Note Méthodologique

Table des matières

[I. Introduction 2](#_Toc178762117)

[I.1. Résultats de classification : erreurs et classifications correctes 4](#_Toc178762118)

[II. Méthodologie 5](#_Toc178762119)

[II.1. Préparation des données 5](#_Toc178762120)

[II.2. Développement des modèles de scoring 6](#_Toc178762121)

[II.3. Evaluation des modèles 7](#_Toc178762122)

[II.3.1. Choix des métriques d'évaluation dans le contexte de la carrière NBA 7](#_Toc178762123)

[II.3.2. Ajustement du seuil de décision 8](#_Toc178762124)

[II.4. Sélection du modèle final 9](#_Toc178762125)

[II.4.1. Performance sur le dataset « Train » 9](#_Toc178762126)

[II.4.2. Performance sur le dataset « Test » 11](#_Toc178762127)

[II.5. Pour aller plus loin 13](#_Toc178762128)

[II.5.1. Quels sont les coûts de l’investisseur ? 13](#_Toc178762129)

[II.5.2. Calcul du Coût Metier Total (CMT) 14](#_Toc178762130)

[II.5.3. Assemblage des modèles (Ensemble Learning) 15](#_Toc178762131)

[III. Interprétabilité globale du modèle 16](#_Toc178762132)

[III.1. Interprétabilité Globale 16](#_Toc178762133)

[IV. Intégration du Classificateur dans un Web Service 17](#_Toc178762134)

[IV.1. En local 17](#_Toc178762135)

[IV.1.1. Serveur Flask 18](#_Toc178762136)

[IV.1.2. Requête cURL 18](#_Toc178762137)

[IV.1. Sur le cloud (PythonAnywhere) 19](#_Toc178762138)

[IV.1.1. Structure du Projet 19](#_Toc178762139)

[IV.1.2. Détails de l'Implémentation 19](#_Toc178762140)

[IV.1.3. Déploiement sur PythonAnywhere 20](#_Toc178762141)

[IV.1.4. Quelques captures d’écran 21](#_Toc178762142)

[Conclusion : Développement d'un Modèle de classification binaire de la longévité des carrières des joueurs de NBA. 22](#_Toc178762143)

# I. Introduction

Ce document présente la méthodologie employée pour **développer un** **modèle de classification binaire permettant de prédire la longévité (carrière) des joueurs débutants en NBA, et d’évaluer la pertinence d’un investissement sur ces joueurs.** Le projet vise à répondre à deux problématiques principales :

1. **Gérer les déséquilibres de classes** : La population de données présente un déséquilibre entre les joueurs ayant une carrière longue (>= 5 ans) et ceux ayant une carrière plus courte (< 5 ans). Cet écart peut affecter la performance des modèles de machine learning, en particulier sur la capacité de prédire correctement les joueurs ayant un fort potentiel de longévité.
2. **Minimiser les coûts d’investissement associés aux erreurs de prédiction** : Les erreurs de prédiction, qu’il s’agisse de faux négatifs (ne pas investir sur un joueur qui aurait eu une carrière longue) ou de faux positifs (investir sur un joueur avec une carrière courte), peuvent entraîner des pertes financières importantes pour les investisseurs. L'objectif est donc de minimiser ces erreurs en fournissant un modèle fiable.

L’ensemble de données utilisé comprend diverses statistiques sportives des joueurs débutants de la NBA, telles que les points par match, les minutes jouées, les rebonds, et d'autres indicateurs de performance clés. Le modèle est destiné à guider les investisseurs cherchant à identifier les talents prometteurs susceptibles de durer plus de cinq ans dans la ligue.

## I.1. Résultats de classification : erreurs et classifications correctes

Dans le cadre de ce projet de prédiction de la longévité des joueurs en NBA, les erreurs de classification peuvent être regroupées en quatre catégories principales :

