HEIG VD
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Machine Learning pour prendre la décision d'un arrêt au stand en Formule 1

Rapport Intermédiaire Travail de Bachelor

Département TIC

Filière Informatique et systèmes de communication

Orientation Informatique logicielle

Nelson Jeanrenaud

26 juillet 2023

Supervisé par :
Pena Carlos Andrés (HEIG-VD)

Hes·so
Haute Ecole Spécialisée de Suisse occidentale

Préambule

Ce travail de Bachelor (ci-après **TB**) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'École.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD
Le Chef du Département

Yverdon-les-Bains, le 26 juillet 2023

Authentification

Je soussigné, Nelson Jeanrenaud, atteste par la présente avoir réalisé seul ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Nelson Jeanrenaud

Yverdon-les-Bains, le 26 juillet 2023

Résumé

L'objectif d'une course automobile est de parcourir un nombre de tours sur un circuit le plus rapidement possible. De plus, dans de nombreuses catégories, les voitures peuvent s'arrêter au stand pour changer de pneus et/ ou se réapprovisionner en carburant. La dégradation des pneus et le poids supplémentaire dû à la charge de carburant ont un impact majeur sur les temps au tour d'une voiture. Décider quand et combien de fois rentrer dans les stands est donc crucial pour les écuries. À cela vienne s'ajouter les notions de neutralisation de la course pour cause d'incident.

Établir la stratégie optimale est un enjeu majeur pour les équipes de course. Il est donc nécessaire de développer un système d'analyse de données et de prédiction pour aider les écuries à prendre les meilleures décisions stratégiques pendant la course. Ce système devra prendre en compte les données de la voiture, les performances des pneus et les conditions de la piste pour fournir des recommandations en temps réel aux équipes de course.

Dans ce travail, nous tentons de créer un tel système pour le championnat du monde de Formule 1. Pour ce faire, nous entraînons des modèles de Machine Learning sur des données historiques de la dernière ère de la formule 1.

Un set de données a été établi à partir des données transmises par la Formule 1 pendant les courses. Ces données ont été préparées de manière à former un ensemble complet et cohérent, permettant d'entraîner des modèles

Après l'expérimentation avec des réseaux de neurones et des réseaux de neurones récurrents, nous avons réussi à développer des modèles capables de capturer les relations entre la stratégie de course et les données de la voiture.

Pour améliorer le système, nous pourrions inclure les facteurs météorologiques, développer une simulation de course pour améliorer la précision des prédictions et tester l'apprentissage par renforcement pour développer des stratégies novatrices.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Objectifs

Les objectifs de ce projet sont de développer un système de prédiction de stratégie de course en temps réel pour les équipes de course, basé sur l'analyse des données de la voiture et des conditions de la piste, et de fournir des recommandations pour optimiser les temps de tour et les arrêts aux stands.

1.1.1 Objectifs fondamentaux

- Implémenter un pipeline de traitement et danalyse de données avec des algorithmes de Machine Learning pour recommander lentrée au stand dune voiture en fonction de la situation de course.
- Établir un dataset adéquat à lentrainement dun tel modèle à partir de multiples sources de données brutes.
- Fournir des prédictions réalistes et interprétables par les experts métiers.
- Le système est réactif et permet des prédictions rapides (moins de 30 secondes).

1.1.2 Objectifs optionnels

- Développer une interface utilisateur pour facilement accéder aux recommandations stratégiques.
- Étendre le modèle pour les conditions météorologiques incertaines comme les chutes de pluie.

1.2 Principe de la stratégie en Formule 1

L'objectif fondamental de la stratégie est de minimiser le temps que l'on prend pour finir tous les tours de course. En formule 1, les écuries doivent choisir entre 3 composés de pneumatique à utiliser. Ces composés nommés tendres, mediums et durs performent différemment le long de la course, les plus tendres offrant une meilleure adhérence, mais s'usant plus rapidement.

Pour illustrer cette section on a simulé en python une course très simplifiée. La simulation considère que la voiture est seule sur la piste, qu'il n'y a pas de variation dans la performance du pilote ou dans l'état du circuit. La course est discrétisée en N tours où chaque temps au tour t_{tour} est défini comme montré dans l'équation 1.1 :

$$t_{tour} = t_{optimal} + t_{carburant} + t_{pneu} \tag{1.1}$$

$t_{optimal}$ est le temps optimal que la voiture peut faire sur le circuit dans les meilleures conditions. $t_{carburant}$ est le temps perdu en raison du poids du carburant embarqué. Au fil de la course ce temps s'amoindrit lorsque que le carburant est consommé. $t_{carburant}$ est calculé comme l'équation 1.2. Où $t_{penalite}$ est le temps au tour perdu par kg. Cette valeur dépend du circuit, mais est approximable à 0.03 seconde pour les circuits de longueur moyenne. [19] [jjn18] p_{depart} est la quantité en kg de carburant au départ de la course, pour cette simulation, nous faisons l'hypothèse que la voiture commence avec le maximum de carburant autorisé, c'est-à-dire 110 kg. $p_{consomme}$ est la quantité de carburant en kg qui est consommé en 1 tour de course, elle est calculée en estimant que la consommation de carburant est constante, soit $\frac{p_{depart}}{N}$. n est le nombre de tours parcourus jusqu'à maintenant.

$$t_{carburant} = t_{penalite} * (p_{depart} - p_{consomme} * n) \tag{1.2}$$

t_{pneu} est le temps perdu en raison de l'adhérence des pneumatiques, elle dépend du composé et de la dégradation du pneu. Le détail est donné dans l'équation 1.3, elle approxime la dégradation des pneus par la formule de calcul des intérêts composés pour représenter la chute de performance que l'on peut observer avec les pneumatiques [ana21]. $t_{performance}$ est le temps perdu comparé à la performance du pneu le plus tendre. Le taux de dégradation du pneu par tour, exprimé en pourcentage par d et le δd l'augmentation de ce taux par tour.

$$t_{pneu} = t_{performance} + d * (1 + \delta d)^{(n-1)} \tag{1.3}$$

La figure 1.1 illustre l'influence de la dégradation sur la performance des différents composés de pneu utilisant la simulation avec des paramètres trouvable dans la table A.1 en annexe. Les pneus tendres sont plus performant initialement avant de rapidement perdre en performance, le même phénomène se produit avec les mediums et les durs, mais après un plus grand laps de temps. Cette différence en performance entre les composés dépend du circuit et est difficile à estimer précisément en amont de la course.

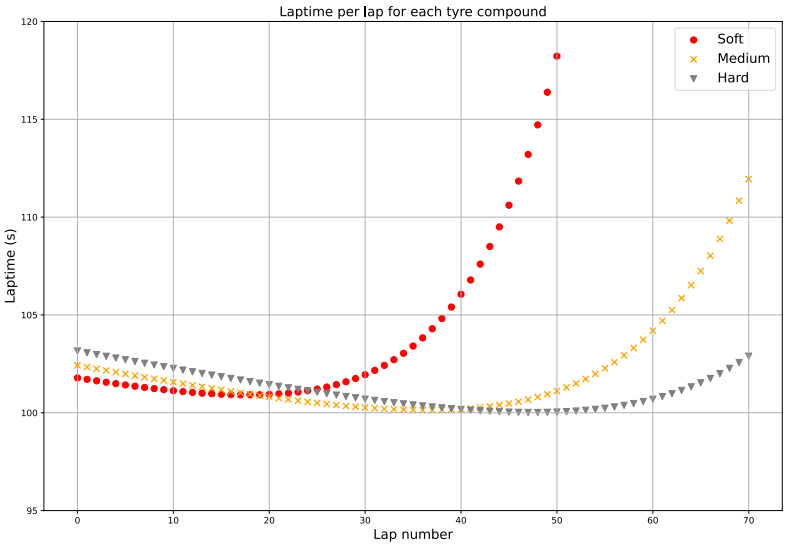

Figure 1.1 – *Comparaison de la dégradation des différents composés de pneu*

Les temps au tour sont successivement additionnés pour obtenir la variable t_{course}, qui représente la durée totale d'une stratégie simulée comme présenté dans l'équation 1.4.

$$t_{course} = \sum_{i=1}^{N} t_{tour}(i) \tag{1.4}$$

On peut observer avec la figure 1.2 que sur une course de 70 tours utiliser les durs est la meilleure stratégie. Cependant, le règlement de la Formule 1 oblige d'utiliser au minimum 2 composés différents par courses. Il faut maintenant trouver le tour optimal où changer de pneumatique pour minimiser le temps de course total. Une fois tous les paramètres établis, cela se ramène à un problème d'optimisation quadratique.

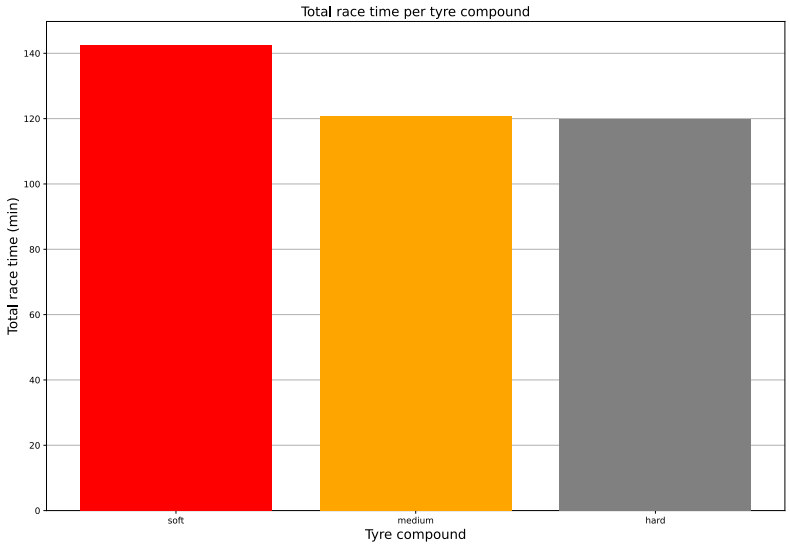

Figure 1.2 – *Comparaison du temps de course des différents composés de pneu*

La figure 1.3, montre l'influence du tour d'arrêt sur le temps total de course. On peut voir que pour notre course, la stratégie de commencer avec des tendres et les remplacer par des mediums après 20 à 30 tours est la plus rapide.

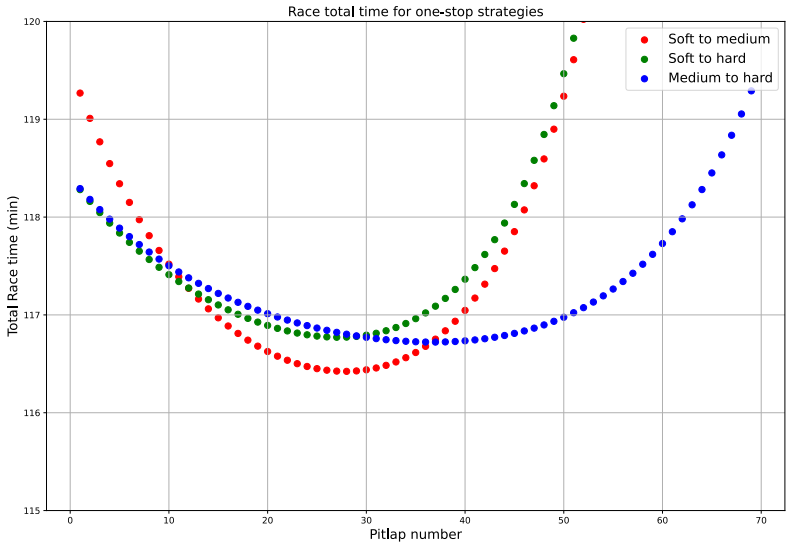

Figure 1.3 – *Comparaison du temps de course de différentes stratégies à 1 arrêt*

Certains événements aléatoires comme les périodes de voiture de sécurité, impact grandement la stratégie. Une voiture de sécurité est déployée suite à un incident ou à la présence de débris sur la piste. Dans cette situation, les voitures doivent rouler à vitesse réduite, ce qui réduit fortement le temps perdu pendant un arrêt au stand. Cela est dû au fait que les voitures doivent toujours rouler à vitesse réduite dans la

voie des stands même quand il n'y a pas de voiture de sécurité, la figure 1.4 illustre ce phénomène. Comparé à la course en figure 1.3 où la voiture de sécurité n'est pas déployée, il est plus intéressant de s'arrêter pendant la période de safety car même si les pneumatiques seront moins performant en fin de course.

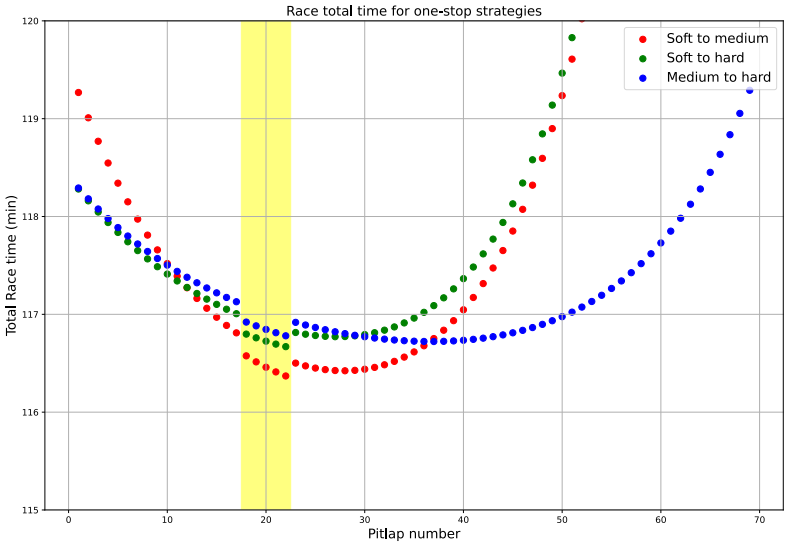

Figure 1.4 – *Comparaison du temps de course de différentes stratégies à 1 arrêt avec une période de voiture de sécurité, en considérant que s'arrêter pendant cette période économise 10 secondes*

D'autres éléments pas pris en compte dans cette simulation impactent la stratégie, notamment les autres voitures sur la piste. Il est souvent intéressant de distancer suffisamment les voitures plus lentes derrière ou attendre qu'elles s'arrêtent avant de s'arrêter soit même pour éviter de perdre du temps à devoir les dépasser sur le circuit. Une autre tactique est celle de l'undercut, utilisée quand un pilote à de la difficulté à dépasser une autre voiture. La technique consiste à s'arrêter plus tôt que prévu pour réduire l'écart avec la voiture poursuivie avec des pneus plus frais. Si elle est bien exécutée l'adversaire se verra dépassé pendant qu'il effectue est dans la voie des stands. C'est dans ce contexte qu'il serait intéressant de développer des méthodes de Machine Learning pour évaluer les décisions stratégiques.

1.3 État de l'art

Les modèles actuellement utilisés par les équipes de course pour établir leur stratégie de course sont évidemment confidentiels. Néanmoins, il existe des articles publiés par des universités notamment Application of Monte Carlo Methods to Consider Probabilistic Effects in a Race Simulation for Circuit Motorsport. [Hei+20a] Cette étude présente une approche de simulation de course qui considère des éléments aléatoires tels que les accidents, les problèmes mécaniques et les variations de performance des pilotes et des mécaniciens.

Un autre article du même institut, Virtual Strategy Engineer : Using Artificial Neural Networks for Making Race Strategy Decisions in Circuit Motorsport. [Hei+20b]

se base sur ce simulateur pour entrainer des réseaux de neurones à la prise de décision stratégiques. On peut noter des décisions intéressantes comme la division de la décision entre 2 modèles, le premier prends la décision de sarrêter et lautre choisis le composé de pneu à utiliser. Ils ont développé des features intéressantes pour l'entrée de leur modèle notamment la catégorisation des circuits en 3 groupes selon le niveau de stress appliqué sur les pneumatiques, la discrétisation de la position en "leader de la course ou non" et de la distance avec la voiture précédente en "Close ahead" qui est vrai si l'écart est inférieur à 1.5 seconde. D'après leur étude, résoudre un problème d'optimisation de minimisation du temps de course dans un simulateur en amont permet de créer une feature représentant le nombre d'arrêts au stand restant pour atteindre la stratégie optimale (1, 2 ou 3 arrêts). Leurs résultats montre qu'une architecture feedforward considérant que le dernier tour de course n'arrive pas à modifier la probabilité d'un arrêt assez rapidement pour obtenir des résultats précis. Ils ont expérimenté avec une architecture récurrente, considérant les 4 derniers tours obtenais de meilleurs résultats, mais faisait des erreurs difficilement compréhensibles. L'architecture retenue est celle d'un réseau hybride avec un seul neurone long short-term memory. Leurs résultats sont présentés en table 1.1 et figure 1.5.

