Projet « Machine Learning »

Teddy NKODIA

Sigma Clermont
Mastère spécialisé Expert en Science des données

Rapport: Prédiction des performances NBA avec Machine Learning

Table of Contents

1. Introduction

1.1 Contexte du projet

La **National Basketball Association (NBA)** est une ligue de basket-ball où chaque équipe dispute **82 matchs en saison régulière**. Les performances des joueurs varient en fonction de nombreux facteurs :

- Leur **forme physique**
- L'**adversaire** qu'ils affrontent
- Le **lieu du match** (à domicile ou à l'extérieur)
- Leur **récente dynamique de jeu**

Anticiper ces performances est **crucial** pour plusieurs acteurs :

- **Les entraîneurs et analystes sportifs** : Ils peuvent utiliser ces prévisions pour affiner les stratégies de jeu.
- **Les fans et journalistes** : Ils peuvent mieux comprendre les performances attendues des joueurs.
- **Les parieurs sportifs** : Un modèle précis pourrait être utile pour établir des pronostics.

1.2 Objectif du projet

L'objectif est de prédire les performances d'un joueur NBA sur un match donné, en fonction de ses statistiques passées et de son contexte de jeu (adversaire, domicile ou extérieur, séries de bons matchs...).

Pour cela, nous avons utilisé des modèles de Machine Learning qui apprennent à partir des données passées pour prédire les résultats futurs.

2. Présentation du jeu de données et de la problématique

2.1 Source des données

Les données sont récupérées en temps réel via nba_api, une API qui fournit les statistiques officielles des matchs NBA.

Nous avons collecté les statistiques des matchs des joueurs pour la saison 2024-25.

2.2 Variables du jeu de données

Les performances d'un joueur de NBA sont influencées par plusieurs facteurs. Pour réaliser une prédiction efficace du nombre de points qu'il marquera lors d'un match, nous avons sélectionné 13 variables d'entrée (features) qui décrivent son contexte de jeu et ses performances récentes.

Les features sont des indicateurs qui permettent de comprendre le comportement et les performances du joueur.

2.2.1 Statistiques de performance du joueur

• REB (Rebounds - Rebonds pris par le joueur)

→ Nombre total de fois où le joueur a récupéré la balle après un tir manqué (offensif ou défensif).
→ Plus un joueur prend de rebonds, plus il a d'opportunités d'affecter le match.

• AST (Assists - Passes décisives du joueur)

→ Une passe décisive est une passe qui mène directement à un panier marqué par un coéquipier.
→ Indique la capacité du joueur à **créer des opportunités pour ses coéquipiers**.

• BLK (Blocks - Contres réalisés par le joueur)

→ Nombre de tirs adverses que le joueur a bloqués.
→ Les pivots et les joueurs défensifs ont souvent un **taux élevé de contres**.

• FG3M (Field Goals 3 Made - Tirs à 3 points réussis)

→ Nombre de tirs à 3 points marqués par le joueur.
→ Un **tir à 3 points réussi** est une action très précieuse en NBA.

• TOV (Turnovers - Ballons perdus)

→ Nombre de fois où le joueur a perdu la balle (passe interceptée, sortie de terrain, etc.).

→ Un **taux élevé de ballons perdus** peut indiquer un joueur sous pression.

• MIN (Minutes jouées durant le match)

→ Temps total passé par le joueur sur le terrain.
→ Un joueur qui joue **beaucoup de minutes** a généralement plus d'impact sur le jeu.

2.2.2 Contexte du match

Ces variables renseignent sur le contexte du match dans lequel le joueur évolue.

• Location (Lieu du match - domicile ou extérieur)

→ Un joueur joue soit **à domicile** ("Home") soit **à l'extérieur** ("Away").
→ Les joueurs performent généralement **mieux à domicile**, soutenus par leurs fans.

• Opponent (Adversaire affronté)

→ Nom de l'équipe adverse.
→ La **qualité défensive** de l'adversaire influence les performances du joueur.

2.2.3 Précision du tir

Ces variables indiquent l'efficacité du joueur lorsqu'il tente un tir :

• FG% (Field Goal Percentage - Précision au tir du joueur : tirs réussis / tirs tentés)

→ Pourcentage de réussite des tirs du joueur.
→ Un **FG% élevé** indique un joueur précis qui optimise ses opportunités.

• FT% (Free Throw Percentage - Précision aux lancers francs : lancers réussis / lancers tentés)

→ Pourcentage de réussite aux lancers francs.
→ Une **faible précision aux lancers** peut être un problème en fin de match.

