**Projet tutoré : Data Science**

**“ Analyse des tirs de joueurs NBA ”**

**Réalisé par :**

* Najat BOUTABRATINE
* Sarah BOUKRIS
* Philippe WEISSBECK

**Encadré par :**

* Kalomé [BOTOWAMUNGU](mailto:kalome.b@datascientest.com)

**Institut :**

**DATASCIENTEST**

Formation continue – Promotion FEV24



Table des matières

[1. Adaptation Daframe 3](#_Toc179295403)

[2.Rapport de modélisation 3](#_Toc179295404)

[LightGBM 4](#_Toc179295405)

[XGBOOST 7](#_Toc179295406)

[Réseaux de neurones 9](#_Toc179295407)

[3. Ouverture 13](#_Toc179295408)

[4. Problèmes rencontrées 13](#_Toc179295409)

# Adaptation Daframe

Lors de nos premiers tests de modélisation, nous nous sommes rapidement rendus compte que le dataframe constitué devait être retravaillé, certaines données étaient manquantes et d’autres avaient une corrélation directe avec la target.

En effet un leak avait été introduit par inadvertance dans les features et nous permettait d’obtenir un très bonne accuracy (>0,8) pour ce type de problématique. Dans les fichiers playByplay, nous avions récupéré les informations ASSIST, qui permet de savoir si une passe décisive a été effectuée, et BLOCK, qui permet de savoir si un tir a été bloqué avant d’atteindre le panier. Le problème de ces features est que leur présence induit directement la réussite d’un tir ou non. Lorsque l’on peut rattacher une passe décisive au tir, celui-ci est forcément réussi. *A contrario*, lorsque le tir est contré, celui-ci est forcément raté. Aussi, nous sommes revenus en arrière dans la constitution de notre dataframe et avons retiré ces variables.

Par la suite nous avons décidé d’introduire l’équipe adverse dans le dataframe. En effet certaines équipes étant par nature meilleures en défense, cela peut réduire les performances au tir d’un joueur (notamment grâce à des contres). Pour cela nous avons calculé le champs vs\_team de sorte que si le joueur joue à domicile, alors vs\_team sera égale à AwayTeam, sinon il sera égal à HomeTeam.

Enfin les performances d’un joueur peuvent être différentes d’un match à un autre, selon sa conditions physique (blessure, fatigue, etc.), selon sa condition morale, ou encore selon l’équipe adverse pouvant avoir une tactique défensive contre un joueur précis. Pour cela nous avons également décidé de garder le champs GameID qui nous permet d’identifier un match précis.

Après la reconstitution de ce dataframe nous avons pu passer dans un premier temps à une vision statistique des données, puis à la modélisation.

# Rapport de modélisation

Le dataframe étant désormais finalisé, nous avons pu nous lancer dans la phase de modélisation.

Modéliser ce type de problème est intrinsèquement compliqué. La réussite d’un tir étant par nature difficile à modéliser. Certaines composantes semblent intuitives dans sa réussite, par exemple, la distance par rapport au panier. D’autres seront tout de suite plus abstraites (fatigue du joueur, confiance au moment du tir, …).

Par ailleurs, notre sélection du top 20 des joueurs, ainsi que la création d’un certain nombre de features a abouti à une volumétrie non négligeable, d’environ 300 000 lignes pour une trentaine de colonnes. Aussi, le choix du modèle a dû intégrer ces critères pour donner de bons résultats tout en maitrisant la durée du traitement.

Après plusieurs tentatives pour analyser lesquels donnaient des résultats probants, trois modèles ont été retenus avant optimisation des hyperparamètres.

* LightGBM
* XGBoost
* Réseaux de neurones

En lieu et place du train\_test\_split habituel, nous avons utilisé une approche de validation croisée appelée StratifiedKfold. Celle-ci prend en compte la répartition des classes dans les ensembles de données. L'objectif principal est de garantir que chaque fold (ou pli) de l'ensemble de données utilisé pour l'entraînement et la validation contient une proportion similaire de chaque classe par rapport à l'ensemble de données global.