Table 1.1 – *Résultats de l'étude "Virtual Strategy Engineer : Using Artificial Neural Networks for Making Race Strategy Decisions in Circuit Motorsport"*

Architecture	F_1 score
Feedforward	0.35
Recurrent	0.9
Hybride	0.59

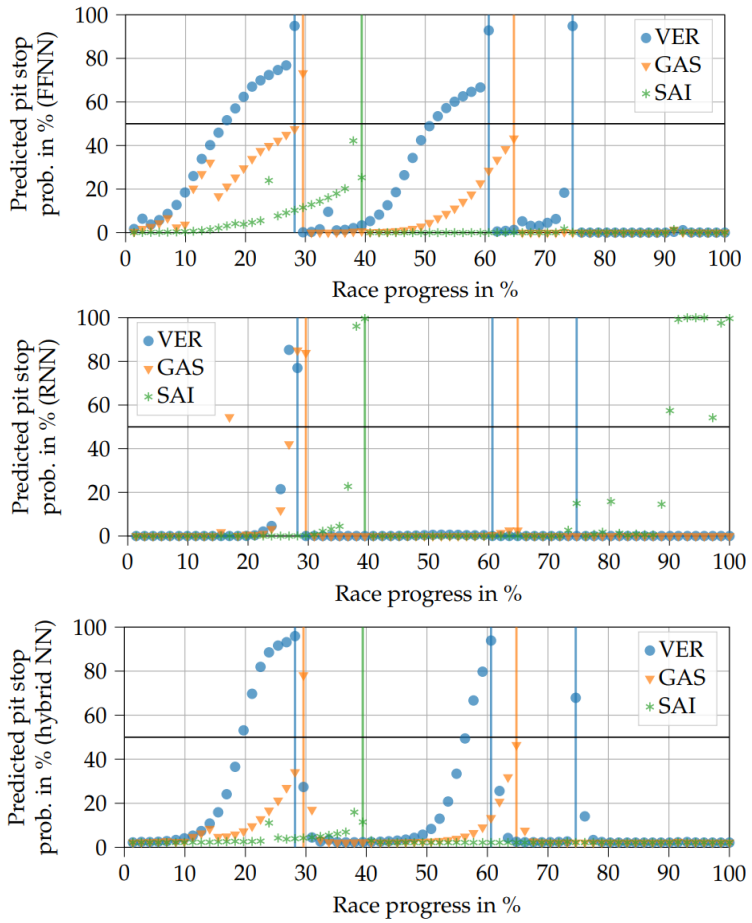

Figure 1.5 – *Progression des probabilités d'arrêt au stand pour trois pilotes prédites par le réseau feed-forward, le réseau récurrent et le réseau hybride en utilisant des données du Grand Prix du Brésil 2019, figure repris du papier "Virtual Strategy Engineer : Using Artificial Neural Networks for Making Race Strategy Decisions in Circuit Motorsport"*

Les 2 travaux "Tire Changes, Fresh Air, and Yellow Flags : Challenges in Predictive Analytics for Professional Racing" [TR14] et "Real-time decision making in motorsports : analytics for improving professional car race strategy"[Cho15] sont des équivalences à notre problématique appliquée au monde de la NASCAR. La NASCAR est une course compétition américaine sur circuits ovales avec des voitures de série modifiées, dans ce contexte, met davantage l'accent sur les courses en peloton, les dépassements sont plus fréquents et le ravitaillement en carburant est un facteur stratégique majeur. Ils apportent cependant une formulation différente intéressante, la variable prédite par ses systèmes est le nombre de positions gagnées ou perdues à la suite d'une décision stratégique.

Finalement, on peut noter l'article Formula-E race strategy development using distributed policy gradient reinforcement learning de l'université de Cranfield [LFA21] qui utilise une approche de reinforcement learning pour établir une stratégie. Cet article présente une méthode de développement de stratégies de course dans le contexte du championnat de Formule E, où les voitures ne changent pas de pneus et où la stratégie est davantage liée à l'économie de la batterie. Cependant, bien que le contexte soit différent de celui proposé dans ce projet et que l'approche reinforcement learning nécessite une simulation précise, la méthode utilisée reste intéressante à noter.

1.4 Méthodologie

La méthodologie adoptée pour cette thèse comprendra les étapes suivantes. Tout d'abord, une analyse approfondie des données disponibles sera réalisée afin d'acquérir une connaissance approfondie de la problématique étudiée. Ensuite, plusieurs formulations du problème, modèles et méthodes seront explorés à travers des expérimentations. Les performances respectives de ces approches seront mesurées à l'aide de différentes métriques pour confirmer leur adéquation avec l'objectif fixé.

Chapitre 2

Données

Cette section est consacrée à l'étude des données disponibles sur les courses de Formule 1 et à la création d'un dataset adapté à l'entrainement d'un modèle de Machine Learning.

2.1 Données existantes

Il existe 2 principales sources de données disponibles sur les courses de Formule 1. La première est le dataset Ergast, qui est un projet open source qui répertorie des données sur les résultats des Grand Prix. La seconde est la Formule 1 elle-même qui diffuse publiquement une certaine partie des données, dites "Live Timing", qui correspondent aux données des capteurs des voitures. Les données ont donc plusieurs temporalités :

— les informations sur un weekend de courses (circuit, saison, etc.)
— les données pour un tour de course (temps au tour, gomme de pneu, etc.)
— les données des capteurs (vitesse, distance avec les autres voitures, etc.)

FastF1 est une librairie Python qui permet facilement d'accéder aux données de l'API Ergast et aux données officielle de la Formule 1 depuis 2018.

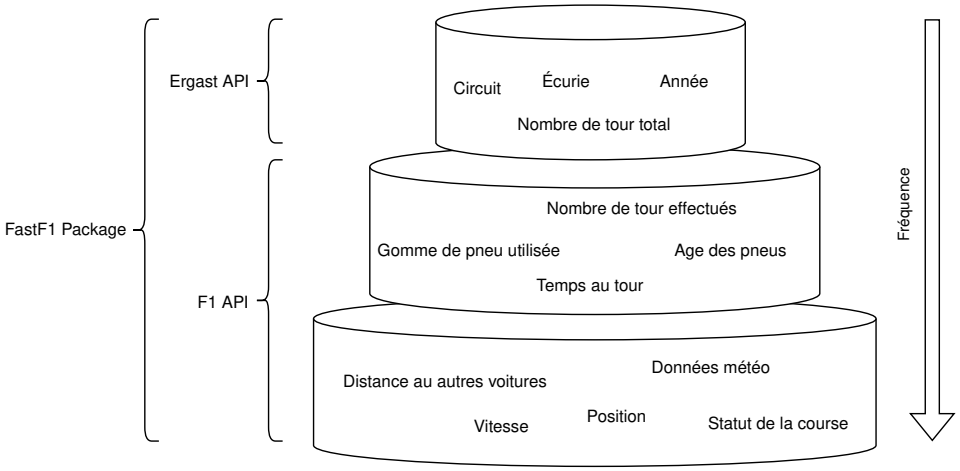

Figure 2.1 – *Diagramme en couches des données*

La quantité entre les catégories de données que nous avons surnommés respectivement données de courses, données de tour et données de télémétrie sont très différentes.

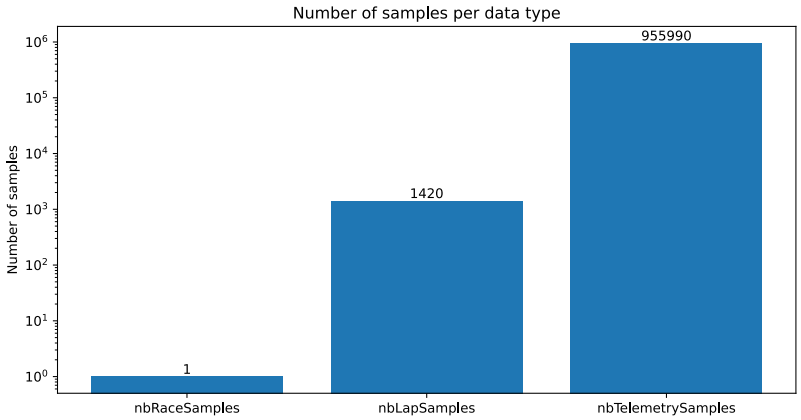

Figure 2.2 – *Nombre de données par courses, pour chaque catégorie de données*

2.1.1 Données de courses

Ces données donnent des informations sur le weekend de course et ne changent pas au cours de ce dernier. La majorité des features de cette catégorie sont connues pendant la course et donc au moment de l'inférence.

Table 2.1 – *Liste d'exemple de features 'données de courses'*

Nom de la feature	Exemple de valeur
season	2021
circuit	Monaco
number of laps	78

Les autres features sont liées au résultat de la course et ne serais pas disponibles pendant la prédiction. Mais elles pourraient être utilisées pendant l'entrainement pour par exemple pondérer plus fortement les stratégies de voitures qui ont de meilleurs résultats.

2.1.2 Données de tour

L'API de la Formule 1 permet d'écouter un flux de données dit "Live Timing". Ce flux contient différents canaux :

Table 2.2 – *Liste des différents canaux fournis dans le flux de données "Live Timing" [Sch]*

Canal	Description
LapNumber	Numéro du tour en cours
Driver	Numéro identifiant du pilote
LapTime	Temps du tour précédent
Stint	Nombre de relais effectués
TotalLaps	Nombre total de tours effectués avec ce set de pneus
Compound	Type de pneus utilisé
New	Indique si le pneu a été neuf au moment de son montage
TyresNotChanged	Signale les arrêts aux stands sans changement de pneus
Time	Temps écoulé depuis le début de la session
LapFlags	Drapeaux survenus pendant le tour

À chaque instant, seulement certains canaux sont actifs. La librairie FastF1 effectue le travail de les agréger sur le parcours d'un tour. Et permet de récupérer ses informations pour tous les tours effectués. Il serait donc possible de s'abonner à ce flux de donner et d'envoyer directement les informations émises en entrée du système de prédiction. Un exemple du flux est présenté dans la Table A.2 en annexe.

2.1.3 Données de télémétrie

L'API de la Formule 1 émet également un flux de données pour les données des capteurs des voitures et des capteurs météorologiques. Les données des capteurs sont envoyés a travers 2 différents flux "Car data" et "Position data" avec des mesures qui sont envoyées environ toutes les 240 millisecondes.

Table 2.3 – *Liste des différents canaux fournis dans le flux de données "Car data" [Sch]*

Canal	Description
Time	Temps écoulé depuis le début de la session
Date	Date exacte de la prise de l'échantillon
Speed	Vitesse de la voiture en Km/h
RPM	Régime moteur
Gear	Rapport de vitesse
Throttle	Pression sur l'accélérateur, exprimé en pourcentage de 0 à 100
Brake	Indicateur de freinage, Vrai si les freins sont enclenchés, faux sinon
DRS	Système de réduction de traînée (Drag Reduction System) Il comporte plusieurs valeurs possibles codifiées : 0-1 = désactivé, 8 = éligible, 10-14 = activé

Table 2.4 – *Liste des différents canaux fournis dans le flux de données "Position data" [Sch]*

Canal	Description
Time	Temps écoulé depuis le début de la session
Date	Date exacte de la prise de l'échantillon
Status	'OnTrack' ou 'OffTrack' (en dehors des limites du circuit)
X, Y, Z	Coordonnées de position relative en mètre

Pour chaque tour de course d'une voiture, il y a alors entre 300 et 1'000 échantillons de données télémétriques en fonction de la longueur du circuit. Ces données offrent un aperçu de la performance de la voiture et permettrait d'estimer la performance du pneumatique. Un exemple de données télémétriques échantillonnées sur un tour de course est donné en figure 2.3.

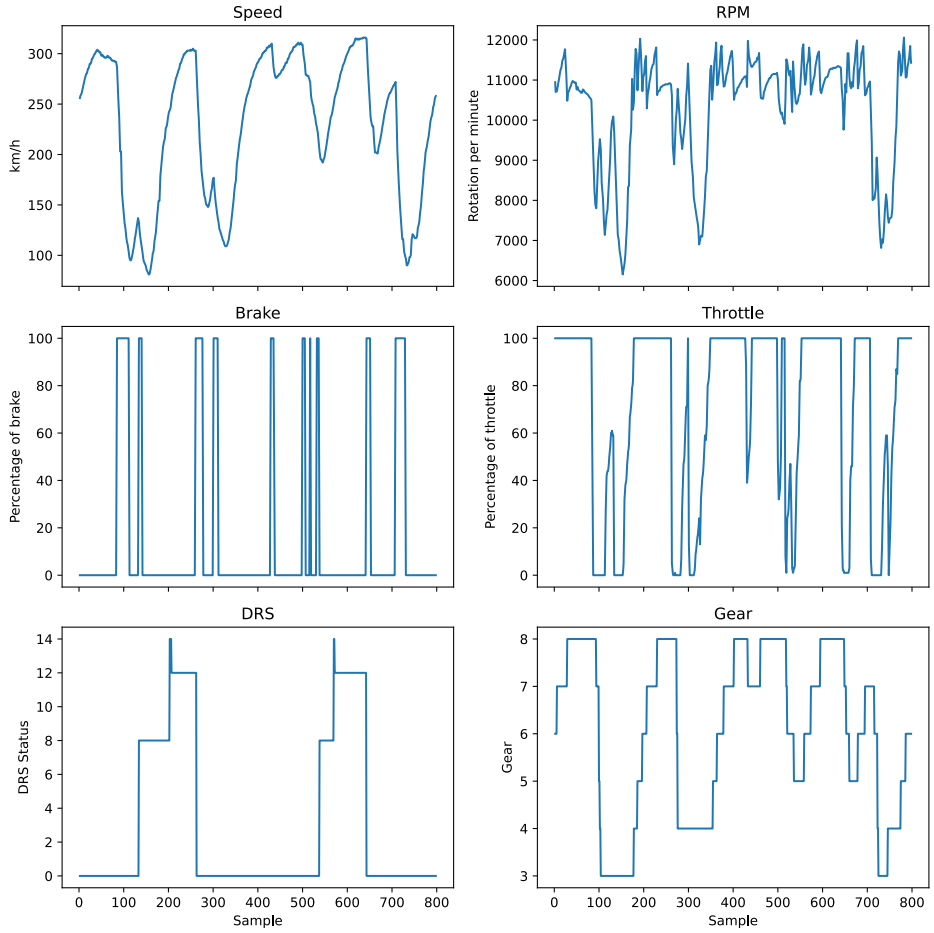

Figure 2.3 – *Exemple de données de télémétrie d'un tour de Carlos Sainz au Grand Prix de Grande-Bretagne 2021*

2.2 Création d'un dataset

Les modélisations explorées dans ce travail prennent en entrée les informations d'un ou plusieurs tours de course. Dans ce contexte, nous avons décidé de discrétiser chaque course en tour pour chaque voiture. À titre d'exemple, une course de 50 tours engendre alors 50 x 20 entrées dans la base de données si les 20 voitures terminent la course.

Le sous-ensemble des champs considérés est présentée en table 2.5. Ces variables retenues résultent d'une phase de sélection dans laquelle les variables non pertinentes, redondantes ou non prédictive ont été éliminées.

Table 2.5 – *Champs retenus dans le dataset*

Champ	Description
LapNumber	Numéro du tour en cours
LapTime	Temps du tour en secondes
DriverNumber	Numéro unique du pilote
Team	Nom de l'écurie
Compound	Composé de pneu utilisé (soft, medium ou hard)
TyreLife	Âge des pneus en nombre de tours
TotalLaps	Nombre de tours totaux de la course
Track	Nom du circuit
TrackStatus	Drapeaux de course apparaissant pendant le tour
Stint	Nombre de relais effectués
DistanceToDriverAhead	Distance en mètre avec la voiture devant
DriverAhead	Numéro unique du pilote devant
Position	Position dans la course
GapToLeader	Distance en secondes avec le pilote en tête
IntervalToPositionAhead	Distance en secondes avec le pilote devant
LapsToLeader	Nombre de tours de retard avec le pilote en tête
PitStatus	In Lap / Out Lap / No Pit

Pour simplifier les opérations de regroupement des tours, 2 colonnes ont étées ajoutées "RoundNumber" et "Year", respectivement le numéro de la course dans la saison et l'année de la course.

