Formula one data analysis

1950-2020

Jérémy GOURDEAU
Antoine HAZART

The datasets

We used the dataset Formula one world championship (1950 - 2020)

link: https://www.kaggle.com/rohanrao/formula-1-world-championship-1950-2020

This dataset contains thirteen csv files with informations about races.

The datasets

We mostly used datasets races, pit stops, drivers, constructors, results. they give us information about tracks, drivers performances, and team performances.

To visualise those informations we had to merge datasets together.

Entrée [17]:
```
circuits_races = pd.merge(data_dict['races'],data_dict['circuits'], on=['circuitId'])
hist = circuits_races.year.hist(bins=len(circuits_races['year'].value_counts()))
```

The datasets

we use drivers dataset to get the name and the id of a driver, constructors to get constructor names, and races dataset to get the name of tracks.

In order to see results, and datas about those sets, others data sets have at least one column that refers to one of the three datasets.

The datasets

Entrée [6]:
```
data_dict['circuits'].sample(1)
```

Out[6]:

	circuitId	circuitRef	name	location	country	lat	lng	alt	url
75	76	mugello	Autodromo Internazionale del Mugello	Mugello	Italy	43.9975	11.3719	\N	http://en.wikipedia.org/wiki/Mugello_Circuit

Entrée [13]:
```
data_dict['pit_stops'].sample(1)
```

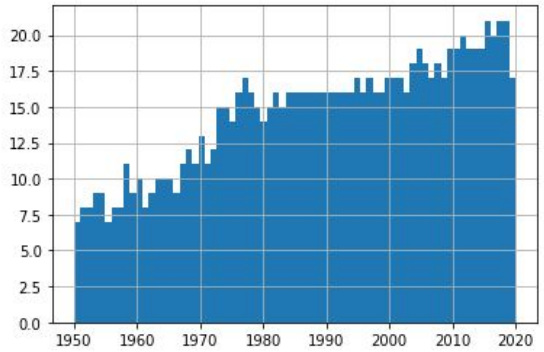
Out[13]:

	raceId	driverId	stop	lap	time	duration	milliseconds
6933	1014	841	2	41	16:12:09	23.507	23507

Here we have samples from pit stops and circuits datasets. We can see that they share one column which permit to identify during which race this stop has been done.

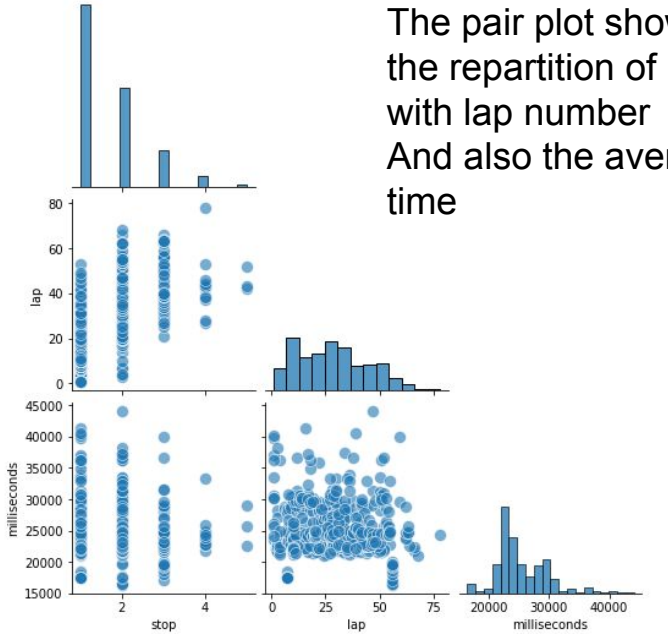
Data visualisation

In our analysis we'll focus on different variables given by the dataset.



Histogram showing how many races per year

The number of races tends to increase over the decades
The biggest decrease was this year

