente progetto si inserisce in questo contesto con l'obiettivo di costruire un sistema predittivo per la Formula 1 che operi su due distinti scenari:

i) Predizione della classifica finale di una gara già dopo i primi giri, utilizzando dati telemetrici e cronometrici dettagliati (`Modello 1`).

ii) Predizione della classifica finale prima dell'inizio della gara, basandosi su informazioni quali la griglia di partenza, i rating di forza (Elo) e lo storico delle performance sul circuito (`Modello 2`).

Il `Modello 1`, progettato per la predizione durante lo svolgimento della gara, opera attualmente su dati storici di competizioni concluse. Questa scelta è dettata dalla natura accademica del progetto e dalla conseguente indisponibilità di un flusso di dati telemetrici e cronometrici in tempo reale.

METODO

Il progetto è stato articolato in diverse fasi, dalla raccolta dati alla costruzione dei modelli e all'integrazione nell'interfaccia.

Acquisizione e Costruzione dei Dataset

Dataset Modello 1

La base dati del progetto è costituita dai dati ufficiali dei Gran Premi di Formula 1. Per l'acquisizione di questi dati, è stata impiegata la libreria Python open-source **FastF1** [(Schaefer, 2025)]. Questa libreria permette di accedere a dati storici e dati di gare attuali relativi a tempi sul giro, telemetria delle vetture (velocità, input del pilota come acceleratore e freno, marcia inserita, uso del DRS), informazioni sui pit stop, e composizione delle mescole degli pneumatici. Sono state considerate le stagioni 2023, 2024 e le gare disputate della stagione 2025. Per ciascuna gara, sono stati processati i dati relativi ai primi 30 giri. L'unità di analisi fondamentale è la performance di un singolo pilota in un dato giro. Lo script `build_dataset.py` (e la sua variante per le gare del 2025 `build_dataset_current_season.py`) esegue questo processo, generando un dataset tabulare dove ogni riga corrisponde a: `(anno, Gran Premio, numero del round, numero del giro, pilota, team) => feature_1, …,`

`feature_N, posizione_finale`. È stato deciso di dividere la costruzione del dataset in due script separati: uno dedicato agli anni precedenti e uno dedicato all'anno corrente. In questo modo, dopo ogni gara è possibile aggiornare il dataset dell'anno attuale, senza il bisogno di ricalcolare quello degli anni precedenti. Questa ottimizzazione è necessaria: senza di essa si impiegherebbe parecchio tempo per rigenerare dati uguali ai precedenti (circa 25k righe).

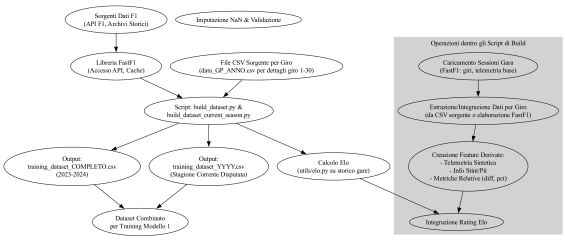

Figure 1: Illustra la pipeline di creazione del dataset.

Feature Overview

- **Dati Cronometrici di Base**: `lap_time`, `sector1_time`, `sector2_time`, `sector3_time`.
- **Dati Relativi al Leader e alla Competizione**:
  - `gap_to_leader_diff`: Differenza (in secondi) rispetto al tempo sul giro del leader in quel giro.
  - `lap_time_pct`: Percentile del tempo sul giro del pilota rispetto a tutti gli altri piloti in quel giro.
  - `gap_to_leader_pct`: Percentile del gap cumulativo dal leader rispetto a tutti gli altri.
- **Telemetria Sintetica** (statistiche aggregate per giro):
  - `speed_avg`: Velocità media.
  - `throttle_pct`: Percentuale di tempo con acceleratore oltre l'80%.
  - `brake_pct`: Percentuale di tempo con freno attivo.
  - `drs_pct`: Percentuale di tempo con DRS attivo.
  - `gear_avg`: Media della marcia inserita.

  Queste feature vengono calcolate anche come valori differenziali rispetto al leader del giro.
- **Informazioni sullo Stint e Pit Stop**:
  - `stint`: Numero dello stint attuale.
  - `lap_in_stint`: Numero di giri completati nello stint.
  - `pit_in`: Flag booleano se il pilota è entrato ai box in quel giro.
  - `pit_out`: Flag booleano se il pilota è uscito dai box in quel giro.
  - `csv_pit_stop_count`: Conteggio dei pit stop fino a quel momento.
- **Informazioni Contestuali**:
  - `compound`: Mescola pneumatico.
  - `track_status`: Stato della pista (es. Safety Car).
- **Rating Dinamici Elo** Per quantificare la forza relativa e lo stato di forma di piloti e team nel corso delle stagioni, è stato implementato un sistema di rating Elo dinamico (logica definita in `utils/elo.py`). Dopo ogni Gran Premio, il rating Elo di un pilota R_A viene aggiornato confrontando il suo piazzamento con quello di ogni altro pilota R_B partecipante, utilizzando la seguente formula di aggiornamento:

$$R_{A'} = R_A + Kc \cdot gc \cdot (S_A - E_A)$$

dove:

- $S_A \in \{0, 0.5, 1\}$ rappresenta il punteggio effettivo ottenuto da R_A contro R_B (1 per vittoria, 0.5 per parità di piazzamento – sebbene raro –, 0 per sconfitta).
- $E_A = \frac{1}{1+10^{\frac{R_B - R_A}{400}}}$ è il punteggio atteso per R_A contro R_B, basato sulla differenza dei loro rating pre-gara.
- K è il "K-factor", un parametro che determina la massima variazione possibile del rating. Nell' implementazione, K è dinamico, assumendo valori maggiori all'inizio della stagione o per piloti con meno storico, per permettere una più rapida convergenza del rating.
- g è un fattore di moltiplicazione che scala l'aggiornamento in base al gap temporale tra i due piloti al traguardo, amplificando l'impatto di vittorie o sconfitte con distacchi marcati.