2.2.4 Indicateurs avancés de tendance

Ces variables permettent d'analyser les tendances récentes des performances du joueur :

• Plus-Minus (Impact du joueur sur le score final de son équipe)

→ Mesure l'impact du joueur sur le match lorsqu'il est sur le terrain.
→ Si un joueur a un **Plus-Minus positif**, cela signifie que son équipe marque plus de points lorsqu'il est en jeu.

• Rolling_PTS_Avg (Moyenne des points du joueur sur les 5 derniers matchs)

→ Moyenne des points marqués par le joueur lors de ses **5 derniers matchs**.
→ Permet d'observer si le joueur est en **hausse de performance** ou non.

• Hot_Streak (Le joueur est-il dans une bonne dynamique ?)

→ Indique si le joueur a marqué **au moins 20% de points en plus** que sa moyenne sur les 5 derniers matchs.
→ Un joueur en **Hot Streak** peut être plus performant que prévu.

2.2.5 Variables cibles

L'objectif du modèle est de **prédire plusieurs statistiques clés** de la performance du joueur lors du prochain match.

• PTS (Points marqués par le joueur)

→ Nombre total de points marqués par le joueur.
→ **Indicateur principal de la performance offensive d'un joueur**.

• REB (Rebonds pris par le joueur)

→ Nombre total de rebonds captés par le joueur (offensifs et défensifs).
→ Important pour **évaluer son impact sur le jeu sous les paniers**.

• AST (Passes décisives du joueur)

→ Nombre de passes décisives effectuées par le joueur.
→ Mesure la capacité du joueur à **créer des opportunités pour ses coéquipiers**.

• BLK (Contres réalisés par le joueur)

→ Nombre de tirs adverses bloqués par le joueur.
→ Indicateur clé de la capacité défensive du joueur.

• FG3M (Nombre de tirs à 3 points réussis)

→ Nombre total de tirs à 3 points réussis par le joueur.
→ Essentiel pour évaluer les shooters spécialisés et l'impact d'un joueur dans le tir extérieur.

Voici un résumé des données utilisées:

Type	Variable	Description
Performance	REB, AST, BLK, FG3M, TOV, MIN	Statistiques individuelles du joueur
Contexte	Location, Opponent	Où et contre qui le match est joué
Précision	FG%, FT%	Efficacité aux tirs
Tendance	Plus-Minus, Rolling_PTS_Avg, Hot_Streak	Dynamique du joueur
Cibles	PTS, REB, AST, BLK, FG3M	Statistiques à prédire

Avec ce jeu de données bien structuré, notre modèle peut prédire plusieurs statistiques clés d'un joueur en fonction de ses performances passées et du contexte du match. Cela nous permet d'obtenir une analyse fine et dynamique du jeu NBA.

3. Présentation des méthodes supervisées testées

Pour prédire les performances des joueurs NBA, nous avons testé plusieurs modèles de machine learning supervisés. Ces modèles ont été sélectionnés en fonction de leur capacité à traiter des données tabulaires, à capturer les tendances de performance des joueurs, et à gérer des prédictions multi-sorties (nombre de points, passes, rebonds, etc.).

Nous avons regroupé ces modèles en quatre grandes catégories :

- Modèles linéaires (Régressions Ridge et Lasso)
- Modèle non paramétrique (K-Nearest Neighbors - KNN)

- Modèle basé sur les marges (Support Vector Machine - SVM)
- Modèles ensemblistes (Random Forest, AdaBoost, Gradient Boosting)

3.1 Modèles linéaires

Les **modèles linéaires** sont souvent utilisés pour les problèmes de régression où la variable cible (*output*) est continue.

Nous avons testé deux modèles :

- **Régression Ridge**
- **Régression Lasso**

3.1.1 Régression Ridge

Principe :

- La régression Ridge est une **régression linéaire pénalisée** par une régularisation **L2**.
- Elle vise à réduire l'impact des variables fortement corrélées, évitant ainsi le surajustement (*overfitting*).
- Elle est particulièrement efficace lorsqu'il y a de nombreuses variables *(features)*.

Pourquoi Ridge dans ce projet ?

- Il est stable et robuste, idéal pour un jeu de données avec plusieurs variables contextuelles et historiques.
- Il limite l'impact des variables trop influentes, ce qui est utile lorsque certaines statistiques varient beaucoup (ex : nombre de tirs à 3 points réussis).

