Document Technique
Prédiction de la Longévité des Joueurs en NBA

Ibtissam LOUKILI

Table des matières

1 Contexte

Dans le monde ultra-compétitif de la NBA, les investisseurs doivent prendre des décisions stratégiques concernant le recrutement et le développement des jeunes talents. Identifier les joueurs avec un fort potentiel est essentiel pour garantir un retour sur investissement optimal et éviter des erreurs coûteuses.

L'évaluation des prospects repose aujourd'hui sur l'expérience des recruteurs et sur des statistiques classiques. Cependant, cette approche reste subjective et ne permet pas de prédire de manière fiable la longévité d'un joueur dans la ligue.

Le risque est double :

- **Manquer un futur talent** et laisser passer une opportunité d'investissement.
- **Investir sur un joueur dont la carrière sera courte**, entraînant des pertes financières importantes.

.

C'est dans ce contexte que ce projet vise à intégrer une approche analytique et prédictive basée sur le *Machine Learning*. À travers l'analyse de statistiques sportives détaillées, nous allons concevoir un modèle permettant de prédire si un joueur aura une carrière de plus de 5 ans en NBA.

Les utilisateurs finaux de cette solution sont principalement :

- **Les investisseurs NBA**, cherchant à maximiser leurs profits en sélectionnant les joueurs les plus prometteurs.
- **Les recruteurs et analystes**, qui pourront s'appuyer sur un modèle prédictif pour appuyer leurs décisions.

2 Objectif

L'objectif principal de ce projet est de développer un modèle de classification performant capable de prédire si un joueur aura une carrière de plus de 5 ans en NBA.

Pour atteindre cet objectif, plusieurs étapes méthodologiques sont mises en place :

1. Prétraitement et Exploration des Données
   - Nettoyage des données : gestion des valeurs manquantes, des valeurs aberrantes et des erreurs de saisie.
   - Analyse exploratoire pour comprendre la distribution des variables et l'équilibre des classes.

2. Entraînement des Modèles
   - Test de plusieurs algorithmes de classification.
   - Optimisation des hyperparamètres pour renforcer la robustesse et la précision des prédictions.

3. Évaluation des Performances et Sélection du Modèle Final
   - Comparaison des modèles selon des métriques adaptées au déséquilibre des classes (*F1-score, recall, précision*).

4. Déploiement et Intégration
   • Développement d'un Webservice sous forme d'**API REST**, permettant d'effectuer des prédictions en temps réel pour un joueur donné.

3 Prétraitement des Données

Cette section détaille les étapes de préparation des données avant l'entraînement du modèle. L'objectif est d'assurer la qualité des données en détectant et en corrigeant les incohérences.

3.1 Statistiques Incluses dans le Jeu de Données

Le jeu de données contient plusieurs statistiques relatives aux performances des joueurs en NBA. Voici la liste des principales variables incluses dans le DataFrame :

— *GP (Games Played)* : Nombre total de matchs joués par le joueur.
— *MIN (Minutes)* : Temps moyen passé sur le terrain par match.
— *PTS (Points)* : Moyenne de points marqués par match.
— *FGM (Field Goals Made)* : Nombre total de paniers marqués.
— *FGA (Field Goals Attempted)* : Nombre total de tirs tentés par un joueur.
— *FG% (Field Goal Percentage)* : Pourcentage de réussite aux tirs (*FGM / FGA*).
— *3P Made (Three-Point Made)* : Nombre de tirs à trois points réussis.
— *3PA (Three-Point Attempts)* : Nombre total de tentatives de tirs à trois points.
— *3P% (Three-Point Percentage)* : Pourcentage de réussite aux tirs à trois points (*3P Made / 3PA*).
— *FTM (Free Throws Made)* : Nombre de lancers francs réussis.
— *FTA (Free Throws Attempted)* : Nombre total de lancers francs tentés.
— *FT% (Free Throw Percentage)* : Pourcentage de réussite aux lancers francs (*FTM / FTA*).
— *OREB (Offensive Rebounds)* : Nombre de rebonds offensifs, c'est-à-dire de ballons récupérés après un tir manqué de son équipe.
— *DREB (Defensive Rebounds)* : Nombre de rebonds défensifs, soit de ballons récupérés après un tir manqué de l'équipe adverse.
— *REB (Total Rebounds)* : Nombre total de rebonds pris par un joueur, défini comme :