Il sistema include inoltre un meccanismo di **decay** percentuale applicato ai rating a fine di ogni stagione, per tenere conto della potenziale perdita di forma o dei cambiamenti nelle dinamiche competitive tra un campionato e l'altro, e specifiche penalità Elo vengono applicate in caso di ritiri dovuti a guasti meccanici.

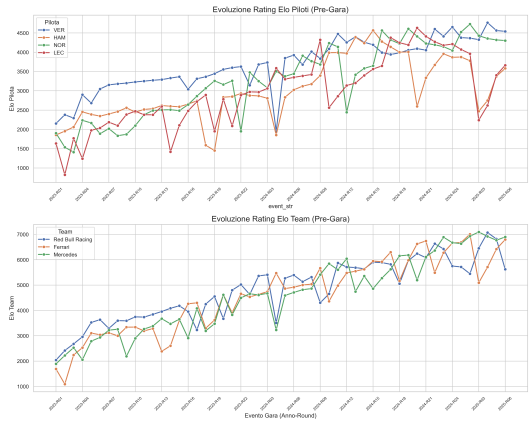

Figure 2: In questo grafico è mostrato l'andamento dell'elo di alcuni piloti e team nella stagione attuale.

Il dataset risultante per il `Modello 1` (`training_dataset_COMPLETO.csv` e il suo aggiornamento per il 2025) contiene circa 28.000 righe, con ogni riga che rappresenta lo stato aggregato delle performance di un pilota a un dato giro.

Dataset Modello 2

Per il `Modello 2` (predizioni pre-gara), lo script `build_dataset_pre_race.py` genera un dataset differente (`pre_race_prediction_dataset.csv`).
Qui, ogni riga rappresenta un pilota prima dell'inizio di una gara.

Le feature includono:

- `starting_grid_position`.
- `elo_driver_pre_race` e `elo_team_pre_race` (ottenuti dallo storico calcolato da `utils/elo.py.compute_elo_history`).
- Statistiche aggregate delle performance del pilota e del team su quel specifico circuito negli anni precedenti (`YEARS_LOOKBACK_M2`), come posizione media finale, posizione media in griglia, media delle posizioni guadagnate/perse, numero di gare disputate, miglior piazzamento e percentuale di giri completati.
- Un flag booleano `is_driver_new_to_circuit`.

Il target rimane `final_position`.

Sviluppo e Training dei Modelli Predittivi

Come precedentemente descritto, sono stati sviluppati due sistemi di modelli distinti.
I modelli sono stati addestrati su un server con una gpu L40S con 46gb di vram.

Modello 1: Predizione Durante la Gara (Primi 30 Giri)

Per questo modello, l'obiettivo è predire la `final_position` basandosi sui dati disponibili fino a un dato giro N (compreso tra 1 e 30). È stato adottato un approccio di ensemble, combinando le predizioni di due potenti algoritmi di **Gradient Boosted Decision Trees (GBDT)**:

- **LightGBM (`lightgbm.LGBMRegressor`)** [(Ke et al., 2017)]:
  - Caratteristiche: **LightGBM** è noto per la sua efficienza e velocità, specialmente su dataset di grandi dimensioni. Utilizza un algoritmo di crescita degli alberi basato su istogrammi (`histogram-based`) e una strategia di crescita `leaf-wise` (invece che level-wise), che spesso porta a una convergenza più rapida e a una migliore accuratezza. È particolarmente efficace con un alto numero di feature. Supporta nativamente l'addestramento su GPU.
  - Preprocessing: Le feature numeriche vengono imputate con la mediana e poi standardizzate (`StandardScaler`). Le feature categoriche vengono imputate con la moda (o una costante "MISSING") e poi trasformate tramite One-Hot Encoding. Questi passaggi sono gestiti da una `sklearn.pipeline.Pipeline` che include un `sklearn.compose.ColumnTransformer`.
  - Hyperparameter Tuning: I parametri di **LightGBM** (`n_estimators, learning_rate, max_depth, num_leaves, parametri di regolarizzazione`) sono stati ottimizzati utilizzando la libreria `Optuna`, minimizzando il **Mean Absolute Error (MAE)** tramite cross-validation con `GroupKFold` (raggruppando per Gran Premio per evitare perdite di informazioni dei gp).
  - Loss Function: Addestrato con `objective='regression_l1'`, che corrisponde a ottimizzare per il MAE.