3.1.2 Régression Lasso

Principe :

- La régression Lasso est une régression linéaire pénalisée par une régularisation L1.
- Contrairement à Ridge, elle peut supprimer complètement certaines variables, ce qui permet de faire de la sélection de features automatiquement.

Pourquoi Lasso dans ce projet ?

- Il nous aide à identifier les variables les plus importantes en supprimant celles qui n'apportent pas d'information utile.
- Il est particulièrement utile si certaines statistiques (ex : plus-minus) ne sont pas toujours pertinentes pour tous les joueurs.

Différence entre Ridge et Lasso

	Ridge	**Lasso**
Type de régularisation	Pénalité **L2** (réduit les coefficients)	Pénalité **L1** (peut annuler des coefficients)
Gestion des variables inutiles	Garde toutes les variables, mais réduit leur poids	Supprime les variables non pertinentes
Utilisation	Idéal pour **éviter l'overfitting**	Idéal pour **sélectionner les meilleures variables**

3.2 Modèle non paramétrique

Les modèles non paramétriques ne font pas d'hypothèses spécifiques sur la forme des données. Ils s'adaptent directement aux exemples passés.

Nous avons testé le modèle des k plus proches voisins (K-Nearest Neighbors - KNN)

3.2.1 K-Nearest Neighbors (KNN)

Principe :

- Ce modèle prédit la performance d'un joueur en comparant ses performances passées avec celles des joueurs similaires.
- Il fonctionne en cherchant les k matchs les plus similaires dans le passé et en utilisant une moyenne des performances observées dans ces matchs pour faire sa prédiction.

Pourquoi KNN dans ce projet ?

- Il est facilement interprétable et ne fait pas d'hypothèses sur la forme des données.

- Il fonctionne bien si des tendances claires se dégagent dans les performances récentes.

Inconvénients :

- Lent sur de grands jeux de données.
- Moins efficace si les performances des joueurs sont très variables.

3.3 Modèles ensemblistes

Les modèles ensemblistes combinent plusieurs modèles de base pour améliorer la précision et réduire les erreurs.

Nous avons testé trois modèles ensemblistes :

- **Random Forest**
- **AdaBoost**
- **Gradient Boosting**

3.3.1 Random Forest

Principe :

- Random Forest utilise plusieurs arbres de décision et effectue une moyenne des prédictions.
- Il réduit la variance et évite l'overfitting des arbres de décision classiques.

Pourquoi Random Forest dans ce projet ?

- Il est très robuste même avec des données bruitées.
- Il permet d'évaluer l'importance des variables (ex : Quel facteur impacte le plus les performances du joueur ?).

Inconvénients :

- Moins efficace si les données sont trop linéaires.

3.3.2 AdaBoost (Adaptive Boosting)

Principe :

- AdaBoost corrige ses erreurs au fil du temps en donnant plus de poids aux erreurs passées.
- Il entraîne une série de modèles faibles, en renforçant à chaque fois les observations difficiles à prédire.

Pourquoi AdaBoost dans ce projet ?

- Il est très efficace quand il y a des relations complexes dans les données.
- Il s'adapte dynamiquement aux joueurs dont les performances sont très irrégulières.

Inconvénients :

- Sensible aux valeurs aberrantes.

3.3.3 Gradient Boosting

Principe :

- Ce modèle fonctionne comme AdaBoost, mais il réduit l'erreur de manière plus progressive et optimisée.
- Il est extrêmement puissant pour les prédictions de variables continues.

Pourquoi Gradient Boosting dans ce projet ?

- Il est souvent le meilleur modèle pour les prédictions numériques.
- Il est capable d'apprendre des tendances complexes qui échappent aux modèles linéaires.

Inconvénients :

- Long à entraîner.
- Risque de surajustement si mal paramétré.

3.4 Comparaison des modèles

Voici une comparaison des modèles cités précedemment

Modèle	Avantages	Inconvénients
Ridge / Lasso	Stables, faciles à interpréter	Peu efficaces sur des données non linéaires
KNN	Simple, fonctionne bien si les tendances sont claires	Lent avec de nombreuses données
SVM	Robuste, capture bien les relations complexes	Sensible aux valeurs aberrantes
Random Forest	Performant, identifie les variables importantes	Moins efficace sur des tendances linéaires
AdaBoost	Très précis, s'adapte aux tendances complexes	Sensible aux erreurs
Gradient Boosting	Très puissant sur les données tabulaires	Risque de surajustement

4. Présentation des Librairies Python Utilisées

Ce projet repose sur plusieurs bibliothèques Python essentielles à la manipulation des données, à l'implémentation des modèles de Machine Learning et à la visualisation des résultats. Chaque bibliothèque joue un rôle spécifique et complémentaire pour assurer le bon déroulement de l'analyse prédictive.