$$REB = OREB + DREB \tag{1}$$

— *AST (Assists)* : Nombre de passes décisives menant directement à un panier marqué.
— *STL (Steals)* : Nombre de ballons volés à l'adversaire.
— *BLK (Blocks)* : Nombre de tirs adverses bloqués.
— *TOV (Turnovers)* : Nombre de fois où un joueur perd la possession du ballon.

Ces variables constituent l'ensemble des caractéristiques utilisées pour l'entraînement du modèle de classification. Elles permettent de capturer les performances individuelles des joueurs afin d'optimiser la prédiction de leur longévité en NBA.

3.2 Gestion des valeurs aberrantes (Outliers)

Dans notre jeu de données, nous avons identifié deux types de valeurs aberrantes :

3.2.1 Les valeurs extrêmes

Les valeurs extrêmes correspondent à des performances très élevées ou très faibles par rapport à la distribution générale des données. Dans le cadre de la NBA, ces valeurs sont souvent légitimes, car il est fréquent d'observer des joueurs aux performances exceptionnelles (très bonnes ou très mauvaises). Nous avons donc choisi **de ne pas les supprimer**, afin de préserver la diversité des profils de joueurs et d'améliorer la représentativité du modèle.

Pourquoi ne pas supprimer ces valeurs ? Éliminer ces joueurs entraînerait une perte d'informations importantes et pourrait biaiser le modèle en l'empêchant d'apprendre des cas extrêmes.

3.2.2 Les valeurs impossibles

Certaines valeurs ne respectent pas les règles fondamentales du basket-ball et doivent être corrigées. Pour les identifier, nous avons appliqué plusieurs contraintes :
- Contraintes sur les pourcentages (compris entre 0 et 100 %)
- Contraintes NBA (règles fondamentales du jeu)

Après suppression de ces valeurs impossibles, la taille du dataset est passée de **1328 à 813 entrées**, soit une réduction significative de 40 % des données. Plutôt que de supprimer ces données, nous avons préféré appliquer des corrections automatiques pour ajuster les valeurs erronées.

En appliquant ces corrections, nous conservons un maximum d'informations tout en garantissant l'intégrité des données, ce qui renforce la fiabilité du modèle.

La suppression des valeurs aberrantes aurait conduit à une réduction drastique des données (**1328 → 813 observations**), ce qui aurait pu nuire à l'apprentissage du modèle. Au lieu de supprimer ces valeurs, nous avons choisi de les analyser et de **corriger celles qui étaient incohérentes** avec les règles NBA. La correction garantit un jeu de données propre tout en **préservant sa représentativité**, permettant au modèle de bien apprendre les cas réels.

4 Analyse Exploratoire des Données (EDA)

L'analyse exploratoire des données (EDA) permet de mieux comprendre la structure du jeu de données avant l'entraînement du modèle. Cette étape comprend l'étude de l'équilibre des classes, la visualisation statistique des variables et la sélection des caractéristiques les plus pertinentes.

4.1 Analyse de l'Équilibre des Classes

L'équilibre des classes est un facteur déterminant dans la performance du modèle de classification. Un déséquilibre important entre les classes peut biaiser l'apprentissage et influencer négativement la qualité des prédictions.

Lorsque les données sont fortement déséquilibrées, le modèle a tendance à privilégier la classe majoritaire, ce qui peut poser plusieurs problèmes :

- La mise à jour des poids du modèle est dominée par la classe majoritaire, tandis que la classe minoritaire influence peu l'optimisation.
- Le gradient de la classe minoritaire est beaucoup plus faible que celui de la classe majoritaire, ce qui ralentit l'apprentissage pour cette classe.
- La convergence du modèle se fait rapidement en faveur de la classe majoritaire, au détriment de la classe minoritaire.