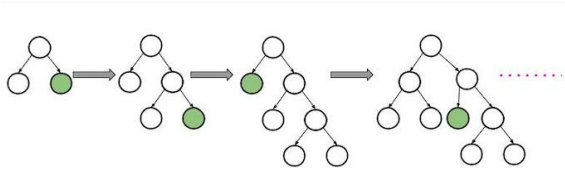

Figure 3: Illustrazione della strategia di crescita degli alberi leaf-wise (foglia per foglia) impiegata da LightGBM. Ad ogni iterazione, viene espansa la foglia (nodo verde) che promette la maggiore riduzione della funzione di loss, portando a una crescita asimmetrica dell'albero e a una convergenza spesso più rapida.

- **CatBoost (`catboost.CatBoostRegressor`)** [(Prokhorenkova et al., 2018)]:
  - Caratteristiche: **CatBoost** è un altro algoritmo GBDT che eccelle nella gestione nativa delle feature categoriche, spesso senza la necessità di pre-processing esplicito come il One-Hot Encoding. Utilizza ordered boosting e permutation-based target encoding per combattere il prediction shift e migliorare la generalizzazione. Supporta anch'esso l'addestramento su GPU.
  - Preprocessing: Le feature numeriche sono imputate e scalate. Le feature categoriche sono imputate (sostituendo i NaN con una stringa placeholder) e poi passate direttamente a CatBoost come stringhe, specificando i loro indici tramite il parametro `cat_features`. La conversione a stringa è stata gestita con un `sklearn.preprocessing.FunctionTransformer` che usa una funzione custom `to_str_type` (definita in `utils_gradio.py` per garantire la serializzazione corretta).
  - Parametri: Sono stati usati parametri di default robusti, con un numero elevato di iterazioni e early stopping, ottimizzando per MAE.

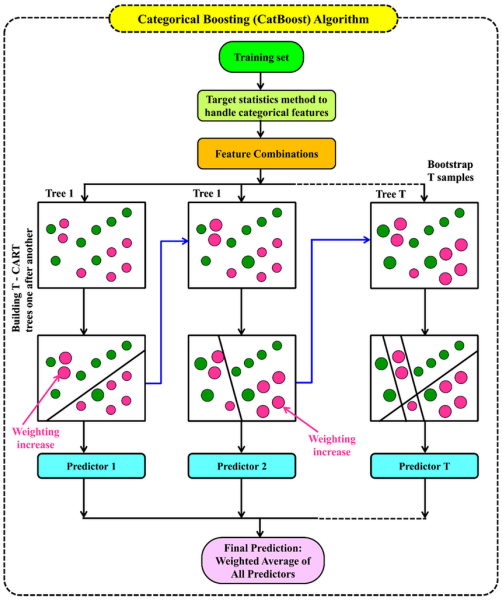

Figure 4: Schema illustrativo dell'algoritmo CatBoost. Vengono evidenziati i passaggi chiave come l'elaborazione delle feature categoriche tramite statistiche del target, la potenziale creazione di combinazioni di feature, e il processo iterativo di boosting dove alberi decisionali sequenziali vengono addestrati ponendo maggiore enfasi sui campioni precedentemente erroneamente predetti. La predizione finale è un aggregato ponderato delle predizioni di tutti gli alberi.

- **Ensemble Ponderato Posizionalmente** (Modello 1)

Per migliorare ulteriormente l'accuratezza delle predizioni del `Modello 1`, è stato implementato un meccanismo di ensemble più sofisticato rispetto alla semplice media aritmetica delle predizioni di **LightGBM** e **CatBoost**. L'approccio adottato è un **ensemble ponderato posizionalmente**, dove il contributo di ciascun modello base alla predizione finale varia dinamicamente in base alla "competenza" storica di quel modello nel predire una specifica posizione di classifica.Il processo si articola nei seguenti passaggi chiave:

1. **Calcolo dei Pesi di Competenza Posizionale**:
   - Questa fase avviene offline, utilizzando le predizioni dei modelli **LightGBM** e **CatBoost** (già addestrati) sul **set di training**.
   - Per ciascuna posizione finale di interesse (es., da 1 a 20), si analizza la performance storica di ogni modello:
     - Si identificano tutte le istanze nel set di training in cui la posizione reale corrispondeva alla posizione corrente in analisi.
     - Per queste istanze, si calcola l'errore assoluto medio (MAE) di ciascun modello.
     - La "competenza" grezza (`skill`) di un modello per quella posizione è definita inversamente proporzionale al suo MAE: `skill = 1 / (MAE + ε)`, dove `ε` è una piccola costante per evitare divisioni per zero.
     - Le `skill` grezze dei due modelli per la medesima posizione vengono poi normalizzate affinché la loro somma sia 1. Questi valori normalizzati costituiscono i **pesi di competenza posizionale** (`w_lgbm_posizioneX`, `w_catboost_posizioneX`).
   - Questi pesi vengono memorizzati in un file JSON (`positional_weights.json`) per essere riutilizzati durante la fase di inferenza, simulando una "cache" della conoscenza storica dei modelli. Se il file non è presente, i pesi vengono ricalcolati.
2. **Applicazione dei Pesi durante l'Inferenza (sul Test Set)**:
   - Quando si deve generare una predizione per un nuovo campione (un pilota in un dato giro) nel test set:
     - Si ottengono le predizioni grezze (valori numerici di posizione) da **LightGBM** (`pred_lgbm`) e **CatBoost** (`pred_cb`).
     - Per determinare quale set di pesi posizionali utilizzare, si stima una **posizione target aggregata**. Questa è calcolata come la media arrotondata delle predizioni `pred_lgbm` e `pred_cb` per il campione corrente: `pos_target_pesi = round((pred_lgbm + pred_cb) / 2)`. Il valore viene limitato (clamped) all'intervallo delle posizioni di interesse.
     - Si recuperano i pesi di competenza specifici per `pos_target_pesi` dal file JSON (es., `w_lgbm_pos_target_pesi` e `w_catboost_pos_target_pesi`). Se per una data `pos_target_pesi` non esistono pesi specifici (es., posizione rara o non vista nel training), si utilizzano pesi di default (es., 0.5 per ciascun modello).
     - La predizione finale dell'ensemble ponderato per il campione è calcolata come: `pred_ensemble = (pred_lgbm w_lgbm_pos_target_pesi) + (pred_cb w_catboost_pos_target_pesi)`. (Una normalizzazione finale viene applicata se la somma dei pesi recuperati non fosse esattamente 1).