Dans une validation croisée K-Fold classique, les données sont simplement divisées en K sous-ensembles de manière aléatoire, ce qui peut entraîner une distribution déséquilibrée des classes dans certains sous-ensembles. Cela peut affecter les performances des modèles, notamment sur des jeux de données déséquilibrés, où certaines classes sont sous-représentées.

Enfin, pour l’optimisation des hyperparamètres, nous avons utilisé une librairie intitulée Optuna. Optuna est une bibliothèque d'optimisation d'hyperparamètres open-source, conçue pour être flexible, efficace et facile à utiliser. Elle permet d'automatiser le processus de recherche des meilleurs hyperparamètres pour les modèles de machine learning, en maximisant (ou minimisant) une métrique donnée, comme la précision, la perte ou toute autre métrique d'évaluation.

Optuna repose sur un concept de **"study"** (étude), qui est un ensemble d'essais (trials). Chaque essai correspond à une exécution de modèle avec un ensemble spécifique d'hyperparamètres. L'utilisateur définit un objectif (une fonction à optimiser) et Optuna se charge de tester différents ensembles d'hyperparamètres pour trouver ceux qui optimisent au mieux cet objectif.

## LightGBM

LightGBM est une implémentation de gradient boosting développée par Microsoft, optimisée pour l'efficacité et la rapidité. Il utilise une approche basée sur les histogrammes pour réduire la complexité temporelle du processus de croissance des arbres. LightGBM est particulièrement adapté pour les grands ensembles de données avec de nombreuses caractéristiques. Parmi les trois modèles retenus, c’est celui qui offre le meilleur couple Accuracy / rapidité.

**Caractéristiques principales :**

* **Efficacité en mémoire et rapidité d'entraînement :** Grâce à l'approche par histogramme, LightGBM peut former des modèles plus rapidement et avec une consommation de mémoire moindre par rapport à d'autres méthodes de boosting.
* **Support de données dispersées (sparse data) :** LightGBM gère efficacement les données ayant beaucoup de valeurs manquantes ou des données catégorielles.
* **Apprentissage en profondeur (Leaf-wise growth) :** Contrairement aux autres implémentations de boosting qui utilisent une approche de niveau (level-wise), LightGBM construit les arbres de manière à maximiser la réduction de perte (leaf-wise growth). Cela permet une meilleure performance, bien qu'il faille faire attention à éviter l'overfitting.

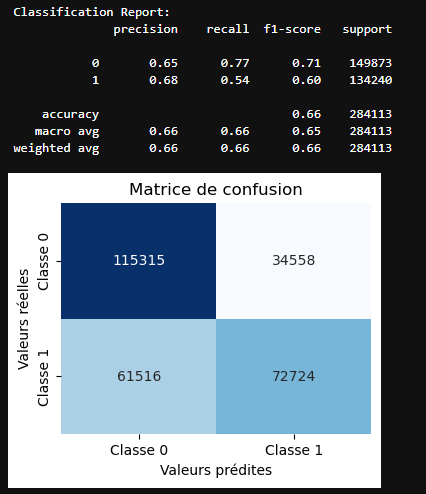
**Applications :** LightGBM est souvent utilisé pour des tâches de classification (binaire, multiclass), de régression, de rang (ranking) et d'autres problèmes en compétition de machine learning.

Nous avons lancé 60 études du modèle afin de définir les hyperparamètres, la sortie est la suivante :

[I 2024-07-29 14:52:56,767] Trial 59 finished with value: 0.6601845080168085 and parameters: {'num\_leaves': 23, 'max\_depth': 12, 'learning\_rate': 0.08041752736856972, 'n\_estimators': 302, 'subsample': 0.9411201509994473, 'colsample\_bytree': 0.5932063689592868}. Best is trial 26 with value: 0.6618458163680879.