Pour les données télémétriques qui sont échantillonnées plus d'une fois par tour, on prend la dernière mesure au moment où la voiture franchit la ligne d'arrivée. Les tours avec des informations manquantes ont été exclus. Les saisons 2019 à 2022 compris ont étés considérés, l'année 2023 étant en cours au moment de la rédaction de ce projet est réservée comme données de test. La base de données contient 87'845 tours de 80 courses et est disponible ici.

2.3 Traitement des données

Les données doivent être préparées pour être utilisées dans le modèle. Les données inutilisable doivent être éliminées et certaines colonnes doivent être transformées.

L'objectif étant de prédire si un pilote va s'arrêter au stand ou non, la colonne PitStatus est utilisée pour créer la variable binaire cible is_pitting qui vaut 1 si le pilote s'arrête au stand et 0 sinon. "PitStatus" est une colonne catégorielle qui prend 3 valeurs possibles : "In Lap", "Out Lap" et "No Pit". "In Lap" signifie que le pilote est entré dans la voie des stands, "Out Lap" signifie qu'il en est sorti et "No Pit" signifie qu'il n'est pas entré dans la voie des stands. Pour créer la variable cible, les valeurs "In Lap" sont considérées comme 1 et les autres comme 0. Les données étant échantillonnées à chaque passage de la ligne d'arrivée, les informations relatives à un tour "In Lap" sont impactées par le temps passé dans la voie des stands. En d'autres termes, les tours "In Lap" et "Out Lap" sont plus longs que les autres tours. Il est impératif donc de décaler la valeur de la variable cible d'un tour vers le haut. Ainsi, la valeur de "PitStatus" au tour n correspond au tour d'entrée au stand du tour $n+1$. De cette manière quand le modèle effectue une prédiction avec les données du tour n, il prédit si le pilote entrera au stand au tour $n+1$. Pour ce faire, les tours ont

étés regroupés par course en utilisant les champs "DriverNumber", "RoundNumber" et "Year". Dans chacun de ces groupes, les tours sont ordonnés par "LapNumber" et les valeurs de "PitStatus" sont décalées d'un tour vers le haut. Pour le dernier tour de course on considère que la voiture n'entrera pas au stand. Les tours "In Lap" et "Out Lap" sont ensuite exclus du dataset car ils ne sont pas représentatifs.

On peut observer en figure 2.4 l'impact de la transformation de la variable cible sur le temps des tours.

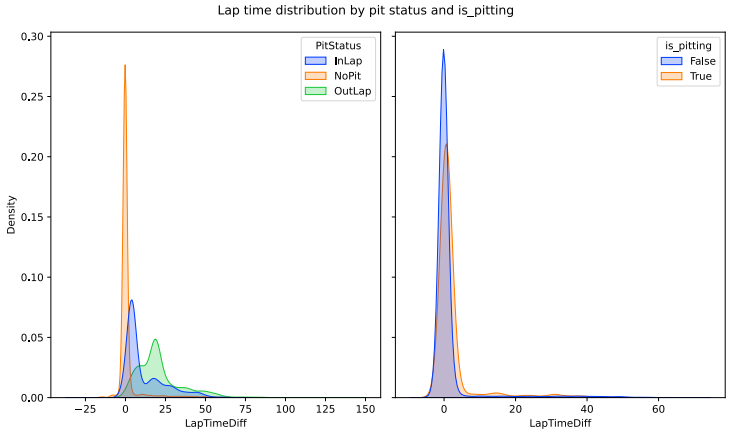

Figure 2.4 – *Distribution de la différence de temps entre les tours et la médiane des tours de la course en fonction de la variable cible.*

Le deuxième problème est le choix des pneumatiques à utiliser. Les données contiennent le composé de pneu utilisé pour chaque tour via la variable catégorielle "Compound". Elle est décalée de 2 tours vers le haut et copiée dans la colonne "NextCompound". Ainsi, la valeur de "NextCompound" au tour n correspond au composé de pneu utilisé au tour $n+2$. La raison de ce décalage est que l'on veut décider du composé de pneu à utiliser au moment ou l'on prend la décision d'entrer au stand. Étant donnée que les informations du tour sont capturées quand la voiture franchit la ligne d'arrivée, le nouveau composé de pneu est connu au moment où la voiture franchit la ligne d'arrivée lors de son tour de sortie des stands.

La gestion de conditions météorologiques changeantent est un problème complexe qui n'est pas abordé dans ce travail. Les courses sous la pluie sont donc exclues du dataset. Pour ce faire, les tours ont étés regroupés par course en utilisant les champs "DriverNumber", "RoundNumber" et "Year". Un groupe représente alors l'intégralité des tours d'une course pour un pilote donné. Les courses qui contiennent des tours avec des pneumatiques pluie sont exclues.

Dans certains cas, les pilotes ne terminent pas la course pour des raisons diverses (accident, problème mécanique, etc.). Ces tours doivent également être exclus du dataset. Pour ce faire, les tours ont encore étés regroupés par course de la même manière. Un groupe est alors considéré comme valide si le nombre de tours dans le groupe est supérieur au nombre de tours total de la course moins 3. Cette marge de 3 tours est ajoutée pour considérer les voitures qui abandonnent dans les derniers tours de la course et les voitures ayant des tours de retards car leur stratégie est quand même pertinente.

Dans d'autres cas, différentes situations peuvent amener des voitures à effectuer des arrêts au stands supplémentaires (pénalité, dégâts, etc.). Ces arrêts supplémentaires

peuvent fausser la prédiction de la stratégie. Nous avons donc décidé de ne pas les considérer. Les tours ont étés regroupés par course comme précédemment et les courses avec plus de 3 arrêts ont étés exclues.

Pour éviter de biaiser le modèle avec des informations comme l'identité du pilote ou de l'écurie, les tours ont étés anonymisés en supprimant les colonnes "DriverNumber" et "Team". En effet, les performances des pilotes et des écuries varient d'une saison à l'autre et il ne faut pas que le modèle apprenne à prédire la stratégie en fonction de ces informations.

La feature TrackStatus communiquée dans l'api officielle contient un ensemble d'entier concaténé dans une chaine de caractères où chaque entier représente un drapeau de course agité au cours du tour. Les différents status et leur entier correspondant sont détaillé dans la table 2.6.

Table 2.6 – *Correspondance des status de piste*

Entier	Correspondance	Description
1	Drapeau vert	La piste est libre
2	Drapeau jaune	La piste est partiellement bloquée, la course continue, mais les dépassements sont interdit dans un ou plusieurs secteurs
4	Voiture de sécurité	La course est neutralisée, les voitures ne peuvent pas dépasser, réduisent leur vitesse et font la queue derrière la voiture de sécurité
5	Drapeau rouge	La course est interrompue, toutes les voitures rentrent au stand
6	Voiture de sécurité virtuelle	La course est neutralisée, les voitures ne peuvent pas dépasser et réduisent leur vitesse
7	Fin de la voiture de sécurité	La voiture de sécurité rentre à la fin de ce tour

Pour l'utiliser, elle a été transformée en 6 features binaires où chaque feature représente la présence d'un statut. Les distributions de la variable avant et après la transformation est observable en figure 2.5.

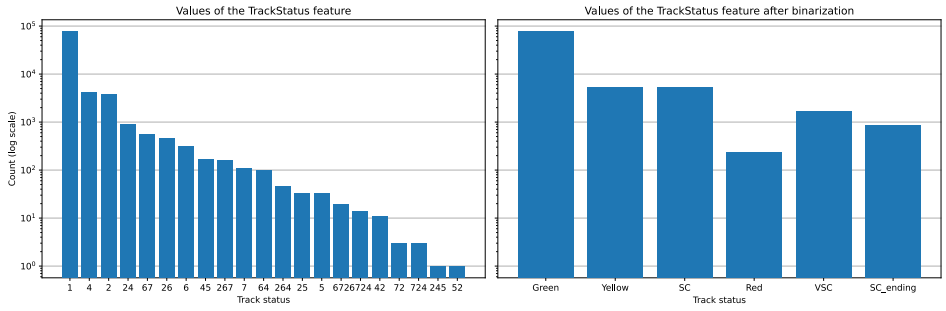

Figure 2.5 – *Distribution de la feature TrackStatus avant et après la discrétisation*

6'455 observations avait des valeurs manquantes pour les colonnes DriverAhead et DistanceToDriverAhead. Il est logique que ces valeurs soient manquantes pour les

tours de la voiture en tête de course. Cependant, en étudiant la distribution des numéros de tour de ces observations, présentés en figure 2.6, on remarque que la plupart des tours sont les premiers de la course. Il n'est pas clair pourquoi ces valeurs sont manquantes, mais il est possible que ce soit un problème de la capture des données. Les premiers tours de la course sont donc exclus du dataset, ce qui ne pose pas de problème car il est très rare d'avoir des arrêts au stand dans les premiers tours. Le reste de la distribution est assez uniforme avec une légère diminution de la fréquence après 55 tours, car la majorité des courses durent entre 50 et 60 tours.

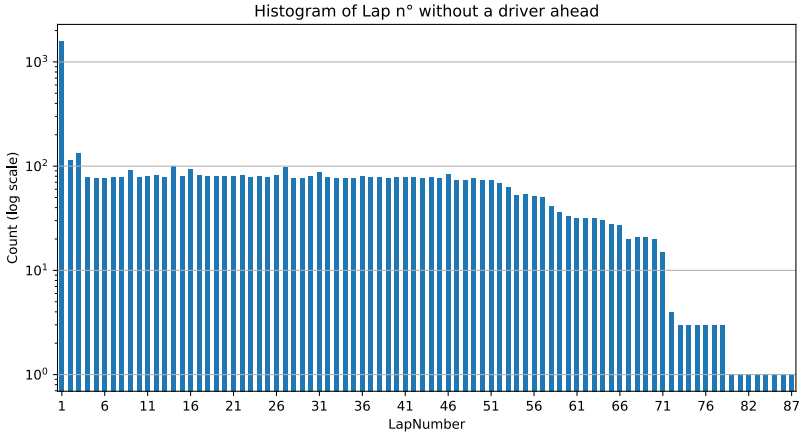

Figure 2.6 – *Distribution de la feature LapNumber pour les tours avec des valeurs manquantes*

En figure 2.7, on voit que beaucoup d'échantillons n'ont pas de valeurs pour la colonne GapToLeader et en observants la figure 2.8, on remarque que ce sont tous des voitures qui ont au moins un tour de retard. Il est difficile et peut intéressant de déterminer l'écart entre une voiture retardataire et le leader, les valeurs manquantes ont donc été remplacées par -1.

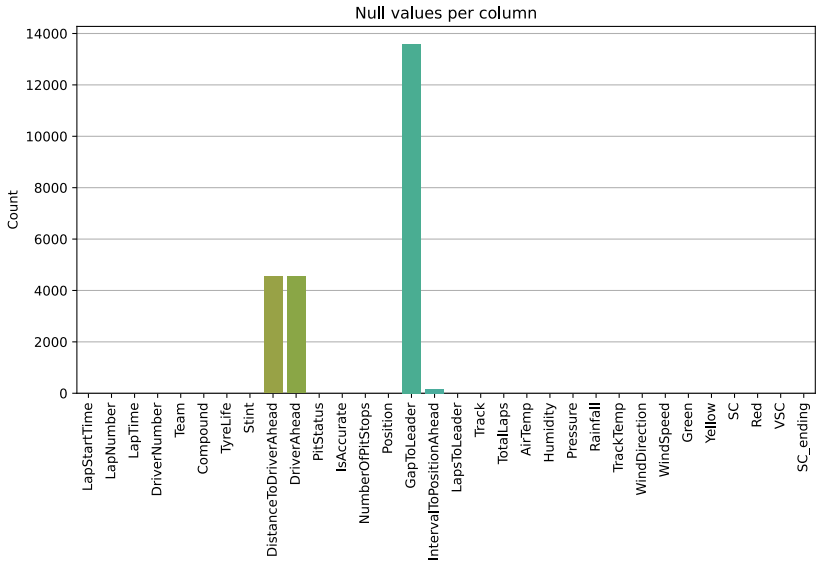

Figure 2.7 – *Valeurs manquantes de chaque colonne, les observations d'IntervalToPositionAhead nulles sont celles de voiture avec un tour de retard sur tout le peloton, celles avec des valeurs manquantes de DistanceToDriverAhead et DriverAhead sont ceux du pilote en tête.*

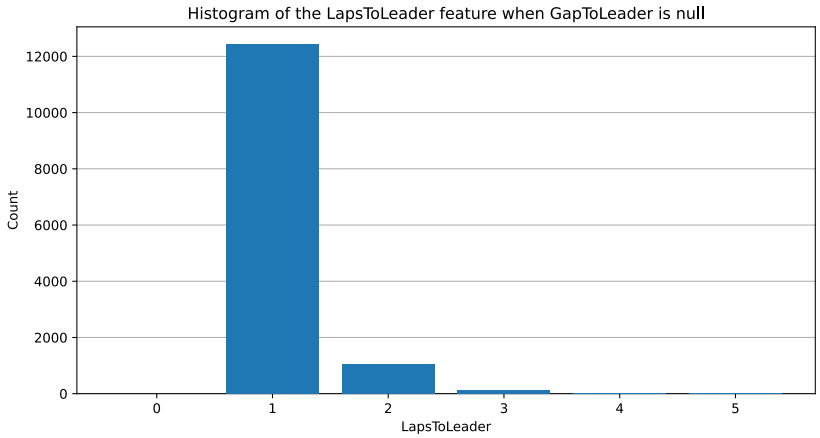

Figure 2.8 – *Histogramme de la caractéristique LapsToLeader lorsque GapToLeader est nul*

Les features Track et Compound de notre dataset nécessitent un prétraitement avant d'être utilisées dans notre modèle. Nous avons utilisé l'encodage one-hot de la bibliothèque scikit-learn pour effectuer cette transformation. L'encodage one-hot est une technique couramment utilisée pour représenter des variables catégorielles sous forme de vecteurs binaires. Par exemple pour la feature Track, en utilisant l'encodage one-hot, on crée des variables binaires pour chaque piste, où une variable est définie à 1 si la course a eu lieu sur cette piste et 0 sinon. L'utilisation de l'encodage one-hot nous permet de représenter ces caractéristiques catégorielles en évitant d'introduire un ordre ou une relation numérique entre les catégories. Pour gérer de manière robuste les nouveaux circuits, le OneHotEncoder à été initialisé avec le paramètre handle unknown='ignore'. De cette manière, le OneHotEncoder ignore

simplement les catégories inconnues et génère un vecteur de zéros correspondant à cette catégorie.

L'intégralité des étapes de prétraitement sont résumées dans la figure 2.9.

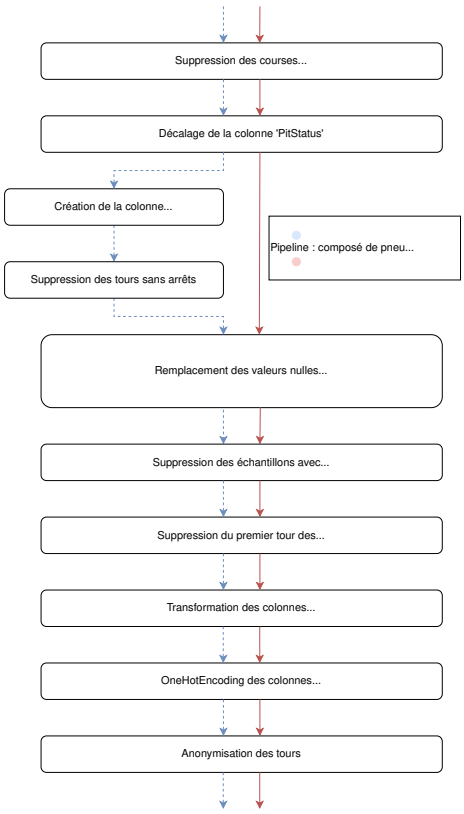

Figure 2.9 – *Pipelines de prétraitement des données, le pipeline de gauche est utilisé pour le modèle de prediction des gommes et celui de droite pour le modèle de prédiction de l'arrêt au stand.*

2.4 Exploration des données

En faisant analyse de forme, le set de données final possède 84'630 lignes et 22 colonnes dont 11 variables quantitatives et 11 qualitatives, la distribution des types de données est en figure 2.10. 1'600 lignes ont été supprimées, car elles possédaient des valeurs manquantes, soit environ 2% du dataset.

