Projet Analyse de données

Noureddine KHELIFAOUI, Noé LALLOUET

April 2020

Introduction

Ce projet porte sur l'étude et l'analyse des courses de Formule 1 qui se sont déroulées depuis 1950 jusqu'à 2019. Nous avions, dès l'annonce du projet, souhaité traiter ce sujet. Nous avons trouvé le dataset à analyser sur le site internet ergast.com. Dans ce projet, nous allons essayer de résoudre trois problèmes :

- Prédire la position à l'arrivée d'un pilote
- Prédire à quelle moment de la course un pilote effectue un arrêt au stand
- Classifier le composé de pneus utilisé par un pilote

Bien entendu, nous clarifierons toutes ces notions dans les pages qui suivent.

Figure 1: Sebastian Vettel remportant le Grand Prix de Singapour, 2019

1 Mise en contexte

1.1 La Formule 1

Il ne fait que peu de doute que nul n'ignore l'existence de la Formule 1, la catégorie reine du sport automobile. Ce championnat, disputé à travers le globe pendant, aujourd'hui, plus de vingt courses, couronne chaque année un pilote *(World Drivers' Championship, ou WDC)* ainsi qu'une écurie *(World Constructors' Championship, ou WCC)*.

Pendant un Grand Prix, environ vingt voitures prennent le départ selon un ordre déterminé préalablement par une séance de qualifications. Chaque écurie aligne deux pilotes sur la grille. Au bout d'un nombre de tours spécifique au circuit, le pilote dépassant le drapeau à damier en premier remporte la course.

De nombreux facteurs sont clés dans la lutte pour la victoire : en voici quelques uns, triés par ordre d'importance.

- **La performance de la voiture :** les disparités entre les différentes monoplaces, bien qu'exprimées en centièmes de seconde, sont la composante primordiale de l'issue d'une course : d'aucuns aiment à dire que les courses de Formule 1 sont remportées non pas sur la piste, mais à l'usine.
- **La position en qualifications :** l'ordre de départ d'un Grand Prix est souvent prémonitoire du classement final.
- **Le talent du pilote :** bien sûr, la performance d'un pilote spécifique est immensément importante dans la route vers la première place.
- **La stratégie :** mise en oeuvre par l'ingénieur de course, une stratégie efficace est toujours la clé d'une bonne performance en course.
- La gestion de l'essence, la gestion des pneus, l'utilisation astucieuse d'arrêts au stand, la réactivité aux évènements de course comme les incidents sur piste ou la tombée de pluie, etc.

1.2 Description du dataset

Le dataset dont nous disposons s'organise en plusieurs fichiers CSV : nous retrouvons, parmi les plus importants :

- **results.csv :** ce fichier répertorie les positions à l'arrivée de tous les pilotes pour toutes les courses de l'histoire de la Formule 1, en plus de leur position de qualification, leur position au championnat, etc.
- **lapTimes.csv :** ce fichier liste les temps de chaque tour pour chaque pilote et pour chaque course de l'histoire du championnat.

Dans les autres datasets, on peut trouver, entre autres, les classements pilote et constructeur, la liste des courses, des pilotes, des résultats de qualifications, etc. En somme, nous pouvons dire que ce dataset est plutôt complet, même si nous verrons plus tard que de nombreuses informations cruciales sont absentes.

2 Prédiction de la position à l'arrivée d'un pilote

Si vous aimez les histoires d'outsiders, la Formule 1 n'est peut-être pas le sport pour vous. Dans la catégorie reine du sport automobile, injuste pour certains, les mêmes écuries ont tendance à dominer le plateau pendant une saison entière, voire même plusieurs : il suffit d'examiner la domination de Michael Schumacher avec Ferrari en 2000-2004 ou bien celle de Lewis Hamilton avec Mercedes en 2014-2019 pour s'en rendre compte.

La position à laquelle un pilote commence la course est préalablement déterminée par une séance de qualifications, dans laquelle la voiture la plus puissante est plus susceptible d'obtenir la *pole position* que les écuries confinées au fond du plateau. Dans ce contexte, on peut essayer de prédire la position à l'arrivée d'un pilote, en se basant sur plusieurs caractéristiques comme son résultat de qualifications ou bien la puissance de sa voiture. Dans cette section, nous allons nous intéresser à ce problème de régression linéaire.

Figure 2: Charles Leclerc remportant le Grand Prix d'Italie, 2019

2.1 Analyse des données

Pour commencer, nous allons nous rendre compte des données dont on dispose, en fonction des facteurs clés pour remporter un Grand Prix vus plus haut. Nous sommes en possession du résultat de qualifications d'un pilote, mais c'est à peu près la seule donnée dont nous disposons.