Questo approccio permette all'ensemble di dare maggior credito al modello che, storicamente, si è dimostrato più abile nel predire l'intorno della posizione stimata per il pilota, portando potenzialmente a predizioni più accurate e contestualizzate. Lo schema di questo processo è illustrato in Figura 5.

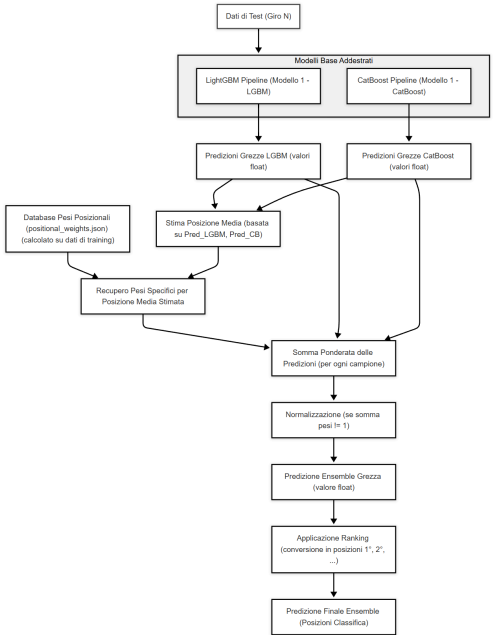

Figure 5: Pipeline Ensamble

Lo script `train_model.py` gestisce il caricamento del dataset combinato (storico + gare 2025 disputate), il preprocessing, il tuning di LightGBM, l'addestramento di CatBoost e la valutazione dell'ensemble. I modelli addestrati e le pipeline di preprocessing vengono salvati su disco utilizzando `joblib`.

Modello 2: Predizione Pre-Gara

Per affrontare lo scenario di predizione antecedente allo svolgimento della competizione, è stato sviluppato un secondo sistema predittivo, denominato `Modello 2`. Questo modello opera intrinsecamente con un insieme di informazioni più limitato rispetto al `Modello 1`, basandosi esclusivamente su dati disponibili prima dell'inizio effettivo della gara.

L'addestramento del `Modello 2` è stato condotto sul dataset `pre_race_prediction_dataset.csv`, specificamente costruito per questo scopo (cfr. Sezione 2.1.2 per i dettagli sulla sua creazione).
Le feature impiegate includono:

- `starting_grid_position`: La posizione sulla griglia di partenza
- `elo_driver_pre_race`: I rating Elo di pilota e team calcolati fino alla gara precedente
- `avg_finish_pos_circuit_prev2y ecc...`: Le statistiche aggregate delle performance storiche del pilota e del team sul circuito specifico (considerando un periodo di lookback definito)
- `is_driver_new_to_circuit`: Un indicatore booleano per i piloti al debutto su quel tracciato
- Il nome del Gran Premio (trattato come feature categorica rappresentativa del circuito).

Dal punto di vista algoritmico, per il `Modello 2` è stato scelto nuovamente un `CatBoostRegressor`. Questa decisione è motivata dalla robustezza dimostrata da tale algoritmo nel `Modello 1`, in particolare per la sua efficace gestione nativa delle feature categoriche e le solide performance ottenibili anche con una configurazione di iperparametri di base.
Sebbene l'architettura interna del `CatBoostRegressor` sia la medesima impiegata nel `Modello 1` (il cui principio di funzionamento è illustrato in Figura 4), il suo addestramento per il `Modello 2` avviene su un diverso set di feature e con un diverso contesto informativo.
Il preprocessing dei dati per il `Modello 2` segue una logica analoga a quella adottata per CatBoost nel `Modello 1`: le feature numeriche sono sottoposte a imputazione dei valori mancanti e successiva standardizzazione, mentre le feature categoriche vengono imputate e convertite in formato stringa affinché l'algoritmo CatBoost possa processarle internamente.
L'obiettivo del training è la predizione della colonna `final_position`, utilizzando come input esclusivamente le feature pre-gara.
L'intero processo di addestramento e salvataggio della pipeline per il `Modello 2` è gestito dallo script `train_model2.py`.