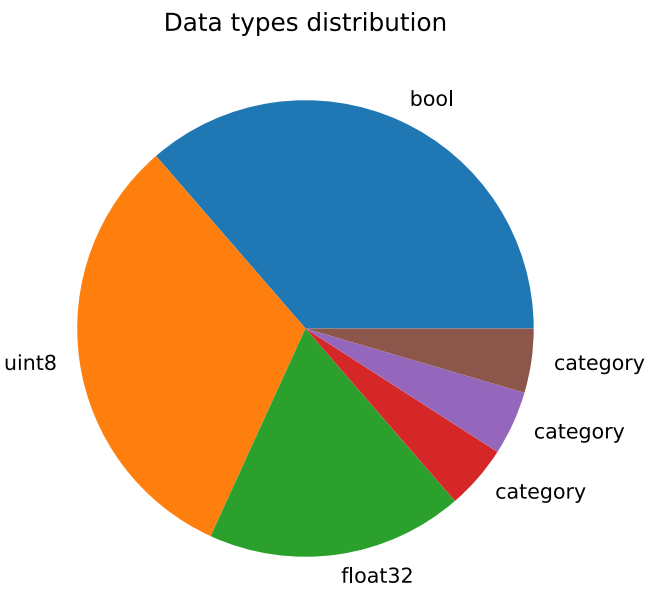

Figure 2.10 – *Distribution des types des features du dataset*

La première variable cible étudiée est is_pitting. Les voitures ne rentrant en moyenne qu'une à trois fois au stand par course, la distribution de cette variable est très déséquilibrée. Seulement 3% de cas positifs, une attention particulière doit donc être accordée à la gestion de ce déséquilibre lors de la phase de modélisation afin d'éviter tout biais potentiel.

Table 2.7 – *Proportion des valeurs de la target is pitting*

Négatif	0.970318
Positif	0.029682

La distribution de la variable Compound est en figure 2.11. On remarque que les pneumatiques les plus utilisés sont les pneumatiques durs et mediums. Les pneumatiques tendres sont moins représentés, étant moins durable les voitures effectuent moins de tours avec. Les intermédiaires et les wets sont utilisés beaucoup moins souvent, car ils ne sont utilisés que dans des conditions météorologiques particulières. Ils ne seront pas présent dans le dataset final, car les courses avec des conditions météorologiques particulières ont été supprimées.

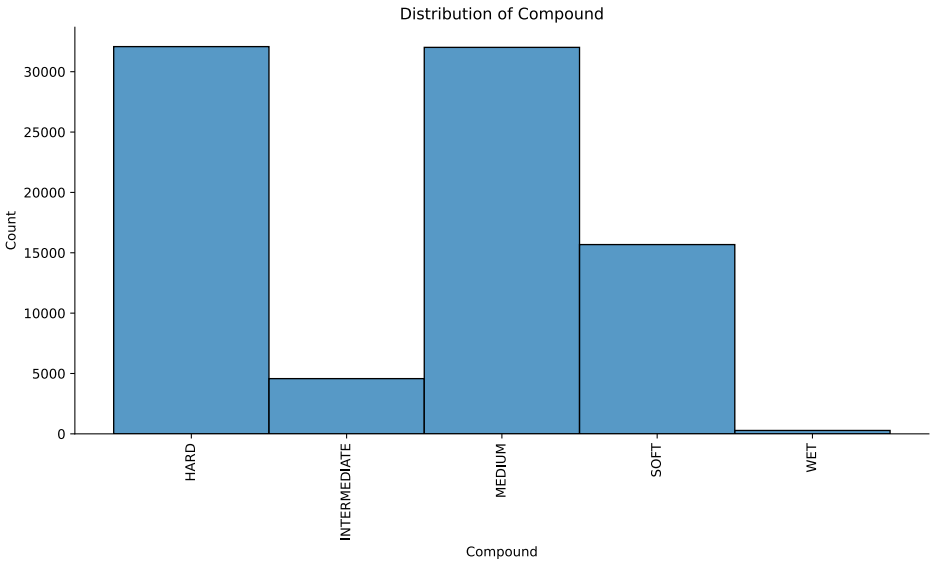

Figure 2.11 – *Distribution de la colonne Compound*

Dans la section suivante, nous examinons la distribution des features du jeu de données. Seules les plus intéressantes sont présentées ici, les autres seront fournies en annexe B.

La distribution en figure 2.12 est unimodal et asymetrique, cela est dû à la nature de l'évolution de l'age des pneumatiques lors de la course. Supposons que dans une course de 60 tours on effectue des arrêts tous les 20 tours, on a trois périodes distinctes pour les âges des pneumatiques : 1-20 tours, 21-40 tours et 41-60 tours. Chaque valeur de 1 à 20 apparaîtra trois fois dans la colonne TyreLife. En revanche, si une voiture ne fait aucun arrêt et parcourt les 60 tours sans changer les pneumatiques, chaque valeur entre 1 et 60 aura seulement un échantillon.

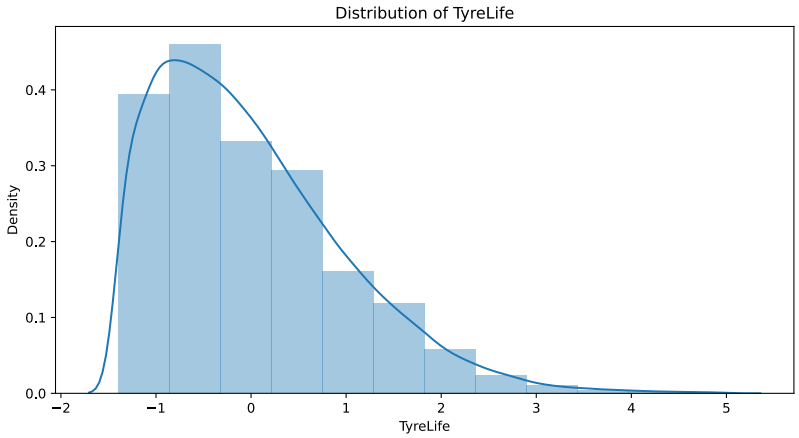

Figure 2.12 – *Distribution de la colonne TyreLife*

La figure 2.13, présente la distribution de la variable Track. Track représente le circuit sur lequel la course a lieu, la distribution est assez déséquilibrée, car certains circuits sont plus courts que d'autres se qu'il fait que les pilotes effectuent moins de

tours sur ces circuits. Et d'autre ne sont pas utilisés tous les ans comme le Mugello qui a été utilisé qu'une fois en 2020.

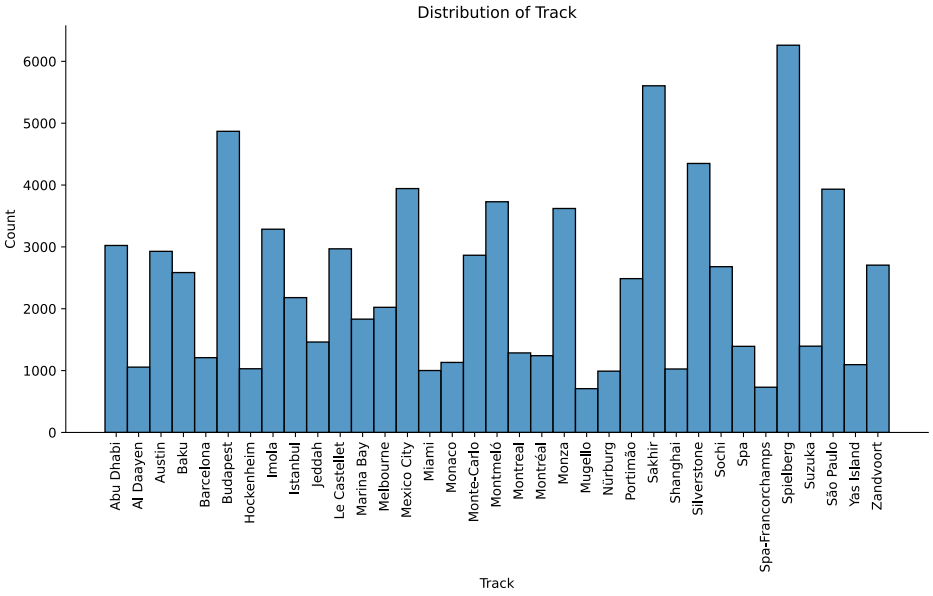

Figure 2.13 – *Distribution de la colonne Track*

Chapitre 3

Expériences

Au cours de ce travail nous utilisons un apprentissage supervisé, l'idée est d'avoir une approche d'imitation learning. C'est à dire que les données historiques sont utilisées pour l'entrainement et les décisions prises par les équipes sont considérées comme ground-truth. On utilise cette méthode car elle nécessite moins de mise en place qu'une approche par renforcement qui nécessiterais la mise en place d'une simulation pour estimer le temps gagné dans les différents scénarios hypothètiques. En revanche l'inconvénient de cette méthode est que les modèles ne peuvent pas surpasser les performances des stratégistes étant donnée qu'ils tentent de les copier. Il est estimé que cela n'est pas un problème dans la mesure où les stratégies employées sont bonnes en moyenne.

3.1 Préparation des données

Les modèles sont entrainés sur les années 2019 à 2022, en réservant l'année 2023 comme set de test afin d'évaluer la capacité du modèle à généraliser sur des données qui ne sont pas inclus dans le set d'entrainement. Étant donné la quantité importante de données, nous avons choisi d'effectuer une division du dataset en un jeu d'entraînement et un jeu de validation avec un ratio de 0,2.

Les étapes de preprocessing ont ensuite été appliquée sur les sets de données individuellement afin d'éviter le data leakage. Ce phénomène se produit lorsque des informations provenant du set de test ou de validation sont incorporées dans le processus d'entraînement du modèle, ce qui fausse les résultats de l'évaluation et conduit à une estimation optimiste des performances du modèle.

3.2 Modélisation

L'objectif étant de déterminer la stratégie d'une voiture, le problème comporte 2 parties :

- La première est de déterminer si la voiture s'arrête ou non.
- La seconde est de déterminer les pneus à utiliser si la voiture s'arrête.

On peut donc modéliser le problème comme une classification binaire suivie d'une classification multi-classes, où la première détermine si la voiture s'arrête ou non et la seconde détermine les pneus à utiliser si la voiture s'arrête. Une autre modélisation possible est de considérer les 2 parties comme un seul problème de classification multi-classes, où le modèle détermine directement les pneus à utiliser et la classe "ne

s'arrête pas" est considérée comme une classe à part entière. Une dernière modélisation possible est de considérer les 2 parties comme 2 sorties indépendantes d'un même modèle si l'algorithme utilisé le permet. Les 3 modélisations sont illustrées par la figure **??**.

Les modélisations basées sur des modèles de régression sont aussi possibles mais elles ne sont pas considérées car selon l'étude de l'état de l'art [Hei+20b] les modèles de classification sont plus performants.

Dans les prochaines sections nous allons détailler les modélisations explorées et les résultats obtenus.

3.3 Classification binaire pour la décision d'arrêt

3.3.1 Random Forest

Le premier algorithme de classification utilisé est Random Forest, c'est un algorithme d'ensemble qui combine plusieurs arbres de décision. Il a été choisit car il est performant sur des sets de données déséquilibrés avec mécanisme de pondération des classes et qu'il est interprétable. L'interprétabilité est un critère important car il permet de comprendre les décisions prises par le modèle se qui vas dans un premier temps nous aider à l'améliorer et dans un second temps à donner confiance aux utilisateurs du modèle.

L'implémentation utilisée est celle de scikit-learn [scia] qui est une librairie open-source de machine learning pour python. Les hyperparamètres du modèle ont été choisis après une recherche par grid search avec une validation croisée sur 4 folds.

Grid search est une méthode d'optimisation des hyperparamètres dans laquelle un modèle est entrainé pour chaque combinaison d'hyperparamètres. Il est donc important de définir une métrique pour estimer la qualité des modèles entrainés.

Métriques

Étant donnés, notre set de données très déséquilibré, nous n'allons pas utiliser l'accuracy ou le score ROC AUC car ils ne sont pas adaptés. Le choix repose entre le F1-score et la balanced accuracy qui sont les 2 métriques les plus communement utilisées pour les sets de données déséquilibrées.

Pour décider quelle métrique favoriser on s'attarde sur leur définition respective à partir des ensembles des classes prédites. La figure 3.1 visualise graphiquement ces ensembles pour le cas d'une classification binaire.

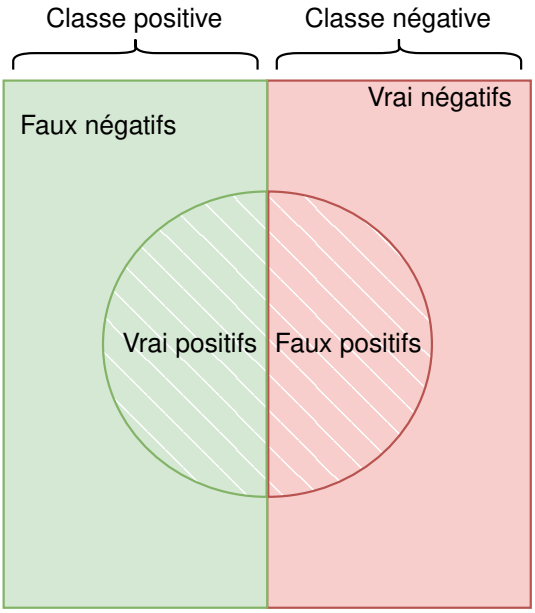

Figure 3.1 – *Visualisation des résultats d'un système de classification binaire*

La balanced accuracy est définie comme la moyenne entre la sensitivité (rappel) et la spécificité :

$$\frac{\text{spécificité} + \text{sensitivité}}{2}$$

La figure 3.2 visualise les ensembles considérés par la spécificité et la sensitivité.

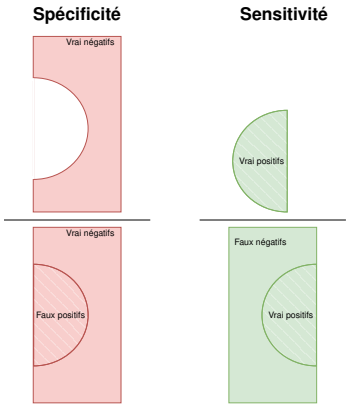

Figure 3.2 – *Visualisation de la spécificité et de la sensitivité*

la sensitivité ou rappel indique combien de fois le modèle a correctement prédit la classe positive parmi tous les échantillons positifs réels. La spécificité est l'équivalent de la sensitivité pour la classe négative, elle mesure la proportion d'échantillons négatifs correctement prédits par le modèle.

Le F1-score est défini comme la moyenne harmonique entre la précision et la sensitivité :

$$F = 2 \cdot \frac{\text{précision} \cdot \text{sensitivité}}{\text{précision} + \text{sensitivité}}$$

La précision indique combien de fois le modèle a correctement prédit la classe positive parmi toutes les prédictions positives faites, La figure 3.3 montre les ensembles considérés par ses 2 métriques.

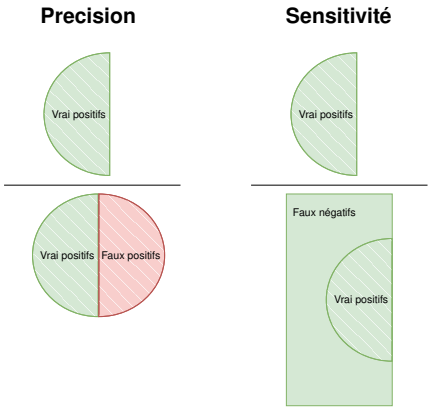

Figure 3.3 – *Visualisation de la précision et de la sensitivité*

On peut observer que le F-score pénalise une précision ou un rappel faible, 2 métriques qui ne considèrent pas les vrai négatifs et qui accordent une grande importance à l'identification de la classe positive. Pour cette raison, le f1 score est souvant utilisé dans les problèmes de détections d'anomalies, car les sets de données sont fortement biasés en faveur de la classe négative. Notre cas est assez similaire à ce cas de figure avec 97% de cas négatifs, le f1 score devrait donc être plus adapté.

En utilisant l'implémentation GridSearchCV de scikit-learn, nous avons effectué une optimisation du f1 score, les hyperparamètres sont en table 3.1.

Table 3.1 – *Paramètres de la recherche GridSearchCV*

hyperparamètre	valeurs
class weight	None, balanced, balanced subsample
n estimator	100, 500, 1000, 2000
max depth	5, 20, None
max features	sqrt, log2, None

La table complète des résultats est en annexe C.1.

Le meilleur modèle obtient un f1 score de 0.13 avec les paramètres en table 3.2. Ce score est très faible, nous allons étudier les performances du modèle sur le set de validation pour comprendre ce résultat.