Afin de remédier à ce problème, nous avons décidé de créer nos propres *features*. Nous avons commencé par essayer de modéliser la puissance d'une écurie, avec la création de deux *features*.

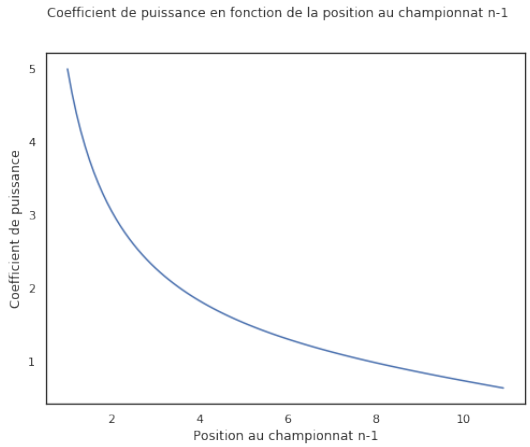


Figure 3: Répartition du coefficient de puissance

- **constructorPower** affecte un coefficient $\mu_1(x)$ à une écurie x, selon la formule suivante: $\mu_1(x) = (tan(0.1i)^{0.7})^{-1}$, avec i la position au championnat constructeur de l'écurie x l'année précédente. Ce coefficient est très grand pour les quelques écuries dominatrices et très petit pour les *backmarkers* : voici un graphique de sa distribution (Figure 3).

- **constructorForm** affecte, de manière analogue à constructorPower, un coefficient $\mu_2(x)$ à une écurie x, à la différence près que ce coefficient évolue en fonction de la progression de la saison : toutes les écuries démarrent sur un pied d'égalité et voient leur coefficient de forme changer au fur et à mesure des courses : pour illustrer ce procédé, nous tracerons l'évolution du coefficient de forme de 3 écuries de milieu de plateau en 2018 (Figure 4).

Nous avons ensuite décidé de modéliser la performance d'un pilote. Cette modélisation est légèrement plus complexe car, en raison de la nature inégale de la Formule 1, d'excellents pilotes peuvent hériter de monoplaces médiocres et, malgré leur talent, achever les courses et les championnats en mauvaise position. Cependant, il ne faut pas oublier que chaque écurie possède deux pilotes : on peut donc les comparer directement. C'est ainsi que nous avons créé un coefficient **driverPerformance** selon la formule suivante : $\mu_3(i) = \frac{1}{1+\exp^{((x_j-x_i)+(r_j-r_i))}}$ avec

- x_i la position moyenne de qualifications du pilote
- x_j la position moyenne de qualifications de son équipier
- r_i le résultat de course moyen du pilote
- r_j le résultat de course moyen de son équipier

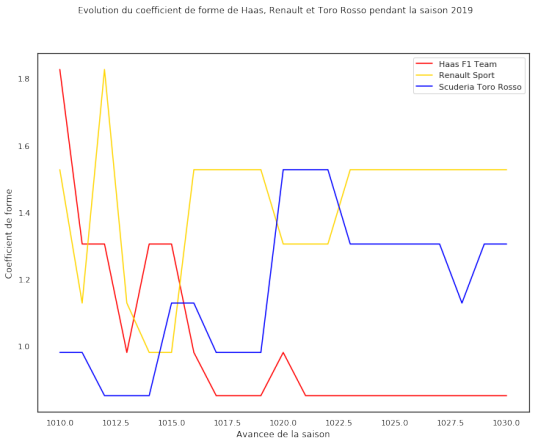
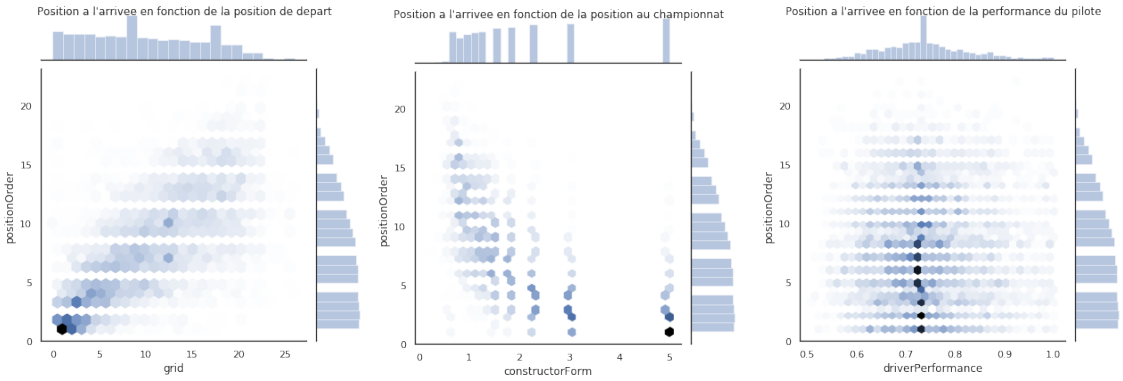


