Projet fil rouge : Analyse des tirs de joueurs NBA

Rapport Final

A. Brahimi, D. Labdouni, W. Maillot, T. Martinez Ostormujof

Table des matières

1 Introduction

Le sport américain est friand de statistiques et la National Basketball Association (NBA), la ligue de basketball la plus compétitive au monde ne déroge pas à cette règle. Bien au contraire, elle est à l'avant-garde de la mise en application des informations récoltées par les données dans les domaines du jeu, du recrutement,etc. On parle même de mariage réussi entre la NBA et la data tant elle y est omniprésente (analyse de performances, prévision de résultats, évaluation de la forme physique des joueurs, aide à la prise de décision diverses et variées,etc).

Du fait de l'accessibilité élevée des données mise à disposition par la NBA (qui propose un package open-source d'APIs : nba_api), une multitude de projets fleurissent sur internet autour de ces données, qu'il s'agisse de fans de basket-ball qui veulent soutirer quelques informations sur leur sport favori ou des scientifiques de la donnée de tous niveaux qui s'essaient sur des jeux de données riches qui offrent de nombreuses possibilités d'exploration.

Ainsi nous nous plaçons dans la lignée de ces projets en ayant pour objectif d'élaborer un modèle de classification qui puissent nous permettre de déterminer la probabilité d'un tir d'être réussi ou non. Il s'agit donc d'un problème de classification.

Pour y parvenir, nous nous serons appuyés essentiellement (mais pas seulement) sur un jeu de données librement disponible qui répertorie les tirs (réussis ou non) des joueurs, leurs localisations sur le terrain, ainsi que d'autres variables et ce, sur une période de plus de vingt saisons (1997 – 2020). Nous enrichirons ces données de variables complémentaires provenant d'autres datasets et qui nous ont semblé pertinentes pour aider à améliorer la qualité et la performance de notre modèle.

Tous les quatre avons une appétence et une connaissance certaine du sport et en particulier de la NBA, nous profitons de ce goût pour ce championnat pour développer notre projet fil rouge autour du monde passionnant qu'est le basketball professionnel. Cela nous offre l'opportunité d'appliquer certains des concepts appris durant notre formation chez DataScientest (promotion de mars 2023).

2 Description des variables explicatives et visualisation des données

2.1 NBA Shot Locations 1997 - 2020.csv dataset

Comme décrit dans l'introduction, l'objectif principal de ce projet est la prédiction des performances de tir des joueurs de la NBA. Pour ce faire, nous avons principalement utilisé un ensemble de données appelé NBA Shot Locations 1997 - 2020.csv qui contient des informations sur tous les tirs effectués dans la NBA entre 1997 et 2020. Une description et une analyse de cet ensemble de données suivent.

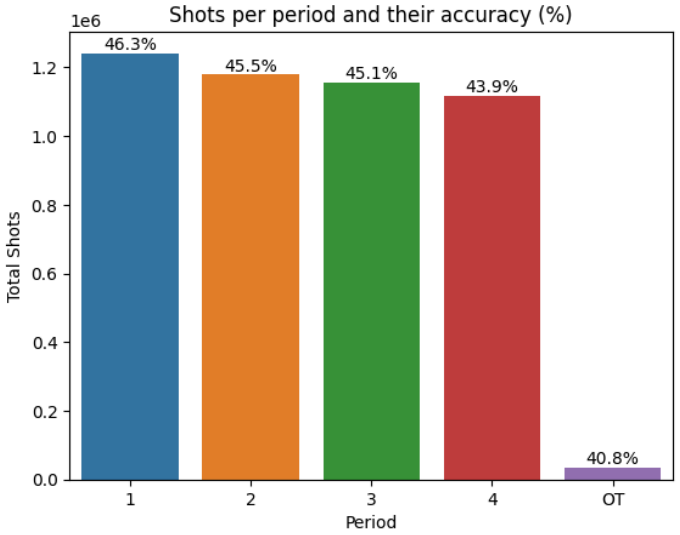
L'ensemble de données contient un total de 22 colonnes et 4729512 lignes. Les colonnes représentent les variables liées aux tirs et chaque ligne représente un tir unique. Il n'y a pas de valeurs manquantes, de sorte que chaque colonne contient exactement le même nombre d'éléments. Dans ce qui suit, nous décrirons les variables de l'ensemble de données, en soulignant celles qui seront prises en compte pour l'entraînement du modèle.

2.1.1 Variable cible

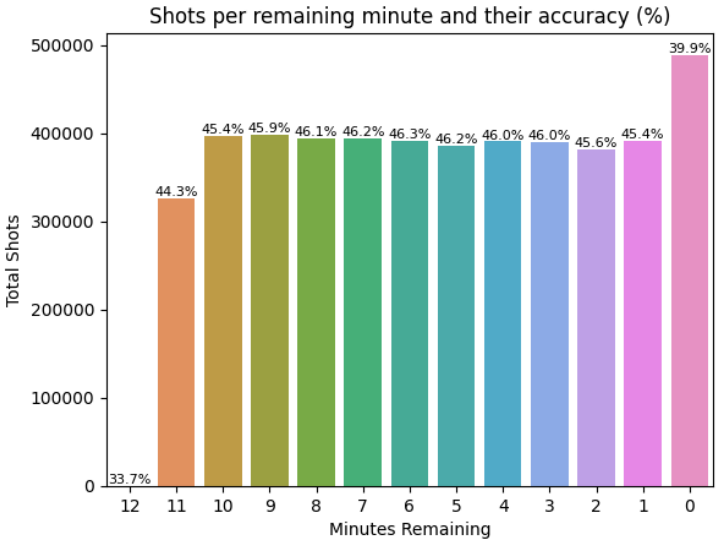
- **Shot Made Flag :** variable de type int64. Elle contient un 0 pour les tirs manqués et un 1 pour les tirs réussis.

2.1.2 Variables explicatives pertinentes pour entraîner notre modèle

- **Player ID :** variable de type int64 qui identifie le joueur qui a effectué le tir. Elle a 2152 valeurs uniques. Cette variable doit être prise en compte car le résultat du tir dépend fortement de chaque joueur.
- **Period :** est une variable catégorielle de type int64 qui indique la période du match dans laquelle le tir a été effectué. Elle contient des valeurs allant de 1 à 8. Les valeurs de 1 à 4 indiquent le temps réglementaire et les valeurs de 5 à 8 indiquent les prolongations. Étant donné que les prolongations ne sont pas très fréquentes et qu'elles sont plus courtes (moins de tirs), nous regroupons tous les tirs des périodes 5 à 8 dans un seul groupe appelé OT (Overtime). Cette variable a un effet sur la précision et le nombre de tirs, comme le montre le graphique ci-dessous, elles diminuent au fur et à mesure des périodes.

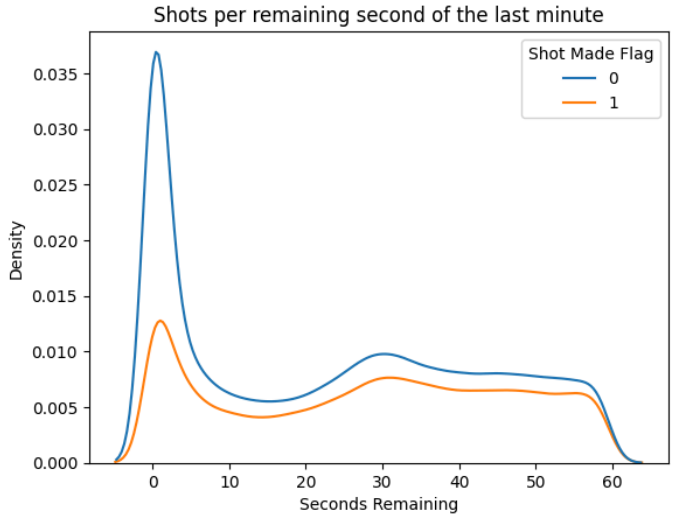


- **Minutes Remaining :** est une variable de type int64 qui indique les minutes restantes jusqu'à la fin de la période en cours. Elle contient des valeurs comprises entre 0 et 12, car les périodes de temps normales durent 12 minutes (les périodes de prolongation durent 5 minutes). Cette variable sera prise en compte car elle a un effet à la fois sur le nombre de tirs et sur la précision. Comme le montre le graphique ci-dessous, à 11 minutes de la fin de la période, le nombre de tirs et la précision sont faibles. À partir des 10 minutes restantes, le nombre de tirs et la précision augmentent et restent constants jusqu'à ce qu'il ne reste plus qu'une minute. Dans la dernière minute, on observe une nette augmentation du nombre de tirs et une nette diminution de la précision. Il convient de noter qu'un faible pourcentage (101 tirs) a été trouvé à 12 minutes de la fin du match. Ces lignes seront éliminées car il ne peut y avoir de tirs avant le début du match.

