Python For Data Analysis Master 1
1st semester - Project

Data Visualisation and Prediction relying on FastF1 Python API.

Authors : *Porterie Benjamin, Touitou Yves*


stc
HONDA
PETRONAS

Space of Data in Formula 1

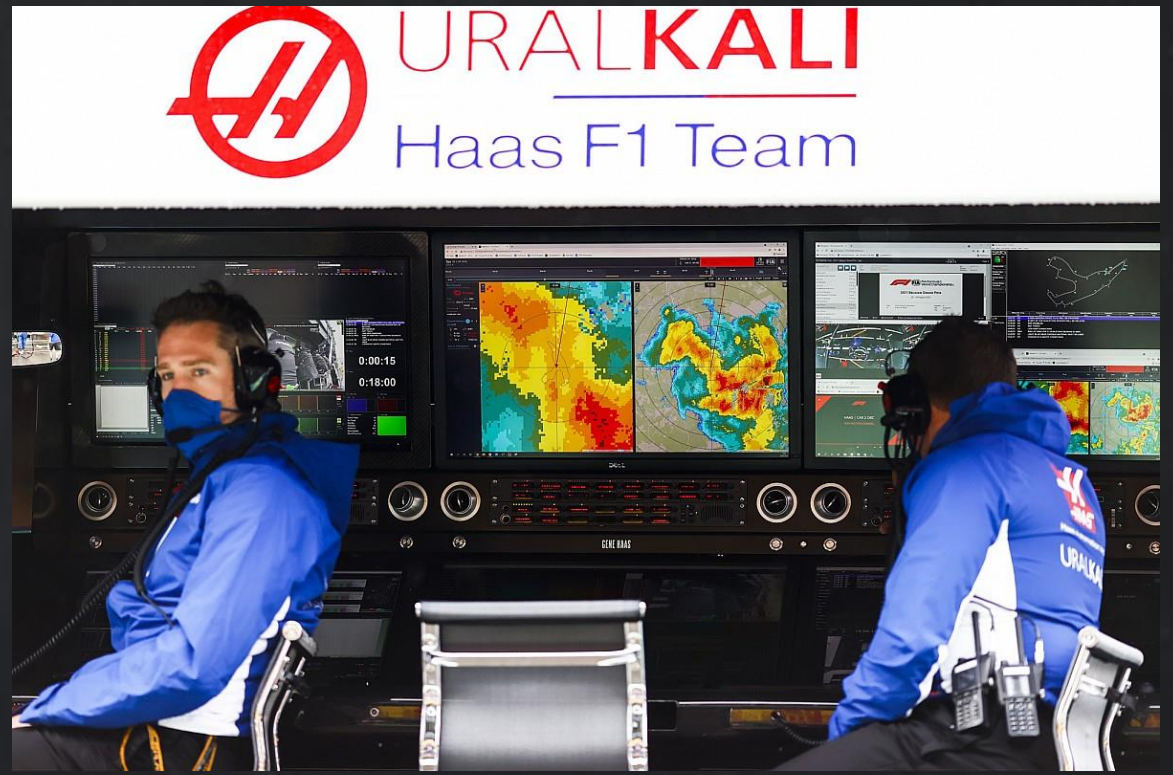
To win races, engineers follow the drivers at each circuit venue to assist them in decision making.

Those engineers on site are complementary with all the engineers back in the formula 1 team’s factory.

Throughout the weekend, even the driver invest time in data analysing to improve his driving skills until Sunday, which is race day.

Two different sorts of Data

- **Internal data** : everything related to the car.
  (Velocity, Shifting, Throttle & Brake pedals, Tyres temperature, Engine status, Gearbox status…)
- **External data** : everything not related to the car.
  (Weather forecasts, Track temperature & grip condition, Track Flags, Other cars light telemetry…)



Fast F1 (2.1.12)

Python API

Core

`Weekend`	Object for accessing weekend specific data.
`Session`	Object for accessing session specific data.
`Laps`	Object for accessing lap (timing) data of multiple laps.
`Lap`	Object for accessing lap (timing) data of a single lap.
`Telemetry`	Multi-channel time series telemetry data
`Driver`	Driver class that provides some information on drivers and their finishing results.

Session class

- Methods

`load_laps` ([with_telemetry, livedata])	Load lap timing information and telemetry data.
`load_telemetry` ([livedata])	Load telemetry data from the API.
`get_driver` (identifier)	Get a driver object which contains additional information about a driver.

- Attributes

`weekend`	Reference to the associated weekend object.
`name`	Name of this session, for example 'Qualifying', 'Race', 'FP1', ...
`date`	Date at which this session took place.
`session_status`	Session status data as returned by *fastf1.api.session_status_data()* as dataframe.
`results`	Race result with driver information.
`laps`	Instance of `Laps` containing all laps from all drivers in this session.
`t0_date`	Date timestamp which marks the beginning of the data stream.
`session_start_time`	Session time at which the session was started.
`car_data`	Dictionary of car telemetry (Speed, RPM, etc.) as received from the api by car number (where car number is a string and the telemetry is an instance of `Telemetry`)
`pos_data`	Dictionary of car position data as received from the api by car number (where car number is a string and the telemetry is an instance of `Telemetry`)
`weather_data`	Dataframe containing weather data for this session as received from the api.
`drivers`	List of all drivers that took part in this session; contains driver numbers as string.