Interfaccia Utente (Gradio)

È stata sviluppata un'interfaccia web interattiva utilizzando la libreria Gradio. L'interfaccia (`app_gradio.py`) permette all'utente di:

- Selezionare l'anno (2023, 2024, 2025) e un Gran Premio.

Per gare disputate (anni 2023, 2024, e gare 2025 con dati disponibili nel dataset del `Modello 1`):

- Selezionare un giro da 1 a 30.
- Visualizzare la situazione reale della gara a quel giro (classifica, gomme, tempi, gap, pit stop).
- Ottenere una predizione della classifica finale utilizzando l'ensemble del `Modello 1`. La tabella dei risultati mostra la posizione predetta, quella reale e la differenza.

Per gare del 2025 non ancora disputate (non presenti nel dataset del `Modello 1` ma elencate in un file di `calendario separato gare_2025.csv`):

- L'interfaccia richiede all'utente di caricare un file .txt contenente la griglia di partenza (un pilota per riga, in ordine di P1, P2, ecc.).
- Una volta caricata la griglia, l'utente può richiedere una predizione pre-gara.

L'applicazione prepara le feature necessarie per il `Modello 2` (posizioni dalla griglia, Elo più recenti disponibili, statistiche storiche sul circuito selezionato) e invoca il `Modello 2` per predire la classifica finale. Viene mostrata solo la classifica predetta. L'interfaccia carica i modelli addestrati (`.joblib`) all'avvio e il dataset combinato per il `Modello 1`, oltre al dataset storico per le feature del `Modello 2` e il calendario 2025.

RISULTATI

La valutazione dei modelli è stata effettuata utilizzando metriche standard per problemi di regressione, sebbene la posizione finale sia intrinsecamente ordinale: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), e R-squared (R^2).

Spiegazione Metriche

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

La MAE calcola la differenza media assoluta tra le previsioni $\hat{y}_i$ e i valori osservati y_i.

- È facile da interpretare (stesse unità della variabile target).
- Pesa in modo lineare tutti gli errori, senza eccessiva penalizzazione degli outlier.

Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

La RMSE è la radice quadrata dell'errore quadratico medio.

- Amplifica gli errori più grandi grazie al quadrato.
- Utile quando si vuole penalizzare in modo marcato le grandi deviazioni.

Coefficiente di Determinazione (R^2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - |(y))^2}$$

Dove $|(y)$ è il valore medio di y_i.

- Indica la quota di varianza spiegata dal modello:
  - $R^2 = 1$ -> previsioni perfette.
  - $R^2 = 0$ -> il modello non migliora la media.
  - $R^2 < 0$ -> il modello performa peggio di un modello baseline che predice sempre la media.
- Adimensionale, consente il confronto tra modelli diversi.

Queste tre metriche verranno riportate congiuntamente per fornire una valutazione completa: MAE per l'errore medio "grezzo", RMSE per evidenziare gli outlier e R^2 per la varianza spiegata.

Risultati Modello 1 (Predizione Durante la Gara)

Sul set di test (20% dei dati, splittato per GP), addestrando sul dataset combinato (2023, 2024, e le prime gare 2025 disputate), sono state ottenute le seguenti performance:

Tabella 1: Metriche di valutazione Modello 1 sul test set.

Modello	MAE	RMSE	R^2
LightGBM	2.3859	3.3718	0.6322
CatBoost	2.4284	3.3936	0.6275
Weighted Ensemble	2.2827	3.0602	0.6950

Un MAE di circa 2.28 per l'ensemble indica che, in media, il modello predice la posizione finale con un errore di circa 2.3-2.4 posizioni, basandosi sui dati dei primi 30 giri. L'R^2 di circa 0.69 significa che l'ensemble spiega quasi il 69% della varianza nella posizione finale. Questi risultati sono considerati buoni data la complessità del problema. **LightGBM** si è dimostrato leggermente superiore a **CatBoost** come modello singolo, e l'ensemble ha fornito un piccolo ma consistente miglioramento in termini di RMSE e R^2.

Risultati Modello 2 (Predizione Pre-Gara)

Sul set di test per il `Modello 2` (addestrato su `pre_race_prediction_dataset.csv`), il `CatBoostRegressor` ha ottenuto:

Tabella 2: Metriche di valutazione Modello 2 (Pre-Gara) sul test set.

Modello	MAE	RMSE	R^2
Pre-Race CatBoost	2.6972	3.8001	0.5657

Come atteso, le performance sono inferiori a quelle del `Modello 1`, data la minor quantità di informazioni disponibili (solo dati pre-gara). Tuttavia, un MAE di 2.7 e un R^2 di 0.56 indicano che il modello è comunque in grado di catturare segnali predittivi significativi basati su griglia, Elo e storico del circuito, performando decisamente meglio di una predizione casuale.

Visualizzazioni delle Predizioni del Modello 1

Per una comprensione più approfondita delle capacità predittive e del comportamento dell'ensemble del `Modello 1` (**LightGBM + CatBoost**), sono state generate le seguenti visualizzazioni diagnostiche basate sulle performance nel set di test.

Feature Importance dei Modelli Componenti

Comprendere quali feature influenzano maggiormente le predizioni è cruciale. Di seguito, vengono presentati i grafici di importanza delle feature per i due modelli base che compongono l'ensemble.