Pour pallier ce problème, nous avons appliqué une technique de pondération de la fonction de perte. Cette méthode consiste à attribuer un poids plus important à la classe minoritaire afin d'équilibrer l'influence des deux classes sur le processus d'apprentissage. Ainsi, l'erreur de la classe minoritaire est mieux prise en compte, ce qui améliore la capacité du modèle à la détecter correctement.

Nous allons tester également, SMOTE une technique qui génère artificiellement des échantillons similaires à la classe minoritaire, ce qui permet au modèle d'avoir plus d'exemples pour apprendre à bien prédire les joueurs à carrière courte.

4.2 Visualisations et Corrélations

Afin d'extraire des informations pertinentes du jeu de données, plusieurs visualisations statistiques et analyses de corrélation ont été réalisées :

- Distribution des variables et étude de la dispersion des données.
- Analyse des corrélations entre les différentes caractéristiques pour détecter les relations significatives.

4.3 Sélection des Variables (Feature Selection)

L'identification des variables les plus pertinentes est une étape clé pour améliorer la robustesse et la performance du modèle. Cette sélection peut être effectuée soit manuellement, soit automatiquement en fonction du modèle utilisé :

• **Méthode Manuelle** : Sélection basée sur l'analyse des corrélations et des connaissances métier.
• **Méthode Automatique** : Certains modèles, tels que *Random Forest* et *XGBoost*, intègrent un mécanisme de sélection des caractéristiques basé sur l'importance des variables dans leurs arbres de décision.

L'objectif est de conserver uniquement les variables ayant un impact significatif sur la prédiction de la longévité des joueurs en NBA, tout en éliminant celles qui sont redondantes ou non pertinentes.

5 Modélisation et Sélection du Modèle

5.1 Choix de la Métrique d'Évaluation

Dans le cadre de ce projet, la sélection des métriques d'évaluation est cruciale, car les erreurs de classification peuvent avoir un impact stratégique et financier significatif. Étant donné que notre dataset est **déséquilibré**, une simple optimisation du **recall** pourrait être trompeuse. En effet, un modèle peut **maximiser le recall au détriment de la précision**, ce qui entraînerait une augmentation des **faux positifs** (prédire qu'un joueur aura une longue carrière alors que ce ne sera pas le cas).

5.1.1 Équilibre entre Recall et Précision

• **Optimiser uniquement le recall de la classe 1 (joueurs ayant une longue carrière)** peut entraîner un excès de faux positifs, ce qui signifie que des joueurs sans réel potentiel seraient retenus à tort.
• **Un recall trop faible pour la classe 0 (joueurs avec une carrière courte)** signifie que le modèle ne détecte pas suffisamment bien les joueurs qui échoueront, ce qui entraîne des erreurs stratégiques et des décisions d'investissement non optimales.

5.1.2 Impact des Erreurs de Classification

Faux négatifs (False Negatives - FN) : *Manquer des joueurs ayant un fort potentiel (classe 1 prédite en classe 0).*
• **Conséquence :** Perte d'opportunités d'investissement sur des talents qui auraient pu exceller.

Faux positifs (False Positives - FP) : *Prédire qu'un joueur aura une carrière longue alors qu'il échouera (classe 0 prédite en classe 1).*
• **Conséquence :** Investir des ressources sur des joueurs qui ne réussiront pas en NBA.

Notre modèle ne doit **pas seulement prédire les joueurs qui réussiront**, mais aussi **éviter les erreurs de sélection coûteuses**. Pour cela, nous devons optimiser **le F1-score des deux classes** :

• **F1-score classe 1** : Maximiser la détection des talents sans générer trop de faux positifs.
• **F1-score classe 0** : S'assurer que le modèle identifie correctement les joueurs qui échoueront, afin d'éviter de mauvais investissements.