Laps class

- Methods

`get_telemetry` ()	Telemetry data for all laps in *self*
`get_car_data` (**kwargs)	Car data for all laps in *self*
`get_pos_data` (**kwargs)	Pos data for all laps in *self*
`get_weather_data` ()	Return weather data for each lap in self.
`pick_driver` (identifier)	Return all laps of a specific driver in self based on the driver's three letters identifier or based on the driver number .
`pick_drivers` (identifiers)	Return all laps of the specified drivers in self based on the drivers' three letters identifier or based on the driver number. This is the same as `Laps.pick_driver()` but for multiple drivers at once. ::.
`pick_team` (name)	Return all laps of a specific team in self based on the team's name .
`pick_teams` (names)	Return all laps of the specified teams in self based on the teams' name. This is the same as `Laps.pick_team()` but for multiple teams at once. ::.
`pick_fastest` ()	Return the lap with the fastest lap time.
`pick_quicklaps` ([threshold])	Return all laps with *LapTime* faster than a certain limit.
`pick_tyre` (compound)	Return all laps in self which were done on a specific compound.
`pick_track_status` (status[, how])	Return all laps set under a specific track status.
`pick_wo_box` ()	Return all laps which are NOT in laps or out laps.
`pick_accurate` ()	Return all laps which pass the accuracy validation check (lap['IsAccurate'] is True).
`iterlaps` ([require])	Iterator for iterating over all laps in self.

- Attributes

`QUICKLAP_THRESHOLD`	Used to determine 'quick' laps.
`base_class_view`	For a nicer debugging experience; can now view as dataframe in various IDEs
`telemetry`	Telemetry data for all laps in *self*

Lap class

- Methods

`get_telemetry` ()	Telemetry data for this lap
`get_car_data` (**kwargs)	Car data for this lap
`get_pos_data` (**kwargs)	Pos data for all laps in *self*
`get_weather_data` ()	Return weather data for this lap.

- Attributes

`telemetry`	Telemetry data for this lap

Telemetry class

- Methods

`join` (*args, **kwargs)	Wraps `pandas.DataFrame.join` and adds metadata propagation.
`merge` (*args, **kwargs)	Wraps `pandas.DataFrame.merge` and adds metadata propagation.
`slice_by_mask` (mask[, pad, pad_side])	Slice self using a boolean array as a mask.
`slice_by_lap` (ref_laps[, pad, pad_side, ...])	Slice self to only include data from the provided lap or laps.
`slice_by_time` (start_time, end_time[, pad, ...])	Slice self to only include data in a specific time frame.
`merge_channels` (other[, frequency])	Merge telemetry objects containing different telemetry channels.
`resample_channels` ([rule, new_date_ref])	Resample telemetry data.
`fill_missing` ()	Calculate missing values in self.
`register_new_channel` (name, signal_type[, ...])	Register a custom telemetry channel.
`get_first_non_zero_time_index` ()	Return the first index at which the 'Time' value is not zero or NA/NaT
`add_differential_distance` ([drop_existing])	Add column 'DifferentialDistance' to self.
`add_distance` ([drop_existing])	Add column 'Distance' to self.
`add_relative_distance` ([drop_existing])	Add column 'RelativeDistance' to self.
`add_driver_ahead` ([drop_existing])	Add column 'DriverAhead' and 'DistanceToDriverAhead' to self.
`calculate_differential_distance` ()	Calculate the distance between subsequent samples of self.
`integrate_distance` ()	Return the distance driven since the first sample of self.
`calculate_driver_ahead` ()	Calculate driver ahead and distance to driver ahead.

- Attributes

`TELEMETRY_FREQUENCY`	Defines the frequency used when resampling the telemetry data.
`base_class_view`	For a nicer debugging experience; can view DataFrame through this property in various IDEs

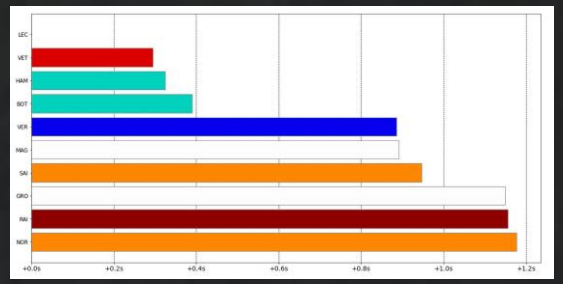
Driver class

- Attributes

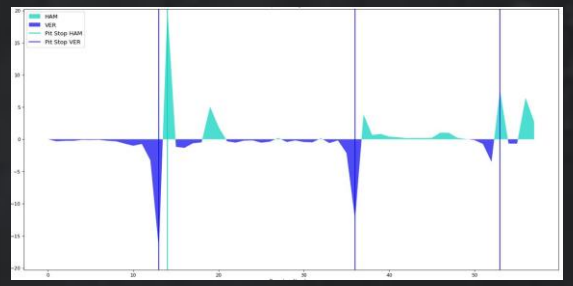
`info`	*Driver.info* contains some more info from the Ergast api
`dnf`	True if driver did not finish
`grid`	Grid position
`position`	Finishing position
`name`	Driver first name
`familyname`	Driver family name
`team`	Team name

Fast F1 Data possibilities

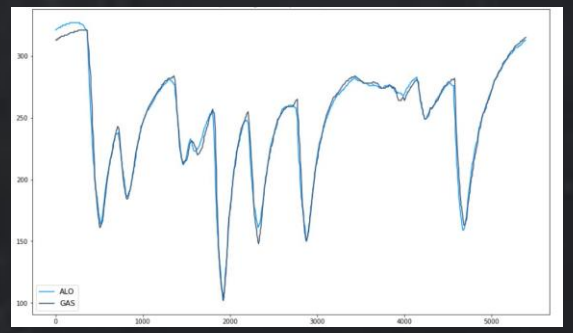
- Sessions results over the years



- Session details lap by lap



- F1 Cars telemetry



Fast F1 Data weaknesses

Data inconstancy in session laps and telemetries reports

Practices results aren’t accessible by “results” from session object and need to be retrieved by session’s laps analysis

---

Some practices sessions aren’t available, only 26 weekends out of 81 (between 2018 and 2021) are complete