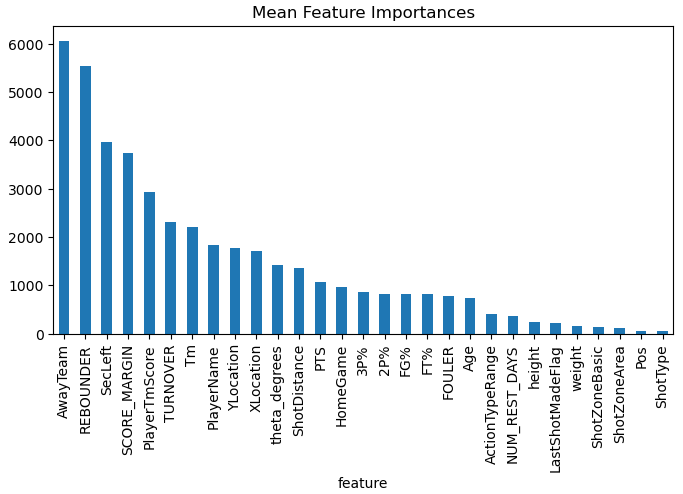
Best hyperparameters: {'num\_leaves': 113, 'max\_depth': 19, 'learning\_rate': 0.03295825041204072, 'n\_estimators': 389, 'subsample': 0.8162306085513036, 'colsample\_bytree': 0.7835908415381942}

En lançant notre modèle final, les résultats sont les suivants :



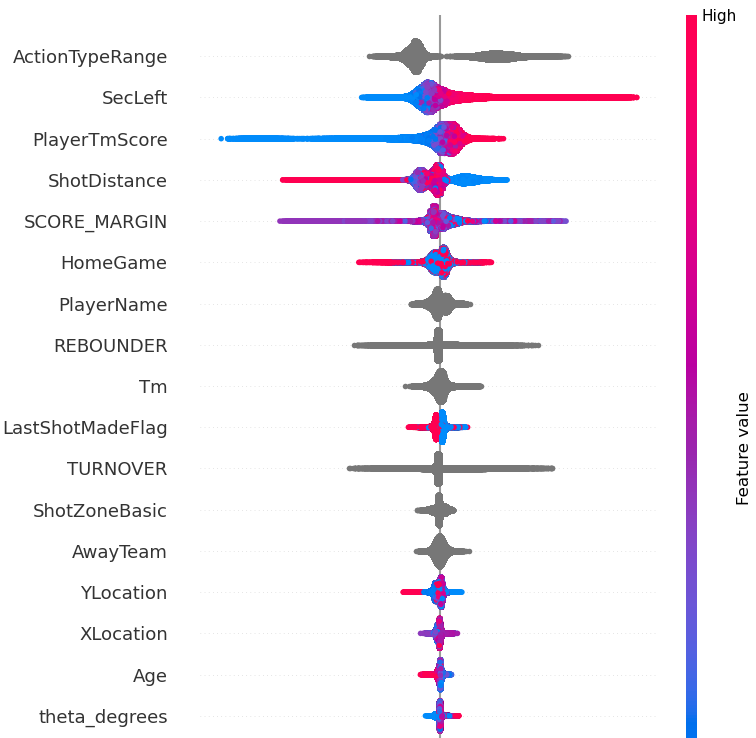
L’accuracy, que nous souhaitons optimiser est de 0,66. Le modèle donne des résultats satisfaisants sur le ShotMadeFlag = 0 avec un f1-score de 0.71 et légèrement inférieur sur la variable ShotMadeFlag =1 avec un score de 0.6.

Les features importance sont les suivantes :



Nous avons ensuite analysé les shap values.

Les **SHAP values** (SHapley Additive exPlanations) sont une méthode d'interprétabilité des modèles de machine learning. Elles permettent d'expliquer les prédictions de modèles complexes en quantifiant l'importance de chaque caractéristique (feature) sur une prédiction donnée. SHAP est basé sur la théorie des jeux, et plus spécifiquement sur la valeur de Shapley, un concept qui provient de l'analyse des contributions équitables des joueurs dans un jeu collaboratif.



L’analyse des Shap values corrobore par exemple la figure 2.15.

Pour la variable SecLeft, plus celle-ci est faible (la fin du match est proche), plus la feature value est faible (probabilité de rater le tir augmente). Cela est dû à la fatigue du joueur en fin de match, la pression lors du tir si le match est serré, etc.