L'objectif n'est pas uniquement d'identifier les joueurs prometteurs, mais aussi d'**éviter de sélectionner des joueurs qui ne performeront pas**. Cela permet de garantir une **gestion efficace des ressources** et une **prise de décision optimisée** pour les investisseurs et les franchises NBA.

5.2 La Fonction de Scoring

L'objectif de la fonction de scoring est d'évaluer la performance des modèles de classification de manière rigoureuse et alignée avec les enjeux du projet. La version initiale se concentrait uniquement sur le **recall global**, ce qui est insuffisant dans un contexte où une classification erronée peut entraîner des **décisions stratégiques coûteuses**.

Afin d'améliorer l'évaluation, nous avons mis en place un calcul détaillé des **précision, recall et F1-score pour chaque classe** (0 et 1). Étant donné que le dataset est **fortement déséquilibré**, il est crucial de mesurer la performance pour chaque classe indépendamment.

• **Optimiser uniquement le recall de la classe 1** peut engendrer trop de **faux positifs**, conduisant à de mauvais investissements.
• **Optimiser uniquement la précision de la classe 1** peut engendrer trop de **faux négatifs**, causant une perte de talents prometteurs.

Il est donc nécessaire de trouver **un équilibre entre précision et recall**. Pour cela, nous avons utilisé le **F1-score** qui permet d'éviter une mauvaise répartition des erreurs.

L'objectif principal est d'optimiser **à la fois la précision et le recall pour les deux classes** afin de garantir un modèle **fiable et exploitable pour la prise de décision en entreprise**.

• La **matrice de confusion** donne une vue d'ensemble, mais elle ne permet pas de comprendre en détail comment le modèle se comporte sur chaque classe.
• Le **rapport de classification** (`classification_report`) permet une évaluation plus détaillée, par classe et par métrique.
• Ce rapport est **indispensable** pour comparer les modèles et justifier les choix effectués.

5.3 Optimisation de la Fonction de Scoring

Afin d'affiner l'analyse des performances des modèles, nous avons développé deux versions améliorées de la fonction de scoring précédente. L'objectif principal de ces améliorations est de **mieux évaluer les performances des modèles sur un dataset**

déséquilibré, en prenant en compte les **deux classes séparément** et en optimisant les hyperparamètres pour améliorer la qualité des prédictions.

5.3.1 Version 1 : Optimisation des Hyperparamètres avec GridSearchCV

Dans la version précédente, les modèles étaient testés avec des paramètres par défaut. Cette nouvelle version introduit une optimisation des hyperparamètres, entraînant une amélioration globale des performances. Pour cela, nous avons intégré **GridSearchCV** dans la fonction de scoring.

- GridSearchCV permet de **tester différentes combinaisons d'hyperparamètres** sur une grille préétablie.
- Cette amélioration nous a permis de **trouver un meilleur équilibre entre recall et précision** pour les deux classes.
- L'optimisation des hyperparamètres permet d'éviter les **problèmes de sous-ajustement ou de surajustement**.

5.3.2 Version 2 : Optimisation Avancée avec Optuna

Bien que GridSearchCV ait permis d'améliorer les performances des modèles, cette approche présente une **limitation majeure** : elle repose sur une grille définie manuellement et explore **toutes les combinaisons possibles** sans prendre en compte la performance des essais précédents. Ce processus peut être **très coûteux en temps de calcul** et ne garantit pas toujours une exploration efficace de l'espace des hyperparamètres.

Pour aller plus loin, nous avons développé une seconde version de notre fonction de scoring en intégrant **Optuna**, un algorithme avancé d'optimisation des hyperparamètres.

