UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Fondamenti di Intelligenza Artificiale
Prof. Fabio Palomba

Formula 1 Predictor

GitHub Repo

0512111084 - Teodoro Grauso
0512112926 - Giovanni Manfredi

Anno Accademico 2023/2024

Indice

1 Introduzione

In questo documento è disponibile la documentazione relativa al progetto universitario sviluppato per il corso "Fondamenti di Intelligenza Artificiale", tenuto dal professore Fabio Palomba.

1.1 Prefazione

La Formula 1, ossia la massima categoria di vetture monoposto da corsa su circuito (definita dalla FIA), ha vissuto negli ultimi anni un'impressionante aumento di spettatori, passando dai circa 300.000 spettatori medi in ogni gara agli oltre 1.200.000 registrati nel 2023. L'aumento di interesse nella Formula 1 ha coinvolto diverse strategie di marketing e di comunicazione, tra cui quella relativa ai social network (in particolare su piattaforme come YouTube e Netflix) che hanno permesso alla F1 di raggiungere un pubblico più vasto e di coinvolgere gli appassionati in modi nuovi e innovativi. Tra le tante è importante citare la serie "Drive to Survive", realizzata da Netflix, la quale ha svolto un ruolo significativo nell'aumentare l'interesse per la Formula 1, soprattutto tra coloro che potrebbero non essere stati tradizionalmente fan dello sport, offrendo un dietro le quinte delle gare e delle vite dei piloti, rendendo il mondo della F1 più accessibile e coinvolgente per il grande pubblico.

L'ammodernamento generale della Formula 1 non si è limitato solo alla comunicazione e al marketing, ma ha coinvolto anche miglioramenti tecnologici, regolamentari e di intrattenimento durante le gare stesse. Questi cambiamenti hanno reso lo sport più competitivo, avvincente e spettacolare, contribuendo così ad attrarre un numero sempre maggiore di spettatori.

1.2 La sfida

Vogliamo realizzare un sistema che possa permettere agli utenti di godere della bellezza della F1 a 360 gradi. Tuttavia, gli anni passati dal più recente cambio regolamentare sono 2 (2022-2023), quindi questo implicherebbe una forte diminuzione dei dati disponibili. Per tali ragioni abbiamo deciso di lavorare sul penultimo, nonchè il più rilevante, cambio regolamentare, ossia quello avvenuto nell'ormai lontano 2014, permettendoci così di avere un range di anni che vanno dall'appunto 2014 sino al 2021.

1.3 Specifica P.E.A.S.

Ecco la formulazione PEAS:

Performance	Predizione del giro svolto dal pilota scelto in un certo circuito e condizioni meteo.
Environment	Rappresentato dalla web-app *Formula 1 Predictor*.
Actuators	Monitor per visualizzare la predizione effettuata.
Sensors	Puntatore attraverso il quale è possibili selezionare il circuito, la condizione meteo e i piloti.

1.3.1 Proprietà dell'ambiente

L'ambiente si compone delle seguenti proprietà:

1. **Completamente osservabile:** i sensori dell'agente forniscono accesso allo stato completo dell'ambiente in ogni momento;
2. **Deterministico:** lo stato successivo dell'ambiente è determinato interamente dall'azione eseguita dall'agente;
3. **Episodico:** nella nostra accezione le interazioni dell'agente con l'ambiente sono episodiche, in quanto ogni previsione riguarda un giro specifico;
4. **Statico:** non cambia mentre l'agente sta deliberando;
5. **Discreto:** l'ambiente è rappresentato da eventi distinti e chiaramente definiti;
6. **Singolo agente:** L'agente è l'unica entità che prende decisioni nell'ambiente; non vi è quindi la presenza di altri agenti indipendenti che influenzano direttamente le azioni dell'agente.

1.4 Ingegneria del Machine Learning: CRISP-DM

È stato scelto il modello CRISP-DM per lo sviluppo della nostra applicazione, il quale risulta essere una costante quando si ha a che fare con progetti di intelligenza artificiale. Esso si compone delle seguenti parti: business understanding, data understanding, data preparation, data modeling, evaluation e deployment. Vediamole una ad una nel dettaglio.

2 Business Understanding

In questa fase ci siamo occupati principalmente nella raccolta dei requisiti e degli obiettivi di business che si intendono raggiungere.