LightGBM (Modello 1)

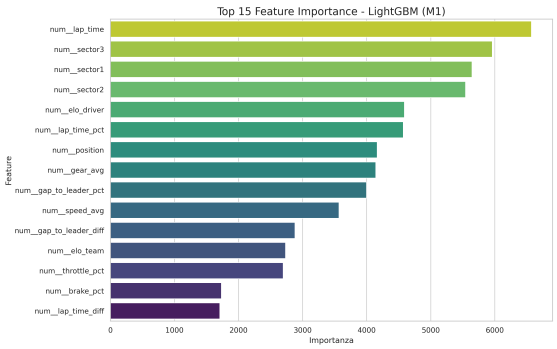

Figure 6: Feature Importance per il modello LightGBM (Modello1). Mostra le top 15 feature che contribuiscono maggiormente alla predizione della posizione finale, basandosi sui dati dei primi 30 giri.

Il grafico (Figura sopra) mostra che variabili cronometriche dirette come `numlap_time` e i tempi dei settori (`numsector1`, `numsector2`, `numsector3`), insieme al rating `numelo_driver`, sono tra i predittori più influenti per **LightGBM**. Questo suggerisce che la performance recente e la forza intrinseca del pilota sono determinanti chiave.

Analisi degli Errori dell'Ensemble (Modello 1)

Distribuzione degli Errori (Istogramma dei Residui)

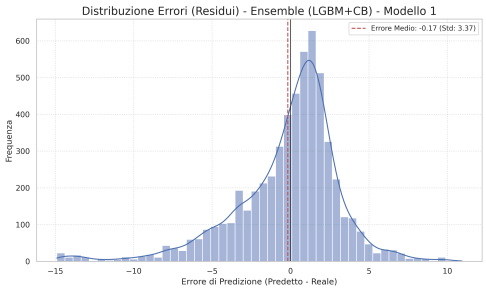

Figure 7: Distribuzione degli errori di predizione (residui) per l'ensemble del Modello 1. L'errore è calcolato come Posizione Predetta - Posizione Reale.

L'istogramma dei residui (Figura sopra) illustra la distribuzione degli errori commessi dall'ensemble del `Modello 1`. Osserviamo che la distribuzione è approssimativamente centrata vicino allo zero, con un errore medio di circa -0.17 (come indicato dalla linea tratteggiata rossa). Questo suggerisce una leggera tendenza del modello a sovrastimare lievemente la posizione finale dei piloti (cioè, predire una posizione leggermente migliore/più bassa di quella effettivamente ottenuta). La deviazione standard degli errori è di circa 3.37 posizioni. La forma della distribuzione, con una campana principale e code che si estendono, indica che la maggior parte delle predizioni ha un errore contenuto, ma occasionalmente si verificano deviazioni più significative.

Scatter Plot: Predetto vs. Reale

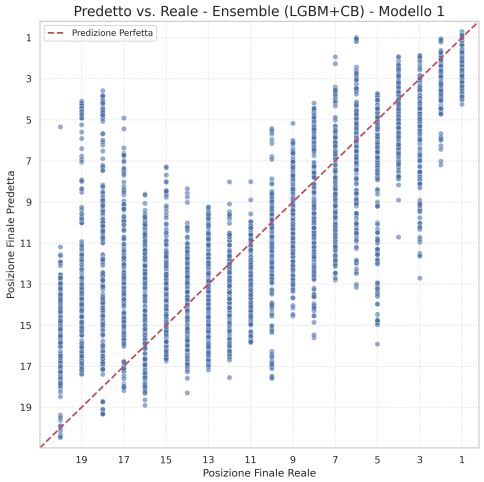

Figure 8: Confronto tra Posizioni Finali Predette e Reali per l'ensemble del Modello 1. La linea tratteggiata rossa rappresenta la predizione perfetta (y=x).

Lo scatter plot (Figura sopra) confronta direttamente le posizioni finali predette dall'ensemble con quelle reali per ciascun pilota nel set di test. I punti che si dispongono lungo la diagonale rossa (y=x) indicano una predizione perfetta. La dispersione dei punti attorno a questa linea fornisce una misura visiva dell'accuratezza del modello. Si può notare come, pur con una certa variabilità, le predizioni tendano a seguire l'andamento delle posizioni reali. Le "bande" orizzontali di punti suggeriscono che per una data posizione reale, il modello può predire un piccolo range di posizioni, riflettendo l'incertezza intrinseca. L'inversione degli assi (P1 in alto a sinistra) è tipica delle rappresentazioni di classifiche.

Distribuzione degli Errori (Istogramma dei Residui) - Modello 2

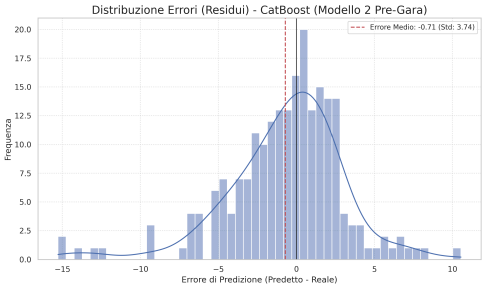

Figure 9: Distribuzione degli errori di predizione (residui) per il Modello 2 (CatBoost Pre-Gara). L'errore è Posizione Predetta - Posizione Reale.