- Contrairement à GridSearchCV, Optuna repose sur une approche **plus intelligente et adaptative**, où les hyperparamètres sont ajustés dynamiquement en fonction des performances obtenues sur les essais précédents.
- Cette approche permet de **trouver les meilleures combinaisons d'hyperparamètres de manière plus efficace et rapide**.
- Optuna a permis d'obtenir des performances **au moins aussi bonnes, voire supérieures**, à celles obtenues avec GridSearchCV.
- Cet outil est particulièrement adapté aux modèles **complexes et gourmands en ressources**, comme **XGBoost ou Balanced Random Forest**, qui nécessitent un ajustement précis pour maximiser leurs performances.

L'amélioration de la fonction de scoring a permis une **meilleure évaluation des modèles** en mettant en avant des métriques adaptées au dataset déséquilibré. Grâce à GridSearchCV et Optuna, nous avons pu affiner les hyperparamètres et **trouver un compromis optimal entre recall et précision** pour les deux classes. Cette approche garantit un modèle **fiable et robuste** pour la prise de décision.

5.4 Choix des Modèles de Classification

Le choix du modèle de classification est une étape clé pour garantir une prédiction fiable des joueurs ayant une carrière de plus de 5 ans en NBA. Étant donné les spécificités du dataset (**déséquilibre des classes, valeurs extrêmes et un nombre limité d'échantillons**), trois modèles ont été sélectionnés pour leur capacité à gérer ces contraintes :

- Random Forest (RF)
- Balanced Random Forest Classifier (BalancedRF)
- XGBoost Classifier

Ces modèles ont été choisis en fonction de leur capacité à gérer des datasets déséquilibrés, leur robustesse face aux valeurs extrêmes et leur performance générale.

5.4.1 Random Forest Classifier (RF)

Le Random Forest est un modèle basé sur un ensemble d'arbres de décision.

- Interprétabilité élevée : Permet d'identifier facilement les variables les plus influentes.
- Robustesse aux valeurs extrêmes et faible risque de sur-ajustement.

Hyperparamètres clés à optimiser
- **n_estimators** : Nombre d'arbres.
- **max_depth** : Profondeur des arbres.
- **min_samples_split, min_samples_leaf** : Contrôle la division des nœuds.
- **max_features** : Nombre de features sélectionnées par split.

5.4.2 Balanced Random Forest Classifier (BalancedRF)

Le Balanced Random Forest est une version améliorée du Random Forest classique, conçue spécifiquement pour traiter les datasets déséquilibrés. Il ajuste dynamiquement l'échantillonnage des classes afin d'éviter un biais en faveur de la classe majoritaire.

- Gestion efficace du déséquilibre des classes grâce à un échantillonnage adaptatif.
- Interprétabilité élevée : Permet d'identifier facilement les variables les plus influentes.
- Robustesse aux valeurs extrêmes et faible risque de sur-ajustement.

Hyperparamètres clés à optimiser
- **n_estimators** : Nombre d'arbres.
- **max_depth** : Profondeur des arbres.
- **min_samples_split, min_samples_leaf** : Contrôle la division des nœuds.
- **max_features** : Nombre de features sélectionnées par split.

5.4.3 XGBoost Classifier

XGBoost est un algorithme de boosting particulièrement efficace pour les datasets déséquilibrés. Grâce à la pondération des classes avec `scale_pos_weight`, il accorde plus d'importance à la classe minoritaire et permet une meilleure séparation des joueurs prometteurs et des joueurs à risque.

- Un des modèles les plus performants pour la classification déséquilibrée.
- Capacité à gérer les valeurs extrêmes et les données bruitées.
- Boosting adaptatif qui corrige progressivement les erreurs du modèle.

Hyperparamètres clés à optimiser

- **n_estimators** : Nombre d'arbres.
- **max_depth** : Profondeur des arbres.
- **learning_rate** : Taux d'apprentissage.
- **subsample, colsample_bytree** : Contrôle la diversité des arbres.
- **scale_pos_weight** : Pour équilibrer les classes.