Table 3.2 – *Paramètres du meilleur modèle*

hyperparamètre	valeur
class weight	balanced subsample
n estimator	2000
max depth	5
max features	sqrt

Les métriques des résultats sont observables en table 3.4 et la matrice de confusion en table 3.3. En raison de la nature déséquilibrée de notre classification, on ne peut pas se fier à la métrique d'accuracy.

Table 3.3 – *Matrice de confusion de l'entrainement du premier modèle, les colonnes représentent les prédictions et les lignes le ground truth.*

	Négatif	Positif
Négatif	10470	3422
Positif	103	281

Table 3.4 – *Matrice de confusion de l'entrainement du premier modèle, les colonnes représentent les prédictions et les lignes le ground truth.*

Classe	Precision	Recall	F1Score
Négatif	0.99	0.75	0.86
Positif	0.08	0.73	0.14

On remarque dans les résultats que le modèle à de la peine à classifier les cas positifs avec beaucoup de faux-positifs.

Analyse des résultats

Premièrement, pour confirmer que le modèle utilise correctement les variables pour la décision nous étudions les raisons derrière ses décisions avec des méthodes d'explications.

Random Forest étant un algorithme interprétable, il nous permet de mesurer en figure 3.4 l'importance des features dans sa classification. L'importance d'une feature est calculée comme la réduction totale de l'entropie normalisée apportée par cette feature.

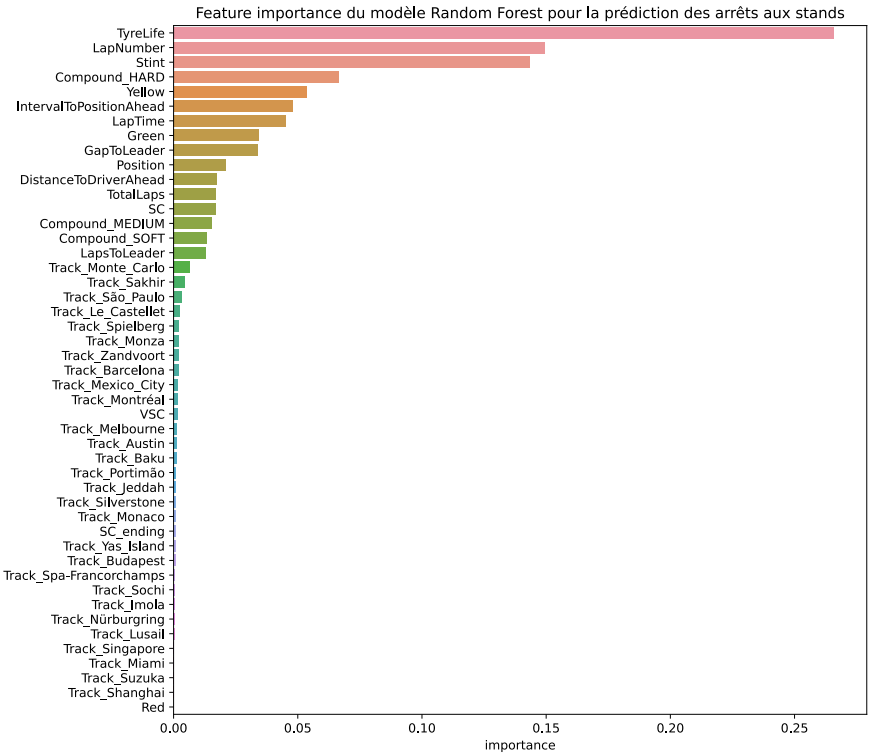

Figure 3.4 – *Importance des features du modèle de Random Forest pour la classification des arrêts au stand.*

Les features TyreLife, LapNumber et Stint sont les plus importantes. Ce qui est cohérent avec nos connaissances sur la stratégie de course. Les features les moins utilisées sont celles liées au circuit, nous supposons que c'est parce que la feature est one-hot encoded se qui crée une grande matrice creuse les features individuelles apporte donc peu d'information au modèle.

Pour comprendre les résultats, nous allons analyser les prédictions du modèle sur le set de validation. Random Forest est un algorithme interprétable localement de lui même mais le grand nombre d'arbre dans notre modèle rend cela plus difficile. Nous utilisons LIME (Local Interpretable Model-agnostic Explanations) [RSG16] un outil d'explication locale, c'est à dire qu'il permet d'expliquer les prédictions pour une instance donnée. Pour ce faire LIME fonctionne de la manière suivante :

1. On génère un ensemble de données autour de l'instance à expliquer en perturbant légerement les features.
2. On prédit la classe de l'ensemble de données généré avec le modèle.
3. On entraine un modèle explicable avec comme features l'ensemble de données généré et comme labels les prédictions du modèle. LIME attribue un poids à chaque instance de l'ensemble de données généré en fonction de la distance avec l'instance à expliquer. Les instances les plus proches sont plus importantes car elles sont plus représentatives de l'instance que l'on cherche à expliquer.

Nous allons étudier les décisions prises pendant la course de Lance Stroll au Grand Prix d'Espagne 2023, on peut observer la progression des probabilités d'arrêt au stand en figure 3.5.

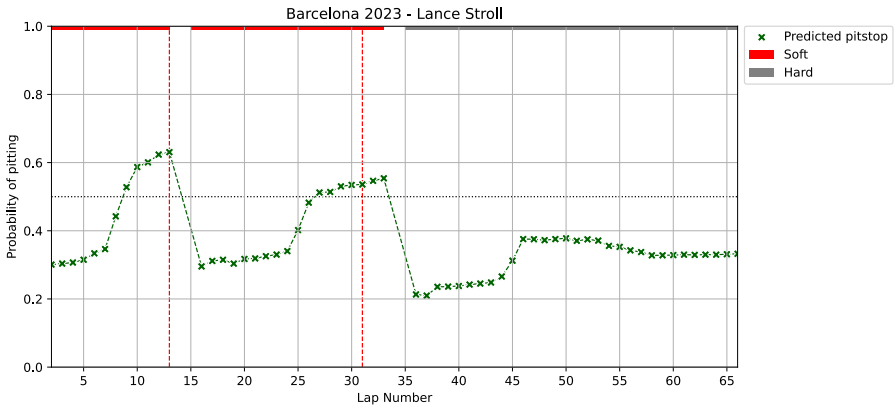

Figure 3.5 – *Progression des probabilités d'arrêt au stand pour le Grand Prix d'Espagne 2023 de Lance Stroll. Les probabilités sont calculées à chaque tour en utilisant les données du tour précédent. Les lignes verticales rouges indiquent la target du modèle, c'est à dire un tour avant l'arrêt au stand réel. Les bares horizontales en haut de la figure indique le composé de pneu utilisé (rouge = tendre, jaune = médium, gris = dur).*

Le modèle semble capturer la relation entre les arrêts au stand et les données de course. Les probabilités d'arrêt au stand augmentent à chaque tour et diminuent drastitquement après un arrêt au stand réel. Les probabilités augmentent également plus faiblement pendant le relai sur pneus durs se qui est cohérent avec la dégradation plus faible de ces pneus. Cependant le modèle ne semble pas pouvoir augmenter la probabilité d'arrêt au stand assez rapidement pour éviter les faux positifs, cela explique les résultats de la matrice de confusion. Ces résultats sont cohérents avec les observations de l'étude [Hei+20b].

Nous allons maintenant étudier les explications de LIME pour les tours 2, 8 et 16 de cette course. Premièrement, on peut observer en figure 3.6 les résultats de LIME pour le tour 2, où le modèle à décidé de ne pas s'arrêter.

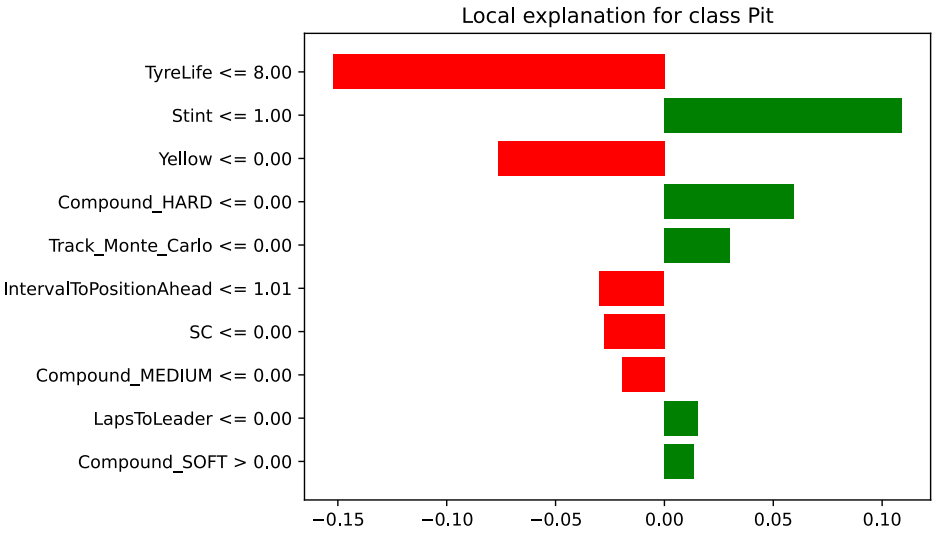

Figure 3.6 – *Résultat de l'algorithme d'explication LIME pour le tour 2 de la course de Lance Stroll à Barcelone en 2023.*

On peut voir que l'age des pneus faible est la raison principale de la décision. En figure 3.7 les résultats de LIME pour le tour 8, le premier tour où le modèle décide de s'arrêter.

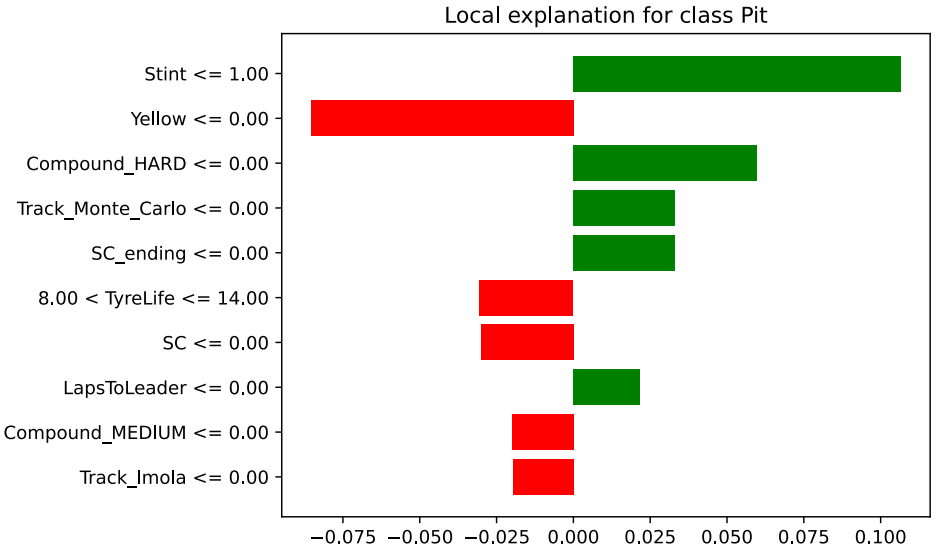

Figure 3.7 – *Résultat de l'algorithme d'explication LIME pour le tour 8 de la course de Lance Stroll à Barcelone en 2023.*

L'age des pneus a augmenté et sont impact négatif sur la décision est plus faible. L'absence de pneumatique dur et le fait que la voiture soit dans son premier relai sont les raisons principales de la décision. En figure 3.8 les résultats de LIME pour le tour 16, le premier tour après l'arrêt au stand.

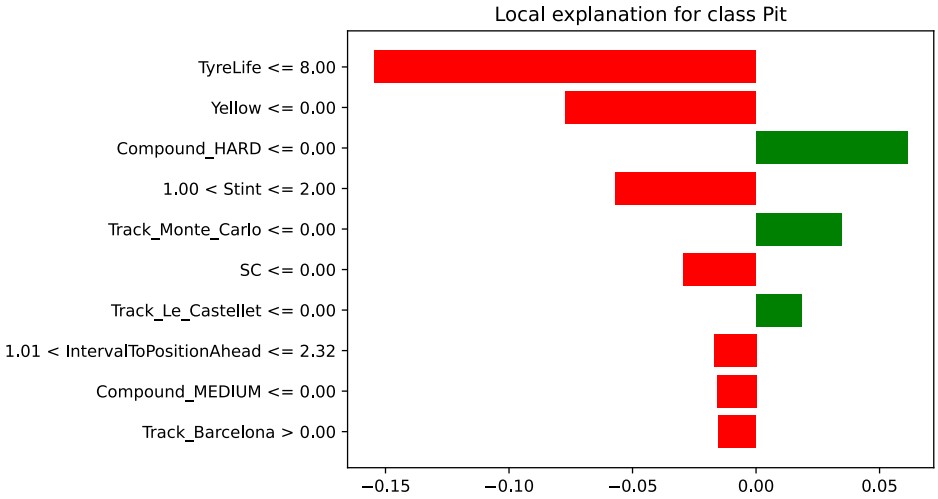

Figure 3.8 – *Résultat de l'algorithme d'explication LIME pour le tour 16 de la course de Lance Stroll à Barcelone en 2023.*

La prédiction est redevenue négative car l'age des pneus est de nouveau faible. De plus, on peut observer l'impact négatif sur la prédiction du second relai. Le modèle a appris que les voitures font au moins deux relais il est donc normal que le second relai ait un impact négatif sur la prédiction. Dans l'ensemble, les features utilisées par le modèle sont cohérentes avec nos connaissances sur la stratégie de course.

Pour étudier le comportement du modèle face au périodes de safety car, on observe la progression des probabilités d'arrêt au stand pour le Grand Prix d'Azerbaïdjan 2023 de Sergio Perez en figure 3.9.

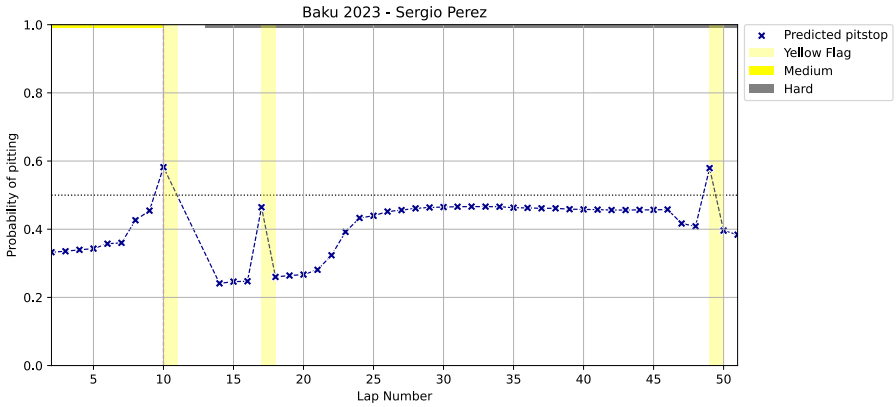

Figure 3.9 – *Progression des probabilités d'arrêt au stand pour le Grand Prix d'Azerbaïdjan 2023 de Sergio Perez. Les zones jaunes indiquent les périodes de drapeau jaune, de safety car ou de virtual safety car.*

La période de drapeau jaune au tour 10 a poussé le modèle à prédire un arrêt au stand pour Sergio Perez. Ce qui est cohérent avec la stratégie de course de l'équipe Red Bull qui a effectué un arrêt au stand pour Sergio Perez au tour 11. En figure 3.10 on peut observer les résultats de LIME pour le tour 10, le drapeau jaune est la raison principale de la décision.