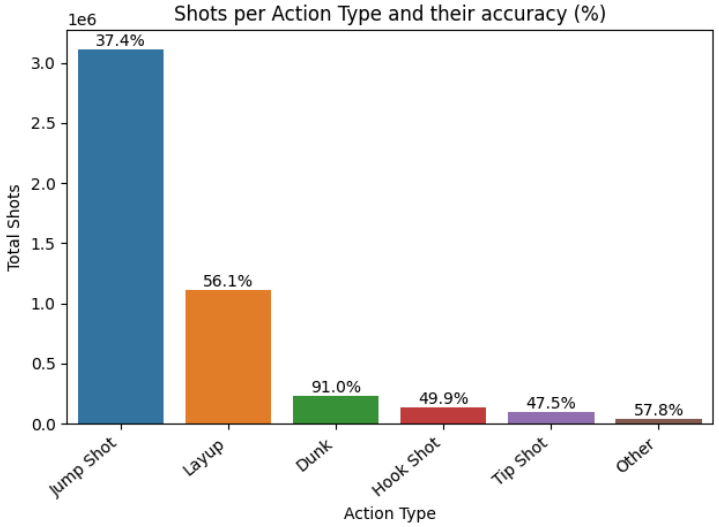


- **Seconds Remaining :** est une variable de type int64 indiquant les secondes restantes jusqu'à la fin de la minute en cours. Cette variable sera prise en compte car elle a un effet à la fois sur le nombre de tirs et sur la précision si l'on considère les secondes restantes dans la dernière minute de la période en cours. Le graphique ci-dessous montre le nombre de tirs par rapport aux secondes restantes dans la dernière minute de la période en cours. Nous pouvons voir que le nombre de tirs manqués (ligne bleue) et le nombre de tirs réussis (ligne orange) suivent approximativement la même tendance jusqu'aux 10 dernières secondes et qu'à partir de là, le nombre de tirs manqués augmente de manière exponentielle tandis que le nombre de tirs réussis augmente beaucoup plus légèrement. Cela indique que la précision

diminue dans les dernières secondes de la période. Entre les variables Minutes Remaining et Seconds Remaining, nous créerons une seule variable continue appelée **Total Seconds Remaining**.

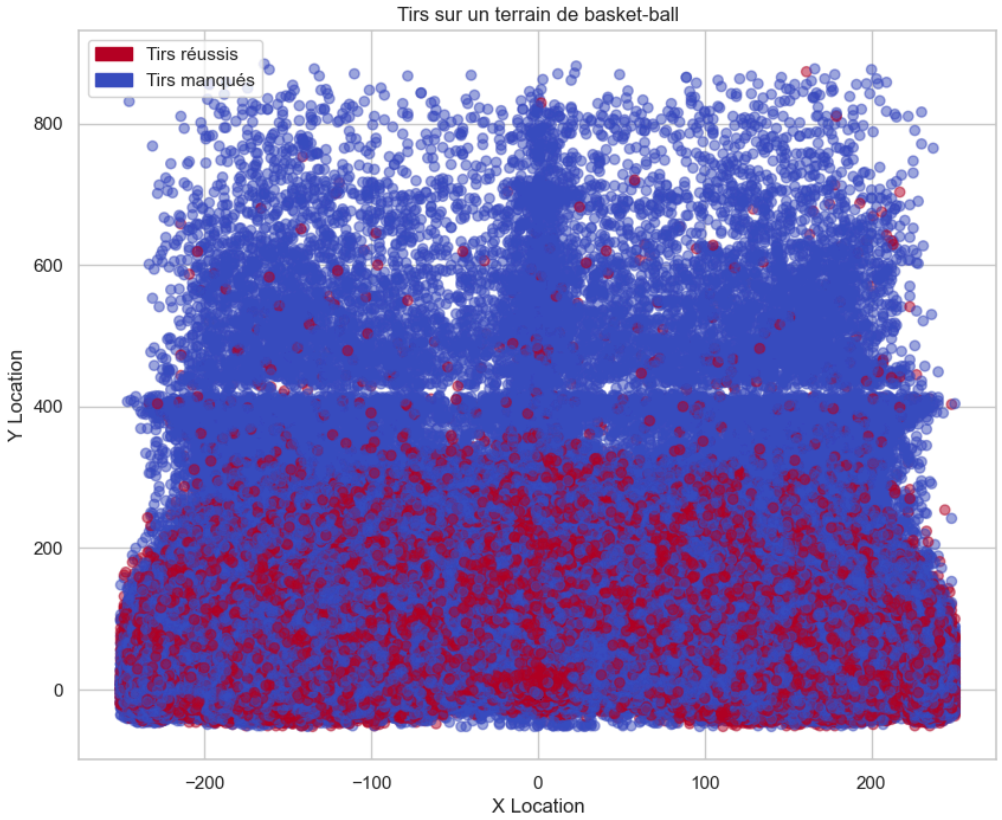


- **Action Type :** est une variable catégorielle de type string indiquant l'action avec laquelle le tir a été effectué. Elle contient un total de 70 types d'actions uniques. Tout d'abord, nous avons regroupé les 70 actions uniques en 7 catégories d'actions similaires, à savoir : Jump shot, Layup shot, Dunk shot, Hook shot, Tip shot, Other et No shot. La catégorie No shot (apparaissant dans 278 lignes) a été éliminée car elle correspond à des tirs effectués après qu'une faute a été signalée et n'est donc pas prise en compte. Il nous reste donc 6 catégories finales, dont l'effet sur le nombre de tirs et la précision est visible dans le graphique ci-dessous. Par conséquent, cette variable sera prise en compte pour la génération de notre ensemble de données.

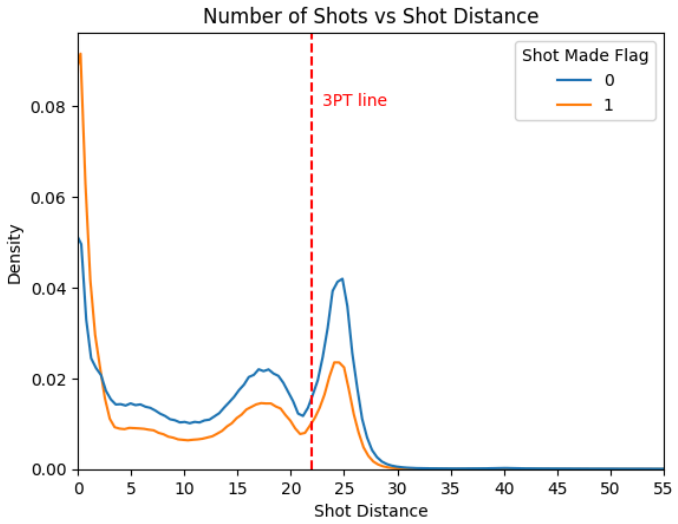


- **Shot Type :** est une variable catégorielle à deux valeurs uniques qui indique si le tir effectué vaut 2 ou 3 points. Il s'agit d'une variable très importante car elle affecte clairement le nombre de tirs et la précision, elle sera donc prise en compte lors de la création de notre ensemble de données. Comme cette variable dépend de la distance à laquelle le tir a été effectué, nous avons constaté quelques incohérences : certains tirs à longue distance valaient 2 points et certains tirs à courte distance valaient 3 points. Au total, 2113 tirs présentant ces caractéristiques ont été trouvés et éliminés.