Our Fast F1 notebooks

Practices_Records.ipynb

⟹ Retrieves practices sessions results by analysing the *Laps* object

Dataset_Creation.ipynb

⟹ Retrieves qualifications and races sessions results by analysing the *Session.Results* data.
Then assemble practices and qualifications/races data into "**allData.csv"**

F1_Visualisation.ipynb

⟹ Offers multiple plot displays to have an overview of an F1 season, an F1 session or compare two drivers performance
Uses "**allData.csv**", *Laps* and *Telemetry* objects

Sunday_Prediction.ipynb

⟹ Gathers all prediction models attempts, correlation and models performance graphs
1) "**allData.csv**" preparation for AI
2) Training, pruning and best model selection
3) Result analysis

allData.csv

Variable	Description
number	Driver’s car number
position	Driver’s session rank
Q1	Qualification 1 session driver’s time
Q2	Qualification 2 session driver’s time
Q3	Qualification 3 session driver’s time
positionText	Driver’s session rank or ‘R’ if driver didn’t finish
points	Driver’s points collected from race day
grid	Driver’s position in race starting grid
laps	Driver’s race laps completed
status	Driver’s race end situation (Finished, Lapped, Car failure…)
year	Grand Prix’s year
gpName	Grand Prix’s name (host country)
gpNumber	Grand Prix’s number within the season
sessionName	‘Practice’, ‘Qualifying’ or ‘Race’
driverId	Driver’s family name
code	Driver’s code (Hamilton is HAM)
DriverNationality	Driver’s Nationality
constructorId	Constructor’s name
constructorNationality	Constructor’s Nationality
fastestLapNumber	Lap number the driver’s made his best lap time within the session
fastestLapRank	Driver’s fastest lap rank
fastestLapAvgSpeed	Driver’s fastest lap’s average speed
fastestLapTime	Driver’s fastest lap time
totalTime	Driver’s time spent in session
TimeInterval	Driver’s gap behind race winner

Project Goal

Can we predict Sunday race results?

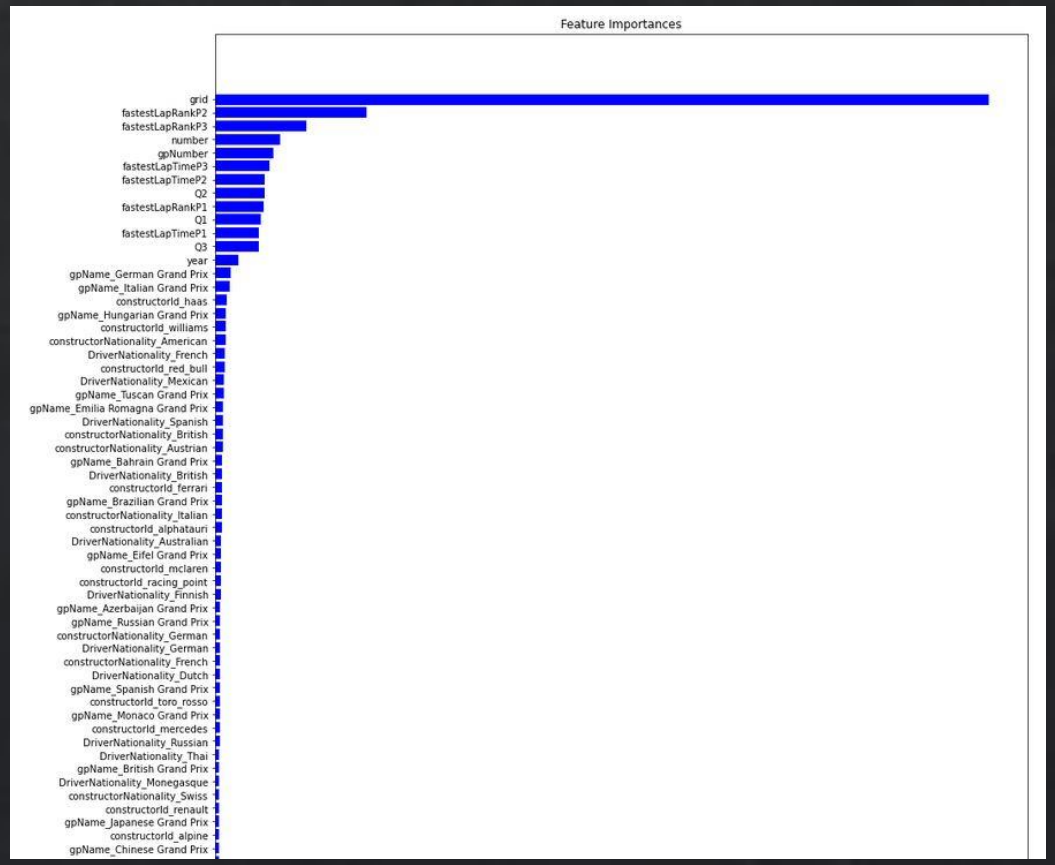
Every weekend, Formula One fans are slightly disappointed when the race ends up the way they predicted and love it when Sunday's event gets hectic.

Can we detect reliable predictors and what are the obstacles predicting Sunday's race result?