* **Vrai positif (True Positive - TP)** : Le modèle prédit correctement qu'un joueur aura une longue carrière (>= 5 ans). Cela signifie qu’un investissement sur ce joueur est justifié, et l'investisseur aurait potentiellement un bon retour sur investissement.
* **Vrai négatif (True Negative - TN)** : Le modèle prédit correctement qu’un joueur aura une carrière courte (< 5 ans). Dans ce cas, l’investisseur fait le bon choix de ne pas investir, évitant ainsi des pertes financières sur un joueur qui n’aurait pas eu de succès durable.
* **Faux positif (False Positive - FP)** : Le modèle prédit qu'un joueur aura une longue carrière (>= 5 ans), mais en réalité, sa carrière est plus courte (< 5 ans). Cela représente un **risque financier**, car l'investisseur engage des ressources sur un joueur qui ne génère pas de retour à long terme. Ce type d’erreur entraîne un coût d’opportunité et une perte potentielle pour l'investisseur.
* **Faux négatif (False Negative - FN)** : Le modèle prédit qu’un joueur aura une carrière courte (< 5 ans), alors qu’en réalité, il aurait eu une carrière longue (>= 5 ans). Cela représente **une opportunité manquée** pour l'investisseur, qui choisit de ne pas investir dans un joueur qui aurait pourtant généré des retours significatifs. Les faux négatifs sont particulièrement coûteux, car les talents prometteurs sont négligés.

**Conséquences dans le contexte d'investissement :**

* **Faux positifs** (FP) entraînent un **coût direct** en ressources investies dans un joueur qui n’apportera pas de retour financier sur le long terme.
* **Faux négatifs** (FN) entraînent un **coût d'opportunité**, où l'investisseur passe à côté de joueurs potentiellement très prometteurs.

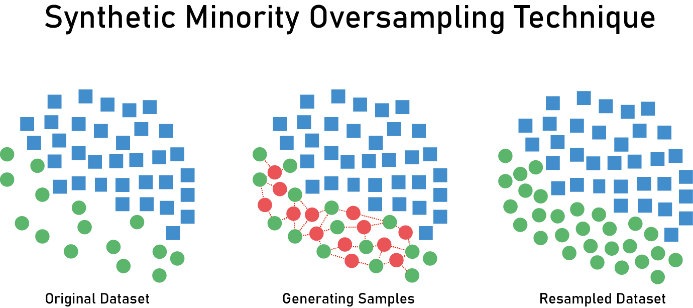
**L'objectif final est de minimiser les erreurs de faux positifs et de faux négatifs pour maximiser le potentiel de retour sur investissement, tout en réduisant les risques financiers.**

# II. Méthodologie

## II.1. Préparation des données

* **Collecte et nettoyage des données :** Nettoyer les données pour éliminer les valeurs manquantes, les incohérences, les duplicates, et les erreurs.
* **Exploration et analyse des données :** Explorer les données pour identifier les distributions des features (et Target), les relations entre les features et les potentiels biais présents.
* **Début de feature engineering avec réduction dimensionelle** (PCA ou T-SNE en 2D pour une optique de visualisation : Avec les features actuelles, peut-on observer sur le scatter plot 2D une distinction entre les deux classes ?)
* **Gestion des déséquilibres de classes (Target) :** Utiliser deux techniques principales pour gérer les déséquilibres de clients :

* **SMOTE (Synthetic Minority Oversampling Technique) :** Générer synthétiquement de nouveaux exemples pour la classe minoritaire (Carrière > 5 ans) afin de créer un ensemble de données plus équilibré.



* **Ajustement du poids des classes :** Ajuster les poids attribués aux exemples de chaque classe lors de l'apprentissage du modèle pour tenir compte de leur importance relative.

## II.2. Développement des modèles de scoring

* **Sélection des modèles :** Choix d’une variété de modèles d'apprentissage automatique adaptés à la classification binaire tels que :
  + **SVC** : Classificateur à noyau qui cherche à séparer les classes par une hyperplane dans un espace de grande dimension.
  + **RandomForest** : Ensemble d'arbres de décision qui combine les prédictions de plusieurs arbres pour améliorer la précision et réduire le surapprentissage.
  + **LogisticRegression** : Modèle de régression qui prédit la probabilité d'appartenance à une classe en utilisant une fonction logistique.
  + **MLPClassifier** : Réseau de neurones multicouche qui apprend des représentations complexes à partir des données via l'apprentissage supervisé.
  + **GradientBoosting** : Algorithme d'optimisation qui construit des modèles de manière séquentielle pour corriger les erreurs des modèles précédents.
  + **XGBoost** : Version optimisée du gradient boosting, particulièrement efficace et rapide, avec des capacités de régularisation pour éviter le surajustement.
  + **AdaBoost** : Algorithme de boosting qui combine plusieurs classificateurs faibles pour créer un classificateur fort, en ajustant les poids des exemples mal classés.
  + **KNeighbors** : Classificateur basé sur la proximité qui classe les points en fonction des classes de leurs voisins les plus proches.
* **Apprentissage des modèles :** Entraîner chaque modèle sur l'ensemble de données nettoyée. Deux tests ont été utilisés :