4.1 nba_api : Récupération des données NBA

La bibliothèque nba_api permet d'accéder aux statistiques officielles des joueurs et des équipes NBA grâce à l'API fournie par la NBA. Elle est particulièrement utile pour récupérer les données des matchs joués par un joueur spécifique durant une saison donnée. Pour utiliser nba_api, il faut d'abord l'installer via pip : ***"pip install nba_api"***

Voici comment récupérer les statistiques de tous les matchs d'un joueur donné pour la saison actuelle :

```python
from nba_api.stats.endpoints import PlayerGameLog
from nba_api.stats.static import players

# Fonction pour obtenir l'ID d'un joueur à partir de son nom
def get_player_id(player_name):
    nba_players = players.get_players()
    for player in nba_players:
        if player["full_name"].lower() == player_name.lower():
            return player["id"]
    return None

# Récupération des matchs d'un joueur
player_name = "LeBron James"
player_id = get_player_id(player_name)

if player_id:
    game_log = PlayerGameLog(player_id=player_id, season='2024-25', season_type_all_star='Regular Season').get_data_frames()[0]
    print(game_log.head())  # Affiche les premières lignes des statistiques des matchs
else:
    print("Joueur non trouvé.")
```

Cette requête retourne un DataFrame contenant les statistiques de tous les matchs joués par le joueur dans la saison 2024-25.

4.2. pandas : Manipulation et Nettoyage des Données

Pandas est une bibliothèque essentielle pour le traitement des données sous forme de DataFrames. Elle permet de charger, nettoyer, transformer et analyser des jeux de données efficacement.

4.3. numpy : Calculs Numériques et Manipulation de Matrices

Numpy est utilisée pour effectuer des calculs mathématiques optimisés et manipuler des tableaux multidimensionnels. Elle est particulièrement utile pour les opérations matricielles dans l'entraînement des modèles.

4.4. scikit-learn : Machine Learning et Prétraitement des Données

scikit-learn est une bibliothèque incontournable pour implémenter des modèles de Machine Learning. Elle inclut des outils pour :

- Le prétraitement des données (normalisation, encodage des variables catégoriques).
- L'entraînement et l'évaluation des modèles.
- L'optimisation des hyperparamètres.

4.5. matplotlib & seaborn : Visualisation des Données

matplotlib est une bibliothèque de visualisation permettant de créer des graphiques statiques.

seaborn est une extension de matplotlib qui facilite la création de visualisations statistiques avancées.

4.6. streamlit : Interface Utilisateur Interactive

streamlit permet de créer une application web interactive pour afficher les prédictions des modèles Machine Learning. Elle simplifie la création d'interfaces utilisateur en Python. Pour utiliser streamlit, il faut d'abord l'installer via pip : ***"pip install streamlit".*** Ensuite l'application streamlit peut etre lancé en éxécutant la commande : ***"streamlit run app.py"*** où app.py est un script contenant le code de l'interface.

Voici ainsi un résumé de toutes les bibliothèques utilisés dans ce projet:

Bibliothèque	Rôle
nba_api	Récupération des données NBA via l'API officielle
pandas	Manipulation et nettoyage des données
numpy	Calculs numériques et manipulation de matrices
scikit-learn	Machine Learning, normalisation et évaluation des modèles
matplotlib & seaborn	Visualisation des données et des résultats
streamlit	Création d'une interface interactive pour visualiser les prédictions

5. Protocole Expérimental

Le protocole expérimental mis en place pour ce projet suit une méthodologie rigoureuse visant à garantir la qualité des prédictions en effectuant un prétraitement des données, une normalisation, ainsi qu'une optimisation des modèles à l'aide de la validation croisée et de la recherche d'hyperparamètres.

5.1. Prétraitement des Données

Le prétraitement est une étape essentielle en Machine Learning. Il permet de transformer les données brutes en un format approprié pour l'entraînement des modèles. Les étapes suivantes ont été réalisées :

5.1.1. Transformation des variables catégoriques en valeurs numériques

Certaines variables, telles que **Location** (domicile ou extérieur) et **Opponent** (adversaire du jour), sont de type catégoriel. Les algorithmes de Machine Learning nécessitent des entrées numériques, il est donc indispensable de les encoder.