Figure 4: Répartition du coefficient de puissance

Une fois les *features* crées, nous pouvons visualiser leur relation avec le résultat que nous cherchons : la position à l'arrivée. Les trois graphiques suivant montrent les correspondances du résultat final avec, dans l'ordre, la position de qualifications, le coefficient de forme d'une écurie, et la performance du pilote.



Résultat de qualifications | Coefficient de forme | Coefficient de performance

Nous remarquons que le résultat de qualifications et le coefficient de forme (et, par extension, le coefficient de puissance) sont de bons indicateurs du résultat final. En revanche, la performance d'un pilote nous donne des résultats mitigés, en raison de sa distribution quasi-symétrique pour deux pilotes d'une même écurie.

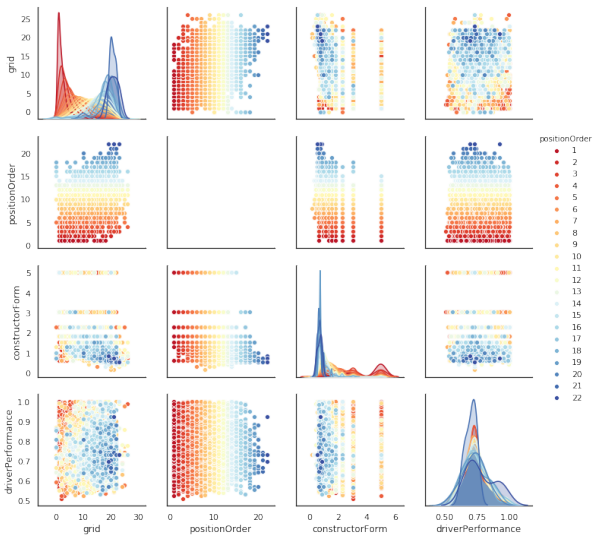


Figure 5: Relation des features avec le résultat

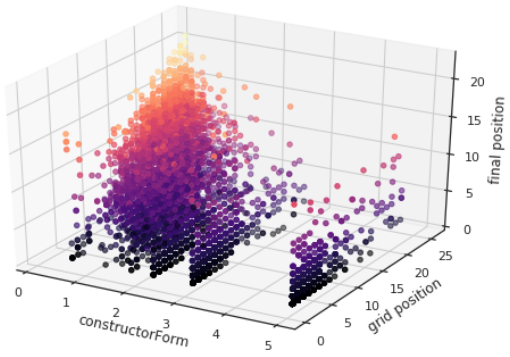


Figure 6: Visualisation 3D des données

Nous pouvons tracer un *pairplot* pour visualiser les relations de chaque feature avec la position à l'arrivée (Figure 5), ainsi qu'une visualisation 3D de la distribution des données (Figure 6).

2.2 Régression linéaire

2.2.1 Descente de gradient

Nous avons commencé par coder une méthode de descente de gradient classique afin de pourvoir prédire la position à l'arrivée. Après quelques essais, nous avons trouvé qu'initialiser une descente de gradient avec **200 itérations** et un taux d'apprentissage égal à **0.01** nous donnait des résultats concluants.