- **X Location et Y location :** sont des variables int64 qui indiquent la position X (largeur du terrain) et Y (longueur du terrain) où le tir a été effectué. X est compris entre -250 et 250 et Y entre -52 et 884. La position X = 0 et Y = 0 indique l'emplacement du panier. Cette variable a un effet sur le nombre et la précision des tirs et sera donc prise en compte dans notre ensemble de données.



- **Shot Distance :** est une variable de type int64 qui indique la distance à laquelle le tir a été effectué. Elle va de 0 (position du panier) à 89. Alors que cette variable est exprimée en pieds, les positions X et Y sont exprimées en 10*pieds. Comme le montre le graphique ci-dessous, le nombre de tirs réussis (lignes orange) et manqués (ligne bleue) varie en fonction de la distance au panier, c'est pourquoi nous tiendrons compte de cette variable.



- **Season Type :** est une variable catégorielle indiquant si le tir a été effectué lors d'un match de la saison régulière ou des playoffs. Cette variable a un impact sur la précision des tirs et sera donc prise en compte.
- **2PT Field Goal_accuracy** et **3PT Field Goal_accuracy :** Ces variables ne figuraient pas à l'origine dans l'ensemble de données, nous les avons générées à partir de la variable Shot Type et Shot Made Flag pour chacun des joueurs. De cette façon, nous pouvons obtenir le pourcentage de doubles et de triples marqués par chaque joueur.

2.1.3 Variables qui ne seront pas prises en compte

- **Game ID et Game Event ID :** il s'agit de variables de type int64 qui identifient le match et l'événement du match où le tir a été effectué.
- **Player Name :** est une variable de type string contenant le prénom et nom de chaque joueur. Il y a 2143 valeurs uniques. Le nombre de valeurs uniques pour Player ID et Player Name n'est pas le même, ce point sera abordé dans la section *Exploration des nouvelles variables*.
- **Team ID** : est une variable de type int64 qui identifie l'équipe qui a tiré. Il y a 30 valeurs uniques.
- **Team Name :** variable catégorielle de type string qui identifie le nom complet de l'équipe qui a tiré. Elle possède 37 valeurs uniques. Le nombre de valeurs uniques pour Team ID et Team Name n'est pas le même. Cela s'explique par le fait que certaines équipes de la NBA ont changé de nom, mais conservent toujours le même ID d'équipe.
- **Shot Zone Basic et Shot Zone Area :** sont toutes deux des variables catégorielles qui indiquent la position dans laquelle le tir a été effectué dans différents groupes. Dans un premier temps, elles ne seront pas prises en compte car les variables X et Y Location pourraient expliquer la même information, en plus d'être des variables numériques qui facilitent l'entraînement.
- **Shot Zone Range :** il s'agit d'une variable catégorielle qui sépare la variable Shot Distance en 5 catégories. Pour l'instant, elle ne sera pas prise en compte car nous préférons utiliser la variable Shot Distance qui est une variable numérique.
- **Game Date :** est une variable de type int64 contenant la date à laquelle le tir a été exécuté au format AAAAMMJJ. Nous avons extrait le mois de l'année mais il n'y a pas de différence dans le résultat du tir en fonction du mois, il ne sera donc pas pris en compte.
- **Home Team et Away Team :** indiquent l'équipe locale et l'équipe extérieure du match au cours duquel le tir a été effectué. Ces deux variables sont des strings mais contiennent l'abréviation de l'équipe et non son nom complet tel qu'il apparaît dans la variable Team Name. Ces variables ne seront pas prises en compte, mais seront utilisées dans la section *Exploration de nouvelles variables* pour créer une nouvelle variable.

2.2 Exploration de nouvelles variables

Notre dataset principal concernant les shots de chacun des joueurs est très intéressant et semble nous fournir plusieurs variables explicatives pertinentes pour déterminer si un tir va être marqué ou non. Cependant, d'autres ensembles de données peuvent contenir des informations intéressantes pour prédire le résultat d'un tir.

2.2.1 Player_data.csv dataset

Le dataset *player_data.csv* couvre la période de 1981 à 2018 et nous fournit des variables concernant les caractéristiques de 4550 joueurs de NBA :

- **name :** est une variable de type string contenant le prénom et nom de chaque joueur.
- **year_start :** est une variable de type int64 indiquant l'année d'entrée du joueur à la NBA.
- **year_end :** est une variable de type int64 indiquant l'année de sortie du joueur à la NBA.

- **position :** est une variable catégorielle de type string indiquant la position du joueur. Elle contient 7 valeurs uniques.
- **height :** est une variable catégorielle de type string indiquant la taille du joueur. Elle a un format "pieds-pouces" par exemple 6-9.
- **weight :** est une variable de type int64 contenant le poids en livres de chaque joueur.
- **birth_date :** est une variable catégorielle de type string indiquant la date de naissance du joueur. Elle a un format “Mois Jour, Année” par exemple June 24, 1968.
- **college :** est une variable de type string qu’indique l’université par laquelle le joueur est passé avant l’entrée en NBA.

En voulant fusionner ces nouvelles données avec notre dataset principal *NBA Shot Locations 1997 - 2020.csv*, nous avons été confrontés à un problème : Le dataset *player_data.csv* ne fournit pas l’ID unique des joueurs mais uniquement leurs noms.

Le dataset *NBA Shot Locations 1997 - 2020.csv* présente 9 noms de joueurs en doublons, c’est à dire 2 ID distincts pour un même nom de joueur. Nous ne pouvons donc pas fusionner les deux dataframes par le nom au risque d’obtenir des données erronées.

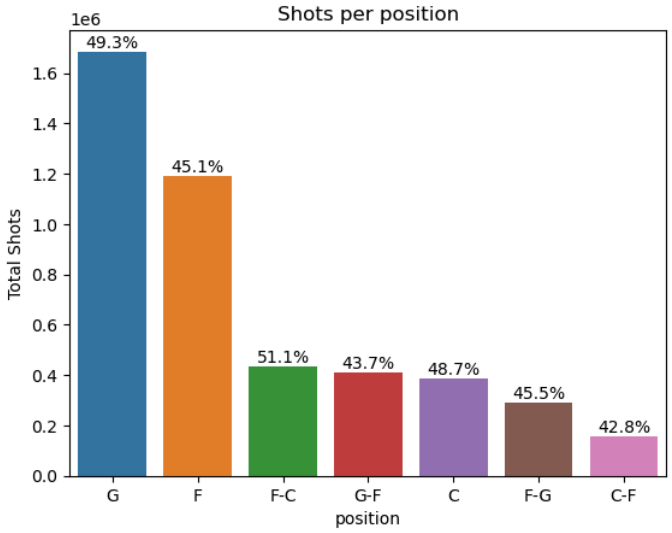
Exemple :

Player Name	ID_1	ID_2
Patrick Ewing	121	201607

En comparant les données, on remarque que les ID les plus petits (*ex : Patrick Ewing = 121*) correspondent aux **year_start** les plus anciennes (*ex : Patrick Ewing = 1986*). Cette logique semble se vérifier et on en conclut donc que l’ID s’incrémente de façon croissante en fonction de l’année d’entrée en NBA. En suivant cette logique, nous pouvons ajouter l’ID correspondant aux joueurs dans le dataset *player_data.csv*. Nous procédons ensuite à la fusion de nos deux datasets.