## II.3. Evaluation des modèles

* **Définition des métriques d'évaluation :** Utiliser deux métriques pour évaluer la performance des modèles :
  + **Recall :** Mesure la capacité du modèle à identifier correctement les joueurs ayant une carrière de 5 ans ou plus (c’est-à-dire les talents prometteurs) parmi l'ensemble des joueurs. Cette métrique est cruciale pour s'assurer que les investisseurs ne passent pas à côté de joueurs à fort potentiel.
  + **AUC\_ROC (Area Under the ROC Curve) :** Mesure de la capacité du modèle à distinguer entre les joueurs ayant une carrière de 5 ans ou plus et ceux ayant une carrière plus courte sur l'ensemble des seuils de probabilité possibles. Cela permet d'évaluer le compromis entre la sensibilité (vrais positifs) et la spécificité (vrais négatifs) du modèle.
* **Comparaison des modèles :** Comparer les performances des différents modèles en se basant sur les métriques d'évaluation définies.
* **Split du dataset nettoyé en 80% Train et 20% Test.**

### II.3.1. Choix des métriques d'évaluation dans le contexte de la carrière NBA

Dans le contexte de la prévision des carrières des joueurs NBA, le choix entre la priorité accordée au **recall**, à l’**AUC ROC** ou à un **équilibre entre les deux** est essentiel et dépend des objectifs stratégiques.

1. **Prioriser le Recall** :
   * Un recall élevé indique que **le modèle est très efficace pour identifier les joueurs susceptibles de réussir une carrière de longue durée**. Cela est crucial pour les investisseurs, car manquer un joueur prometteur peut entraîner des pertes significatives.
2. **Mettre l'accent sur l'AUC ROC** :
   * Une AUC ROC signifie que **le modèle est bon pour distinguer entre les joueurs ayant des carrières longues et ceux avec des carrières plus courtes.** Cela peut réduire le risque d'investir dans des joueurs qui ne performent pas comme prévu, minimisant ainsi les pertes financières.
3. **Chercher un équilibre** :
   * Une approche équilibrée permet de maximiser à la fois la détection des talents tout en maintenant un faible taux de faux positifs. **Cela est recommandé si l'on souhaite identifier un nombre substantiel de joueurs potentiels sans surinvestir dans des talents incertains.**

**Conclusion**

Dans le contexte des investissements liés aux carrières NBA, il est préférable de **chercher un équilibre**.

### II.3.2. Ajustement du seuil de décision

Dans les modèles de classification binaire, la probabilité prédite par le modèle est généralement comparée à un seuil (souvent fixé à 0,5 par défaut) pour déterminer si une observation appartient à la classe 0 ou à la classe 1. Cependant, dans le cadre de ce projet, où les faux positifs et les faux négatifs entraînent des coûts financiers différents, **il est pertinent d'ajuster ce seuil pour optimiser le modèle en fonction des objectifs d'investissement.**

Pour déterminer le seuil optimal, j'ai tracé la courbe **Precision-Recall**. Cette courbe est particulièrement utile lorsque les classes sont déséquilibrées, ce qui est pertinent dans notre cas, car le nombre de joueurs qui restent plus de 5 ans dans la ligue est inférieur à ceux qui n'y parviennent pas.

L'image ci-dessous montre la courbe Precision-Recall avec l'aire sous la courbe (Average Precision (AP) = 0.83). Le seuil de décision final de 0,47 a été sélectionné en fonction d'un compromis entre la **précision** (limiter les faux positifs) et le **recall** (limiter les faux négatifs). Ce seuil est indiqué par le point rouge sur la courbe.

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

Description générée automatiquement

**L'ajustement du seuil a permis de trouver un compromis optimal pour réduire le nombre de faux positifs sans compromettre excessivement le recall des talents potentiels, maximisant ainsi le retour sur investissement pour les investisseurs.**

## II.4. Sélection du modèle final

### II.4.1. Performance sur le dataset « Train »

Ci-dessous le tableau récapitulatif des performances des différents classifieurs d’après les conditions initiales (Test 1 à 2) pour le dataset « Train ».