2.2.2 Optimisation directe

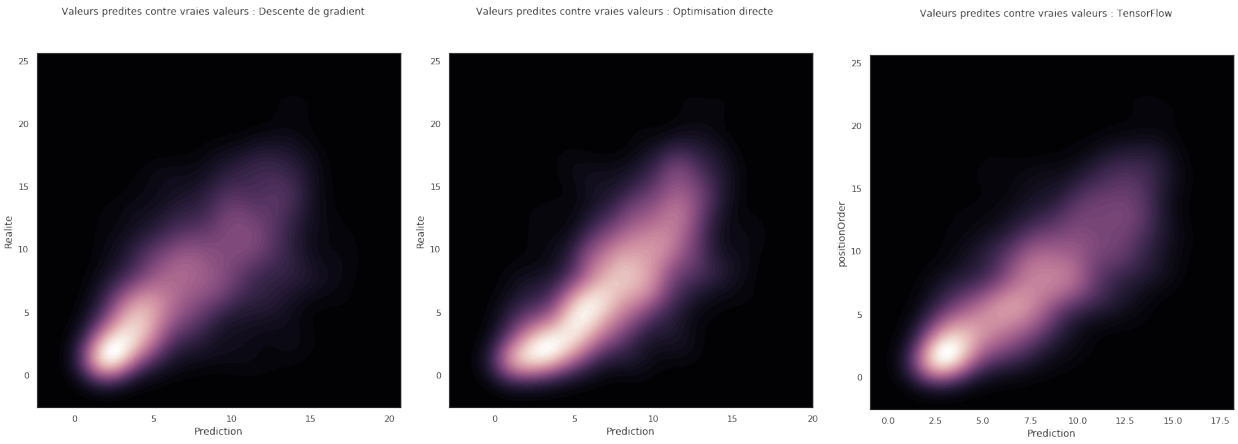
Afin de coder une seconde méthode pour prédire la position à l'arrivée, nous avons utilisé des résultats appris en cours, notamment en *Méthodes Numériques : Optimisation* et en *Fondements du Machine Learning*. Nous procédons à une optimisation directe avec la formule suivante : $X^T\beta = Y \iff \beta = ((X^TX)^{-1}X^T)^{-1}Y$. Ici, pas de méta-paramètres : la formule nous prodigue une optimisation directe sans avoir besoin de régler quoi que ce soit.

2.3 TensorFlow

Enfin, nous avons souhaité comparer nos résultats avec ceux d'une librairie puissante : TensorFlow. A travers Keras, nous avons créé un modèle à trois couches de 64 unités toutes trois munies de la fonction d'activation Relu, optimisé avec l'algorithme RMSprop. **100 epochs** nous ont permis d'obtenir des résultats satisfaisants avec ce modèle.

2.4 Comparaison des différentes méthodes

Nous pouvons juger des performances de chacun de ces algorithmes en tracant plusieurs graphiques. Tout d'abord, nous allons vérifier que les algorithmes arrivent réellement à produire des résultats.



Descente de gradient | Optimisation directe | TensorFlow

Nous pouvons ensuite visualiser les erreurs, en termes de positions, de chaque méthode, grâce à un graphique en boîtes (Figure 7), ainsi que nous rendre compte de la distribution des erreurs sur la figure 8.

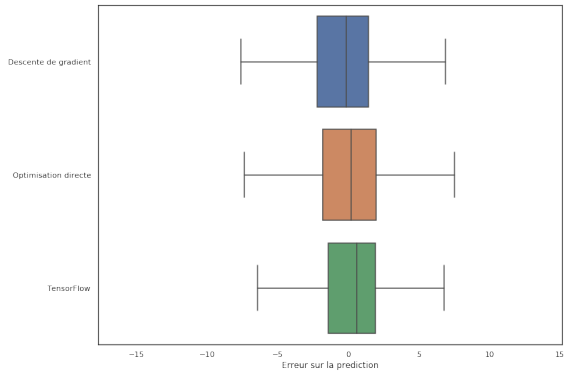


Figure 7: Erreur sur la position

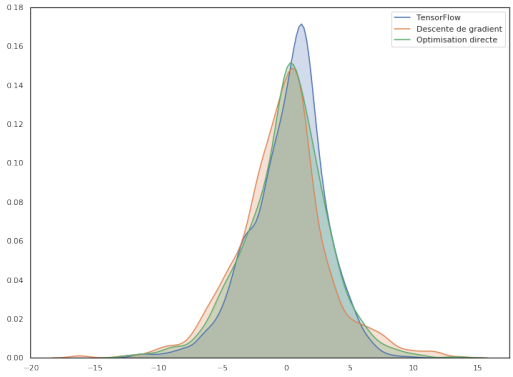


Figure 8: Distribution de l'erreur

Nous pouvons, pour conclure, tenter de prédire les résultats d'une seule course, afin de nous rendre compte des performances des trois algorithmes dans une situation réelle. Nous avons pour cela choisi le Grand Prix d'Abu Dhabi 2017.

Pilote	Resultat Final	Optimisation	Gradient	TensorFlow
Valtteri Bottas	1	1	1	5
Lewis Hamilton	2	2	2	2
Sebastian Vettel	3	4	3	1
Kimi Räikkönen	4	3	5	4
Max Verstappen	5	7	6	6
Nico Hülkenberg	6	9	7	7
Sergio Pérez	7	6	8	8
Esteban Ocon	8	8	9	10
Fernando Alonso	9	14	11	9
Felipe Massa	10	11	10	11
Romain Grosjean	11	16	16	15
Stoffel Vandoorne	12	15	13	12
Kevin Magnussen	13	12	14	14
Pascal Wehrlein	14	19	18	17
Brendon Hartley	15	18	20	20
Pierre Gasly	16	17	17	18
Marcus Ericsson	17	20	19	19
Lance Stroll	18	13	15	16
Carlos Sainz	19	10	12	13
Daniel Ricciardo	20	5	4	3

On remarque que les algorithmes performent très raisonnablement ! Daniel Ricciardo et Carlos Sainz sont mal catégorisés car ils ont tous deux abandonné la course, une éventualité que le modèle ne prédit pas.