First prediction dataframe

```
df_pred_postion.columns
```

```
['number', 'position', 'grid', 'year', 'gpName', 'gpNumber',
 'DriverNationality', 'constructorId', 'constructorNationality', 'Q1',
 'Q2', 'Q3', 'fastestLapRankP1', 'fastestLapTimeP1', 'fastestLapRankP2',
 'fastestLapTimeP2', 'fastestLapRankP3', 'fastestLapTimeP3'],
```



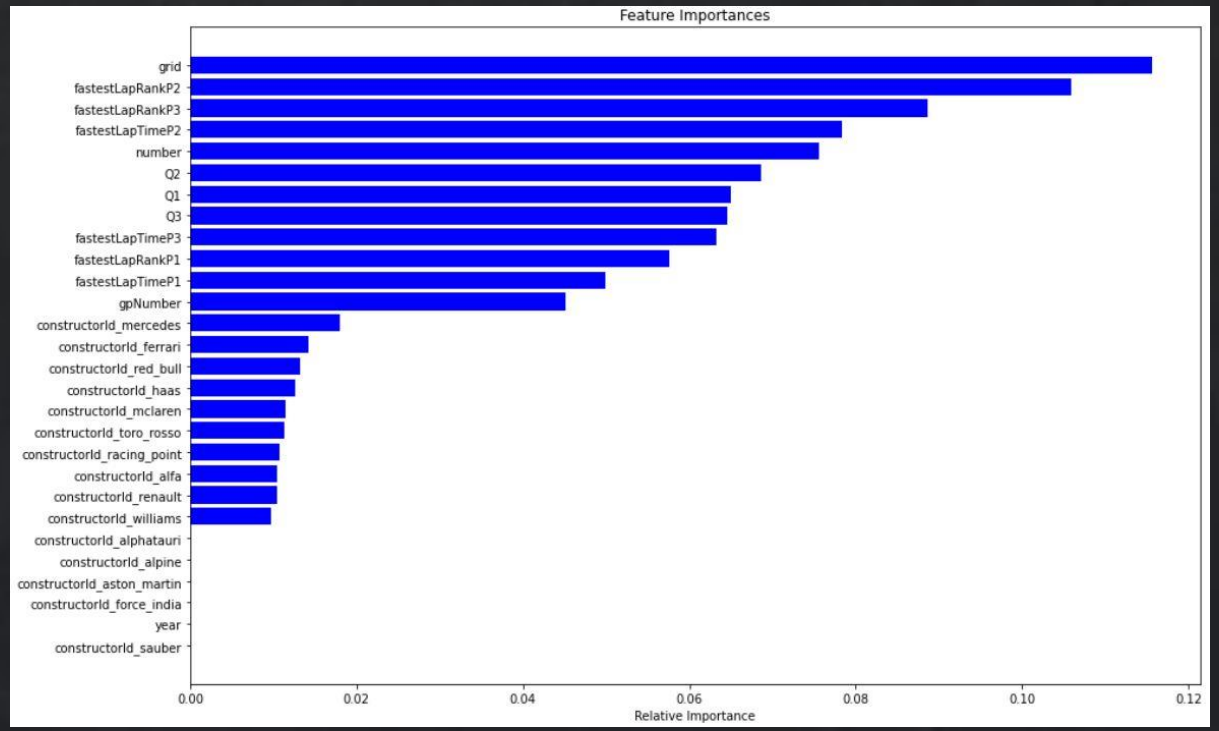
Second prediction dataframe

Insignificant variables have been dropped to lighten the predicting dataframe

Variables dropped : 'DriverNationality', 'constructorNationality', 'gpName'

```
df_features2.columns
```

```
['number', 'position', 'grid', 'year', 'gpNumber', 'constructorId', 'Q1',
 'Q2', 'Q3', 'fastestLapRankP1', 'fastestLapTimeP1', 'fastestLapRankP2',
 'fastestLapTimeP2', 'fastestLapRankP3', 'fastestLapTimeP3'],
```



Predicted data shape

Returned data needs to be an array of positions, from 1 to the number of drivers taking part in Sunday's race. All those positions need to be distinct.

```
y_pred
```

```
[ 5.,  4.,  7.,  2.,  1., 12.,  6., 16., 11., 17.,  8., 15., 14.,
 18., 10., 13., 20., 19.,  3.,  9.])
```

Prediction conjectural performance

Random estimation mean accuracy : 1 / Sunday's race drivers number

Between 2018 and 2021, F1's grid was composed of 20 drivers ⟹ $P_{mean} = 1/20 = \mathbf{5\%}$