2.1 Obiettivi di business

In linea con il rinnovamento e il crescente interesse, citato nella prefazione, abbiamo pensato di creare un'applicazione che possa stimolare ulteriormente l'interesse verso questo sport, dando la possibilità agli utenti di simulare delle vere e proprie gare di F1. L'obiettivo è quello di realizzare un'applicazione in grado di simulare un'intera gara di F1 prevedendo, sulla base di parametri scelti dall'utente, l'ordine di arrivo dei piloti in griglia. I parametri selezionabili dall'utente sono:

- Piloti: è possibile selezionare fino a a 20 piloti diversi (ossia il massimo consentito in una gara di F1) e simulare la gara;
- Circuito: viene selezionato il circuito in cui i piloti guideranno;
- Condizioni meteo: viene selezionata la condizione meteo preferita;
- Numero di giri: si sceglie il numero di giri della gara (minimo 1 e realisticamente massimo 100).

2.2 Analisi del problema

Analizzando gli obiettivi prefissati è possibile notare come l'agente dovrà essere in grado di predire variabili continue, quindi avremo a che fare con la regressione. Inoltre, la presenza della condizione meteo ci permette di intuire che avremo a che fare un tipo di regressione multipla. Sulla base di queste considerazioni, possiamo concludere affermando che la soluzione più conveniente sia quella di modellare un agente di machine learning, in particolare un regressore.

2.3 Tecnologie ed i tool necessari

Per raggiungere gli obiettivi prefissati abbiamo utilizzato il linguaggio di programmazione python con le segenti librerie:

- pandas: Libreria per la manipolazione e l'analisi dei dati;
- sklearn.model: Parte di scikit-learn, libreria per il machine learning in Python, che fornisce modelli e algoritmi, tra gli altri, anche per la regressione;

- sklearn.metrics: Parte di scikit-learn, offre metriche per valutare le prestazioni dei modelli di machine learning;
- sklearn.ensemble: Parte di scikit-learn, fornisce implementazioni di metodi di apprendimento ensemble;
- joblib: Libreria per il salvataggio e il caricamento di oggetti Python (come modelli addestrati) su disco;
- openmeteo_requests: Libreria per effettuare richieste di dati meteorologici attraverso API.
- numpy: Libreria fondamentale per la computazione scientifica in Python.

3 Data Understanding

In questa fase ci siamo occupati di analizzare i datasets necessari al raggiungimento degli obiettivi.

3.1 Acquisizione dei dataset

I dataset utilizzati sono disponibili sulla piattaforma Kaggle, *Formula 1 World Championship (1950 - 2023)*. All'interno della pagino sono disponibili diversi datasets, ognuno riguardante specifiche informazioni, come ad esempio: risultati delle gare, piloti, circuiti, costruttori etc...

3.2 Analisi dei dataset

Dopo l'acquisizione dei datasets, è stato cruciale eseguire un'analisi preliminare per comprende la struttura di ogni singolo file `.csv` e identificare quali sono i dataset rilavanti ai fini progettuali. Cominciamo quindi ad elencare i dataset disponibili:

- `circuits.csv`: Circuiti dove si svolgono le gare di F1;
- `constructor_results.csv`: Risultati delle gare del campionato costruttori;
- `constructor_standings.csv`: Classifica finale del campionato costruttori;
- `constructors.csv`: Costruttori in F1;
- `driver_standings.csv`: Classifica finale del campionato piloti;
- `drivers.csv`: I piloti della F1;
- `lap_times.csv`: I tempi sul giro in F1;
- `pit_stops.csv`: Pit stop in F1;
- `qualifying.csv`: Qualifiche in F1;
- `races.csv`: Le gare di F1;
- `results.csv`: Risultati delle gare di F1;
- `seasons.csv`: Stagioni della F1;
- `sprint_results.csv`: Risultati delle gare sprint di F1;
- `status.csv`: Mappatura dei vari stati.