Encodage avec LabelEncoder

La transformation a été réalisée via LabelEncoder, qui attribue un entier unique à chaque catégorie :

```python
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

df["Location"] = label_encoder.fit_transform(df["Location"])  # Convertit "Home" et "Away" en 0 et 1
df["Opponent"] = label_encoder.fit_transform(df["Opponent"])  # Encode chaque adversaire avec un numéro unique
```

Ainsi :

"Home" devient **0** et "Away" devient **1**.

"LAL", "BOS", "NYK", etc., sont convertis en des entiers uniques.

5.1.2. Gestion des valeurs manquantes

Certaines variables peuvent contenir des valeurs manquantes (ex : pourcentage de tirs à 3 points si aucun tir n'a été tenté). Ces valeurs doivent être remplacées pour éviter des erreurs lors de l'entraînement.

Utilisation de SimpleImputer

L'approche choisie consiste à remplacer les valeurs manquantes par la **moyenne** de la colonne correspondante.

```python
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="mean")
df[["FG%", "FT%", "Plus-Minus"]] = imputer.fit_transform(df[["FG%", "FT%", "Plus-Minus"]])
```

Cela permet de garder une cohérence dans les données sans supprimer des lignes potentiellement utiles.

5.1.3. Normalisation des données

Les modèles de Machine Learning sont sensibles aux échelles des variables. Par exemple, le nombre de minutes jouées (MIN) peut être bien plus grand que les valeurs de FG%, ce qui pourrait biaiser les modèles. Il est donc nécessaire d'uniformiser ces échelles.

Utilisation de StandardScaler

Le StandardScaler de scikit-learn permet de ramener toutes les valeurs à une distribution centrée réduite ($\mu = 0, \sigma = 1$).