5.5 Évaluation des Modèles & Résultats

L'objectif principal de ce projet est de prédire avec précision les joueurs ayant une carrière de plus de 5 ans en NBA, tout en prenant en compte les risques liés aux erreurs de classification. Plus précisément, nous cherchons à :

- **Minimiser les faux positifs (FP)** → *Éviter d'investir sur des joueurs qui ne performeront pas* (Classe 0 mal classée en Classe 1).
- **Minimiser les faux négatifs (FN)** → *Ne pas manquer des talents prometteurs* (Classe 1 mal classée en Classe 0).
- **Équilibrer Précision et Recall** afin de garantir une prise de décision optimisée.

L'évaluation des modèles est donc basée sur plusieurs métriques essentielles, dont :

- **Précision** : capacité du modèle à ne pas générer de faux positifs.
- **Recall** : capacité du modèle à détecter correctement les joueurs ayant une longue carrière.
- **F1-score** : mesure de compromis entre précision et recall.
- **Matrice de confusion** : visualisation des erreurs de classification.

6 Intégration du Modèle dans un Webservice REST

6.1 Objectif de l'API REST

Afin de permettre une utilisation pratique du modèle de prédiction, nous avons intégré notre classificateur sous forme d'un webservice REST. Cette API permet d'envoyer les statistiques d'un joueur et de recevoir une prédiction indiquant si ce joueur aura une carrière de plus de 5 ans en NBA.

6.2 Technologies Utilisées

L'API a été développée avec les technologies suivantes :
— **FastAPI** : Framework Python performant pour la création d'API REST.
— **Uvicorn** : Serveur ASGI léger permettant de déployer FastAPI.
— **Joblib** : Bibliothèque utilisée pour charger le modèle entraîné.
— **Jinja2** : Moteur de templating pour afficher les résultats de prédiction sur une interface web.

6.3 Architecture de l'API

L'API est constituée de trois composants principaux :

1. **Un endpoint POST "/predict"** permettant de soumettre les statistiques d'un joueur et d'obtenir une prédiction.
2. **Un formulaire web** permettant de saisir les données du joueur via une interface utilisateur.
3. **Un moteur de templates** affichant la prédiction sur une page HTML.

6.4 Interface Web pour l'Utilisateur

L'interface utilisateur est implémentée à l'aide d'un formulaire HTML permettant de saisir les statistiques du joueur.

Une fois la prédiction effectuée, les résultats sont affichés sur une nouvelle page HTML.

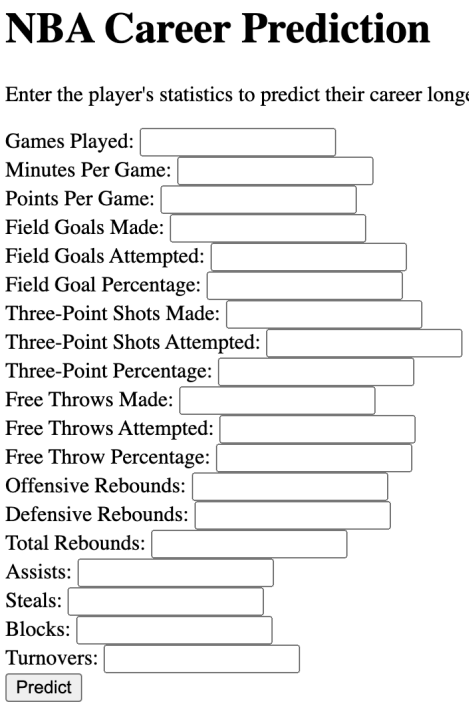

FIGURE 1 – Interface pour saisir les statistiques du joueur

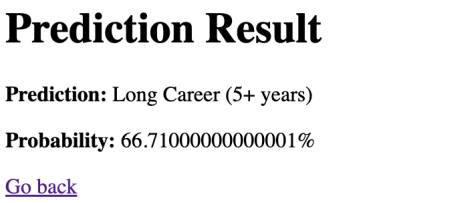

FIGURE 2 – Interface d'affichage des résultas