Dopo un'attenta analisi sono stati individuati i datasets rilevanti, ececo quindi la lista aggiornata (in verde i datasets rilevanti e in rosso quelli irrilevanti):

- `circuits.csv`: Circuiti dove si svolgono le gare di F1;
- `constructor_results.csv`: Risultati delle gare del campionato costruttori;
- `constructor_standings.csv`: Classifica finale del campionato costruttori;
- `constructors.csv`: Costruttori in F1;
- `driver_standings.csv`: Classifica finale del campionato piloti;
- `drivers.csv`: I piloti della F1;
- `lap_times.csv`: I tempi sul giro in F1;
- `pit_stops.csv`: Pit stop in F1;
- `qualifying.csv`: Qualifiche in F1;
- `races.csv`: Le gare di F1;
- `results.csv`: Risultati delle gare di F1;
- `seasons.csv`: Stagioni della F1;
- `sprint_results.csv`: Risultati delle gare sprint di F1;
- `status.csv`: Mappatura dei vari stati.

4 Data Preparation

In questa fase ci siamo occupati principalmente di preparare i dati per le fasi successive del CRISP-DM.

Come mostrato nell'analisi del dataset, abbiamo a disposizione davvero tanti datesets diversi tra loro, ognuno rappresentante specifiche informazioni. Per poter procedere quindi, abbiamo bisogno di *fondere* tutti i datasets in verde, ossia quelli rilevanti, in un unico dataset. Ecco il risultato:

raceId	driverId	lap	position_x
time_x	milliseconds_x	driverRef	number_x
code	forename	surname	dob
nationality	url_x	year	round
circuitId	name_x	date	time_y
url_y	fp1_date	fp1_time	fp2_date
fp2_time	fp3_date	fp3_time	quali_date
quali_time	sprint_date	sprint_time	resultId
constructorId	number_y	grid	position_y
positionText	positionOrder	points	laps
time	milliseconds_y	fastestLap	rank
fastestLapTime	fastestLapSpeed	statusId	status
circuitRef	name_y	location	country
lat	lng	alt	url

Tale dataset si compone di ben **56 colonne** e oltre **551.000 righe**.

4.1 Feature selection

Interveniamo ora e sistemiamo la situazione del dataset eliminando informazioni non necessarie o ridondanti, migliorando la leggibilità del dataframe finale. Prima di tutto è necessario rimuovere le colonne irrilevanti ai fini della predizione. Procediamo quindi con la rimozione delle colonne, ottenendo il csv così composto:

raceId	driverId	lap	circuitId
forename	surname	name_x	date
positionOrder	time	time_y	statusId
status			

Procediamo ora a rinominare i campi per renderli più coerenti e uniamo le celle relative al nome e cognome del pilota in un unica cella. Ecco il dataset finale correttamente unito:

raceId	driverId	driver_name	circuitId
circuit_name	race_date	time_race	lap
position_lap	time_lap	statusId	status

Tale dataset si compone di ben **14 colonne** e oltre **166.000 righe**.

4.1.1 Meteo

Un'importante feature che abbiamo voluto inserire nella nostra applicazione è stata quella della scelta delle condizioni meteo. Le gare di F1 infatti sono fortemente influenzate dalle condizioni meteorologiche e, di conseguenza, dalle condizioni della pista. Il dataset, così come è, non ci fornisce informazioni riguardo le condizioni meteo, tuttavia presenta un'importante informazione, ossia la data e l'ora dello svolgimento della gara.

Partendo da questo ci siamo subito messi alla ricerca di API in grado di fornirci informazioni su quella che è la condizione meteo; È stata individuata quindi la seguente: Open-Meteo Weather API come la migliore API al caso nostro. Abbiamo quindi modificato ulteriormente il dataset aggiungendo due colonne:

raceId	driverId	driver_name	circuitId
circuit_name	race_date	time_race	lap
position_lap	time_lap	statusId	status
weather_code	weather_description		

Il weather_code indica uno specifico weather_description, il quale contiene numerose descrizioni, ognuno per ogni possibile tipo di condizione meteo, ad esempio: Clear sky, Mainly clear, Partly cloudy, Freezing drizzle, Drizzle, Freezing rain, Rain, etc...

Ai fini progettuali è inutile mantenere tutte queste micro indicazioni, è infatti più logico e sensato raggrupparlo in macro indicazioni evitando che l'utente possa "perdersi" tra le numerose indicazioni (ad esempio Drizzle, che sta per pioviggine, è stata raggruppata nella grande categoria: Rain). Ecco quindi le tre grandi categorie individuate:

- Clear Sky, ossia *cielo sereno*;
- Cloudy, ossia *nuvoloso*;
- Rain, ossia *pioggia*.