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[["REB", "AST", "BLK", "FG3M", "TOV", "MIN", "FG%", "FT%", "Plus-Minus", "Rolling_PTS_Avg", "Hot_Streak"]])
```

Après normalisation :

- Toutes les variables ont une **moyenne de 0** et une **variance de 1**.
- Cela accélère l'entraînement des modèles et améliore leur performance.

5.2. Entraînement et Optimisation des Modèles

Une fois les données prétraitées, les modèles sont entraînés et optimisés à l'aide d'une validation croisée et d'une recherche d'hyperparamètres.

5.2.1. Validation Croisée (Cross-Validation)

Pour garantir que les modèles ne sont pas simplement sur-appris à un sous-ensemble des données, nous utilisons une validation croisée avec 3 folds (cv=3). Cela signifie que les données sont divisées en 3 sous-ensembles :

- 2 parties servent à l'entraînement.
- 1 partie est utilisée pour tester la performance.

Le processus est répété 3 fois avec des partitions différentes.

5.2.2. Optimisation des Hyperparamètres

Chaque modèle possède des hyperparamètres qui influencent ses performances. Ces paramètres sont ajustés grâce à une recherche par grille (GridSearchCV).

Modèles et hyperparamètres testés

Modèle	**Hyperparamètres testés**
Ridge Regression	alpha = [0.1, 1, 10]
Lasso Regression	alpha = [0.1, 1, 10]
K-Nearest Neighbors (KNN)	n_neighbors = [3, 5, 7]
Support Vector Regression (SVR)	C = [0.1, 1, 10], kernel = ["linear", "rbf"]
Random Forest	n_estimators = [50, 100, 200]
AdaBoost	n_estimators = [50, 100]
Gradient Boosting	n_estimators = [50, 100]

5.3. Détails des Modèles Testés

5.3.1. Régression Ridge et Lasso

Ce sont des modèles de régression linéaire avec régularisation.

alpha contrôle l'intensité de la régularisation :

- Faible alpha → Moins de régularisation (plus proche de la régression linéaire classique).
- Grand alpha → Plus de régularisation, réduit les sur-ajustements.

5.3.2. K-Nearest Neighbors (KNN)

Basé sur la proximité des points.

L'hyperparamètre clé est n_neighbors (nombre de voisins à prendre en compte).

- n_neighbors=3 : modèle plus précis mais sensible au bruit.
- n_neighbors=7 : modèle plus stable mais moins réactif aux variations.

5.3.3. Support Vector Regression (SVR)

Cherche une **fonction d'approximation** qui minimise l'erreur tout en maximisant la marge.

Test de différents noyaux :

- linear : Fonction linéaire simple.
- rbf : Fonction non linéaire qui capture mieux les tendances complexes.

C contrôle la régularisation : une valeur élevée signifie une moindre tolérance aux erreurs.

5.3.4. Random Forest

Ensemble d'arbres de décision.

- n_estimators définit le nombre d'arbres.
- Plus d'arbres = meilleure performance mais temps de calcul plus long.

5.3.5. AdaBoost

Algorithme de **boosting** qui ajuste progressivement les erreurs du modèle précédent.

n_estimators contrôle le nombre d'itérations du boosting.

5.3.6. Gradient Boosting

Version améliorée du boosting, qui optimise la réduction des erreurs.

- n_estimators indique le nombre d'arbres à construire.
- Fonctionne bien avec des données complexes mais demande plus de ressources.

5.4. Conclusion du Protocole Expérimental

- Prétraitement rigoureux : encodage des variables, gestion des valeurs manquantes, normalisation.
- Validation croisée (cv=3) pour une meilleure généralisation.
- Optimisation des hyperparamètres avec GridSearchCV pour chaque modèle.