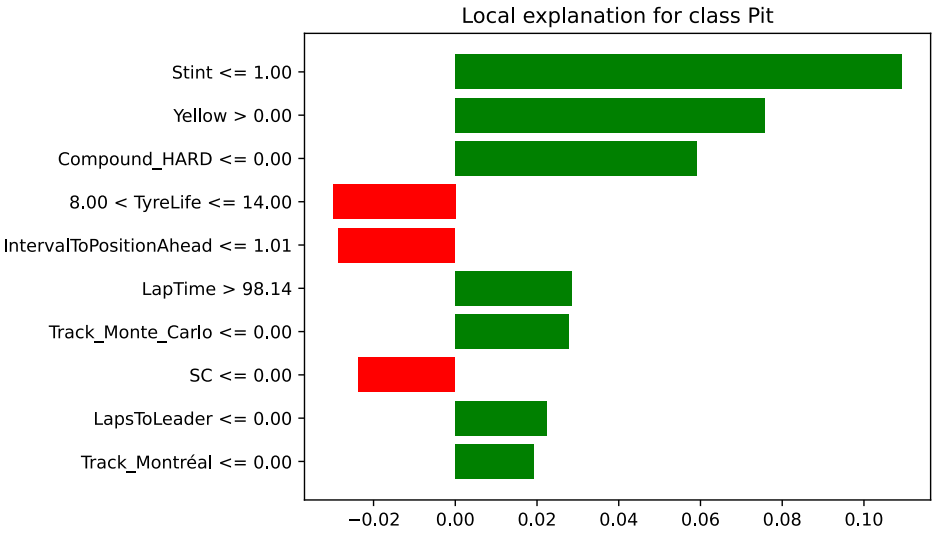

Figure 3.10 – *Résultat de l'algorithme d'explication LIME pour le tour 10 de la course de Sergio Perez à Bakou en 2023.*

Cependant la période de drapeau jaune au tour 49 a également poussé le modèle à prédire un arrêt au stand alors qu'il n'y en a pas eu. Cela est dû à un manque d'information sur la raison du drapeau jaune. La première période de drapeau jaune est due à un accident et donc est fortement suseptible d'entrainer une voiture de sécurité. Alors que la deuxième période de drapeau jaune est due à une sortie de piste sans conséquence et donc n'entraine pas de voiture de sécurité. Ces information ne sont pas disponibles dans les données car elles sont observables uniquement en regardant les images de la course. En figure 3.11 on peut voir que le drapeau jaune est encore une fois la cause de la décision.

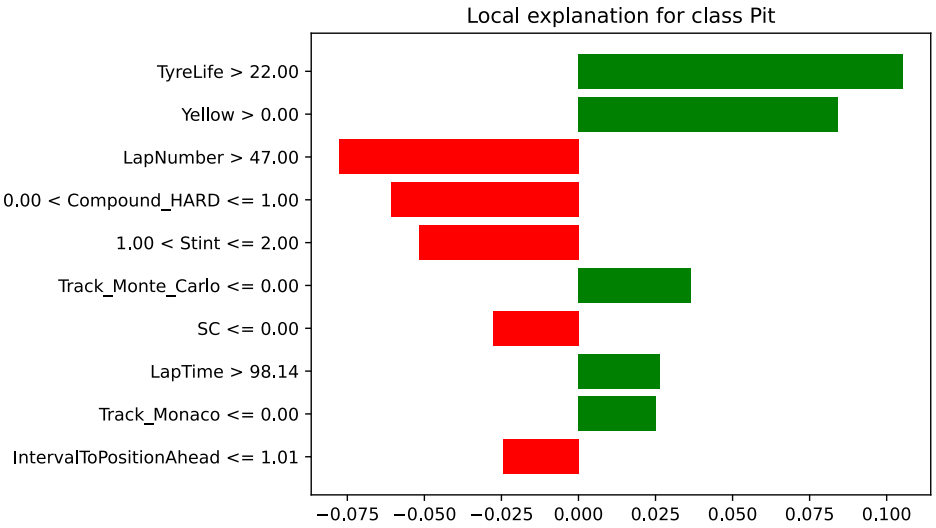

Figure 3.11 – *Résultat de l'algorithme d'explication LIME pour le tour 49 de la course de Sergio Perez à Bakou en 2023.*

3.3.2 Réseaux de neurones

Les réseaux de neurones sont des modèles plus puissant que Random Forest. Nous allons donc les tester sur le problème de prédiction d'arrêt au stand pour voir si ils sont capables d'obtenir de meilleurs résultats. Pour ce faire nous avons utilisé la libraire Keras [Cho+15] qui permet de construire des réseaux de neurones en Python. Nous avons testés plusieurs architectures de réseaux de neurones et nous avons obtenu les meilleurs résultats avec l'architecture suivante 3.5 :

Couche	Nombre de neurones	Fonction d'activation
Couche d'entrée	64	ReLU
1ère couche cachée	64	ReLU
Dropout	0.2	-
2ème couche cachée	64	ReLU
Dropout	0.2	-
3ème couche cachée	64	ReLU
Dropout	0.2	-
Couche de sortie	1	Sigmoid

Table 3.5 – *Architecture du réseau de neurones utilisé pour la prédiction d'arrêt au stand.*

Les couches Dropout sont des couches qui désactivent aléatoirement un certain pourcentage de neurones à chaque itération. Le réseau de neurones est donc obligé de s'adapter à l'absence de certains neurones et ne peut pas se reposer sur un seul neurone. Cela nous permet de réduire l'overfitting du modèle et d'améliorer sa capacité de généralisation. Nous avons également utilisé la technique de Early Stopping qui permet d'arrêter l'entrainement du modèle si la performance sur le jeu de validation ne s'améliore plus. Nous avons choisi une batch size assez grande de 256 pour s'asssurer que chaque batch contiennent des arrêts au stand. Comme avec les Random Forest, nous avons réservé 20% des données pour le jeu de validation et nous avons entrainé le modèle sur les 80% restants. Vous pouvez retrouver en table 3.6 les hyperparamètres utilisés pour l'entrainement du modèle.

Hyperparamètre	Valeur
Optimizer	Nadam
Loss	Binary Crossentropy
Learning rate	0.001
Epochs	300
Batch size	256

Table 3.6 – *Hyperparamètres utilisés pour l'entrainement du réseau de neurones.*

Pour palier au problème de déséquilibre des classes, nous avons utilisé l'argument class_weight de la fonction fit de Keras. Cet argument permet de donner un poids à chaque classe pour le calcul de la loss. Les poids des classes ont étés calculés avec la méthode compute_class_weight [scib] de la librairie scikit-learn. Les pondérations résultantes sont 0.5 pour la classe 0 et 17 pour la classe 1.

Nous avons obtenu les résultats suivants en table 3.8 avec la matrice de confusion en figure 3.7 :

Table 3.7 – *Matrice de confusion du réseau de neurones.*

	Négatif	Positif
Négatif	10053	2665
Positif	121	247

Table 3.8 – *Métriques de performance du réseau de neurones.*

Classe	Precision	Recall	F1Score
Négatif	0.99	0.79	0.88
Positif	0.08	0.67	0.15

Les résultats sont très similaires à ceux obtenus avec les Random Forest. Mais en étudiant le comportement du modèle sur les données de validation, nous avons remarqué un meilleur comportement. En figure 3.12 vous pouvez voir la différence de comportement entre les deux modèles pour la course de Carlos Sainz sur le Grand Prix de Miami en 2023.

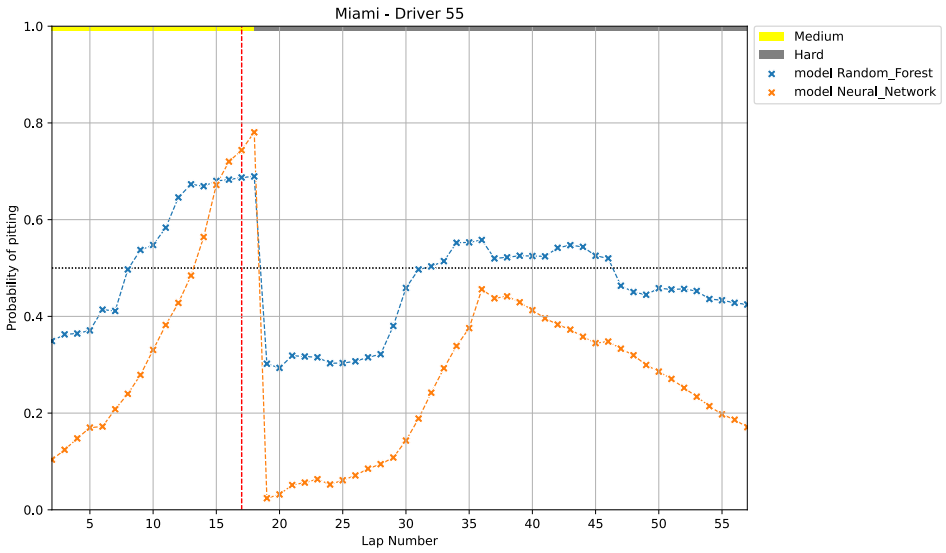

Figure 3.12 – *Prédiction d'arrêt au stand pour la course de Carlos Sainz sur le Grand Prix de Miami en 2023.*

Ici le modèle de Random Forest commence à prédire des arrêt au stand dès le 8ème tour, alors que le modèle de réseau de neurones ne commence à prédire des arrêts au stand qu'à partir du 14ème. Le modèle de réseau de neurones semble pouvoir augmenter sa probabilité d'arrêt au stand plus rapidement que le modèle de Random Forest ce qui est un comportement plus utile pour la stratégie de course. En étudiant les résultats de LIME sur le 8ème tour de la course en figure 3.13 et 3.14, on peut voir que le réseau de neurones accordre plus d'importance à l'age des pneus -0.3 contre -0.15 pour le modèle de Random Forest.

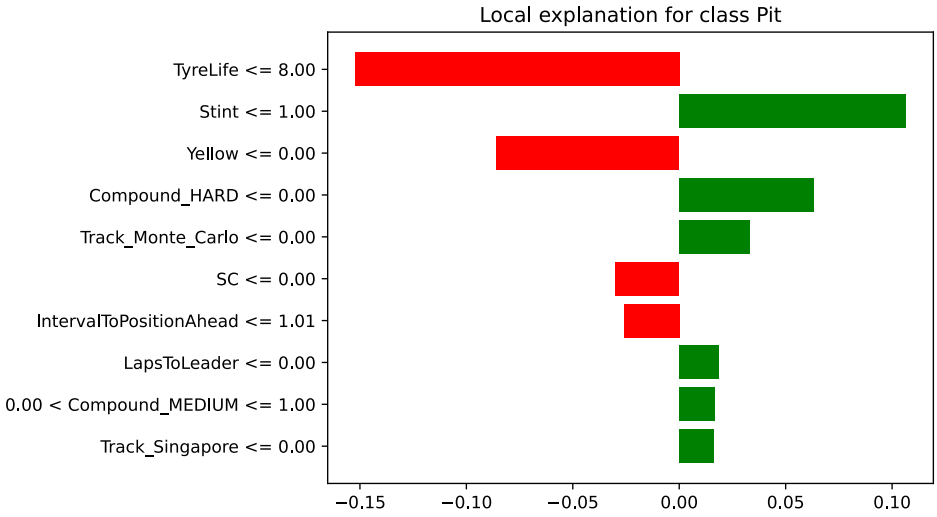

Figure 3.13 – *Résultats de LIME pour le modèle de Random Forest sur le 8ème tour de la course de Carlos Sainz sur le Grand Prix de Miami en 2023.*

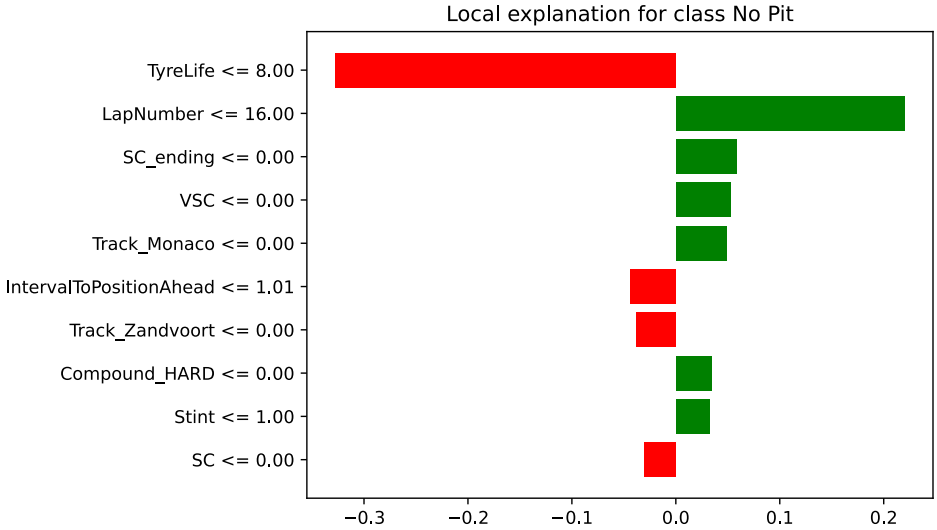

Figure 3.14 – *Résultats de LIME pour le modèle de réseau de neurones sur le 8ème tour de la course de Carlos Sainz sur le Grand Prix de Miami en 2023.*

Malgré les métriqes de performance similaires, le modèle de réseau de neurones semble avoir un meilleur comportement.

3.3.3 Séries temporelles

Les précédentes expériences utilisaient uniquement le dernier tour pour prédire la probabilité d'arrêt au stand. Cependant, il sera plus intéressant d'utiliser les données de plusieurs tours car cela permettra de mieux capturer les tendances.

La formulation du problème devient alors une prédiction de séries temporelles. On considère alors les données des tours comme une séquence. Cette formulation est plus naturelle car les données sont une discrétisation par tour de données de capteurs qui

sont des séries temporelles. Elles sont enregistrées de façon séquentielle et sont donc naturellement ordonnées et corrélées.

Il faut cependant grouper les données par course et par voiture pour avoir des séquences de données cohérentes. On peut observer en figure 3.15 la nature séquentielle des données pour un sous ensemble de features.

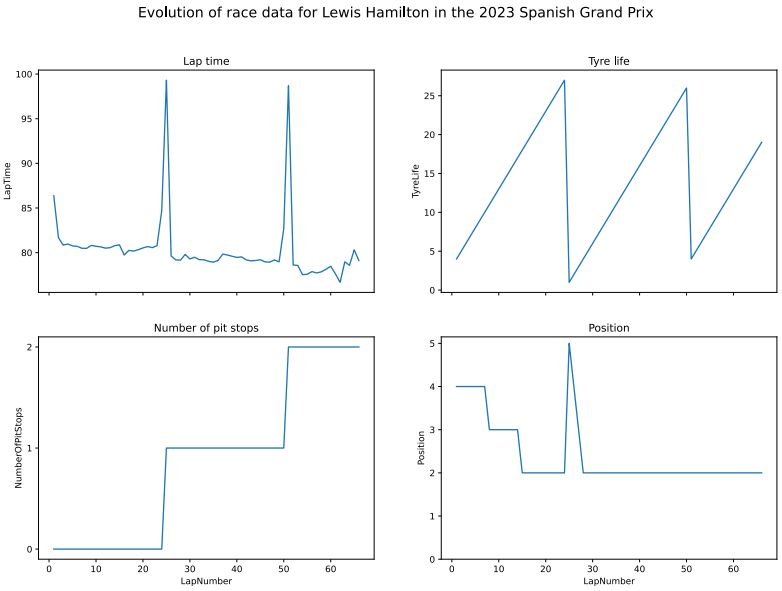

Figure 3.15 – *Exemple de séries temporelles pour la course de Lewis Hamilton sur le Grand Prix d'Espagne en 2023.*

L'étape de préparation des données est plus complexe que pour les modèles précédents. Il faut maintenant créer des séquences de données de taille fixe à partir des données de chaque course. Les données des course de chaque voiture doivent être dans le même set de données (entrainement ou validation). Le processus de séparation des données est le suivant :

1. On regroupe les données avec les colonnes *Year*, *RoundNumber* et *DriverNumber* comme clé.
2. On mélange ces clés puis on les sépare en 80% pour l'entrainement, 20% pour la validation.
3. On crée les 2 sets de données en utilisant les clés séparées.

La séparation n'est plus exactement fidèle au pourcentage spécifié car on sépare les courses et non les tours. Mais avec notre grand nombre de données cela ne pose pas de problème. Ensuite, on crée les séquences de données de taille fixe à partir des données de chaque course. La cible de chaque séquence est la colonne "is_pitting" du prochain tour, par exemple, la cibe d'une séquence comprenant les tours de 10 à 15 est la valeur de "is_pitting" du tour 16.

Pour cette modélisation nous avons utilisé un réseau de neurones récurrents (RNN) avec des cellules LSTM. Les réseau de neurones récurrents sont un type de réseau de neurones adapté aux séries temporelles. Les connexions entre les neurones forment des cycles 3.16, ce qui permet d'utiliser les sorties précédentes comme entrées. Cela permet aux informations passées d'avoir un impact sur les prédictions futures.