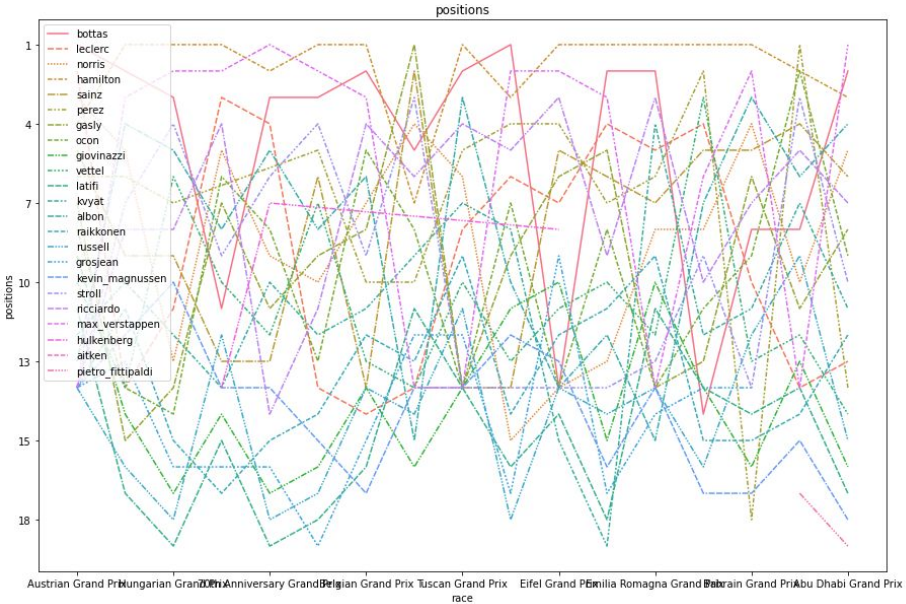


Pair plot of pit stops data

The pair plot shows us the repartition of pit stops with lap number
And also the average time

Data visualisation

We thought about comparing position of each race for each driver

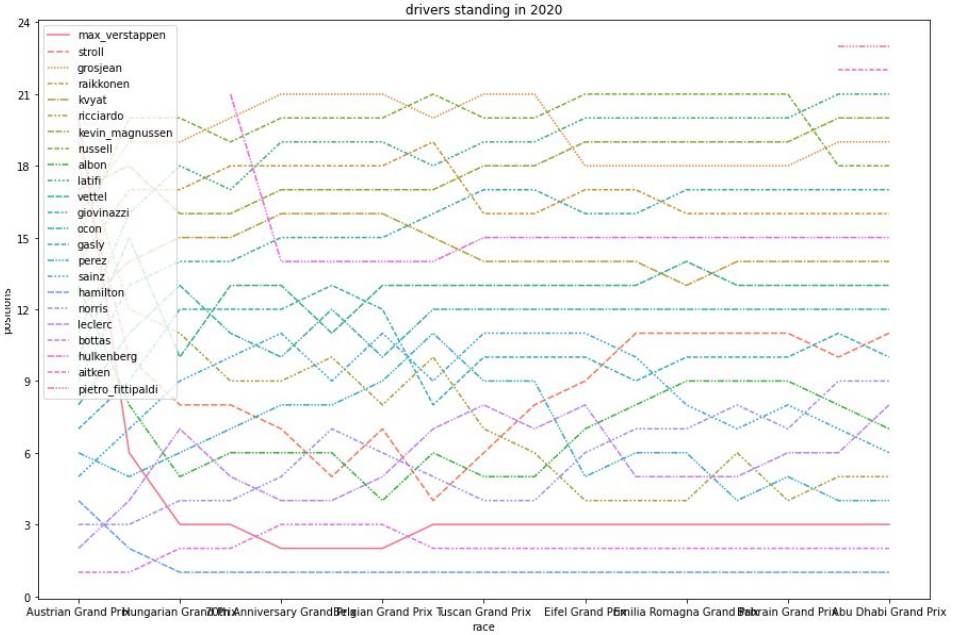


Obviously this graph is difficult to read, but we're still able to see hamilton's dominance at the top

To make a graph similar but easier to read we thought about comparing their ranking with total scores (next slide)

Data visualisation

We thought about comparing pilots scores ranking evolution over a season

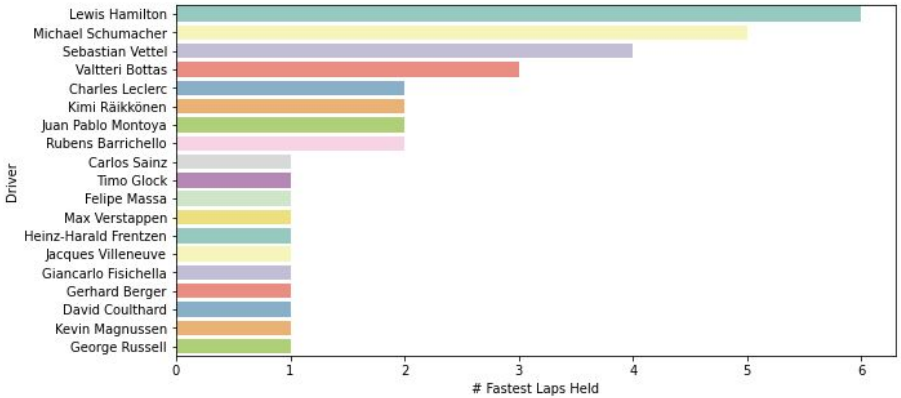


Here we plot the evolution in the players ranking. We can see the dominance of the top 3, at the bottom of the graph.