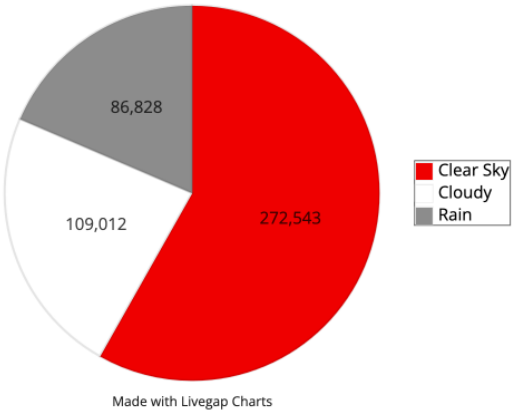


Figure 1: Grafico a torta relativo alle condizioni meteo

4.2 Data cleaning

Dopo la corretta fusione dei dataset, abbiamo creato uno script in grado di eseguire una serie di operazioni per pulire e manipolare i dati al fine di renderli più coerenti e adatti all'analisi successiva.

4.3 Feature scaling

Prima di procedere con il bilanciamento dei dati è fondamentale dividere il nostro dataset in due gruppi:

- Training set: composto dalle istanze utilizzate per addestrare l'algoritmo di regressione;
- Test set: composto dalle istanze su cui verrà valutata l'accuratezza delle predizioni dell'algoritmo di regressione, dopo essere stato addestrato con il training set.

La dimensione scelta dei due gruppi è: **training test**, pari all'78%, e **test set**, pari al 22%. Abbiamo creato una funzione chiamata `Oversampling` che: calcola il massimo numero di giri svolti da ogni pilota e, sulla base di questo valore, aggiunge giri ai piloti fino al raggiungimento di una soglia (`max_giri - 800`) vicina al massimo. Ecco lo script:

```
import pandas as pd
import random

def oversampling(dataset):
```

```
    # Numero di giri da raggiungere
    max_laps = dataset.groupby('driverId').size().max()

    # Bilanciamento del dataset
    for driver_id, group in dataset.groupby('driverId'):
        num_laps = len(group)
        desired_laps = random.randint(max_laps - 800, max_laps)
        if num_laps < desired_laps:
            # Add laps
            additional_laps = desired_laps - num_laps
            additional_samples = group.sample(n=additional_laps,
replace=True)
            dataset = pd.concat([dataset, additional_samples])

    return dataset
```

4.4 Data balancing

Come detto più volte, il focus del nostro progetto è il giro compiuto da ogni pilota. Ecco la situazione prima di un possibile bilanciamento:

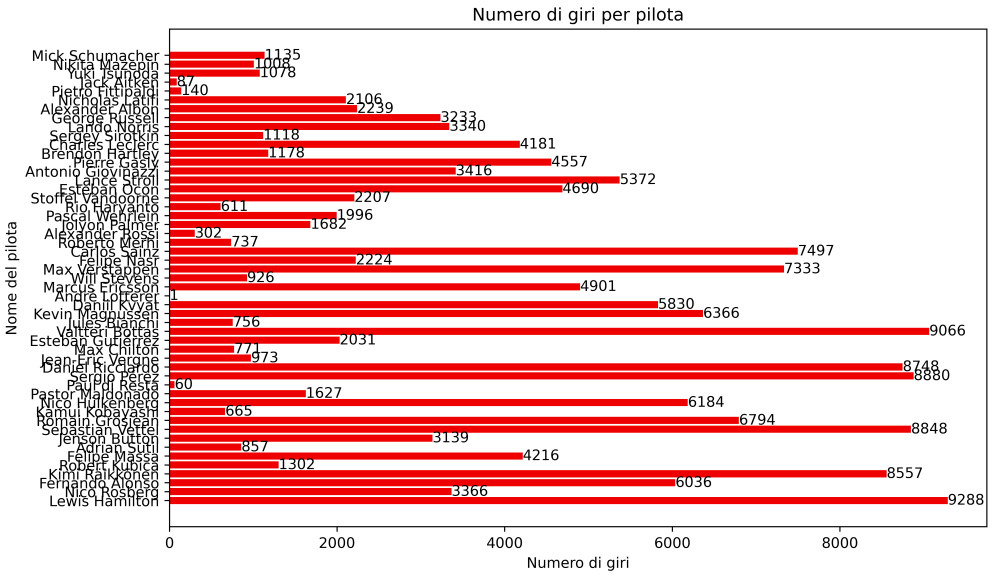


Figure 2: Numero di giri effettuati per ogni pilota

Notiamo come ci sono piloti con un numero di giri davvero basso, questo perchè è molto probabile che il cambio regolamentare coincida con le ultime gare di questi piloti (prima del ritiro). Un scelta importante è quindi quella di rimuoverli:

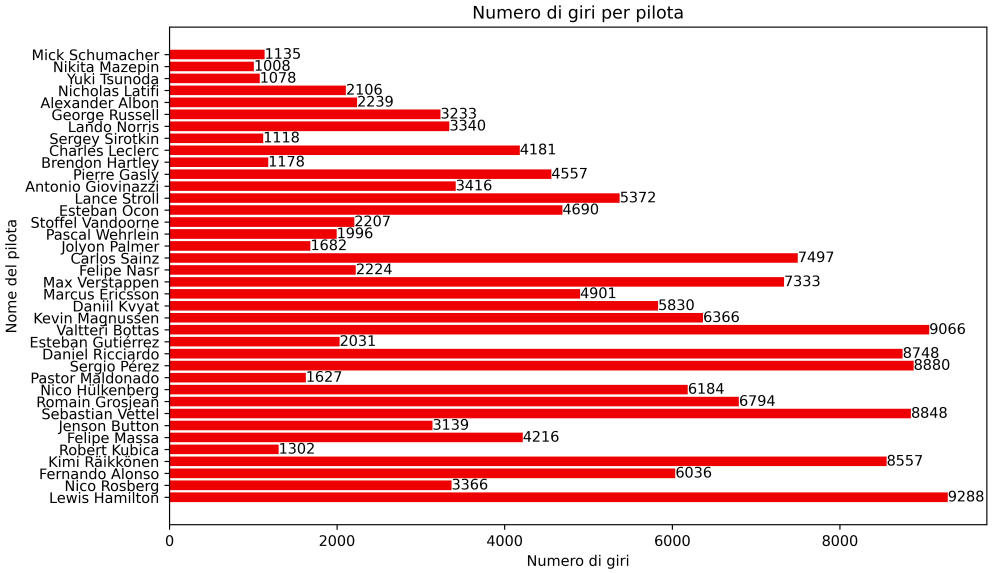


Figure 3: Rimossi i piloti con un numero di giri inferiore a 1000

È stato scelto un valore ragionevole e realistico come 1000 per rimuovere i piloti dal dataset. Notiamo ora che il dataset è sbilanciato, sono quindi necessarie delle operazioni di bilanciamento. Due sono le possibili strade:

- **Oversampling**: aggiunta di istanze al dataset;
- **Undersampling**: rimozione di istanze dal dataset.

Nel nostro caso abbiamo optato per l'*oversampling*, tecnica che ci permette, nel nostro caso, di duplicare alcuni giri dei piloti. Questo non influisce negativamente sulle performance poiché è molto probabile che un pilota, nelle stesse condizioni e con le stesse vetture, faccia, l'anno dopo, un tempo simile.