Les nouvelles variables qui nous semblent pertinentes pour entraîner notre modèle sont :

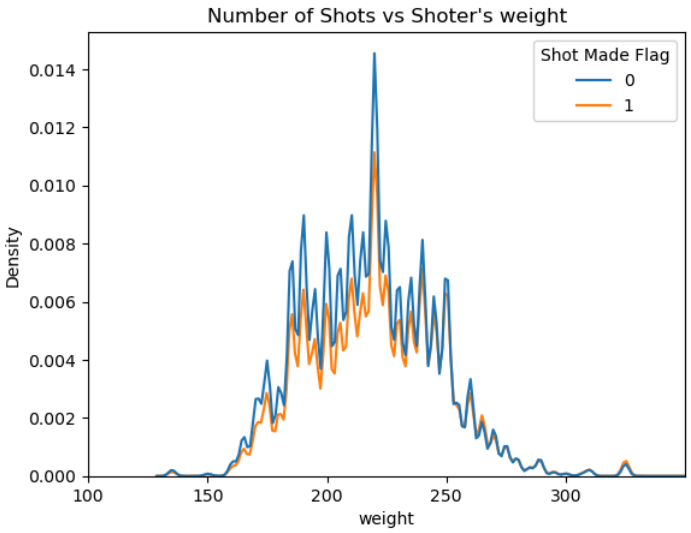
- **Position** : nous remarquons sur le graph ci-dessous que le nombre et la précision des tirs changent selon le poste du joueur qui a effectué le tir.



- **Height** : comme cette variable a un format "pieds-pouces" on le convertit uniquement en pieds afin d’obtenir une valeur numérique float64. Pour cela, on sépare pieds et pouces dans 2 colonnes distinctes, on convertit les pouces en pieds sachant que 1 pouce = 0.0833 pieds et on additionne les 2 valeurs pour obtenir notre valeur numérique finale qui correspond à la

mesure de la taille en pieds. On constate une croissance de la précision des tirs lorsque la taille du joueur se trouve entre 6.33ft et 7.08ft. A moins de 6.33ft la précision varie autour de 42% alors qu'au-dessus de 7.08ft la précision varie autour de 50%.

- **Weight** : nous constatons une augmentation de la précision (les courbes bleue et orange se chevauchent) au fur et à mesure que le joueur devient plus lourd. Ceci peut s'expliquer par le fait que les joueurs plus lourds jouent généralement près du panier.



- **Age** : calcule du reste entre la date de naissance du joueur et la date du tir. On constate une précision autour de 45% pour les joueurs âgés entre 21 ans et 36 ans. En dehors de cette tranche d'âge, la précision se dégrade.

2.2.2 Teams.csv dataset

D'autre part, l'ensemble de données *teams.csv* nous fournit deux variables importantes telles que Team ID et l'abréviation de son nom respectif pour chaque équipe de la NBA, tel qu'il apparaît dans les colonnes Home Team et Away Team dans *NBA_Shot_Locations_1997_-_2020.csv*. Avec ces données, nous avons créé une nouvelle variable **Shot_Team** qui décrit si le tir a été effectué par l'équipe extérieure ou l'équipe à domicile.

2.2.3 Ranking.csv dataset

La réussite d'un tir de la part d'un joueur ne dépend pas seulement de ses facultés personnelles et de la position sur le terrain, mais aussi de l'aide de ses coéquipiers (par exemple : un écran bien placé pour libérer le tireur, une passe décisive d'une bonne qualité, une stratégie d'isolation pour amener à un match-up avantageux etc…).

Il nous a semblé compliqué d'intégrer à nos données le contexte précis de chaque shot. En revanche le classement de l'équipe du tireur à la date du match joué peut nous servir d'indicateur sur la performance collective de l'équipe et par conséquent de l'impact que celle-ci pourrait avoir sur la réussite d'un tir. Nous avons donc retenu la variable suivante :

- **W_PCT** : pourcentage de victoires de l'équipe sur la saison au moment du match joué.

2.2.4 Variable on fire

Nous partons du postulat que les joueurs ont des phases de réussite plus importantes que d'habitude sur certains matchs. En effet, il apparaît que des joueurs réussissent mieux leurs tirs sur une séquence donnée, peut-être parce qu'ils se sentent en confiance, parce qu'ils sont portés par les

acclamations des spectateurs, parce qu'ils ressentent une excitation dans des situations sous pression ou parce qu'ils ont su exploiter une faille dans la défense adverse par exemple.

Quelle qu'en soit la raison, il nous a semblé intéressant d'identifier ces phases qui correspondent à une hausse de leurs réussite au tir, car cela pourra certainement nous aider à mieux prédire si une prochaine tentative sera réussie ou ratée.

Pour cela nous avons paramétré une nouvelle variable nommée "on_fire" qui sera un booléen indiquant si le joueur est dans une phase de réussite ou non. L'idée est de vérifier si pour un même joueur on a N matchs consécutifs avec une augmentation de de la précision de +X% par rapport à sa moyenne habituelle. Si c'est le cas alors tous les tirs des N matchs en question seront tagués comme "on_fire".

Afin de ne pas taguer des matchs en "on_fire" lorsqu'il y a très peu de tirs qui ont été tentés et réussis, nous définissons également un seuil 'S' pour le déclenchement de cette variable.

- **on_fire** : prend la valeur 1 si le joueur a tenté au moins S tirs dans chacun des N matchs consécutifs avec une réussite de +X% par rapport à sa moyenne de réussite en carrière. Avec :
    - S = nombre de tir minimum tenté pour que la variable on_fire puisse prendre la valeur 1.
    - X = augmentation minimum du pourcentage par rapport à la moyenne de "Shot Made Flag" du joueur sur tous les tirs observés pour que la variable on_fire puisse prendre la valeur 1.
    - N = nombre de matchs consécutifs minimum avec une augmentation de réussite de X% pour que la variable on_fire puisse prendre la valeur 1.

2.2.5 Vérification de l'incohérence après la fusion

Suite à l'ajout de ces nouvelles variables nous faisons une vérification et nous obtenons quelques valeurs aberrantes pour l'âge car 3 joueurs ont entre 3 et 16 ans en NBA. Pour cause, il s'agit de noms de joueurs qui apparaissent en double dans *player_data.csv* mais qui n'apparaissent qu'une seule fois dans *NBA Shot Locations 1997 - 2020.csv*.

Affichage des ID des joueurs dans *NBA Shot Locations 1997 - 2020.csv* :

Player Name	ID
Glenn Robinson	121
Gary Payton	56
Tim Hardaway	896

L'ID relativement petit semble indiquer qu'il s'agit des joueurs de la 1ère génération. Pour vérifier cette hypothèse, et voir s'il n'y a pas deux joueurs différents avec le même nom qui auraient été regroupés par erreur sous le même ID dans *NBA Shot Locations 1997 - 2020.csv*, nous vérifions la plage des années jouées dans *NBA Shot Locations 1997 - 2020.csv* :

- Glenn Robinson : [1997 1998 1999 2000 2001 2002 2003 2004 2005]
- Gary Payton : [1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007]
- Tim Hardaway : [1997 1998 1999 2000 2001 2002 2003]

Il semble clair qu'il s'agit des joueurs Glenn Robinson, Gary Payton et Tim Hardaway de la première génération qui est présente dans le dataframe *NBA Shot Locations 1997 - 2020.csv*. Nous supprimons donc les 3 joueurs de la 2ème génération dans *player_data.csv* et procédons à une nouvelle fusion des deux datasets.

3 Génération du dataset pour l'entraînement du modèle

Après avoir effectué une analyse exploratoire générale pour tous les tirs de notre ensemble de données, nous devrons effectuer un petit filtre pour restreindre notre ensemble de données afin de faciliter l'entraînement. Pour ce faire, nous prendrons la recommandation faite par l'organisme de formation pour entraîner un modèle pour les 20 meilleurs joueurs de la NBA du 21ème siècle.