3 Prédiction des arrêts au stand

Dans la Formule 1 moderne, les pilotes sont tous obligés d'utiliser au moins deux composés de pneus pendant une course : pour ce faire, ils effectuent des arrêts au stand, ou *pit-stop*. Ces manoeuvres, mettant en jeu plus de 14 mécaniciens, s'effectuent en environ deux secondes, suite à quoi le pilote reprend sa course. Nous avons remarqué que le dataset dont nous disposons ne nous fournit les informations de pit-stop (*à quel tour un pilote a-t-il réalisé un arrêt au stand*) que pour environ 200 des 1030 courses proposées : une volonté est donc d'être capables de remplir ce dataset. Pour chaque tour, nous noterons 0 si le pilote ne s'arrête pas et 1 si le pilote s'arrête.

3.1 Analyse des données

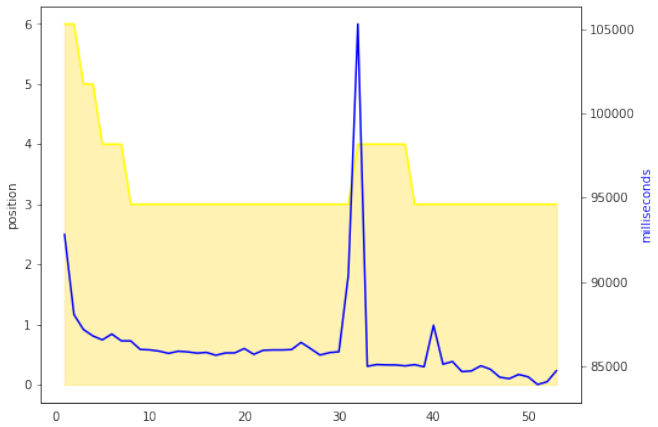


Figure 9: Résumé de course de Sebastian Vettel, Italie 2017

Quelles données permettent de détecter un arrêt au stand? Cette question n'est pas évidente, car aucune *feature* ne nous permet de détecter facilement la présence d'un pit-stop pour un tour donné. Nous avons par conséquent eu l'intuition suivante : un arrêt au stand prend, selon les circuits, de 20 à 35 secondes, en raison de la limitation de vitesse dans la voie des stands. Ainsi, la présence d'un pit-stop devrait être signifiée par une brusque augmentation du temps au tour. Notre intuition est confirmée par la Figure 9 : le pit-stop de Sebastian Vettel au tour 31 s'est effectivement manifesté par un pic sur les temps au tour.

Nous avons donc, pour mettre en évidence cette singularité, créé une feature qui n'existait pas dans le dataset, que nous avons appelé **lapDelta**. Nous avons tout d'abord créé deux *features* intermédiaires :

- **previousLapDelta** est le rapport du temps au tour n avec le temps au tour $n-1$
- **nextLapDelta** est le rapport du temps au tour n avec le temps au tour $n+1$

Enfin, **lapDelta** est la moyenne arithmétique de ces deux résultats à laquelle nous avons appliqué la fonction $f(x) = 10 * (1 - \exp^{(1-x)})$ afin d'amplifier les valeurs supérieures à 1 et de minimiser les valeurs proches de 1.

3.2 Prédiction

3.2.1 Régression Logistique

Nous pouvons remarquer que les labels ne sont pas du tout équitablement répartis dans le dataset : en effet, étant donné que les pilotes réalisent uniquement entre 1 et 3 arrêts au stand par course, nous

remarquons que les labels positifs représentent uniquement 4% du dataset. Durant nos premiers essais, nous avons remarqué qu'une régression logistique avait du mal à prédire un label positif dans ce contexte d'inégalité : c'est pourquoi nous avons choisi de restreindre le dataset en choisissant uniquement 10% de labels négatifs parmi l'intégralité des données : cette technique est appelée *undersampling*.