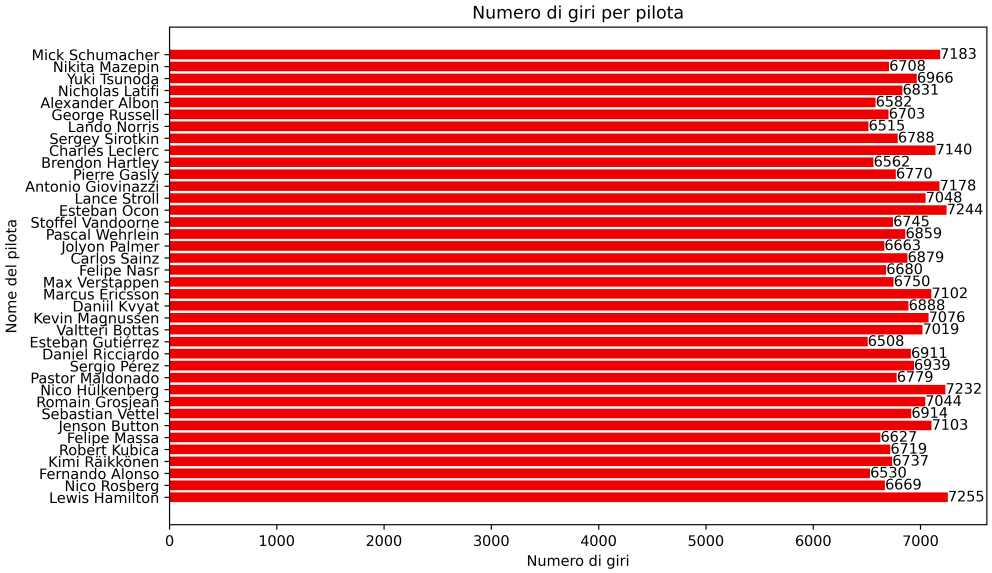


Figure 4: Oversampling

Poiché siamo pienamente soddisfatti del risultato ottenuto, possiamo passare alla prossima fase.

5 Data Modelling

Ora che i nostri dataset sono finalmente pronti possiamo passare alla fase di *modelazione*.

5.1 Scelta dell'algoritmo da utilizzare

Come anticipato nell'analisi del problema è possibile notare come l'agente dovrà essere in grado di predire il giro, quindi avremo a che fare con la **problemi di regressione**, ossia un'istanza del più generico **problema di apprendimento supervisionato**.

Esistono centinaia di regressori, pertanto è necessario testarne più di uno e, per quanto possibile, scegliere il migliore. Nella nostra accezione abbiamo posto l'attenzione sui seguenti quattro regressori:

- Gradient Boosting
- Decision Tree
- Random Forest
- Extra Trees Regressor

5.2 Addestramento

Ecco qui l'algoritmo correttamente addestrato in python:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
import oversampling
import joblib

# Caricamento del dataset
data = pd.read_csv("./merged_dataset_races.csv")

# Divisione del dataset in test e train
train_data, test_data = train_test_split(data, test_size=0.22,
    random_state=42)

```

```
train_data = oversampling.oversampling(train_data)

X_train_resampled = train_data[['driverId', 'circuitId', 'lap', 'weather_code']]
y_train_resampled = train_data['time_lap']

X_test = test_data[['driverId', 'circuitId', 'lap', 'weather_code']]
y_test = test_data['time_lap']

# Addestramento del modello RandomForestRegressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_resampled, y_train_resampled)