3.1 Sélection des 20 meilleurs joueurs

Pour cela nous nous sommes appuyés sur le dataset Season_Stats.csv qui fournit plusieurs statistiques intéressantes sur les joueurs pour chaque saison NBA. Voici les variables prises en compte pour déterminer le classement des 20 meilleurs joueurs :

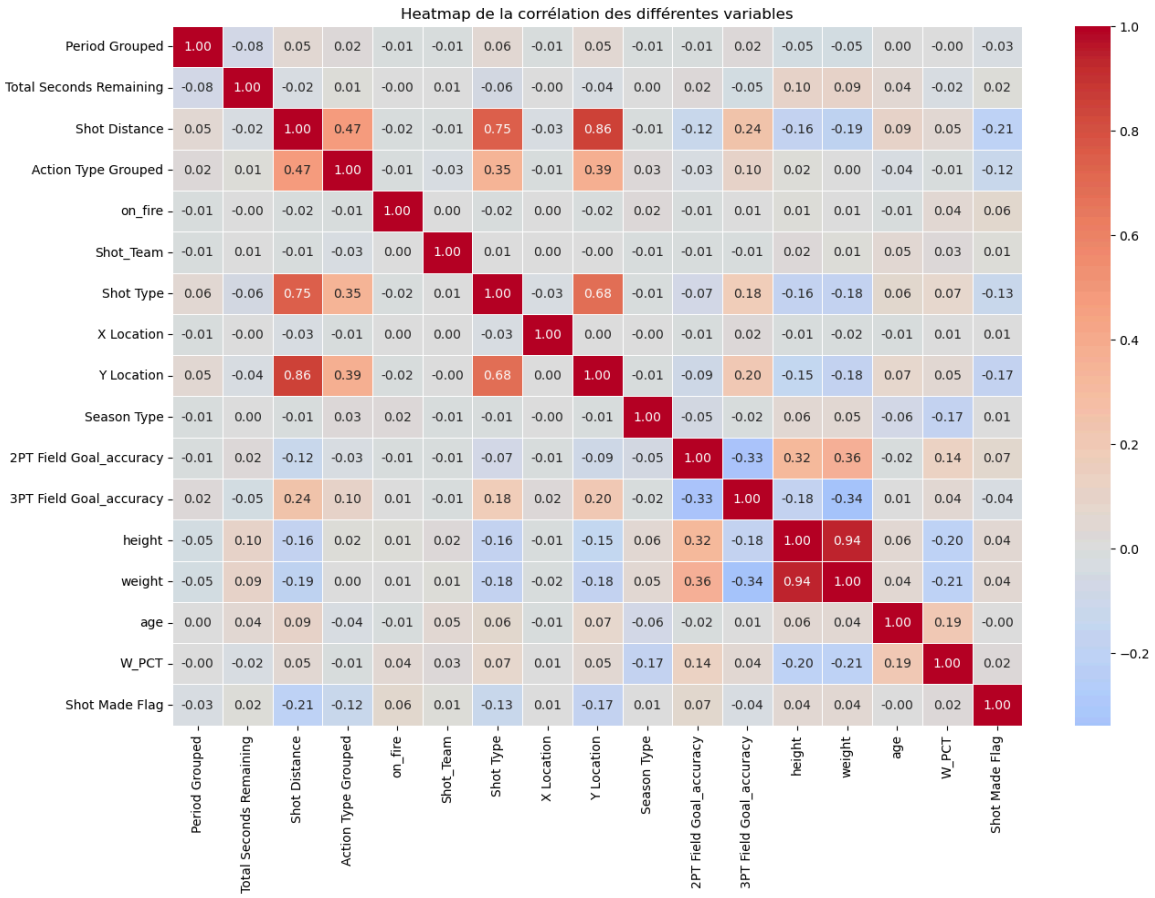
- **G** : Le nombre de matchs joués par joueur pour chaque saison.
- **AST** : Le nombre d'assist (passe décisive) effectués par joueur pour chaque saison.
- **TRB** : Le nombre de rebonds total pris par joueur pour chaque saison.
- **PTS** : Le nombre de points marqués par joueur pour chaque saison.
- **PER** : Player Efficiency Rate. Cette variable mesure la contribution d'un joueur par minutes jouées.

Après avoir calculé pour chaque joueur les PPG (moyenne des points par matchs), RPG (moyenne des rebonds par matchs) et APG (moyenne des assists par match), nous avons attribué des poids à chacune de ces variables tel que : Score = 0.3*PPG + 0.1*APG + 0.1*RPG + 0.5*PER. Notre sélection des 20 meilleurs joueurs correspond alors aux 20 joueurs ayant le score le plus élevé.

3.2 Dataset final

Notre dataset a une dimension de 248528 lignes, et les 17 colonnes détaillées ci-dessous :

['Period Grouped', 'Total Seconds Remaining', 'Shot Distance', 'Action Type Grouped', 'on_fire', 'Shot_Team', 'Shot Type', 'X Location', 'Y Location', 'Season Type', '2PT Field Goal_accuracy', '3PT Field Goal_accuracy', 'height', 'weight', 'age', 'W_PCT', 'Shot Made Flag']



3.3 Pre-processing

1. **Séparation de notre dataset en features et target** : X (features) et y (target = ‘Shot Made Flag’)
2. **Équilibrage des données** : Pour entraîner au mieux notre modèle nous avons rééquilibré nos données afin d’obtenir autant de shoots réussis que de shoots ratés. Pour cela nous avons effectué un under sampling à l’aide de ‘RandomUnderSampler’.
3. **Séparation de nos données en un ensemble d'entraînement et un ensemble de test** : à l’aide d’un ‘train_test_split’ nous avons séparé nos données en un ensemble d'entraînement X_train, y_train qui correspond à 80% de nos données, puis X_test, y_test pour l’ensemble de test qui correspond à 20% de nos données. Nous avons conservé une sélection aléatoire pour ce découpage.
4. **Gestion des variables catégorielles** : Nous avons encodés nos variables catégorielles avec un ‘get_dummies’.
5. **Standardisation** : Afin de gérer nos variables numériques continues ayant des ordres de grandeurs différents les unes des autres, nous avons standardisé les données à l’aide de ‘StandardScaler’.

Suite à ces manipulations nous obtenons alors 118964 lignes avec une target = 1 et 118964 lignes avec une target = 0. Nous passons de 16 colonnes de features à 29 colonnes. X_train contient 190342 lignes et X_test contient 47586 lignes.

4 Entraînement des modèles

Une fois que nous avions obtenu un ensemble de données exploitable, nous avons été confrontés au choix du modèle. Étant donné notre problème de classification binaire, nous nous sommes tout naturellement tournés vers l'algorithme le plus populaire, la régression logistique, en raison de sa

facilité d'implémentation et de sa rapidité. Dans un premier temps, nous avons exécuté le modèle sans ajuster ses hyperparamètres, ce qui a donné une précision de 0,628. Ensuite, nous avons appliqué une technique d'optimisation en utilisant la validation croisée avec une grille de recherche (GridSearchCV). Les meilleurs hyperparamètres identifiés ont amélioré légèrement notre modèle, atteignant une précision de 0,631.

Nous avons ensuite exploré d'autres algorithmes plus complexes et plus longs à exécuter. Les modèles SVM et KNN ont rapidement été abandonnés en raison de leur temps d'exécution excessif. Les modèles LightGBM et CatClassifier se sont montrés performants mais plus difficiles à interpréter.

Les modèles que nous avons entraînés ainsi que leur précision sont énumérés dans le tableau ci-dessous.