Whereas places 4-11 are challenged and always changing

Data visualisation

To go further, we tried to find information by merging several parts of the dataset

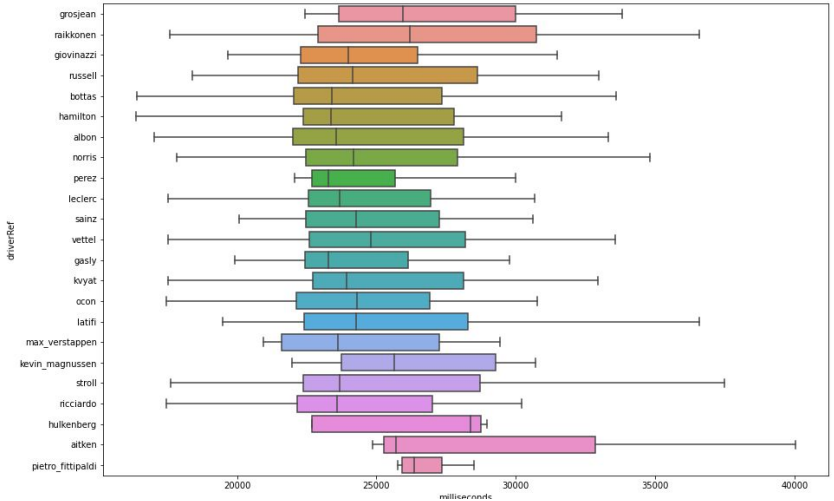


By merging races, drivers,circuits and lap times dataset, we're able to retrieve the drivers who hold the fastest laps on a circuit

We can see that Hamilton just passed on first position, beating Schummarer

Data visualisation - Pit stops

Since pit-stops are the base of a race strategy we were interested in their analysis



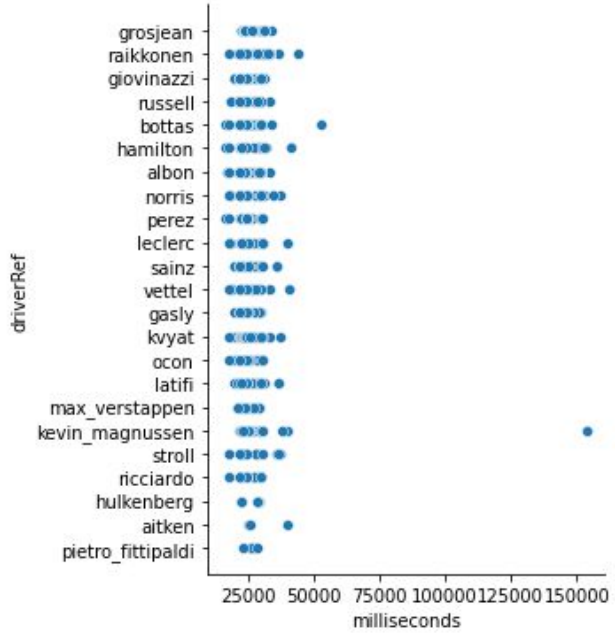
Here are the box plots of pit stop times for a season

The ranges are quite large, this is due to the differences we can get between the sizes of the track in the stop area, depending on the circuits.

This will have to be taken into account for pit stops prediction

Data visualisation - Pit stops

Since pit-stops are the base of a race strategy we were interested in their analysis



Also, it was very important to get rid of the outliers, since there was a lot, because of the abandons at the pit stop.

To see the outliers we plotted the data with this graph, and reducing the range until no outliers were found

Data Prediction- Pit stops

We will try to predict if a driver will stop at a given lap

To do this, we had to create an optimal dataset with enough columns to represent the pilot state at any time of the race

By merging races and pit-stops, we get the following dataframe :

	year	round	circuitId	driverId	stop	milliseconds	lap
0	2011	1	1	153	1	26898	1
1	2011	1	1	30	1	25021	1
2	2011	1	1	17	1	23426	11
3	2011	1	1	4	1	23251	12
4	2011	1	1	13	1	23842	13
...	...	...	...	...	...	...	...
8025	2020	17	24	20	1	22040	35
8026	2020	17	24	849	2	22384	35
8027	2020	17	24	817	1	22123	39
8028	2020	17	24	825	2	23098	47
8029	2020	17	24	850	3	23217	48

Data Prediction- Pit stops

We will try to predict if a driver will stop at a given lap

To get a more precise model, we added 2 columns, storing how many previous laps we did, and the number of laps since last stop

We also had to replace every stop value (which could be 2, or 3) by 1 or 0. Since this is what we want to predict