Nous avons créé un algorithme de régression logistique classique par descente de gradient : après plusieurs essais, nous avons conclu que des méta-paramètres adaptés sont un nombre d'itérations de **200** et un taux d'apprentissage de **0.05**. Nous pouvons voir, en Figure 10, la matrice de confusion résultant de l'apprentissage par régression logistique : le nombre de faux positifs est égal à 0, ce qui est encourageant. Cependant, le nombre de faux négatifs est légèrement grand. Nous avons également calculé plusieurs métriques permettant de juger l'efficacité de notre modèle comme l'accuracy, la précision, le rappel ou encore le score F1. Le tableau suivant nous permet de nous rendre compte de ces valeurs : nous remarquons que les métriques obtenues sont tout à fait acceptables. Toutefois, nous devons chercher à encore améliorer ces résultats.

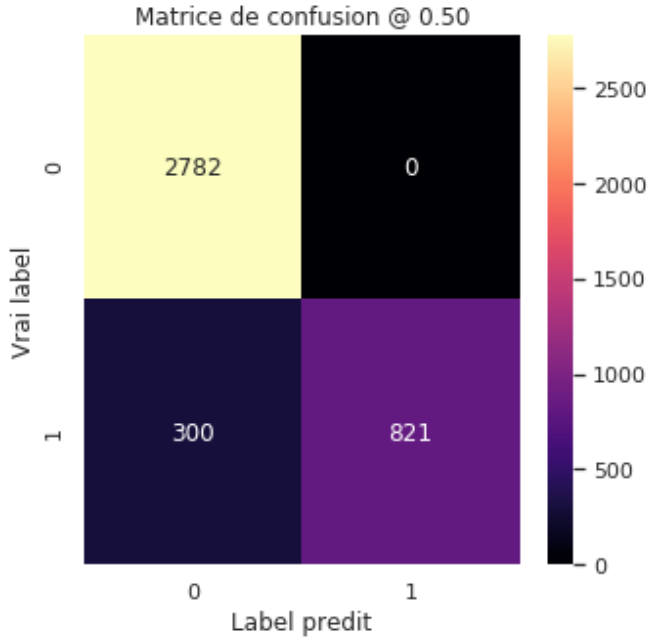
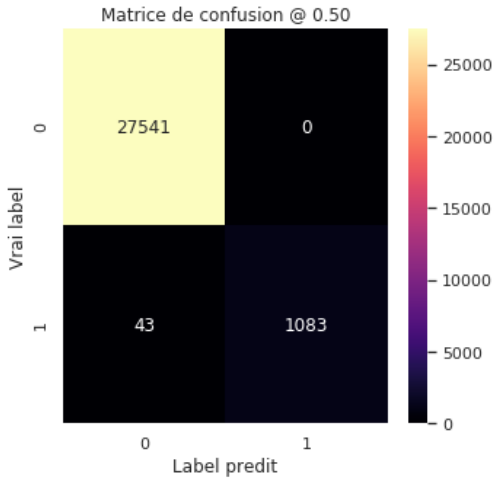


Figure 10: Matrice de confusion, régression logistique

Métrique	**Score**
Précision	1.0
Rappel	0.732
Accuracy	0.932
Score F1	0.845
Taux de faux positifs	0.0

3.2.2 TensorFlow

Encore une fois, nous allons comparer notre modèle créé *"à la main"* avec un modèle créé avec Keras. Ici, nous avons choisi une simple couche de 16 unités avec la fonction d'activation Relu, munie en plus une régularisation Dropout de paramètre 0.5, avant d'utiliser la fonction sigmoïde pour prédire une probabilité. Nous utiliserons ici l'algorithme d'optimisation Adam. Grâce à la puissance intrinsèque de ce modèle, nous pouvons également nous passer d'*undersampling* et donner directement à l'algorithme notre dataset.



Métrique	**Score**
Précision	1.0
Rappel	0.962
Accuracy	0.999
Score F1	0.981
Taux de faux positifs	0.0

Figure 11: Matrice de confusion, Keras

Les résultats donnés par Keras sont très clairement supérieurs à ceux de notre régression logistique : comme nous pouvons le voir sur la figure 11, l'écrasante majorité des arrêts au stand ont été détectés.

Encore une fois, afin de tester notre algorithme, nous avons essayé de prédire les arrêts au stand du Grand Prix d'Abu Dhabi 2017 : comme indiqué sur la figure 12, tous les pit-stops ont été correctement identifiés : c'est une réussite. Nous pouvons conclure que la détection d'un arrêt au stand est possible, grâce à la création d'une *feature* représentative.