Modèle	Accuracy	GridSearchCV	(≈) Temps (sec)
Régression logistique	0.628	non	3
	0.629	**oui**	190
Random Forest	0.622	non	30
	0.642	**oui**	+5000
XGBoost	0.641	non	83
CatBoostClassifier	0.642	non	30
LightGBM	0.642	non	19

Voici un résumé de certaines de nos recherches d'hyper-paramètres :
Régression Logistique :

- grille de recherche GridSearchCV : {'C': np.logspace(-4, 4, 50)}
- Meilleurs paramètres retenus : max_iter=500 et 'C'= 0.0029470517025518097

Random Forest :

- grille de recherche GridSearchCV : {'n_estimators': [100, 500, 1000, 1500], 'max_depth' : [8, 10, 12, 15, 17, 20]}
- Meilleurs paramètres retenus : max_depth= 15, n_estimators= 1000

En tenant compte de l'accuracy, du rappel, du temps d'exécution et de l'interprétabilité des différents modèles nous avons choisi de conserver le **XGBoost** comme modèle central de notre projet.

5 Résultats et interprétation du modèle le plus performant

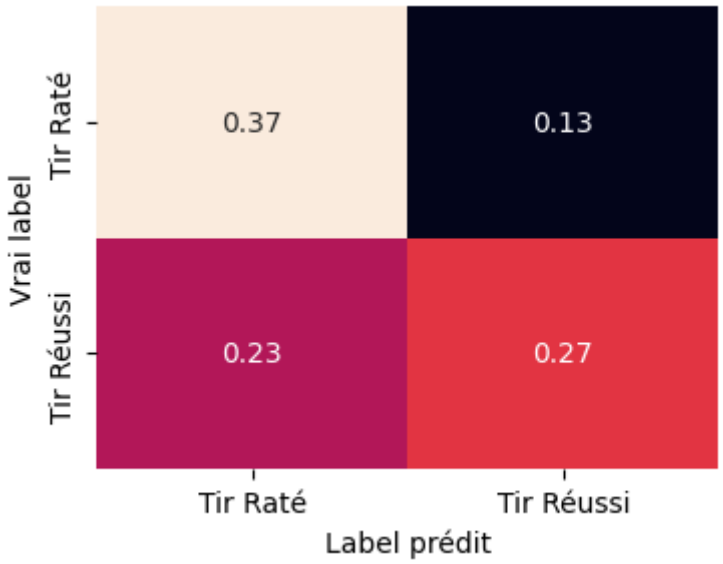
Comme nous l'avons vu dans la section précédente, le modèle qui a donné le meilleur résultat en termes de précision est le XGBoost. C'est pourquoi nous allons nous concentrer sur ce modèle afin d'analyser ses performances plus en détail et d'interpréter les résultats.

Avec le XGBoost, nous avons obtenu une accuracy de 0.641. Bien que cette valeur de précision ne soit pas très élevée, elle n'est pas non plus négligeable, étant donné que la prédiction du résultat d'un tir de basket-ball est une tâche très compliquée qui dépend de nombreux facteurs, souvent impossibles à quantifier. Derrière cette accuracy, nous pouvons également analyser certains paramètres importants tels que la précision et le rappel, qui peuvent également être combinés dans le score F1. Ces valeurs sont présentées dans le tableau ci-dessous.

Classe	Precisión	Rappel	F1-score
Tir Rate	0.61	0.75	0.67
Tir Réussi	0.68	0.53	0.60

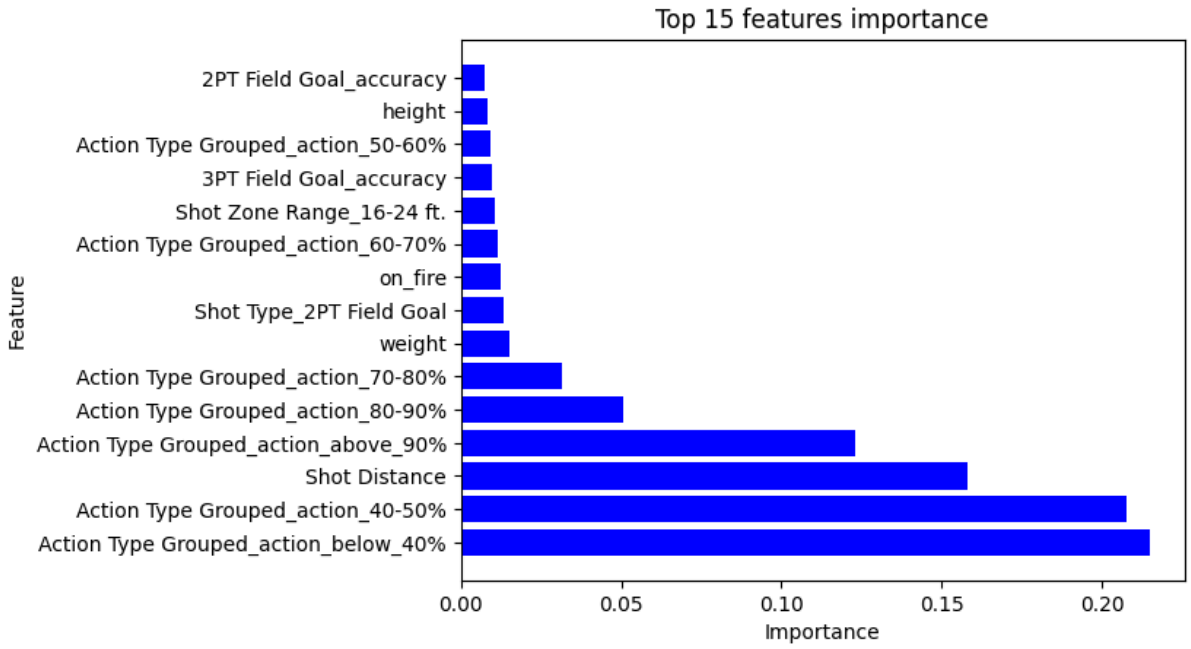
Nous pouvons constater que le modèle obtient de meilleurs résultats pour les tirs ratés que pour les tirs réussis (score F1 plus élevé pour la classe des tirs ratés que pour la classe des tirs réussis). D'autre part, nous pouvons également détecter une surestimation des tirs ratés, puisque le rappel pour la classe des tirs ratés est beaucoup plus élevé que pour la classe des tirs réussis.

Cette surestimation des tirs ratés peut être clairement observée dans la matrice de confusion normalisée, où l'on constate un pourcentage élevé de tirs réussis qui ont été prédits comme des ratés (quadrant gauche en bas).