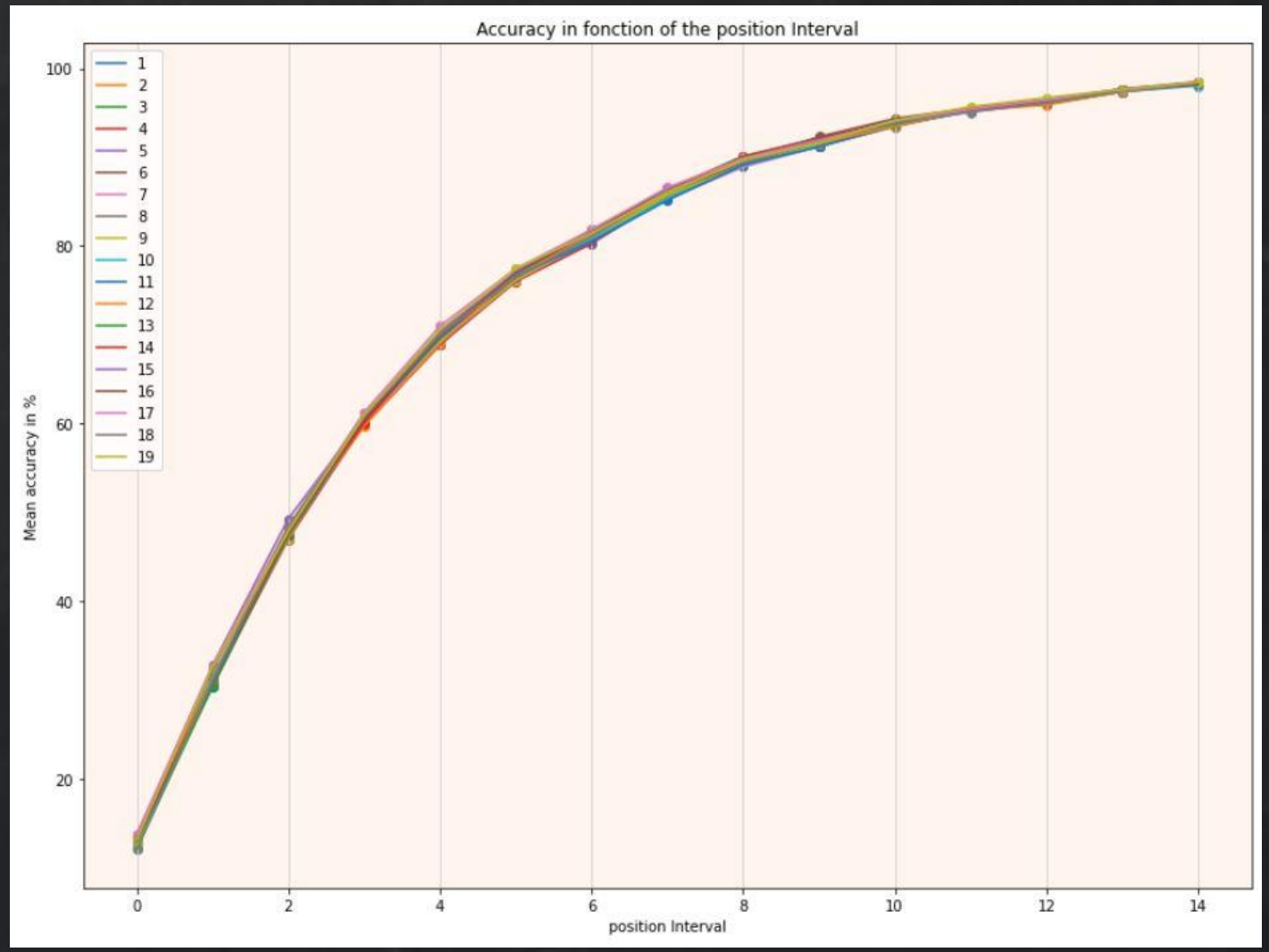
On Sundays, several types of unexpected moments can completely modify the race finishing order, making it difficult to be predicted :

- *Contact between drivers*
- *Contact with walls (turn misjudgement)*
- *Car failures*
- *Weather changes*
- *Drivers strategy efficiency*
- *Tyre wear higher than expected*

We expect our prediction accuracy to be better than the random one but still being low because of those moments, sometimes making the race result a mess.

Prediction practical performance

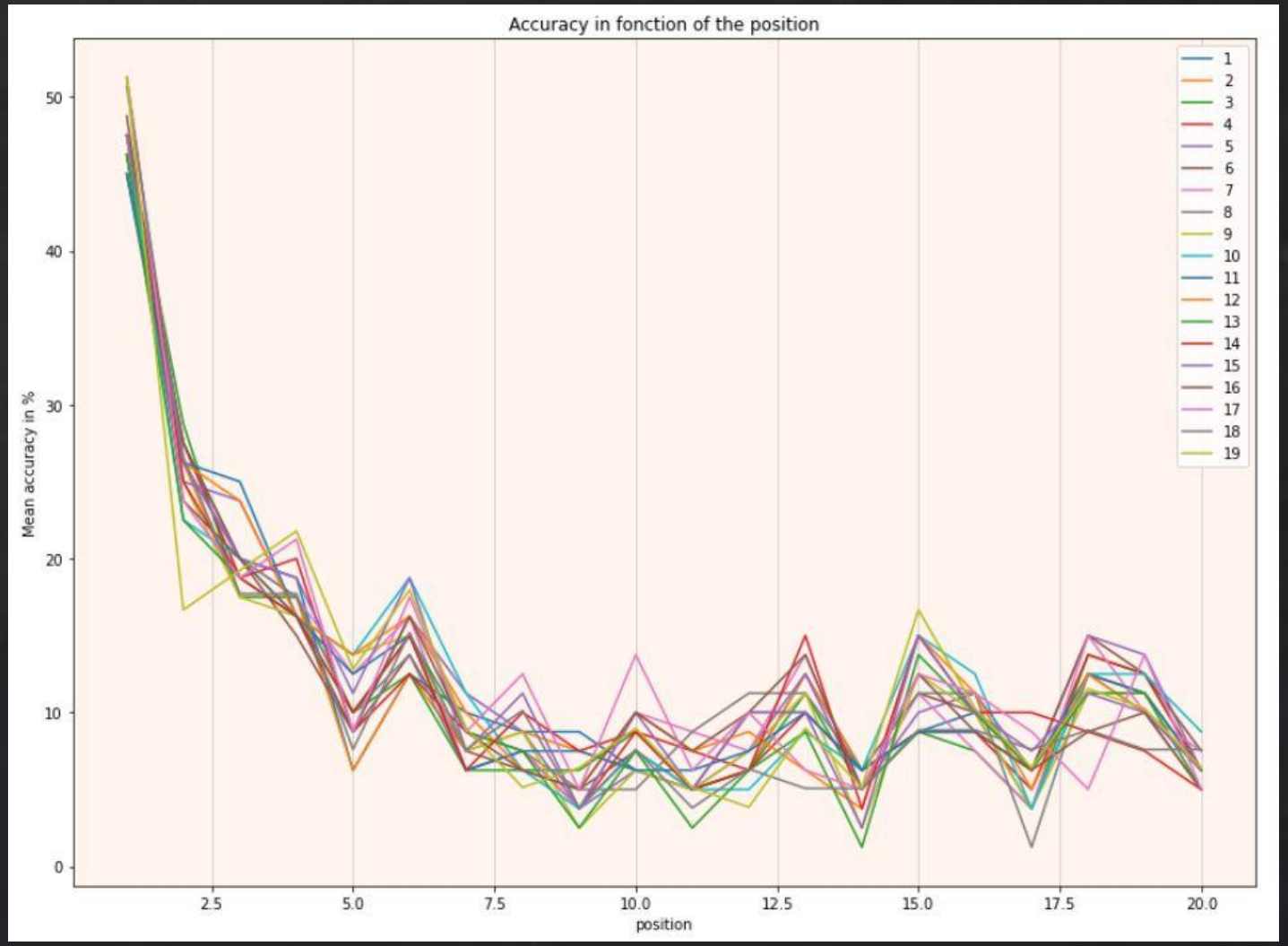
- This graph shows our models global performance with a customised prediction's margin error (-/+).