Dans l’autre sens, si nous regardons la variable ShotDistance, le phénomène inverse est observé. Plus cette variable est faible (le tir est proche du panier), plus la probabilité de réussir le tir augmente (feature value tend vers le rouge).

## XGBOOST

**Présentation :** XGBoost est un autre modèle de gradient boosting très populaire, souvent utilisé dans des compétitions de machine learning comme celles de Kaggle. Il est connu pour ses performances élevées, sa robustesse et sa flexibilité. XGBoost améliore les implémentations classiques du boosting grâce à des optimisations telles que le calcul distribué, la régularisation et la gestion efficace de la mémoire.

**Caractéristiques principales :**

* **Régularisation :** XGBoost introduit une régularisation explicite (L1 et L2) pour éviter le surapprentissage (overfitting), ce qui lui permet d'être plus robuste sur des jeux de données bruyants.
* **Gestion des valeurs manquantes :** XGBoost a un mécanisme intégré pour gérer les valeurs manquantes, en adaptant automatiquement les splits d'arbres.
* **Parallélisation et calcul distribué :** L'algorithme supporte l'entraînement en parallèle et la distribution du calcul sur plusieurs machines, ce qui le rend idéal pour les grands ensembles de données.
* **Optimisation des performances :** XGBoost utilise des techniques d'optimisation basées sur des approximations par gradient pour accélérer le processus de construction des arbres.

**Applications :** XGBoost est utilisé dans des problèmes de classification, régression, rang et survie, et est réputé pour ses performances sur les jeux de données structurés/tabulaires.

Les résultats en sorti d’Optuna sont les suivants :

[I 2024-07-29 16:06:43,167] Trial 19 finished with value: 0.6478010664695634 and parameters: {'booster': 'dart', 'lambda': 0.545250915868964, 'alpha': 0.1862682433889052, 'max\_depth': 14, 'eta': 0.0732125737589444, 'gamma': 0.34258422729834104, 'subsample': 0.8260454036152651, 'colsample\_bytree': 0.7269577376450733}. Best is trial 14 with value: 0.6507224187388909.

Best hyperparameters: {'booster': 'dart', 'lambda': 0.7611599117364898, 'alpha': 0.2694478668585406, 'max\_depth': 12, 'eta': 0.08760207662576841, 'gamma': 0.24105373659758264, 'subsample': 0.7806768513525376, 'colsample\_bytree': 0.9515248736781333}

Les features importances sont les suivantes :

Une image contenant texte, capture d’écran, diagramme, Tracé

Description générée automatiquement

Les résultats sont assez proches de ceux obtenus grâce à LightGBM :

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

## Réseaux de neurones

**Présentation :** Les réseaux de neurones sont des modèles d'apprentissage automatique inspirés du fonctionnement du cerveau humain. Ils sont composés de couches de neurones artificiels (ou unités) organisées en couches d'entrée, couches cachées et couches de sortie. Chaque neurone est une unité de calcul qui reçoit des entrées pondérées, applique une fonction d'activation, et produit une sortie transmise aux neurones des couches suivantes.

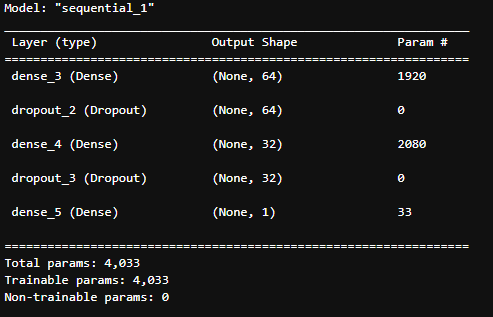
**Caractéristiques principales :**

* **Flexibilité et expressivité :** Les réseaux de neurones peuvent modéliser des relations complexes et non linéaires entre les variables. Ils sont particulièrement puissants pour des tâches telles que la reconnaissance d'images, le traitement du langage naturel (NLP) et les séries temporelles.
* **Capacité de généralisation :** Avec suffisamment de données et une architecture appropriée, les réseaux de neurones peuvent apprendre des représentations générales des données, ce qui leur permet de bien généraliser sur des exemples non vus.