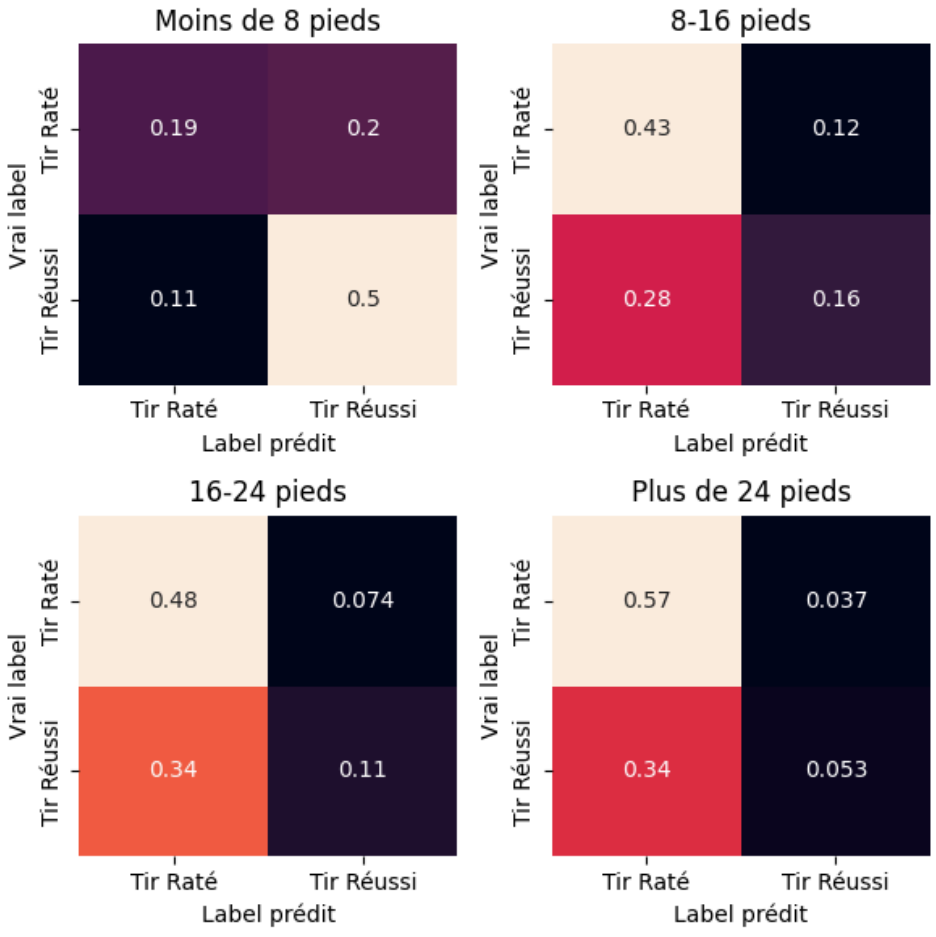


Pour mieux comprendre les résultats obtenus, nous pouvons les interpréter en analysant quelles sont les variables les plus importantes pour le modèle et quels résultats il donne dans des situations particulières.

Le graphique ci-dessous montre l'importance des variables pour notre modèle. Il est clair que le modèle accorde une grande importance à deux types de variables : la variable catégorielle Action Type et la variable continue Shot Distance. Il est clair que ces variables sont étroitement liées l'une à l'autre, puisque, en fonction de la distance au panier, le joueur tentera un type d'action ou un autre. Comme on peut le voir, les deux variables les plus importantes sont les actions de tir qui sont le plus souvent manquées, ce qui peut être l'une des raisons pour lesquelles le modèle fonctionne mieux pour prédire les tirs ratés.



Compte tenu du fait que la Shot Distance est une variable très importante pour notre modèle, nous pouvons analyser la manière dont il classe les tirs en fonction de la distance par rapport au panier. Pour ce faire, nous allons segmenter la variable Shot Distance en différents groupes qui sont couramment utilisés en NBA, à savoir les tirs à une distance inférieure à 8 pieds, les tirs entre 8 et 16 pieds, les tirs entre 16 et 24 pieds et les tirs de plus de 24 pieds (ces derniers correspondent aux tirs à trois points). Nous observons ci-dessous les matrices de confusion normalisées correspondant à chaque cas.



Il est clair que pour les tirs de plus près (moins de 8 pieds) le nombre de tirs réussis classés comme manqués est faible (0.11) et l'erreur la plus fréquente consiste à classer les tirs manqués comme des tirs réussis (0.2). C'est logique puisqu'en général, plus le joueur est proche du panier, plus il est facile de marquer. Dans ce cas, le modèle surestime davantage les tirs réussis. Au contraire, à mesure que l'on s'éloigne du panier, cette tendance s'inverse fortement et la plupart des erreurs proviennent de tirs réussis classés comme manqués (0.34 pour les tirs de plus de 16 pieds).

6 Perspectives pour améliorer la performance de notre modèle

6.1 Modèle à un seul joueur

Il est évident que les joueurs que nous avons sélectionnés pour créer notre ensemble de données sont tous très bons et ont tendance à réaliser de bonnes performances match après match. Cependant, chacun d'entre eux possède des caractéristiques particulières qui les rendent plus ou moins forts dans certains types de tirs. C'est pourquoi nous nous sommes demandés ce qui se passerait si nous entraînions des modèles individuels pour chaque joueur.

Pour ce faire, nous allons présenter les résultats de deux grands joueurs de la NBA, LeBron James et Stephen Curry. Pour LeBron James, l'ensemble de données contient un total de 21058 tirs, tandis que pour Curry, nous avons 13316 tirs. Comme pour l'ensemble de données initial, les données ont été équilibrées et séparées avec un pourcentage de 80 % pour l'entraînement et de 20 % pour le test. Dans les deux cas, nous avons entraîné un XGBoost. Le tableau suivant montre les résultats obtenus en termes d'accuracy, précision, rappel et score F1.

Joueur	Accuracy	Classe	Precisión	Rappel	F1 score
LeBron James	0.684	Tir Rate	0.65	0.8	0.72
		Tir Réussi	0.74	0.57	0.64
Stephen Curry	0.675	Tir Rate	0.67	0.68	0.68
		Tir Réussi	0.68	0.67	0.67

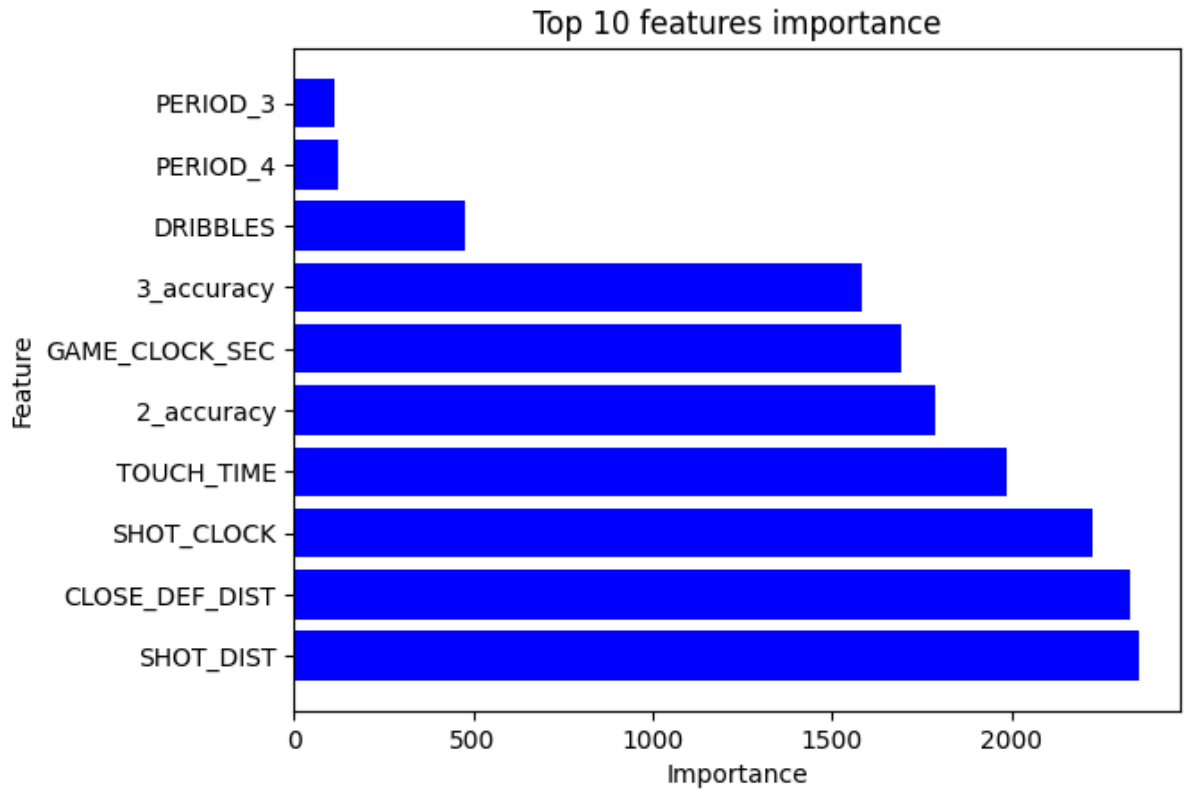
La première chose que nous pouvons observer est que l'accuracy augmente considérablement, passant de 0.641 avec le modèle à 20 joueurs à 0.684 et 0.675 pour LeBron James et Curry respectivement. Bien que le modèle de Curry soit un peu moins précis, il est important de noter qu'il classe les tirs manqués et les tirs réussis avec la même précision, ce qui n'est pas le cas de LeBron James, qui surestime également les tirs manqués. Ce résultat équilibré pour le modèle de Curry est dû au fait qu'il est probablement le meilleur tireur à trois points de l'histoire, de sorte que son efficacité en matière de tirs à longue distance est très élevée et que le modèle n'a pas tendance à estimer que les tirs à longue distance seront manqués par ce joueur.