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Classifier** | **Recall** | **AUC ROC** | **Average** | **Optimal Threshold** |
| **Test1** | RandomForest | 0.77 | 0.83 | 0.80 | 0.48 |
| **Test1** | GradientBoosting | 0.79 | 0.80 | 0.79 | 0.43 |
| **Test1** | XGBoost | 0.73 | 0.82 | 0.78 | 0.48 |
| **Test1** | LogisticRegression | 0.77 | 0.77 | 0.77 | 0.46 |
| **Test2** | GradientBoosting | 0.75 | 0.80 | 0.77 | 0.47 |
| **Test2** | RandomForest | 0.72 | 0.82 | 0.77 | 0.50 |
| **Test2** | MLPClassifier | 0.74 | 0.77 | 0.75 | 0.46 |
| **Test2** | AdaBoost | 0.73 | 0.77 | 0.75 | 0.50 |
| **Test1** | SVC | 0.73 | 0.77 | 0.75 | 0.49 |
| **Test2** | XGBoost | 0.71 | 0.79 | 0.75 | 0.51 |
| **Test2** | SVC | 0.70 | 0.77 | 0.73 | 0.50 |
| **Test2** | LogisticRegression | 0.69 | 0.77 | 0.73 | 0.51 |
| **Test1** | MLPClassifier | 0.66 | 0.77 | 0.71 | 0.54 |
| **Test2** | KNeighbors | 0.65 | 0.76 | 0.70 | 0.60 |
| **Test1** | KNeighbors | 0.63 | 0.76 | 0.70 | 0.60 |
| **Test1** | AdaBoost | 0.63 | 0.74 | 0.69 | 0.50 |

**TOP 3 modèles (RECALL)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Classifier** | **Recall** | **AUC ROC** | **Average** | **Optimal**  **Threshold** |
| **Test1** | GradientBoosting | 0.79 | 0.80 | 0.79 | 0.43 |
| **Test1** | **RandomForest** | **0.77** | **0.83** | **0.80** | **0.48** |
| **Test1** | LogisticRegression | 0.77 | 0.77 | 0.77 | 0.46 |

**TOP 3 modèles (AUC ROC)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Classifier** | **Recall** | **AUC**  **ROC** | **Average** | **Optimal**  **Threshold** |
| **Test1** | **RandomForest** | **0.77** | **0.83** | **0.80** | **0.48** |
| **Test1** | XGBoost | 0.73 | 0.82 | 0.78 | 0.48 |
| **Test2** | RandomForest | 0.72 | 0.82 | 0.77 | 0.50 |

**TOP 3 modèles (EQUILIBRE)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test** | **Classifier** | **Recall** | **AUC**  **ROC** | **Average** | **Optimal**  **Threshold** |
| **Test1** | **RandomForest** | **0.77** | **0.83** | **0.80** | **0.48** |
| **Test1** | GradientBoosting | 0.79 | 0.80 | 0.79 | 0.43 |
| **Test1** | XGBoost | 0.73 | 0.82 | 0.78 | 0.48 |

**Conclusion sur le choix du modèle final :** RandomForest avec les conditions du Test1 (SMOTE + Feature Engineering).

**Confusion Matrix :**

### II.4.2. Performance sur le dataset « Test »

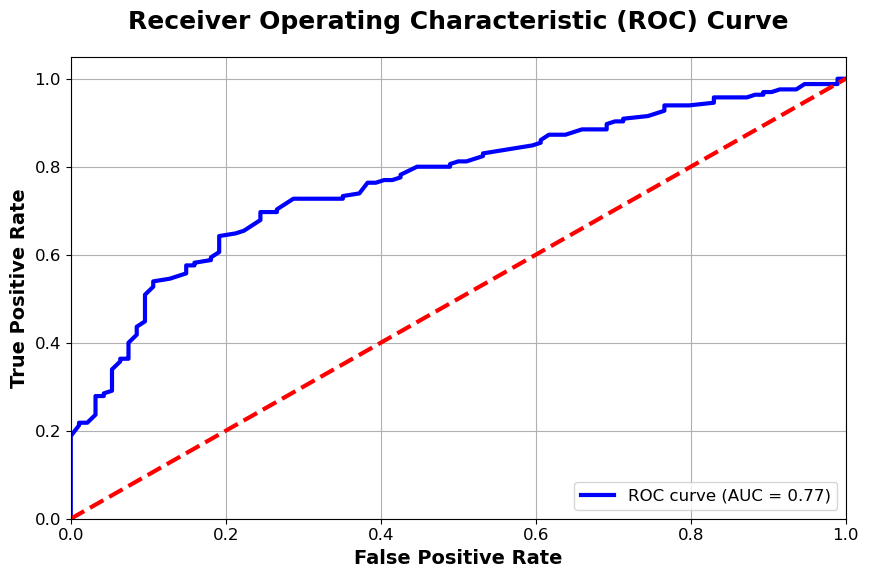
Dans cette partie, le modèle évalué est RandomForest qui a subi une opération d’hyperpameter tuning dans les conditions du Test1 (SMOTE + Feature Engineering).