Prediction practical performance

- We can observe that our models can make very good predictions concerning the first drivers and the podium.

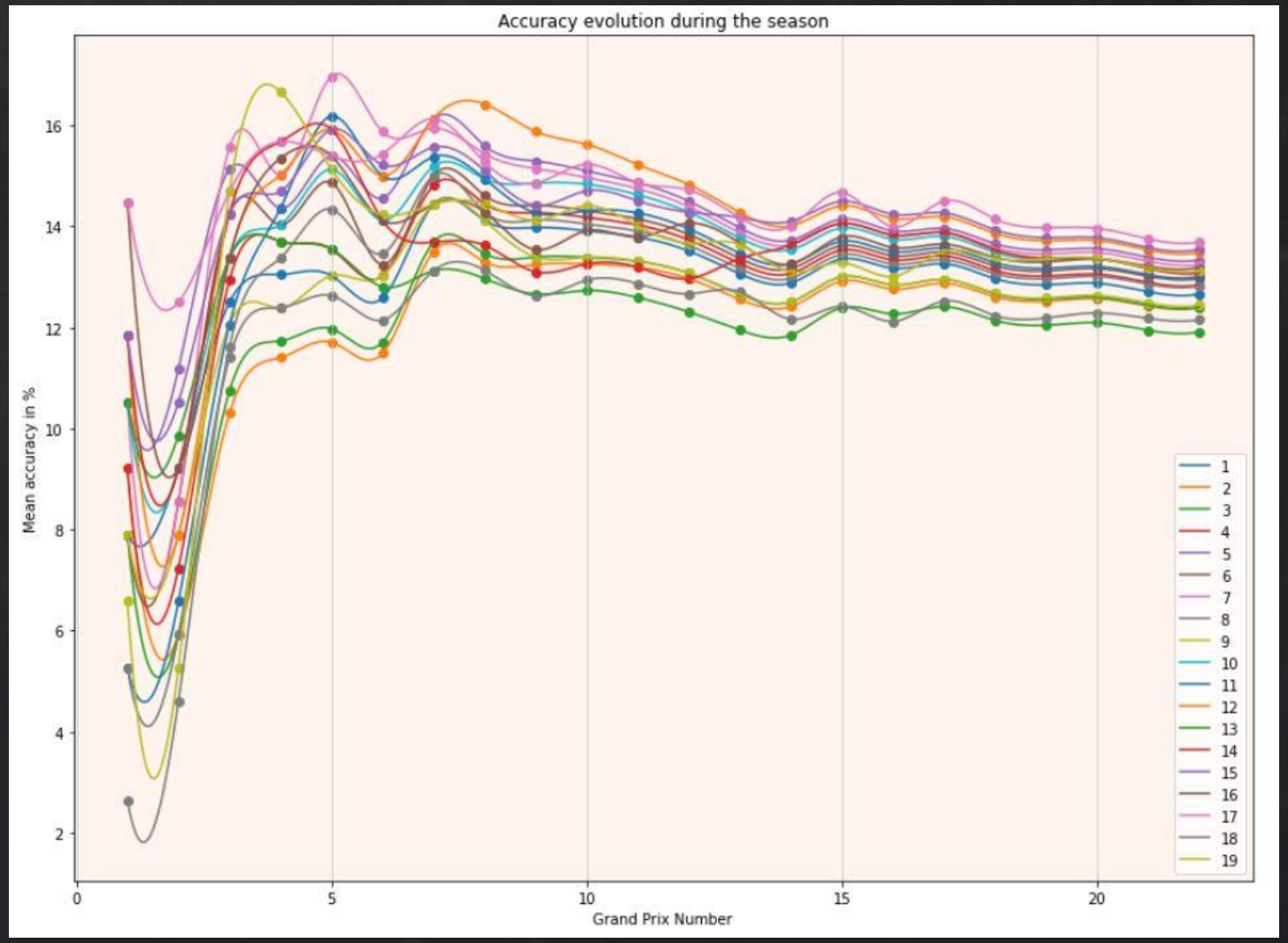
This is explained by Mercedes, Ferrari and Red Bull domination over the midfield when they were fighting for race wins throughout the last few years.



Prediction practical performance

- This graph shows our models global performance as we make progress into the season.

It tends to stabilize because as the season goes by, our predictive models lean on previous Grand Prix to sharpen the prediction.