The number of the stop is now stored in the column laststops

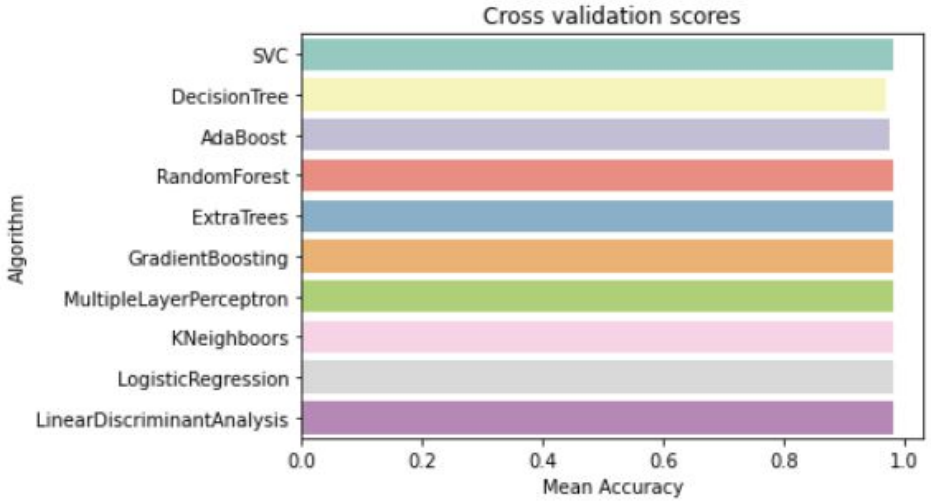
	circuitId	driverId	lap	position	stop	laststops	sincelaststop
0	1	20	1	1	0	0.0	0
1	1	20	2	1	0	0.0	1
2	1	20	3	1	0	0.0	2
3	1	20	4	1	0	0.0	3
4	1	20	5	1	0	0.0	4
...	...	...	...	...	...	...	...
490899	24	815	4	17	0	0.0	3
490900	24	815	5	16	0	0.0	4
490901	24	815	6	15	0	0.0	5
490902	24	815	7	15	0	0.0	6
490903	24	815	8	14	0	0.0	7

Data Prediction- Pit stops

We will try to predict if a driver will stop at a given lap

We then tried several classifiers algorithms
With a low number of kfold
The best classifier found was the gradient boosting one

	CrossValMeans	CrossValerrors	Algorithm
5	0.983676	0.000018	GradientBoosting
0	0.983665	0.000003	SVC
8	0.983665	0.000003	LogisticRegression
6	0.983609	0.000059	MultipleLayerPerceptron
9	0.983576	0.000010	LinearDiscriminantAnalysis
7	0.983240	0.000005	KNeighboors
3	0.982879	0.000010	RandomForest
4	0.981967	0.000026	ExtraTrees
2	0.977671	0.000033	AdaBoost
1	0.970139	0.000415	DecisionTree



Data Prediction- Pit stops

Using grid search, we then tried to optimize our gradient boosting classifier

```
GBC = GradientBoostingClassifier()
gb_param_grid = {'loss' : ["deviance"],
              'n_estimators' : [100,200,300],
              'learning_rate': [0.1, 0.05, 0.01],
              'max_depth': [4, 8],
              'min_samples_leaf': [100,150],
              'max_features': [0.3, 0.1]
              }

gsGBC = GridSearchCV(GBC,param_grid = gb_param_grid, cv=kfold, scoring="accuracy", n_jobs= 4, verbose = 1)

gsGBC.fit(X_train,y_train)

GBC_best = gsGBC.best_estimator_
```

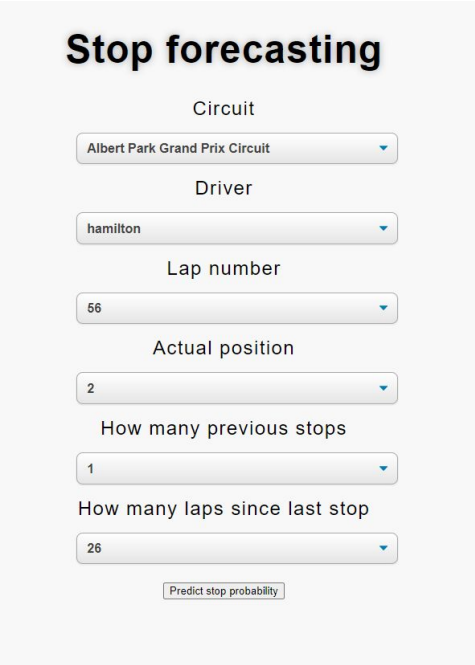
```
Fitting 2 folds for each of 72 candidates, totalling 144 fits

0.983782462467433
```

The score is now even higher than the previous one (but only by 0.001)

Data Prediction- Pit stops

After pickling the model, all we needed is to start building the flask API



Note that a stop is a rare event, so instead of printing just a 0 or a 1, decided to print the stop probability

Most of the time the model will predict a value around 0