6.2 Ajout de nouvelles variables

Bien que notre ensemble de données soit assez complet, certaines variables importantes qu'il serait très utile d'intégrer ne sont pas facilement disponibles sur internet. Nous avons trouvé sur Kaggle un ensemble de données qui contient des informations sur les tirs de la saison 2014-2015 telles que la distance à laquelle le défenseur était le plus proche du tireur, le nombre de secondes pendant lesquelles le tireur a eu le ballon en main avant de tirer ainsi que le nombre de dribbles, le nombre de secondes restantes sur le chronomètre de possession (chaque possession au basket-ball dure au maximum 24 secondes), entre autres. Comme ces informations ne sont pas disponibles pour tous les tirs de notre jeu de données original, nous avons entraîné un modèle basé sur ce nouveau jeu de

données. Il contient un total de 128069 tirs, que nous avons nettoyées, équilibrées, séparées en entraînement et test avec un rapport de 80-20 et entraînées avec un XGBoost.

Le résultat obtenu (0,6 d'accuracy) n'est pas aussi bon que celui obtenu avec l'ensemble de données original, peut-être parce qu'il s'agit d'un ensemble de données avec tous les joueurs qui ont joué dans la saison (pas les 20 meilleurs) et aussi parce que certaines variables comme le Action Type n'étaient pas disponibles dans ce nouvel ensemble de données. Mais ce qu'il faut noter, c'est l'importance des variables, comme le montre le graphique ci-dessous. Nous pouvons voir que le modèle donne une grande importance à la distance au panier (SHOT_DIST), mais aussi à la distance du défenseur le plus proche (CLOSE_DEF_DIST), aux secondes sur le chronomètre de possession (SHOT_CLOCK) et aux secondes pendant lesquelles le joueur a eu le ballon en main avant de tirer (TOUCH_TIME). Cela nous amène à penser que si nous pouvions combiner ces informations avec notre ensemble de données original, il serait possible d'augmenter encore l'accuracy de notre modèle.



6.3 Modèle selon le type de tir

Une autre technique à laquelle nous avons pensé pour augmenter la précision de notre modèle consistait à séparer notre ensemble de données par type de tir, et donc à entraîner un modèle pour prédire les doubles et un autre pour prédire les triples. Dans ce cas, notre ensemble de données de doubles comporte un total de 18 448 tirs et l'ensemble de données de triples (évidemment plus petit) comporte un total de 43 154 tirs. Ici aussi, les données ont été séparées en entraînement et test avec un rapport de 80-20 et entraînées avec un XGBoost. Les résultats pour les deux modèles sont présentés dans le tableau suivant.

Modele	Accuracy	Classe	Precisión	Rappel	F1 score
2 Points	0.650	Tir Rate	0.63	0.74	0.68
		Tir Réussi	0.69	0.56	0.61

3 Points	0.559	Tir Rate	0.56	0.58	0.57
		Tir Réussi	0.56	0.53	0.55

Dans ce cas, nous pouvons observer que l'accuracy augmente légèrement pour le modèle de tir à deux points, passant de 0,641 à 0,65. Comme avec le modèle pour tous les tirs, nous observons une surestimation des tirs manqués avec un rappel beaucoup plus faible pour les tirs réussis. En revanche, pour le modèle de tir à trois points, la précision diminue fortement, passant à 0,559. En conclusion, ce type de séparation des données ne conduit pas à une grande amélioration comme nous l'avons obtenu lors de la séparation par joueur.

7 Conclusion et perspectives

Au cours de ce projet, nous avons parfois été confrontés à des obstacles. Tout d'abord notre dataset d'origine était très volumineux (plus de 4 Millions de lignes), et ce dernier ne comportait pas toutes les variables que nous aurions souhaité avoir. Les autres dataset complémentaires utilisés ne couvraient pas toujours les mêmes années que notre dataset principal et ne contenaient que très rarement les identifiants des joueurs, ce qui a rendu les fusions de tableaux plus compliquées que prévu. D'autre part, il nous a été difficile de trouver d'autres données accessibles qui selon nous auraient permis d'améliorer les performances de nos modèles (par exemple : la distance du défenseur le plus proche qui ne couvrait qu'une seule année). Avec plus de temps et d'expérience, de telles données auraient probablement pu êtres acquises via des API ou du Web Scrapping. Et enfin, nous avons été limités par les puissances de calcul de nos machines, nous avons donc rapidement abandonné l'idée du deep learning et n'avons pas pu rechercher convenablement les meilleurs hyperparamètres pour chacun de nos modèles du fait d'un temps d'exécution trop long.

Toutefois il s'est avéré que nous avions au sein de notre groupe de bonnes connaissances de la NBA et du basket en général, ce qui a permis de fluidifier nos échanges mais surtout de converger ensemble sur les différentes variables d'importance à intégrer à nos données. Nous nous sommes répartis les différentes tâches naturellement en fonction des souhaits et des compétences de chacun et notre projet a été rythmé par des échanges réguliers par visioconférence à fréquence d'une à deux fois par semaine, en plus du point hebdomadaire organisé par le chef de projet.

En conclusion, notre analyse révèle la réelle complexité de la prédiction des tirs réussis au basket-ball. Après avoir testé différents algorithmes tels que la régression logistique, gradient boosting, LGBM, etc., notre modèle, basé sur XGBoost, est celui qui a présenté la meilleure précision. Cependant il a montré une précision variable en fonction de la classe de tir : avec une meilleure performance pour la prédictions des tirs manqués. Une explication plausible de ce phénomène réside dans l'importance plus élevée accordée par le modèle aux types d'action de tir (Action Type) qui sont les plus souvent manqués, en particulier lorsque ces variables sont corrélées avec la distance de tir (Shot Distance).

Notre analyse de la performance du modèle a révélé une meilleure précision pour les tirs plus proches du panier. Cela suggère que notre modèle pourrait bénéficier d'une prise en compte plus nuancée de la distance de tir, potentiellement en formant des sous-modèles spécifiques pour différentes gammes de distances.

De plus, nous avons constaté une amélioration significative de la précision de notre modèle lorsque nous avons entraîné des modèles individuels pour chaque joueur, pour le modèle Lebron James une accuracy de 0.684 est obtenue. Ces résultats suggèrent que les caractéristiques individuelles d'un joueur peuvent jouer un rôle significatif dans la prédiction des tirs. Cela peut aussi faciliter l'apprentissage et la compréhension du modèle. Dans le futur, il serait intéressant d'explorer cette voie en formant des modèles individuels pour un plus grand nombre de joueurs.

Nous avons également identifié plusieurs variables potentiellement utiles qui ne sont pas actuellement incluses dans notre ensemble de données, telles que la distance du défenseur le plus proche, le temps de possession du ballon et le nombre de dribbles avant le tir. Bien que l'ajout de ces variables ait nécessité l'entraînement d'un modèle sur un nouvel ensemble de données, l'importance attribuée à ces variables par le modèle suggère qu'elles pourraient être bénéfiques si elles étaient intégrées à notre ensemble de données original, malheureusement nous ne possédons uniquement qu'un jeu de données sur la saison 2014-2015. L'intégration de ces variables dans notre modèle pourrait très probablement améliorer encore sa précision.

En somme, notre analyse a non seulement fourni un modèle capable de prédire avec une précision raisonnable les tirs réussis et manqués, mais a également révélé des informations précieuses sur les facteurs qui influencent ces résultats. En dépit des défis inhérents à la prédiction des résultats dans un sport aussi dynamique et complexe que le basket-ball, cette étude a montré qu'il est possible de tirer des enseignements précieux et exploitables des données disponibles.