Evaluation du modèle final sur le dataset « Test » :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

La courbe de ROC du meilleur modèle sur dataset « Test » est alors :



## II.5. Pour aller plus loin

L'introduction d'une métrique d’évaluation : Coût Métier Total (CMT) présente des avantages, en particulier dans un contexte financier :

* **Quantification de l'impact financier** : Le CMT permet de quantifier les conséquences financières des faux positifs et des faux négatifs. L'évaluation de la performance du modèle est ainsi plus pertinente et plus facile à mettre en œuvre pour les investisseurs.
* **Optimisation de la stratégie d'investissement** : En utilisant le CMT, il est possible d’évaluer différents modèles et de choisir celui qui minimise le risque financier, ce qui permet d'affiner la stratégie d'investissement.

### II.5.1. Quels sont les coûts de l’investisseur ?

* **Coût d'un Faux Positif (FP)** : C'est le montant que l'investisseur perd en pariant sur un joueur qui n’a pas la carrière attendue (< 5 ans).
  + **Salaire du joueur** (ou prime d’engagement).
  + **Coût d’opportunité** (argent investi qui aurait pu être utilisé ailleurs).
  + **Frais liés au scouting et à la formation**.
* **Coût d'un Faux Négatif (FN)** : C’est le montant que l'investisseur perd en ne pariant pas sur un joueur qui aurait eu une carrière longue (>= 5 ans).
  + **Revenus manqués** (ex. contrats, performances sportives générant des revenus).
  + **Coût d'opportunité** (ne pas avoir investi dans un joueur à fort potentiel).

### II.5.2. Calcul du Coût Metier Total (CMT)

Le CMT est une somme pondérée des coûts associés aux faux positifs et aux faux négatifs. L’idée est d’estimer combien chaque erreur coûte en termes financiers et de les additionner pour obtenir une évaluation du coût total des erreurs.

**Exemple :**

Imaginons qu’un expert dans le domaine a pu nous conseiller sur le sujet. Avec son aide, nous avons pu déterminer le coût d’un FP et d’un FN.

* Coût d’un FP : 2 millions d’euros
* Coût d’un FN : 5 millions d’euros

Maintenant, si après avoir testé le modèle, nous obtenons :

* 10 FP
* 4 FN

Alors le Coût Métier Total est de :

**Il est alors possible de comparer les modèles grâce aux pertes financières de l’investisseur. Le choix du modèle est celui qui minimise le CMT.**

**Note :**

Si nous sommes amenés à utiliser des datasets de taille différente par la suite, il est intéressant de normaliser la formule du CMT.

La nouvelle formule devient alors :

**Note 2 :**

Je ne rajouterai pas cette métrique ni son optimisation dans le code puisque je ne suis pas sûr du coût des FP et des FN. L’exemple a été donné à titre indicatif seulement.

### II.5.3. Assemblage des modèles (Ensemble Learning)

Une approche intéressante pour améliorer (potentiellement) la performance du modèle est d'assembler les prédictions de plusieurs classificateurs. En combinant les forces de différents algorithmes, il est possible de réduire les erreurs de prédiction tout en augmentant la robustesse du modèle. Voici quelques méthodes d’assemblage couramment utilisées :

* **Bagging (Bootstrap Aggregating)** : L'algorithme Random Forest est un exemple de bagging, où plusieurs arbres de décision sont entraînés sur des échantillons aléatoires du dataset. Chaque arbre effectue une prédiction et les résultats sont agrégés (généralement via une moyenne ou un vote majoritaire) pour produire une prédiction plus stable et précise.
* **Boosting** : Les algorithmes comme AdaBoost et Gradient Boosting fonctionnent en entraînant des modèles successifs qui corrigent les erreurs des modèles précédents. Ce processus permet de créer un classificateur global plus puissant.
* **Stacking** : Le stacking consiste à entraîner plusieurs modèles de classification (par exemple, Random Forest, SVC, Logistic Regression, etc.) et à combiner leurs prédictions à l'aide d'un méta-modèle. Ce méta-modèle apprend à partir des prédictions des premiers modèles pour produire une prédiction finale.
* **Voting Classifier** : Une méthode simple mais efficace pour combiner les prédictions consiste à utiliser un Voting Classifier. Dans ce cas, plusieurs modèles sont entraînés indépendamment, et les prédictions sont agrégées soit par vote majoritaire (pour la classification), soit par moyenne des probabilités (pour la régression). Cela permet d'obtenir une prédiction plus robuste en tirant parti des forces de chaque classificateur.