L'istogramma dei residui per il `Modello 2` (Figura sopra) mostra una distribuzione degli errori con un errore medio di circa -0.71 e una deviazione standard di 3.74. Similmente al `Modello 1`, si osserva una tendenza a sovrastimare leggermente la posizione finale (predizioni migliori/più basse rispetto al reale). La dispersione degli errori è, come atteso, maggiore rispetto al `Modello 1`, riflettendo la maggiore incertezza intrinseca nella predizione pre-gara. Nonostante ciò, la distribuzione mantiene una forma approssimativamente campanulare attorno al suo errore medio.

Scatter Plot: Predetto vs. Reale - Modello 2

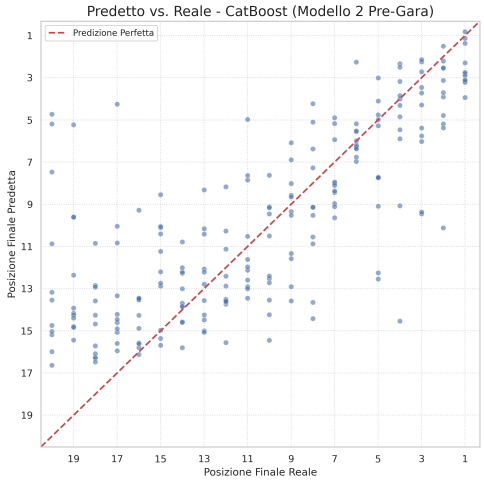

Figure 10: Confronto tra Posizioni Finali Predette e Reali per il Modello 2 (CatBoost Pre-Gara). La linea tratteggiata rossa rappresenta la predizione perfetta (y=x).

Lo scatter plot che confronta le posizioni predette dal `Modello 2` con quelle reali (Figura sopra) evidenzia una correlazione positiva, sebbene con una dispersione dei punti più accentuata rispetto a quella osservata per il `Modello 1`. Questo risultato è coerente con la natura delle informazioni disponibili: basandosi solo su dati pre-gara (griglia di partenza, Elo, storico sul circuito), il `Modello 2` cattura alcune tendenze generali ma non può tenere conto delle dinamiche specifiche che si sviluppano durante la competizione. La maggior parte delle predizioni si colloca comunque in una fascia qualitativamente ragionevole rispetto ai valori reali, confermando l'utilità del modello pur con i suoi limiti intrinseci.

Problemi Incontrati e Soluzioni Adottate

Preprocessing e Pipeline Sklearn:

La corretta definizione dell'ordine dei trasformatori e la gestione dei tipi di output (DataFrame vs NumPy array) all'interno di `ColumnTransformer` e `Pipeline` hanno richiesto attenzione, specialmente per passare gli indici corretti delle `cat_features` a **CatBoost** quando inserito in una pipeline.

Serializzazione e Deserializzazione Modelli (joblib)

La serializzazione delle pipeline scikit-learn con `joblib` ha presentato sfide quando queste includevano trasformatori custom basati su funzioni definite localmente (come `to_str_type` per **CatBoost**). L'errore comune `AttributeError: Can't get attribute` si verificava al momento del caricamento del modello in un contesto diverso da quello di training (es. l'interfaccia Gradio), poiché `joblib` non riusciva a rintracciare la funzione nel suo scope originale. La soluzione implementata è stata quella di definire tali funzioni in un modulo Python separato (`utils_gradio.py`), importandole sia nello script di training che in quello di inferenza. Questo ha garantito un riferimento stabile e rintracciabile, risolvendo i problemi di deserializzazione e sottolineando l'importanza di considerare l'ambiente di deployment durante il salvataggio dei modelli.

Calcolo e Gestione dell'Elo

L'integrazione di un sistema di rating Elo dinamico è stata cruciale per quantificare la forza evolutiva di piloti e team. La sfida principale è stata ottenere lo snapshot Elo corretto per ogni punto dati: l'Elo post-gara precedente per il `Modello 1` (predizione durante la gara) e l'Elo pre-gara corrente per il `Modello 2` (predizione pre-gara). Questo è stato risolto modificando la funzione di calcolo Elo (`compute_elo_history`) per registrare lo stato dei rating dopo ogni evento. La funzione è stata inoltre resa robusta per gestire la mancanza di dati da **FastF1** per gare future o incomplete, permettendo un calcolo continuo dell'Elo basato sui dati disponibili e l'applicazione corretta del decay di fine stagione.

Feature Engineering e Selezione per il Modello 2 (Pre-Gara)

Per il `Modello 2`, che opera con informazioni limitate disponibili prima della gara, è stato necessario un attento feature engineering. Le feature chiave includono la posizione sulla griglia di partenza, i rating Elo pre-gara di pilota e team, e statistiche aggregate delle performance storiche del pilota e del team su quel specifico circuito (anni precedenti), come posizione media finale/griglia e posizioni guadagnate/perse. È stato aggiunto un flag per i piloti al debutto sul circuito, con imputazione di valori di default per le loro statistiche storiche. La selezione è stata guidata dall'ipotesi di combinare forma attuale (Elo), potenziale di partenza (griglia) e affinità storica col tracciato.