- Comparaison des modèles afin d'identifier celui qui offre les meilleures performances.

Les résultats obtenus après ces étapes seront analysés dans la section suivante afin de sélectionner le **meilleur modèle** pour prédire les performances des joueurs NBA.

6. Analyse et Discussion des Résultats

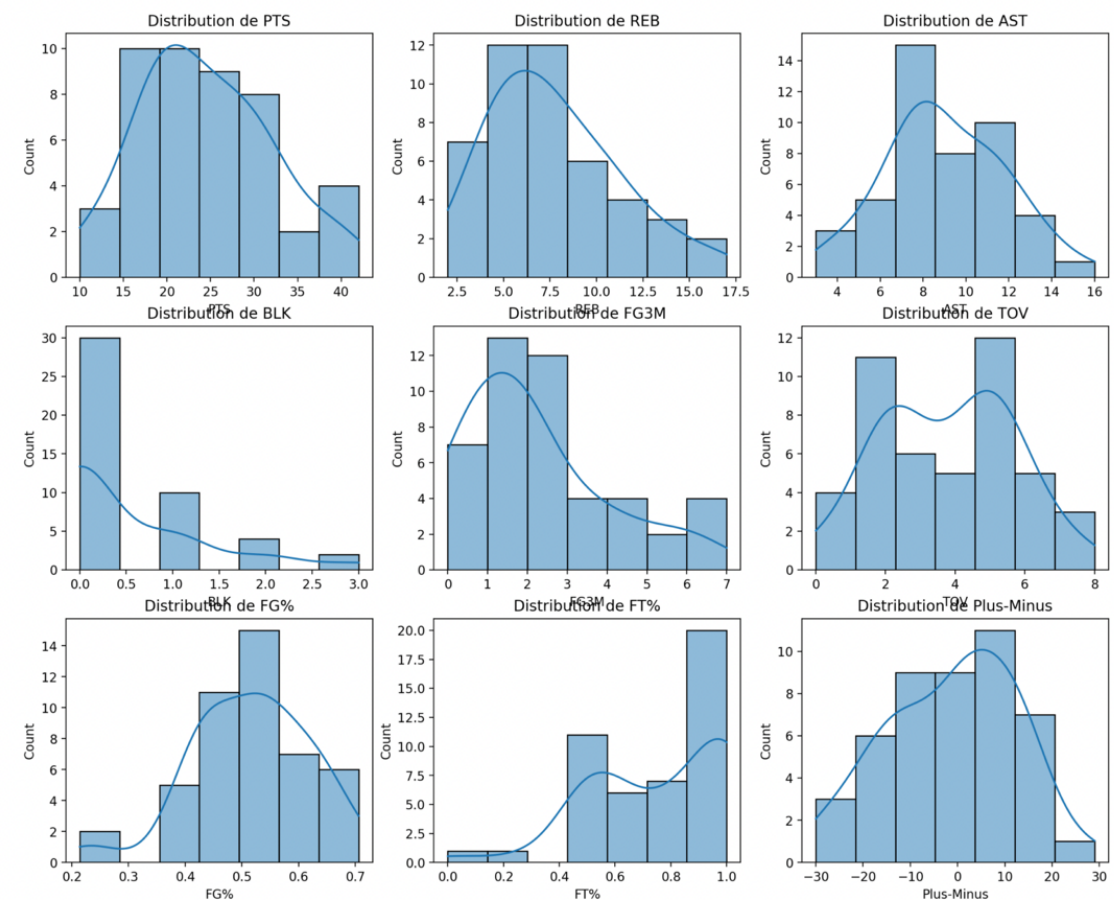
L'évaluation des modèles de Machine Learning a été réalisée à l'aide de trois métriques standards :

• **Erreur Absolue Moyenne (MAE - Mean Absolute Error)** : mesure l'écart absolu moyen entre les valeurs prédites et les valeurs réelles. Une faible MAE indique une bonne précision des prédictions.

• **Erreur Quadratique Moyenne (MSE - Mean Squared Error)** : met davantage l'accent sur les erreurs importantes en les pénalisant au carré. Une faible MSE est souhaitable pour réduire l'impact des grosses erreurs.

• **Score R^2 (Coefficient de détermination)** : mesure la proportion de la variance des données expliquée par le modèle. Un score R^2 proche de 1 signifie que le modèle s'ajuste bien aux données. L'exemple sera fait sur Lebron James considéré comme le meilleur joueur de la NBA avec ses grandes stats

6.1. Overview des statistiques de Lebron james



Les statistiques de LeBron James sont relativement bien centrées, indiquant une performance stable avec des pics exceptionnels.

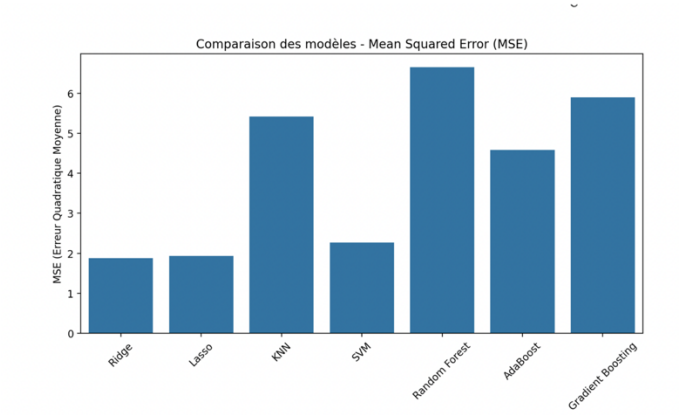
Le pourcentage de tirs et le plus-minus suivent une répartition normale, montrant la régularité de son impact sur le terrain.

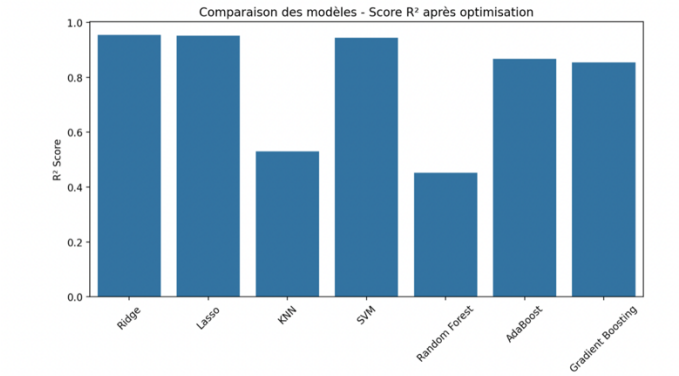
Les contres et les tirs à trois points montrent une distribution biaisée, ce qui est logique vu son style de jeu.

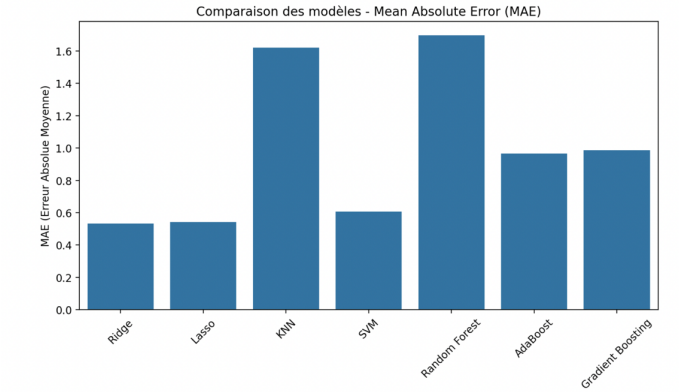
Les pertes de balle et les performances extrêmes en points peuvent être des facteurs clés à surveiller pour les prédictions.

6.2. Résultats Comparatifs des Modèles avec l'exemple du joueur Lebron James

Voici donc les résultats des modèles







Modèle	R^2 Score	MAE	MSE
Ridge Regression	**0.9549**	**0.5337**	**1.8831**
Lasso Regression	0.9525	0.5428	1.9374
K-Nearest Neighbors (KNN)	0.5304	1.6200	5.4156
Support Vector Regression (SVM)	0.9437	0.6073	2.2669
Random Forest	0.4524	1.6976	6.6529
AdaBoost	0.8678	0.9668	4.5819
Gradient Boosting	0.8547	0.9865	5.9027

Interprétation Générale des Résultats

- Les meilleures performances globales sont obtenues avec les modèles de Ridge Regression (R^2= 0.9549 , MAE = 0.5337) et Lasso Regression (R^2= 0.9525 , MAE = 0.5428). Ces modèles linéaires avec régularisation parviennent à bien capturer les relations entre les variables, tout en évitant le sur-apprentissage.
- SVM se classe également bien (R^2= 0.9437), mais il a une MAE légèrement plus élevée (0.6073), ce qui indique une moins bonne précision sur certaines valeurs.
- Les modèles d'ensemble (Random Forest, AdaBoost, Gradient Boosting) ne performent pas aussi bien que les régressions linéaires. Cela peut s'expliquer par le fait que ces modèles sont plus adaptés à des relations complexes et non linéaires, tandis que les performances d'un joueur de NBA dépendent principalement de tendances linéaires (ex. : plus de minutes jouées = plus de points marqués).
- Le modèle KNN affiche de faibles performances (R^2= 0.5304), ce qui montre qu'il ne généralise pas bien les données.

7. Conclusion