# Salvataggio del modello
joblib.dump(rf_model, 'rf_model.pkl')
```

6 Evaluation

Lo scopo dell'Evaluation è valutare i risultati prodotti dall'algoritmo selezionato e definire aspetti non molto chiari prodotti dalla fase di Modeling.

6.1 Valutazione degli algoritmi

Proseguiamo quindi a valutare ognuno degli algoritmi individuati e a trarre le dovute conclusioni.

6.1.1 Gradient Boosting

Fornito dalla libreria `sklearn.ensemble`, **Gradient Boosting**, il primo regressore testato, ha prodotto i seguenti risultati:

Gradient Boosting	
Mean Absolute Error	11.791618631878723
Accuracy	0.10723481572237503

I risultati prodotti non ci soddisfano a pieno, dunque procediamo a valutare le performance degli altri modelli.

6.1.2 Decision Tree Regressor

Il prossimo regressore si chiama **Decision Tree Regressor**, fornito dalla libreria `sklearn.ensemble`:

Decision Tree Regressor	
Mean Absolute Error	7.487555514758704
Accuracy	0.23752717047597838

Notiamo come in questo caso i risultati prodotti sono migliori rispetto al modello *Decision Tree Regressor*. Non fermiamoci qui e continuiamo con il prossimo.

6.1.3 Random Forest Regressor

Il prossimo modello è chiamato **Random Forest Regressor**, fornito anch'esso dalla libreria `sklearn.ensemble`:

Random Forest Regressor	
Mean Absolute Error	6.864292037016726
Accuracy	0.4079788133234922

I risultati ottenuti sono nettamente migliori dei precedenti, ottenendo un'accuratezza migliore di circa due volte rispetto al *Decision Tree Regressor* precedentemente tastato. Continuiamo ora con l'ultimo regressore.

6.1.4 Extra Trees Regressor

L'ultimo modello è chiamato **Extra Trees Regressor**, fornito sempre dalla libreria `sklearn.ensemble`:

Extra Trees Regressor	
Mean Absolute Error	7.572745384638995
Accuracy	0.268625021295138

I risultati ottenuti non sono migliori rispetto al *random forest*, quindi questo modello verrà scartato a prescindere.

6.2 Considerazione sugli algoritmi testati

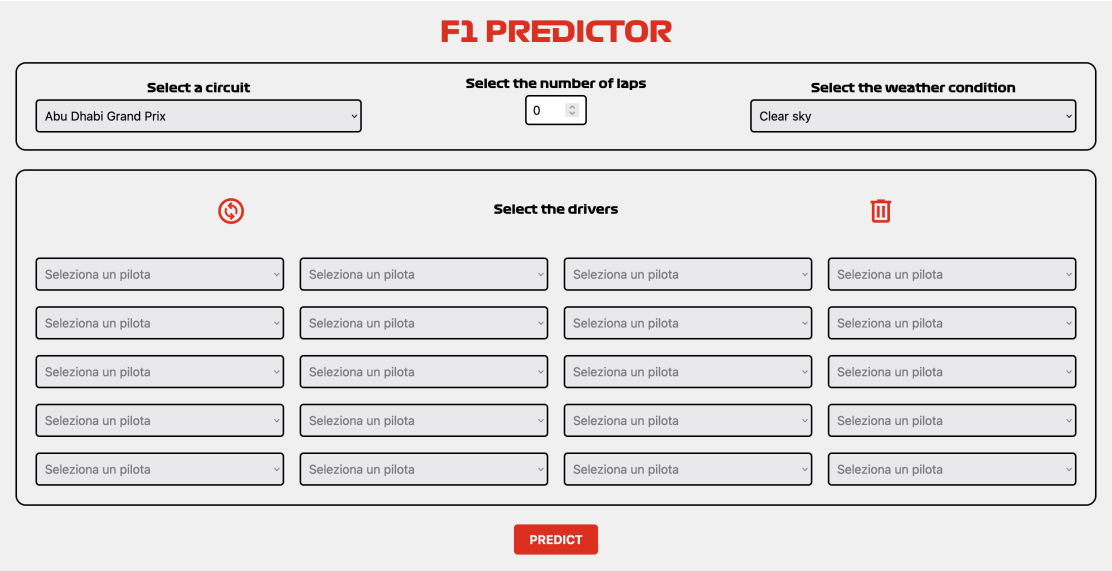
È evidente come il modello migliore sia stato il *Random Forest Regressor*, dunque la scelta ricadrà su di lui. I risultati ottenuti sotto il punto di vista dell'*accuratezza* non sono eccellenti, tuttavia possiamo ritenerci soddisfatti in quanto, il valore del *Mean Absolute Error* è di circa 6 secondi, decisamente tollerabile.

7 Deployment

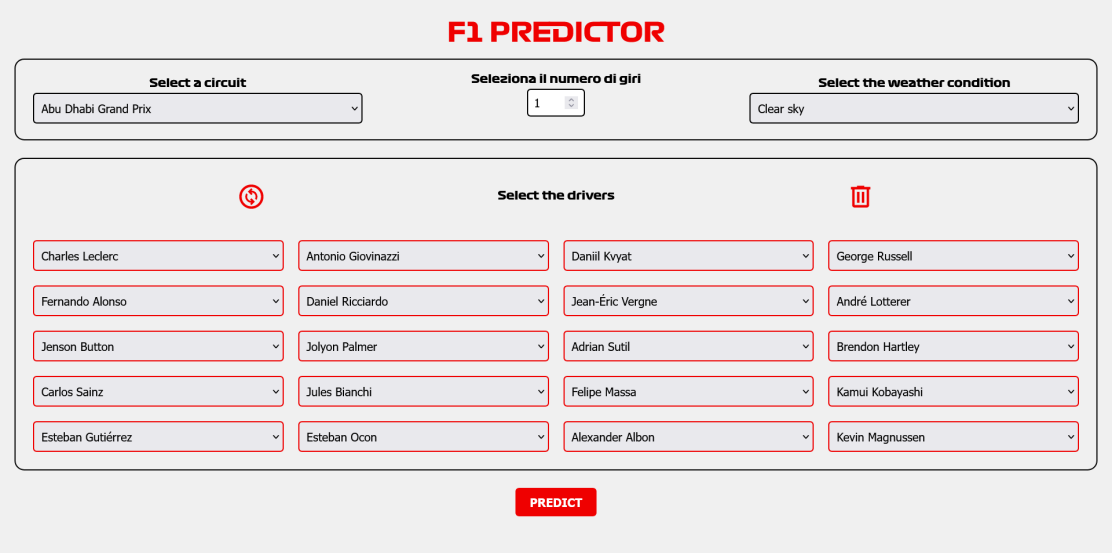
Finalmente si è arrivati alla fine, qui l'obettivo è *semplicemente* rendere operabile l'intero sistema.

7.1 L'applicazione in esecuzione

Ecco come si presenta la nostra applicazione:



Come è possibile subito notare presenta un'interfaccia grafica minimale ed intuitiva. La prima sezione è quella relativa ai *parametri* modificabili dall'utente. Qui potrà infatti selezionare il circuito e le condizioni meteo dal menù a tendina, inoltre potrà scegliere il numero di giri totali della gara (numero compreso tra 1 e 100). La seconda sezione invece è dedicata alla selezione dei piloti, i quali possono andare da un minimo di 1 ad un massimo di 20 (quando l'utente sceglie il pilota, la box relativa si illuminerà di rosso per evidenziare la scelta fatta).