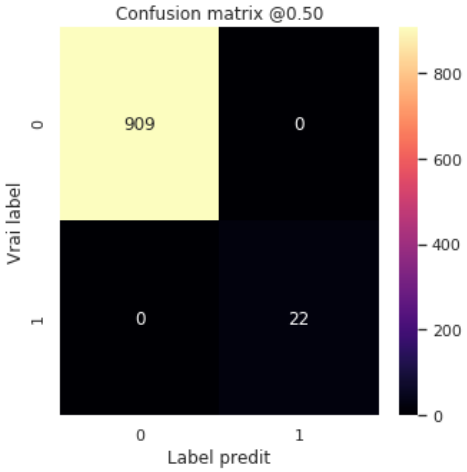


Figure 12: Matrice de confusion, Abu Dhabi 2017

4 Classification des composés de pneus

4.1 Présentation du problème

Le dernier problème auquel nous nous sommes intéressés est la classification des différents composés de pneus utilisés pendant les Grands Prix de Formule 1. En Formule 1, cinq types de pneus peuvent être utilisés pendant une course :

- Soft : ce pneu, extrêmement rapide, possède la durée de vie la plus courte.
- Medium : ce pneu est moyennement rapide et possède une durée de vie moyenne.
- Hard : ce pneu est le plus lent mais le plus durable.
- Inter : ce pneu est conçu pour être utilisé sur circuit humide.
- Wet : ce pneu est utilisé dans le cas d'un circuit trempé, c'est le plus lent de tous.

Dans le dataset, nous n'avons aucune information sur le composé de pneu utilisé par un pilote: chercher à les classifier de cette manière relèverait du domaine de l'*apprentissage non supervisé*. Il est raisonnable de penser que réussir à classifier les pneus dans ce scénario est irréaliste : nous avons donc utilisé un ensemble d'apprentissage très restreint, ne contenant que les données de pneus pour une saison, recueillies à la main.

L'objectif est de trouver, pendant une course et pour un pilote donné, quels composés de pneus il a utilisé. La différence entre deux composés de pneus peut être observée sur deux plans : la durée du relais (longévité du pneu) et le temps au tour (adhérence du pneu).
Nous convînmes de prendre en compte les considérations suivantes :

- Un pilote Williams ayant chaussé des softs peut avoir, au même moment de la course, un temps au tour supérieur à un pilote Ferrari équipé de hards en raison de la différence de puissance entre les deux monoplaces. Nous avons donc dû modéliser le problème en considérant cette disparité.
- Au fur et à mesure de l'avancement de la course, les voitures s'allègent en raison de la diminution de la masse d'essence dans le réservoir : c'est pourquoi le Meilleur tour est très souvent obtenu en fin de course : nous avons donc pris en compte le nombre de tours, sous forme de pourcentage, pour classifier les pneus.
- Naturellement, un temps au tour ne peut être représentatif que pour un circuit donné : c'est pour cela qu'il convient de stocker l'identifiant du circuit sous forme de one-hot.

En fait, l'histoire est plus complexe : Pirelli, le fabricant de pneus officiel de la Formule 1, propose une gamme de 5 composés de pneus pour la saison. Ils possèdent chacun un identifiant, qui s'étend de C1 (gomme la plus dure) à C5 (gomme la plus molle).

Chaque weekend, trois composés sont sélectionnés parmi les 5 disponibles et sont affublés des noms *soft, medium* et *hard*. Ainsi, si le choix de pneus est $(C1, C2, C3)$ pour un weekend w_1 et $(C3, C4, C5)$ pour un weekend w_2, le pneu *soft* de w_1 est exactement le même composé que le *hard* de w_2 !

C'est pourquoi nous avons dû associer, pour chaque course, aux pneus leur composé exact et procéder à une classification en cinq classes et non pas trois comme l'intuition nous le suggèrait.

4.2 Analyse des données

Afin de prédire le composé de pneu, nous avons créé trois *features* principales :

- **race_completion** indique le moment de la course auquel le relais a été effectué, sous forme de pourcentage.
- **stint_duration** représente la durée en tours d'un relais.
- **stint_avg_laptime** est la moyenne des temps au tour d'un relais divisée par le temps de qualifications du pilote en question : prendre ce rapport nous permet d'éliminer la composante inégalitaire des monoplaces puisqu'un pilote n'est comparé qu'à ses propres performances.