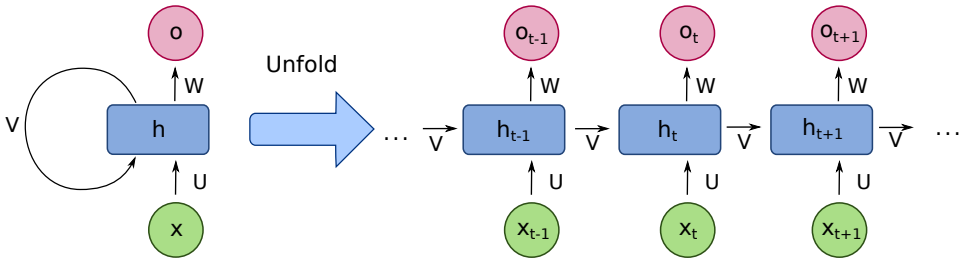

Figure 3.16 – *Exemple de réseau de neurones récurrents.[fde17]*

Une des limitations des réseaux de neurones récurrents est le "vanishing gradient problem" où les gradients deviennent trop petits pour être utiles. Les LSTM (Long Short Term Memory) sont une variante des RNN qui permettent de résoudre ce problème. L'architecture utilisée est en table 3.9.

Table 3.9 – *Architecture du réseau de neurones récurrents.*

Couche	Nombre de neurones	Fonction d'activation
LSTM	64	Tanh
Dropout	0.1	
LSTM	64	Tanh
Dropout	0.1	
Dense	64	ReLu
Dropout	0.1	
Dense	64	ReLu
Dense	1	Sigmoid

Vous pouvez retrouver en table 3.10 les hyperparamètres utilisés pour l'entrainement du modèle. Contrairement au réseaux de neurones classiques nous avons observés de meilleures résultats avec un learning rate plus faible et un batch size plus grand.

Hyperparamètre	Valeur
Optimizer	Nadam
Loss	Binary Crossentropy
Learning rate	0.0005
Epochs	100
Batch size	1024

Table 3.10 – *Hyperparamètres utilisés pour l'entrainement du réseau de neurones.*

On peut voir en table 3.11 et 3.12 les résultats de ce modèle.

Table 3.11 – *Matrice de confusion du réseau de neurones récurrents.*

	Négatif	Positif
Négatif	8914	2607
Positif	91	242

Table 3.12 – *Métriques de performance du réseau de neurones récurrents.*

Classe	Precision	Recall	F1Score
Négatif	0.99	0.77	0.87
Positif	0.08	0.73	0.15

On peut observer une très légère amélioration par rapport au modèle précédent. Cependant, le modèle est toujours très biaisé vers la classe négative. En figure 3.17 on peut voir la comparaison des modèles sur les données du Grand Prix d'Espagne 2023 de Max Verstappen.

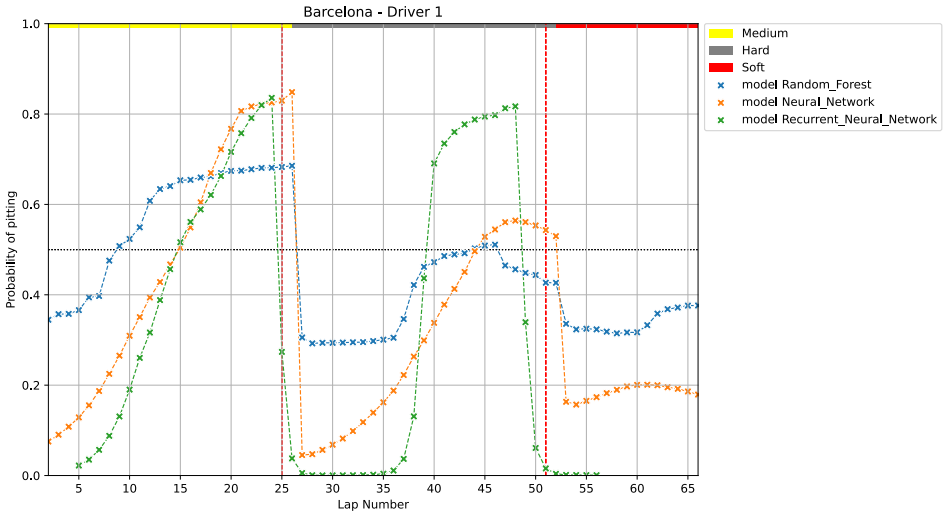

Figure 3.17 – *Comparaison des modèles sur les données du Grand Prix d'Espagne 2023 de Max Verstappen.*

On peut observer que le modèle récurrent a un comportement similaire au modèle précédent mais avec des prédictions plus confiantes. Cependant, il semble sujet a des prédictions assez étrange comme en figure

3.4 Classification multi-classes pour le choix des pneus

Dans cette section nous allons nous intéresser à la classification multi-classes pour le choix des pneus. Elle serait utilisée en combinaison avec la classification binaire pour la prédiction des arrêts aux stands. Le but est que une fois une précition positive pour un arrêt aux stands, on puisse prédire le type de pneus qui sera utilisé.

La préparation des données est similaires à la classification d'arrêt aux stands à la différence que la cible est la colonne "NextCompound". De plus on ne garde que les tours où un arrêt aux stands a eu lieu. On réduit grandement le nombre d'échantillons (environ de 69'400 à 1'900) mais le problème devient plus simple car les classes sont plus équilibrées, comme on peut le voir en figure **??**.

Comme avec la classification binaire, nous avons expérimenté avec des modèles Random Forest et des réseaux de neurones. Le modèle Random Forest a été entrainé avec les hyperparamètres en table 3.13.

Table 3.13 – *Paramètres du meilleur modèle Random Forest pour le choix des pneus.*

hyperparamètre	valeur
class weight	balanced subsample
n estimator	2000
max depth	5
max features	sqrt

On peut voir en table 3.14 et 3.15 les résultats de ce modèle.

Table 3.14 – *Matrice de confusion du meilleur modèle Random Forest pour le choix des pneus. Les colonnes représentent les prédictions et les lignes les vraies valeurs.*

	Soft	Medium	Hard
Soft	83	7	1
Medium	59	63	7
Hard	68	27	69

Table 3.15 – *Métriques de performance du meilleur modèle Random Forest pour le choix des pneus.*

Classe	Precision	Recall	F1Score
Soft	0.71	0.70	0.70
Medium	0.65	0.49	0.56
Hard	0.9	0.42	0.57

Les résultats sont assez particulier, le modèle prédit souvent la classe Soft, on a beaucoup de faux positifs pour cette classe. La classe Hard est rarement prédite mais quand elle l'est, elle est souvent correcte. En figure 3.18 on peut voir la comparaison des prédictions du modèle avec les vraies valeurs.

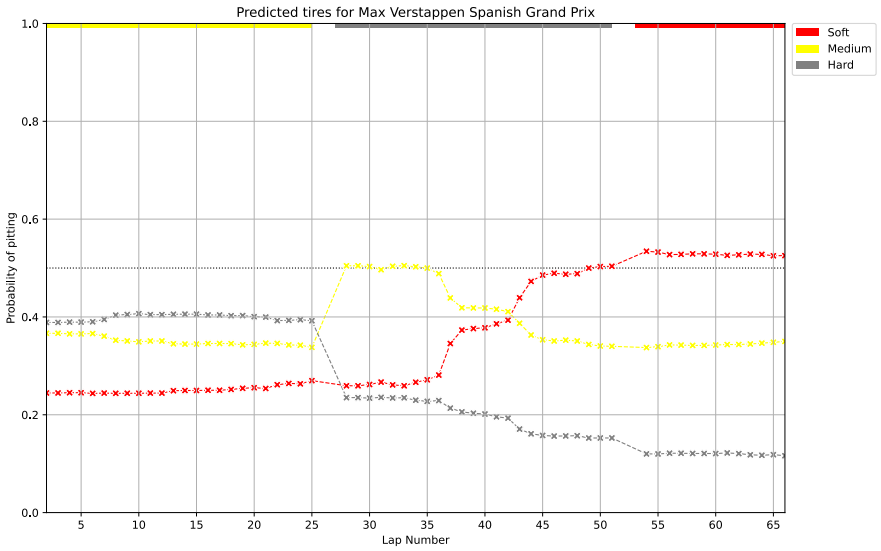

Figure 3.18 – *Prédiction du modèle de Random Forest pour le choix des pneus du grand prix d'Espagne 2023 de Max Verstappen.*

En début de course, le modèle prédit la classe Hard se qui fait sens car c'est le pneu le plus résistant sur lequel on serait le plus susceptible de faire un long relais jusqu'à la fin de la course. Puis en milieu de course le modèle choisit des pneus Medium avant de choisir des pneus Soft à la fin de la course. Cela semble être un comportement logique pour un pilote qui cherche à faire le moins d'arrêts aux stands possible.

3.5 Autre modélisation

Nous avons expérimentés les autres modélisation dans lesquelles les deux problèmes sont traités en même temps. En théorie cela permettrait d'avoir un modèle plus performant car les décisions sont liées. Deux pistes explorées sont celle d'une classification avec 4 classes ou les 2 colonnes "is_pitting" et "NextCompound" sont combinées en une seule cible codifiée de la manière suivante :

Classe	Valeur
Pas d'arrêt aux stands	[1, 0, 0, 0]
Arrêt aux stands avec pneus Soft	[0, 1, 0, 0]
Arrêt aux stands avec pneus Medium	[0, 0, 1, 0]
Arrêt aux stands avec pneus Hard	[0, 0, 0, 1]

L'autre piste est celle d'un réseau de neurones avec 2 sorties, une pour chaque problème. L'une des sorties est une classification binaire pour la prédiction des arrêts aux stands soit un seul neurone avec une fonction d'activation sigmoïde. L'autre sortie est une classification multi-classes pour le choix des pneus soit 3 neurones avec une fonction d'activation softmax. Chacune des sorties possède sa propre fonction de perte. Une telle architecture est illustrée en figure 3.19.

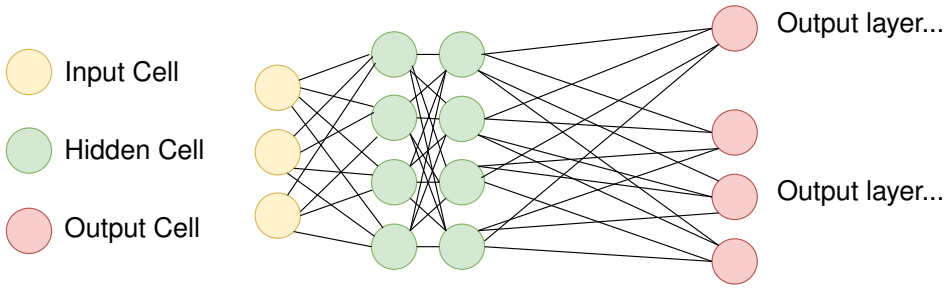

Figure 3.19 – *Architecture d'un réseau de neurones avec 2 sorties.*

Ces deux pistes n'ont pas donné de résultats concluants. Les modèles sont bien moins performants que ceux présentés dans les sections précédentes. Cela peut être dû au fait que la sur-représentation de la classe "Pas d'arrêt aux stands" rend difficile l'apprentissage des classes du choix des pneus.

3.6 Métrique de performance

Suite à l'analyse des résultats des différents modèles explorés dans ce chapitre, on a pu observer que les métriques qui considérait les prédictions comme des résultats booléens (True positif, False positif, ...) étaient assez peu représentatives de la performance des modèles.

En effet, dans son utilisation finale, le modèle n'as pas besoin d'être restraint à une prise de décision binaire avec des seuils de probabilité. Il peut simplement donner une probabilité d'arrêt aux stands pour chaque tour de la course. C'est alors aux ingénieurs de l'équipe de décider si ce seuil est suffisamment élevé pour justifier un arrêt aux stands.

Il fait donc plus sens d'utiliser des métriques qui considèrent les prédictions comme des probabilités et qui sont calculées sur l'ensemble des prédictions d'une course, le scénario d'utilisation du modèle final. L'objectif est de faire la différence entre un modèle ne prédirait pas du tout un arrêt aux stands et un modèle qui l'aurait prédit avec quelques tours de retard.

La métrique proposée pondère

Chapitre 4

Conclusion

Au moment de ce rendu intermédiaire, un premier modèle a été entrainé, il présente des résultats relativement médiocres mais cohérent avec l'état de l'art actuel dans le domaine. Cependant, il s'agit maintenant d'entreprendre différentes pistes pour améliorer les prédictions. Explorer différents algorithmes tel que les réseaux de neurones, notamment les réseaux de neurones récurrents et la modélisation du problème tel qu'un time series. Analyser les résultats pour découvrir les features les plus prédictives et potentiellement en créer de nouvelles plus informatives.

Nelson Jeanrenaud

Bibliographie

[TR14] Theja Tulabandhula et Cynthia Rudin. « Tire Changes, Fresh Air, and Yellow Flags : Challenges in Predictive Analytics for Professional Racing ». In : *Big Data* 2.2 (2014). PMID : 27442303, p. 97-112. doi : 10.1089/big.2014.0018. eprint : https://doi.org/10.1089/big.2014.0018. url : https://doi.org/10.1089/big.2014.0018 (cf. p. 7).

[Cho+15] Francois Chollet et al. *Keras*. 2015. url : https://github.com/fchollet/keras (cf. p. 33).

[Cho15] Christopher Choo. « Real-time decision making in motorsports : analytics for improving professional car race strategy ». Thèse de doct. Jan. 2015 (cf. p. 7).

[RSG16] Marco Tulio Ribeiro, Sameer Singh et Carlos Guestrin. « "Why Should I Trust You ?" : Explaining the Predictions of Any Classifier ». In : (2016), p. 1135-1144 (cf. p. 28).

[fde17] fdeloche. *Structure of RNN*. 2017. url : https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg (cf. p. 37).

[jjn18] Vyssion & jjn9128. *Hurry up and weight*. Mai 2018. url : https://www.f1technical.net/features/21637. (accessed : 20.05.2023) (cf. p. 2).

[19] *Formula One Race Strategy*. Supported by McLaren Racing Limited. 2019. url : https://www.stem.org.uk/rxstz. (accessed : 20.05.2023) (cf. p. 2).

[Hei+20a] Alexander Heilmeier et al. « Application of Monte Carlo Methods to Consider Probabilistic Effects in a Race Simulation for Circuit Motorsport ». In : *Applied Sciences* 10.12 (2020). issn : 2076-3417. doi : 10.3390/app10124229. url : https://www.mdpi.com/2076-3417/10/12/4229 (cf. p. 5).

[Hei+20b] Alexander Heilmeier et al. « Virtual Strategy Engineer : Using Artificial Neural Networks for Making Race Strategy Decisions in Circuit Motorsport ». In : *Applied Sciences* 10.21 (2020). issn : 2076-3417. doi : 10.3390/app10217805. url : https://www.mdpi.com/2076-3417/10/21/7805 (cf. p. 5, 24, 29).

[ana21] "Part time analyst". *F1 Strategy Analysis*. Sept. 2021. url : https://theparttimeanalyst.com/2021/09/29/f1-strategy-analysis/. (accessed : 20.05.2023) (cf. p. 2).

[LFA21] Xuze Liu, Abbas Fotouhi et Daniel J. Auger. « Formula-E race strategy development using distributed policy gradient reinforcement learning ». In : *Knowledge-Based Systems* 216 (2021), p. 106781. issn : 0950-7051. doi : https://doi.org/10.1016/j.knosys.2021.106781.

URL : https://www.sciencedirect.com/science/article/pii/S0950705121000447 (cf. p. 7).

[Sch] Philipp SCHAEFER. *FastF1*. URL : https://docs.fastf1.dev/api.html. (accessed : 20.07.2023) (cf. p. 10, 11).

[scia] SCIKIT-LEARN. *Random Forests*. URL : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#. (accessed : 01.07.2023) (cf. p. 24).

[scib] SCIKIT-LEARN. *sklearn.utils.class_weight.compute_class_weight*. URL : https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html. (accessed : 26.07.2023) (cf. p. 33).