Quando tutto è pronto, l'utente non deve fare altro che cliccare sul pulsante "Predict" e osservare i risultati ottenuti. Ecco un esempio di output:

PILOTA	TEMPO PREVISTO
1) Alexander Albon	00:01:49.1
2) Charles Leclerc	00:01:50.5
3) Daniel Ricciardo	00:01:51.1
4) Jean-Éric Vergne	00:01:51.1
5) Brendon Hartley	00:01:52.9
6) Carlos Sainz	00:01:54.0
7) Antonio Giovinazzi	00:01:54.6
8) George Russell	00:01:54.8
9) Felipe Massa	00:01:55.3
10) Esteban Ocon	00:01:55.4
11) Jenson Button	00:01:55.9
12) Adrian Sutil	00:01:55.9
13) Daniil Kvyat	00:01:56.1
14) Jolyon Palmer	00:01:56.1
15) André Lotterer	00:01:56.5
16) Fernando Alonso	00:01:57.1
17) Jules Bianchi	00:01:58.1
18) Kevin Magnussen	00:01:58.2
19) Kamui Kobayashi	00:01:58.3
20) Esteban Gutiérrez	00:01:58.6

7.2 Istruzioni d'uso

Per eseguire l'applicazione è necessario scaricare l'intero progetto dalla repository GitHub: Formula 1 Predictor e avere installato **Python** sul proprio dispositivo. Fatto questo è necessario installare tutte le librerie qui citate (ad ogni modo presenti nel file `requirements.txt` all'interno della repository). Fatto questo non resterà altro che eseguire il file `app.py` ed usare l'applicazione!

8 Conclusioni

Di seguito sono disponibili le riflessioni personali, l'elenco degli acronimi utilizzati (glossario) e i link (riferimenti) utilizzati per la stesura della documentazione.

8.1 Riflessioni

Il progetto *F1 Predictor*, con il suo obiettivo di predire una gara di F1, si presenta come un'interessante sfida nel campo della previsione sportiva. Tuttavia, riflettendo sui risultati ottenuti, emerge la consapevolezza che vi è ancora margine per migliorare le prestazioni del modello. Inoltre, l'introduzione di nuove feature come l'esperienza di un pilota o la percentuale di incidenti, potrebbe arricchire significativamente la comprensione del contesto e, di conseguenza, la precisione delle previsioni. Per concludere: siamo convinti che attraverso un continuo processo di aggiornamento e raffinamento del modello, è possibile perseguire l'obiettivo di fornire previsioni sempre più precise e utili agli appassionati e, chi lo sa, magari anche agli addetti ai lavori del mondo delle corse automobilistiche. ☺

8.2 Glossario

- F1 → Formula 1
- FIA → Federazione Internazionale dell'Automobile
- CRISP-DM → Cross-Industry Standard Process for Data Mining
- CSV → Comma-separated values
- API → Application programming interface

8.3 Riferimenti

- GitHub: Formula 1 Predictor
- Formula 1 - Wikipedia
- Formula 1 World Championship (1950 - 2023)
- Open-Meteo Weather API

Grazie per l'attenzione!

0512111084 - Teodoro Grauso
0512112926 - Giovanni Manfredi