**Applications :** Les réseaux de neurones sont utilisés dans une variété de domaines : vision par ordinateur, reconnaissance vocale, traduction automatique, robotique, médecine, finance, et bien d'autres. Ils sont devenus un standard pour les tâches nécessitant une compréhension fine et complexe des données, comme l'identification d'objets dans des images ou la prévision de séquences.

A défaut d’utiliser Optuna, nous avons essayé différents modèles afin d’analyser les résultats.

Dans un premier temps, nous avons implémenté un modèle relativement simple :

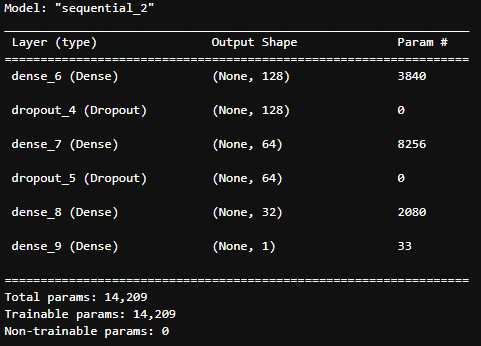


Après 20 Epochs, les résultats semblent relativement satisfaisant :

Epoch 20/20

5683/5683 [==============================] - 8s 1ms/step - loss: 0.6338 - accuracy: 0.6468 - val\_loss: 0.6240 - val\_accuracy: 0.6544

Ensuite, nous avons densifié notre modèle en ajoutant des neurones dans les premières couches ainsi qu’une couche dense en fin de modèle :



Après 50 Epochs, les résultats sont les suivants :

Epoch 50/50

5683/5683 [==============================] - 11s 2ms/step - loss: 0.6214 - accuracy: 0.6562 - val\_loss: 0.6188 - val\_accuracy: 0.6604

Ceux-ci sont légèrement meilleurs, mais la différence n’est pas très importante.

Nous avons ensuite lancer un troisième modèle :

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

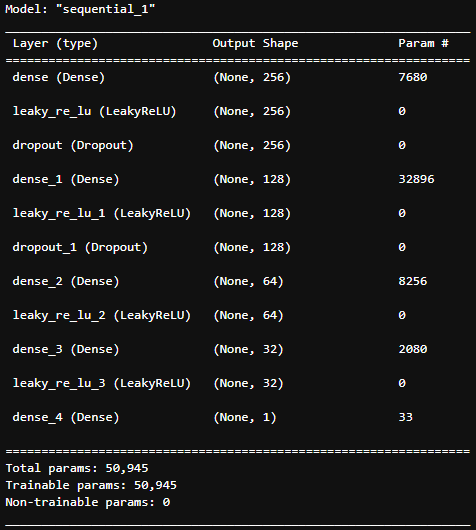
Nous y avons notamment inclus des couches **LeakyReLU** (Leaky Rectified Linear Unit) qui sont un type de fonction d'activation non linéaire. Elles sont utilisées pour introduire de la non-linéarité dans les réseaux de neurones, ce qui permet aux modèles d'apprendre des relations plus complexes dans les données et éviter le surapprentissage.

Après 50 Epochs, les résultats sont quasi identiques au modèle précédent :

Epoch 50/50

5683/5683 [==============================] - 11s 2ms/step - loss: 0.6217 - accuracy: 0.6556 - val\_loss: 0.6196 - val\_accuracy: 0.6620

Enfin, nous avons essayé un dernier modèle plus lourd, avec plus d’Epochs :



Aussi, nous avons implémenté un callback early\_stopping qui va surveiller la val\_loss et arrêter le modèle si celui-ci ne s’améliore pas après 10 epochs :



Nous avons ensuite lance 100 Epochs, afin d’essayer d’améliorer encore les résultats de notre modèle. Les résultats sont les suivants :

Epoch 53/100

5683/5683 [==============================] - 27s 5ms/step - loss: 0.6239 - accuracy: 0.6610 - val\_loss: 0.6214 - val\_accuracy: 0.6653

L’accuracy s’améliore légèrement mais semble stagner autour de 0,66.