Nous avons obtenu, pour le dataset recueilli, les valeurs suivantes pour *stint_duration* et *stint_avg_laptime* :

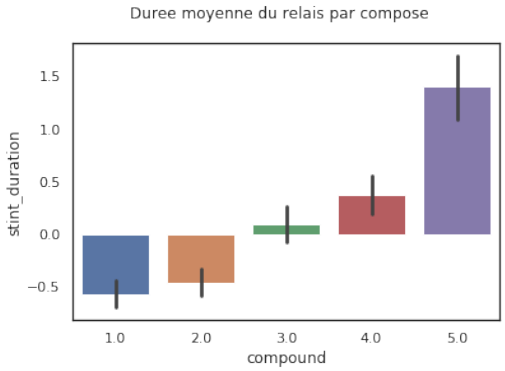


Figure 13: Durée moyenne du relais

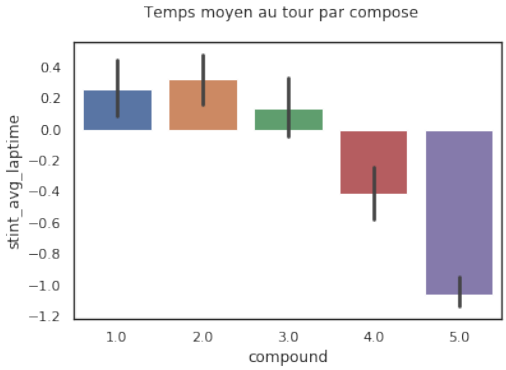


Figure 14: Temps au tour du relais (rapport)

4.3 Classification

Afin de classifier les composés de pneus, nous avons décidé d'utiliser une méthode de classification bien connue, que nous avons déjà eu l'occasion de coder dans le cadre de cet enseignement: la *Random Forest*. Nous avons essayé de nombreux méta-paramètres avant de sélectionner les suivants : notre Random Forest possède 111 arbres et met en jeu 3 *features* par coupe, avec un minimum de 2 éléments par coupe.

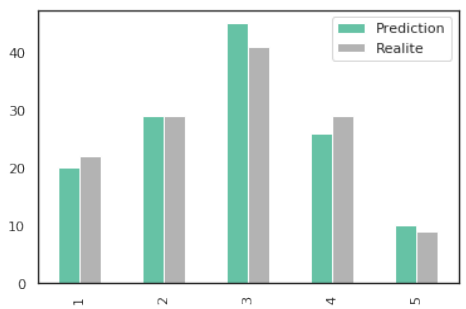


Figure 15: Prédiction et réalité : composé de pneus

Nous pouvons déduire de la figure 15 que la classification a fonctionné : la Random Forest a réussi à trouver le composé de pneus exact dans 70% des exemples de l'ensemble de test : comme nous pouvons le voir sur le graphique, les distributions prédites sont similaires à la réalité.

Encore une fois, afin de valider nos résultats, nous avons testé notre algorithme sur une unique course : le Grand Prix de Grande-Bretagne 2019. Comme nous pouvons le voir sur la figure 16, la classification se révèle difficile : en effet, les pneus se comportent différemment sur chaque circuit et nous n'avons pas eu l'occasion d'entraîner le modèle sur le circuit de Silverstone. Afin d'obtenir une prédiction plus précise, il aurait fallu un dataset plus complet afin d'obtenir les informations de composés sur de nombreuses courses : il est en effet très laborieux de récolter ces données qui ne sont pas disponibles sur Internet mais seulement sur les rediffusions officielles appartenant à *Formula One Group*, auxquelles nous avons pu accéder grâce à une souscription payante.

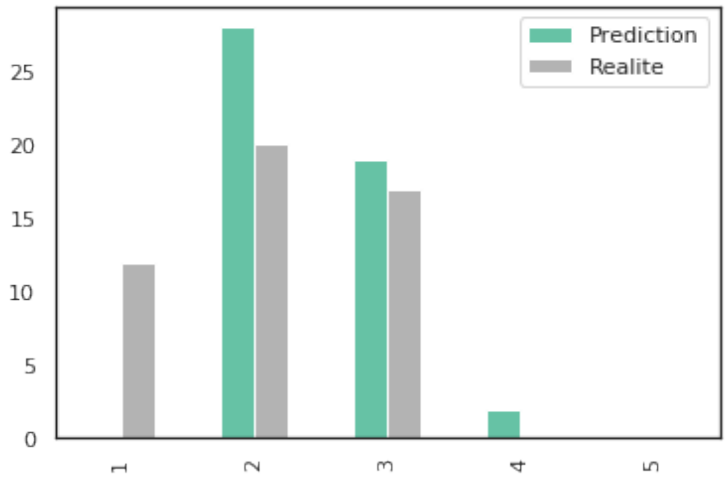


Figure 16: Prédiction et réalité : Silverstone 2019

5 Conclusion

Dans ce projet, nous avons attaqué trois problèmes différents dans l'univers de la Formule 1. Le dataset est somme toute assez complet : il ne fait nul doute que de nombreux autres problèmes peuvent être abordés.

Nous avons codé quatre méthodes d'analyse de données : une descente de gradient, une optimisation directe, une régression logistique et une Random Forest. Nous avons également eu l'occasion de manipuler Keras et d'attester de la puissance de cette librairie.