Limiti del Dataset e Generalizzazione

Nonostante l'uso di dati da più stagioni, il dataset presenta un numero finito di esempi di dinamiche di gara. La Formula 1 è soggetta a **concept drift** dovuto a cambiamenti regolamentari, evoluzione tecnica e variazioni nelle performance dei team/piloti. Questo implica che i modelli addestrati potrebbero richiedere un retraining periodico con dati recenti per mantenere l'accuratezza predittiva su stagioni future. La validità del `Modello 1` è inoltre intrinsecamente legata ai primi 30 giri di gara. Infine, la qualità delle predizioni dipende dalla completezza e accuratezza dei dati sorgente forniti da **FastF1**, e sebbene gli script di costruzione del dataset gestiscano i dati mancanti, l'assenza di informazioni non può essere interamente compensata.

CONCLUSIONI

Il presente progetto ha dimostrato con successo la fattibilità di sviluppare sistemi predittivi per i risultati dei Gran Premi di Formula 1, utilizzando un approccio basato sull'apprendimento automatico e su dati telemetrici, cronometrici e contestuali.
L'efficacia dei modelli è stata validata attraverso metriche quantitative.
L'approccio a due modelli si è rivelato strategico: il `Modello 2`, sebbene intrinsecamente meno preciso (MAE $\approx$ 2.7, R^2 $\approx$ 0.56) a causa della limitatezza delle informazioni pregara (griglia, Elo, storico circuito), fornisce comunque predizioni preziose e migliori di un'ipotesi casuale, estendendo l'applicabilità del sistema a scenari di gara non ancora disputate.
L'utilizzo di un ensemble di modelli GBDT (**LightGBM** e **CatBoost**) per il `Modello 1` ha generalmente offerto una leggera superiorità in termini di robustezza e generalizzazione rispetto ai singoli componenti.

Tuttavia, il sistema presenta alcuni limiti intrinseci:

- La precisione del `Modello 2` è notevolmente inferiore a quella del `Modello 1`, riflettendo la difficoltà di predire un evento complesso con sole informazioni statiche pregara. Entrambi i modelli dipendono criticamente dalla qualità e disponibilità dei dati da **FastF1** dati mancanti o imprecisi possono inficiare la qualità delle feature e, di conseguenza, delle predizioni.
- Inoltre, il `Modello 1` è attualmente limitato all'analisi dei primi 30 giri, e la sua estendibilità a fasi successive della gara richiederebbe un'analisi e un addestramento specifici.
- Infine, il fenomeno del **concept drift**, tipico di ambienti dinamici come la F1, necessita di un retraining periodico dei modelli per mantenere performance ottimali nel tempo.

Per quanto riguarda i possibili **sviluppi futuri**, diverse direzioni appaiono promettenti:

- Un tuning più approfondito degli iperparametri del `Modello 2` o l'esplorazione di feature alternative potrebbero migliorarne l'accuratezza
- L'integrazione di dati contestuali aggiuntivi, come le condizioni meteorologiche previste o aggiornamenti specifici sulle vetture, potrebbe arricchire ulterioremente i modelli.
- L'estensione del `Modello 1` a un numero maggiore di giri o a fasi specifiche della gara (es. post pit-stop) rappresenta un'evoluzione naturale.
- Sebbene non previsto in questo progetto accademico, l'adattamento del sistema per l'utilizzo con flussi di dati in tempo reale, qualora disponibili, lo trasformerebbe in uno strumento di analisi live di grande potenziale.
- Infine, l'esplorazione di tecniche di ensemble più sofisticate, come lo stacking, potrebbe ulteriormente affinare le capacità predittive del sistema.

In conclusione, il progetto ha raggiunto gli obiettivi prefissati, fornendo modelli funzionanti e un'interfaccia utente interattiva, e ponendo solide basi per future ricerche e miglioramenti nel campo dell'analisi predittiva applicata alla Formula 1.

Bibliografia

Bunker, R., Yeung, C., Susnjak, T., Espie, C., & Fujii, K. (2023). A Comparative Evaluation of Elo Ratings- and Machine Learning-based Methods for Tennis Match Result Prediction. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, *237*(4), 305–316. https://doi.org/10.1177/17543371231212235

Hvattum, L. M., & Arntzen, H. (2010). Using Elo Ratings for Match Result Prediction in Association Football. *International Journal of Forecasting*, *26*(3), 460–470. https://doi.org/10.1016/j.ijforecast.2009.10.002

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems 30*, 3146–3154. https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased Boosting with Categorical Features. *Advances in Neural Information Processing Systems 31*, 6639–6649. https://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features

Schaefer, P. (2025, March). *FastF1: A Python package for accessing and analyzing Formula 1 results, schedules, timing data and telemetry.*

Wang, K., Wang, L., & Sun, J. (2022). The Data Analysis of Sports Training by ID3 Decision Tree Algorithm and Deep Learning. *Proceedings of the 2022 Conference on Artificial Intelligence in Sports*, 78–85. https://doi.org/10.1145/ai-sports.2022.0123

Wiechno, W., Bartosik, B., & Duch, P. (2023). Time Series and Deep Learning Approaches for Predicting English Premier League Match Outcomes. *International Journal of Sports Science and Engineering*, *15*(4), 210–225. https://doi.org/10.1016/ijsse.2023.04.005

Yang, S., Luo, L., & Tan, B. (2021). Research on Sports Performance Prediction Based on BP Neural Network. *Journal of Computational Sports Science*, *10*(1), 45–58. https://doi.org/10.1155/jcss.2021.00045