# Ouverture

Au cours de ce projet, un travail conséquent a été mené pour explorer et évaluer divers modèles de machine learning. Chaque modèle a été ajusté en optimisant minutieusement les hyperparamètres afin d'obtenir des performances optimales avec le jeu de données disponible. Cette démarche nous a permis d’atteindre un niveau d’accuracy satisfaisant, tout en identifiant les limites inhérentes aux données actuelles et aux modèles testés.

Néanmoins, dans l’optique de poursuivre cette amélioration, plusieurs pistes d’enrichissement peuvent être envisagées. Tout d'abord, l'ajout de nouvelles variables explicatives serait une direction prometteuse. Actuellement, les modèles se basent uniquement sur les données quantifiables, mais de nombreux facteurs qui influencent la réussite d’un tir au basket sont complexes à transcrire numériquement. Par exemple, la fatigue du joueur au moment de l’action, la pression exercée par le défenseur, ou encore l’ambiance générale dans la salle sont des éléments déterminants, mais difficiles à capturer avec les variables traditionnelles. Le défi consisterait donc à développer des indicateurs plus avancés ou à collecter des données supplémentaires qui permettent de mieux modéliser ces aspects contextuels.

Par ailleurs, l’adoption de modèles plus lourds et sophistiqués ou des modèles basés sur des architectures plus complexes, pourrait aussi constituer une piste d'amélioration. Ces modèles pourraient être en mesure de capturer des relations plus subtiles dans les données et d’améliorer ainsi l’accuracy, bien qu’ils requièrent une puissance de calcul plus importante et un travail supplémentaire pour éviter le surapprentissage.

En somme, bien que le travail accompli ait abouti à des résultats significatifs, les perspectives d'amélioration restent nombreuses et variées, qu'il s'agisse d’enrichir les données disponibles ou de recourir à des modèles plus avancés.

# Problèmes rencontrées

Lors de la réalisation de ce projet nous avons pu rencontrer quelques difficultés et doutes.

Lors de nos tests de modélisation, nous avons constitué différents dataframe, jusqu’à arriver à celui qui a été sélectionné. L’un des dataframe (non sélectionné) nous a permis d’avoir une accuracy « un peu trop bonne » et a mis en doute nos calculs.

Les features étant :

Une image contenant texte, menu, capture d’écran

Description générée automatiquement

Dans ce dataframe les champs Ylocation et Xlocation ont été supprimés, en effet l’angle ayant été calculé et la distance connue, ces informations peuvent être considéré comme redondantes.

L’angle est aussi calculé en valeur absolu, et le distinguo entre la droite et la gauche, ou le centre est connu par le champs zone.

La distance exacte n’est pas non plus connue, la feature ShotZoneRange nous permet d’avoir les différentes tranches. Et les pourcentages de blocages et de passes décisives pour un joueur au cours d’une saison ont été introduit.

Avec ce dataframe et le model LightGBM nous obtenons une accuracy de 0,88 qui nous parait un peu trop bonne pour être valable.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, Rectangle, Police

Description générée automatiquement

Avec les mean features :

Une image contenant texte, capture d’écran, Tracé, ligne

Description générée automatiquement

De plus parmi les difficultés rencontrées, un test mathématique a été effectué sur les données. En effet nous avons les champs Xlocation et YLocation constituant l’abscisse et l’ordonné par rapport au panier, ainsi que le champs shotDistance correspondant à la distance avec le panier.

Grace à un calcul simple suivant le théorème de Pythagore, Xlocation^2 + YLocation^2 devrait être égale à shotDistance^2. Cependant cette règle n’est pas respectée par quelques 3000 lignes, ce qui met en doutes la cohérence de certaines données.

Par ailleurs, les contraintes personnelles / professionnelles de chacun ont contribué à des difficultés organisationnelles et logistiques. Certains membres ont pu avoir des problèmes personnels assez important, et le projet a dû s’organiser autrement.