Annexes

Annexe A

Première annexe

Table A.1 – *Paramètres de la simulation utilisées pour les examples en introduction*

N	70
$t_{optimal}$	95
$t_{penalite}$	0.06
p_{depart}	110
$d_{performance}(tendre)$	0
$d(tendre)$	0.2
$\delta d(tendre)$	0.1
$d_{performance(medium)}$	0.75
$d(medium)$	0.08
$\delta d(medium)$	0.08
$d_{performance}(dur)$	1.5
$d(dur)$	0.06
$\delta d(dur)$	0.07
$t_pitstop$	25
$t_pitstop_safety_car$	15

Table A.2 – *Flux "Live Timing" de la voiture 31 pendant le Grand Prix de Monaco 2021*

LapNumber	Driver	LapTime	Stint	TotalLaps	Compound	New	TyresNotChanged	Time	LapFlags
2	31	00 :01 :20.154000	0	-	-	-	-	00 :35 :51.209000	1
3	31	00 :01 :19.421000	0	-	-	-	-	00 :37 :05.565000	-
-	31		0	3	-	-	-	00 :37 :05.940000	-
4	31	00 :01 :18.494000	0	-	-	-	-	00 :38 :24.008000	-
-	31		0	4	-	-	-	00 :38 :25.951000	-
5	31		0	-	-	-	-	00 :39 :42.570000	-
-	31		0	5	-	-	-	00 :39 :45.937000	-
-	31		0	6	-	-	-	00 :41 :05.970000	-
-	31		0	7	-	-	-	00 :42 :20.948000	-
8	31	00 :01 :17.967000	0	-	-	-	-	00 :43 :37.668000	-
-	31		0	8	-	-	-	00 :43 :40.948000	-
9	31		0	-	-	-	-	00 :44 :56.051000	-
-	31		0	9	-	-	-	00 :45 :00.953000	-
-	31		0	10	-	-	-	00 :46 :15.975000	-
11	31	00 :01 :17.770000	0	-	-	-	-	00 :47 :32.098000	-
-	31		0	11	-	-	-	00 :47 :35.967000	-
12	31	00 :01 :17.677000	0	-	-	-	-	00 :48 :49.775000	-

Annexe B

Deuxième annexe

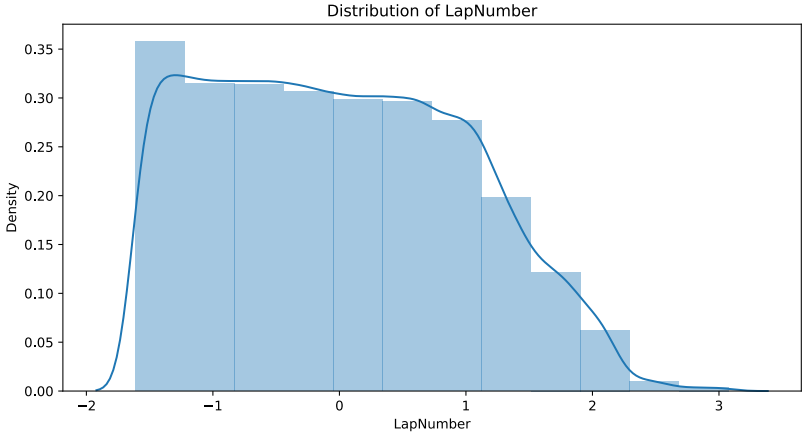

Figure B.1 – *Distribution de la colonne LapNumber. On remarque que la distribution est très uniforme, car la variable augmente linéairement au fil de la course. On remarque par contre qu'elle est asymétrique, car certaines voitures ne terminent pas la course et certaines courses sont plus longues que d'autres.*

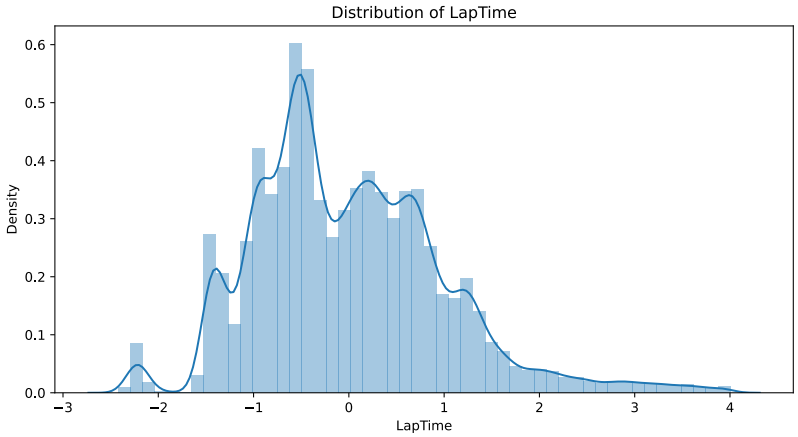

Figure B.2 – *Distribution de la colonne LapTime.*

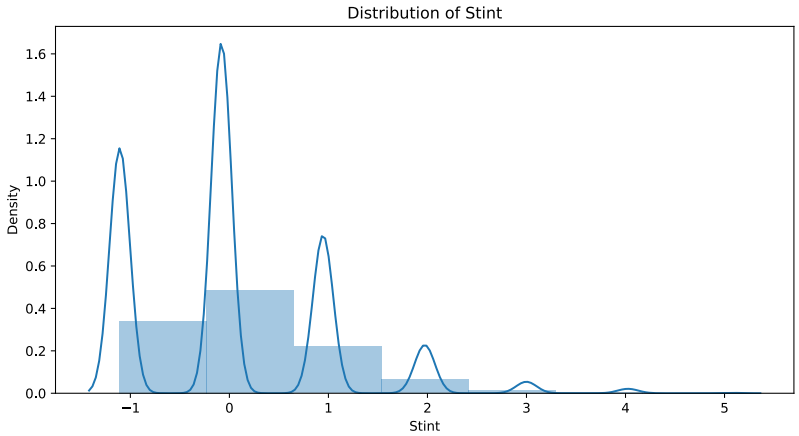

Figure B.3 – *Distribution de la colonne Stint. Elle est similaire à celle de TyreLife pour les mêmes raisons.*

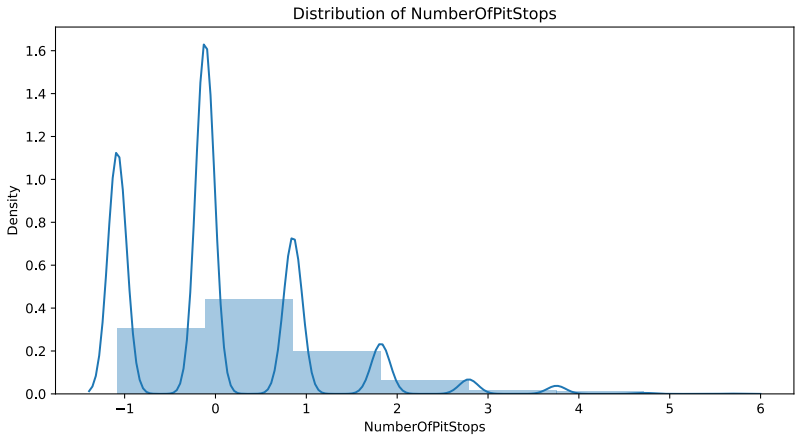

Figure B.4 – *Distribution de la colonne NumberOfPitStops. Elle est identique à Stint car NumberOfPitsStops = Stint - 1.*

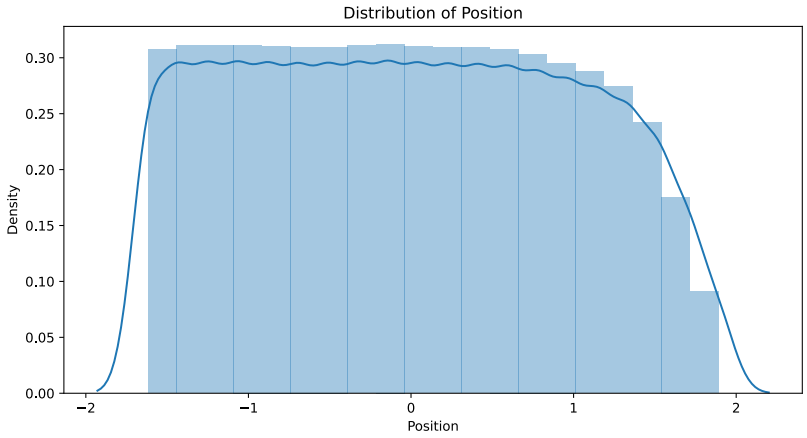

Figure B.5 – *Distribution de la colonne Position. Elle est uniforme avec une asymétrie à droite en raison des voitures qui se retirent avant la fin des courses.*

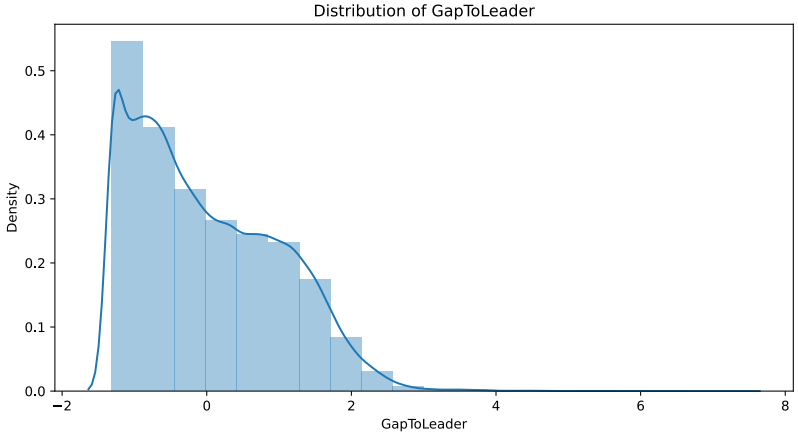

Figure B.6 – *Distribution de la colonne GapToLeader, La distribution est très concentrée autour de 0 étant donné que le leader de la course est toujours à 0. La distribution est très asymétrique à droite, ce qui est logique, car la plupart des pilotes sont assez près du leader au début de la course.*

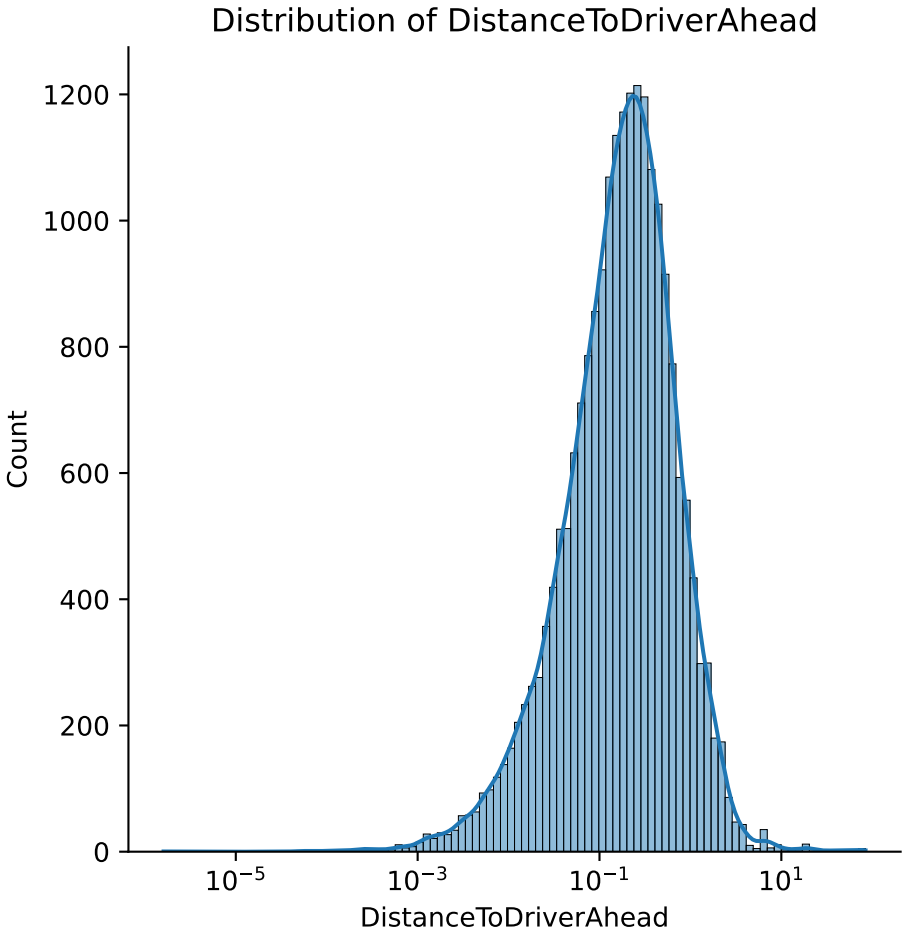

Figure B.7 – *Distribution de la colonne DistanceToDriverAhead.*

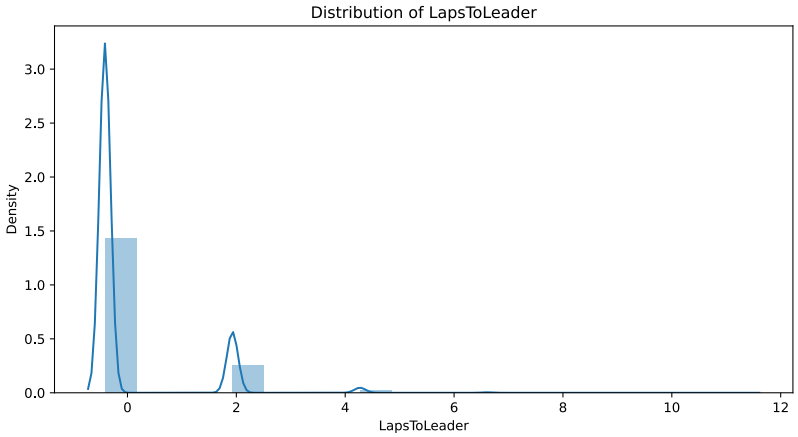

Figure B.8 – *Distribution de la colonne LapsToLeader*

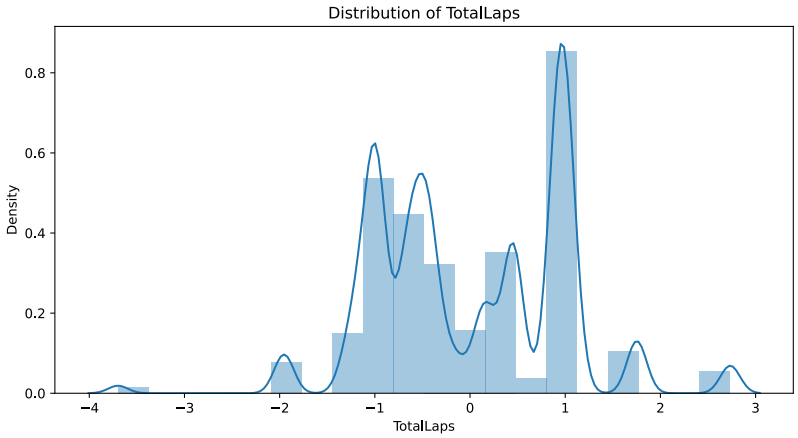

Figure B.9 – *Distribution de la colonne TotalLaps*

Annexe C

Troisième annexe

Table C.1 – *Résultats du GridSearchCV pour le modèle RandomForestClassifier optimisant le score F1.*

rang	class weight	max depth	max features	n estimators	score
1	balanced subsample	5.0	sqrt	2000	0.130011
2	balanced	5.0	sqrt	500	0.129973
3	balanced subsample	5.0	sqrt	500	0.129856
4	balanced subsample	5.0	log2	2000	0.129681
5	balanced	5.0	log2	2000	0.129667
6	balanced	5.0	sqrt	2000	0.129547
7	balanced subsample	5.0	sqrt	1000	0.129496
8	balanced	5.0	sqrt	1000	0.129489
9	balanced subsample	5.0	log2	1000	0.128940
10	balanced subsample	5.0	log2	100	0.128932
11	balanced	5.0	log2	1000	0.128561
12	balanced subsample	5.0	sqrt	100	0.128288
13	balanced subsample	5.0	log2	500	0.127931
14	balanced	5.0	log2	500	0.127563
15	balanced	5.0	sqrt	100	0.127500
16	balanced	5.0	log2	100	0.126691
17	balanced subsample	5.0	None	2000	0.122646
18	balanced subsample	5.0	None	1000	0.122618
19	balanced	5.0	None	2000	0.122528
20	balanced	5.0	None	1000	0.122481
21	balanced subsample	5.0	None	100	0.122207
22	balanced subsample	5.0	None	500	0.122206
23	balanced	5.0	None	500	0.122176
24	balanced	5.0	None	100	0.121999
25	balanced	20.0	log2	100	0.090315
26	balanced	20.0	log2	500	0.087494
27	balanced	20.0	None	100	0.086376
28	balanced subsample	20.0	None	100	0.084230
29	balanced subsample	20.0	None	500	0.083750
30	balanced subsample	20.0	log2	2000	0.083263
31	balanced	20.0	None	1000	0.083231
32	balanced subsample	20.0	None	1000	0.083107
33	balanced subsample	20.0	None	2000	0.082549
34	balanced subsample	20.0	log2	500	0.082350
35	balanced	20.0	None	2000	0.082121
36	balanced	20.0	None	500	0.081996
37	balanced	20.0	log2	2000	0.081599