Prediction model choice

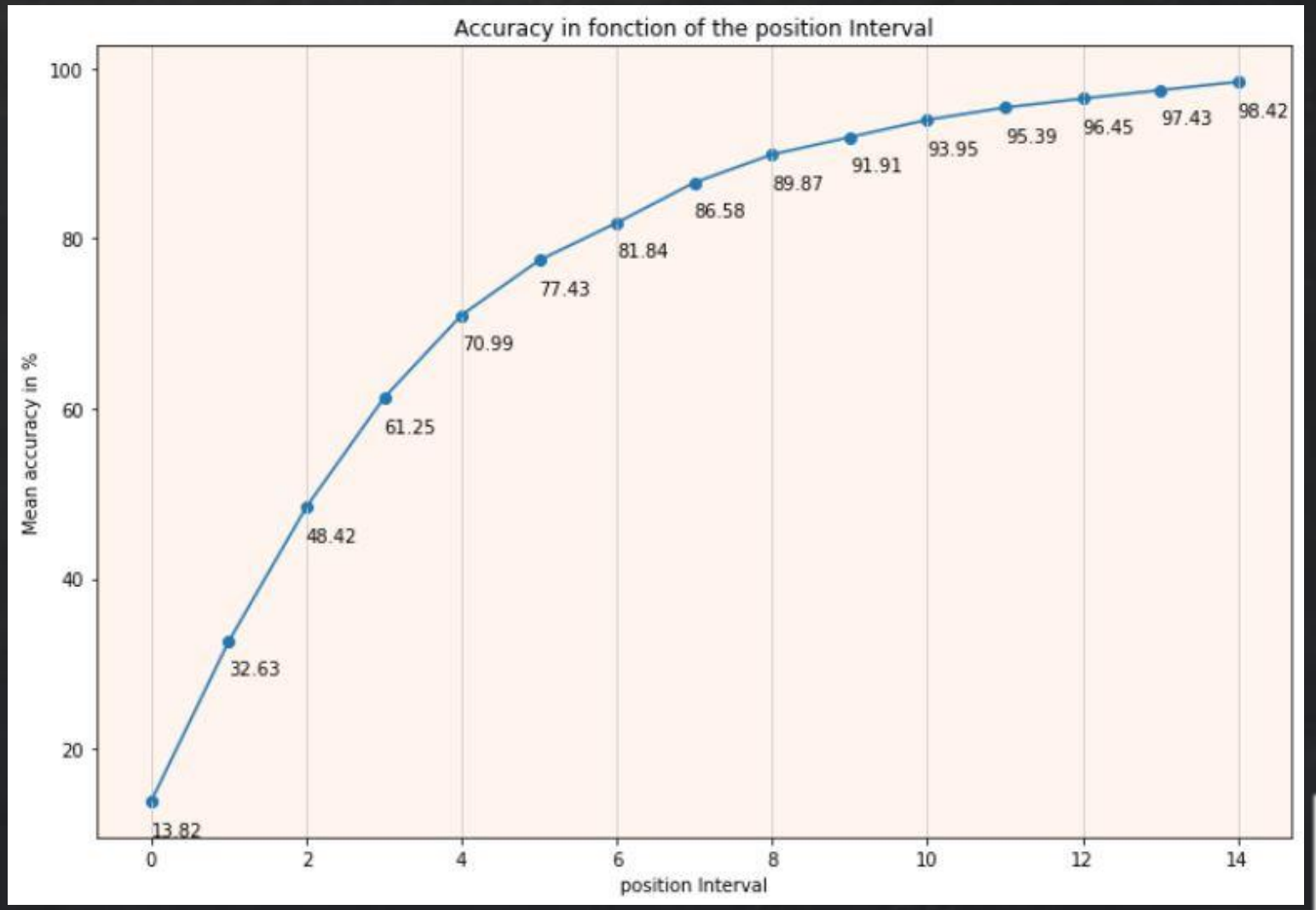
After comparing models' accuracies, we decided to use a Random Forest Regressor, with an Output Ranking transformation.

Optimum model parameters :

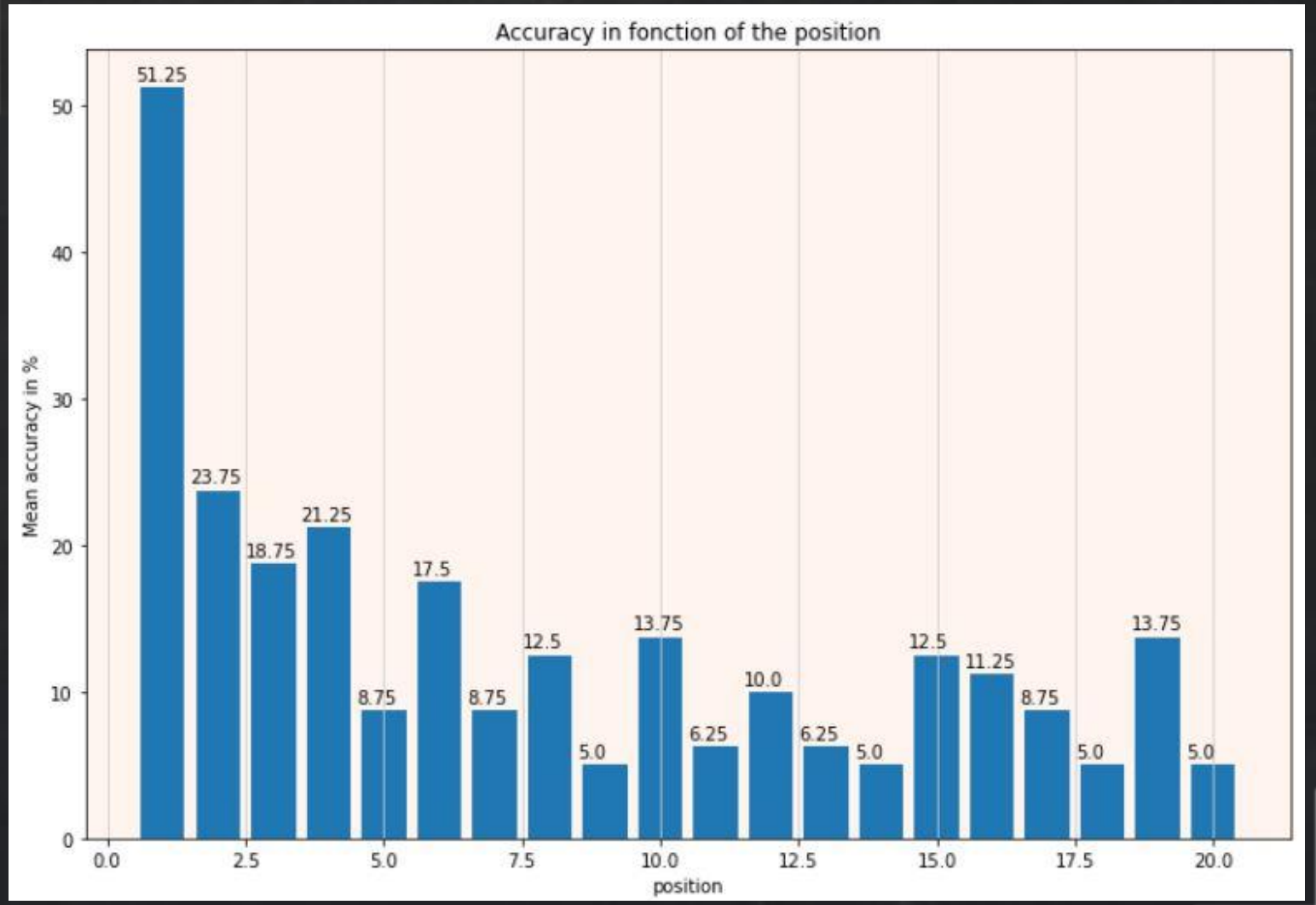
- Random State = 47
- Number of previous weekends considered = 17