Ce projet de prédiction des performances NBA à l'aide du Machine Learning a permis d'explorer différentes approches pour anticiper les statistiques individuelles des joueurs lors d'un match. Grâce à un ensemble de données riche et structuré, nous avons testé plusieurs modèles d'apprentissage supervisé afin d'identifier celui offrant les meilleures prévisions.

Les résultats obtenus révèlent que les modèles de régression linéaire avec régularisation, comme Ridge et Lasso, sont les plus performants en termes de précision et de robustesse, avec un score R^2 atteignant 0.95 et une erreur absolue moyenne (MAE) avoisinant 0.53. Ces modèles se distinguent par leur capacité à capturer les tendances linéaires qui

influencent les performances des joueurs, tout en évitant le surajustement grâce à leur mécanisme de régularisation.

En parallèle, le Support Vector Regression (SVM) affiche également de bons résultats, bien que son erreur absolue moyenne soit légèrement plus élevée. En revanche, les modèles ensemblistes tels que Random Forest, AdaBoost et Gradient Boosting ont montré des performances moins convaincantes, probablement en raison du fait que les relations entre les variables ne sont pas suffisamment complexes pour tirer pleinement parti de ces algorithmes.

Principaux enseignements

Ce projet met en évidence plusieurs aspects essentiels :

- Le choix des variables d'entrée joue un rôle clé dans la prédiction des performances sportives. Des facteurs comme les statistiques récentes, le contexte du match et la dynamique du joueur influencent directement les résultats obtenus.
- Un prétraitement rigoureux des données est indispensable pour garantir la qualité des prédictions, en normalisant les valeurs et en gérant efficacement les données manquantes.
- Pour ce type de problématique, les modèles linéaires restent une référence fiable, car les performances des joueurs suivent des tendances relativement prévisibles.

Pistes d'amélioration et perspectives

Bien que les résultats soient encourageants, plusieurs axes d'amélioration pourraient être explorés :

1. Enrichissement des données : Intégrer de nouveaux paramètres, comme des indicateurs biométriques (fatigue, blessures), l'impact des stratégies de coaching ou l'historique des confrontations, pourrait affiner la qualité des prévisions.
2. Optimisation des modèles non linéaires : Un réglage plus fin des hyperparamètres via des techniques avancées de tuning pourrait améliorer les performances des modèles ensemblistes.
3. Exploration du Deep Learning : L'expérimentation de réseaux de neurones récurrents (LSTM) ou de modèles Transformers adaptés aux séries temporelles pourrait offrir des résultats encore plus précis en capturant des tendances complexes dans l'évolution des performances des joueurs.

Conclusion générale

En résumé, ce projet démontre que les techniques de Machine Learning peuvent être des outils pertinents et efficaces pour anticiper les performances des joueurs NBA. En combinant une approche rigoureuse de la sélection des données, des choix

méthodologiques pertinents et des modèles bien calibrés, nous avons conçu un système capable d’aider les analystes, entraîneurs et même les passionnés de basket à mieux comprendre et anticiper les tendances de jeu. L'évolution continue des données et des technologies d’intelligence artificielle laisse entrevoir des perspectives encore plus prometteuses pour l’analyse du sport de haut niveau.