Because of the nature of our problem, we must train that model for every new prediction.

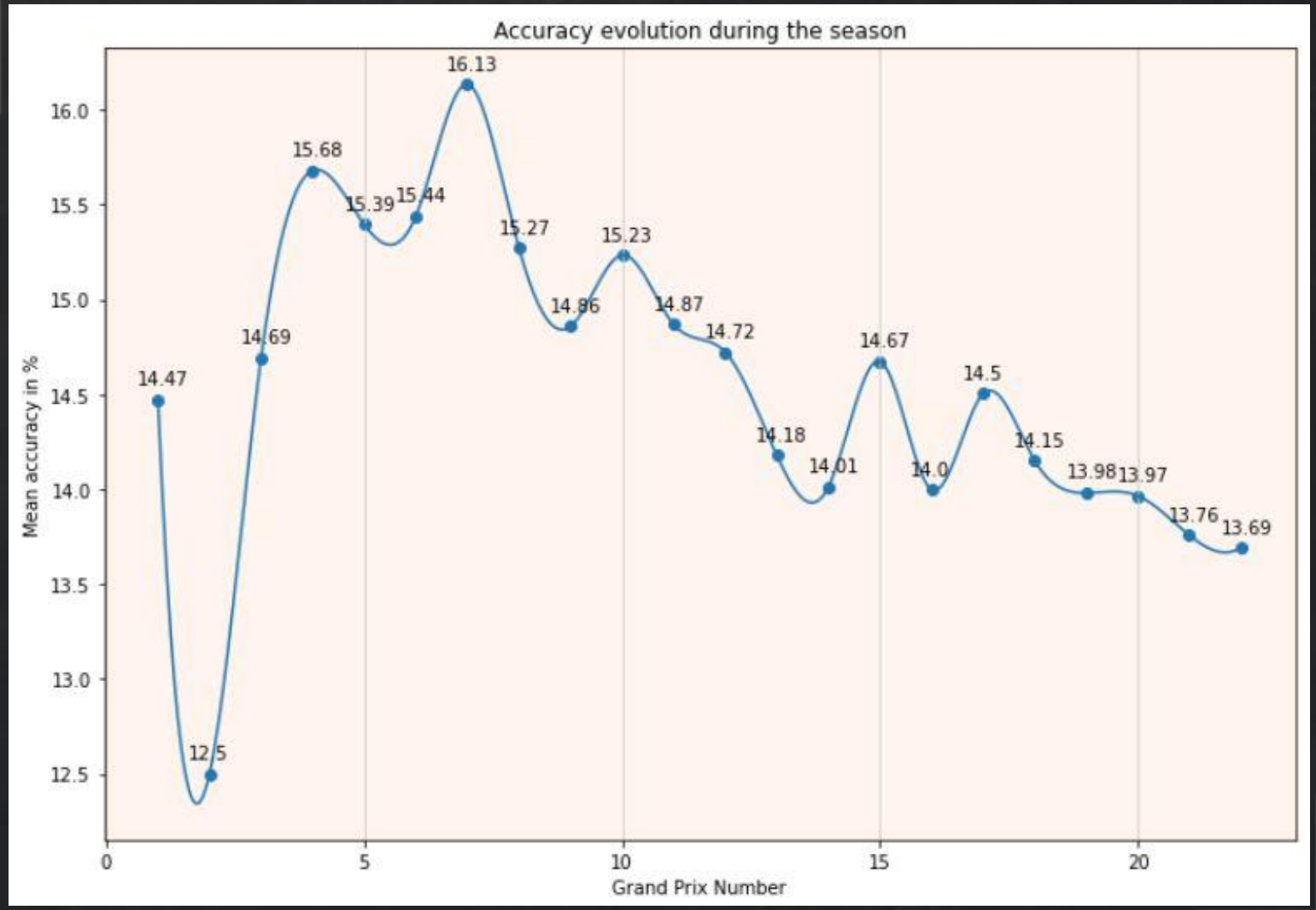
Model accuracy (position interval)



Model accuracy (function of position)



Model accuracy (over the season)



Appendix

Weekend overview to understand each session

Race weekend timeline

-- Traditional Format --

Friday

Practice 1/2 (P1/P2)

Saturday

Practice 3 (P3)
Qualification (Q)

Sunday

Race (R)

-- 2021 Sprint Format --

Friday

Practice 1 (P1)
Qualification (Q)

Saturday

Practice 2 (P2)
Sprint Race (SR)

Sunday

Race (R)