# III. Interprétabilité globale du modèle

L'interprétabilité des modèles d'apprentissage automatique est cruciale pour comprendre leur fonctionnement et pour s'assurer qu'ils ne prennent pas de décisions discriminatoires ou biaisées.

## III.1. Interprétabilité Globale

L'interprétabilité globale permet d'identifier les features qui ont le plus d'impact sur les prédictions du modèle.

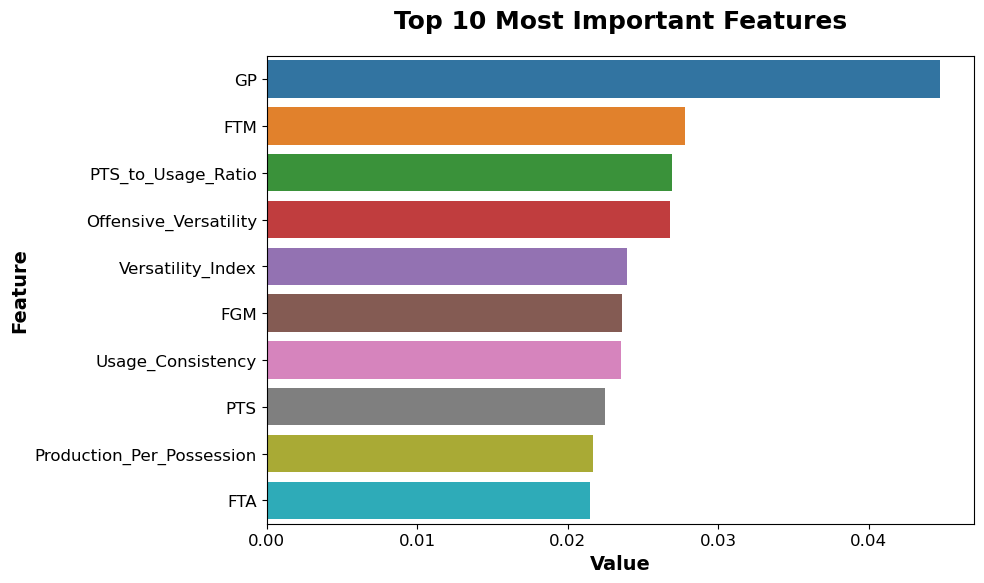


Figure 4: Analyse de l’importance des features

Dans ce projet, l'analyse de l'importance des features a révélé que les features les plus importantes pour la prédiction sont :

* GP
* FTM
* PTS\_to\_Usage\_Ratio
* Offensive\_Versatility
* Versality\_Index

# IV. Intégration du Classificateur dans un Web Service

## IV.1. En local

L'objectif de cette phase est d'intégrer le classificateur entraîné dans un service web afin de permettre aux utilisateurs d'effectuer des prédictions de manière simple et efficace. Cette intégration est réalisée via une API REST, accessible à distance.

**Étapes de l'Intégration**

1. **Création de l'API avec Flask** : Un framework léger, Flask, est choisi pour construire des applications web en Python, permettant ainsi de développer l’API. Ce choix facilite la création rapide d'un service web avec une structure simple. L'API est lancée sur le port 5000 et fournit un point d'entrée principal, accessible via la route /.
2. **Chargement des Ressources** : Pour que l'API puisse faire des prédictions, le modèle de machine learning préalablement entraîné est chargé à l'aide de la bibliothèque joblib. Cela inclut également le chargement des noms de caractéristiques et du scaler, qui normalise les données d'entrée pour garantir la cohérence des prédictions.
3. **Définition des Routes** : Plusieurs routes sont définies pour gérer les requêtes. La route principale (/) informe les utilisateurs que le service est opérationnel, tandis qu'une route spécifique (/predict) est mise en place pour recevoir des requêtes de prédiction. La route /predict accepte uniquement les requêtes de type POST contenant les caractéristiques des joueurs sous forme de JSON.
4. **Traitement des Requêtes** : Lorsqu'une requête POST est reçue sur la route de prédiction, la validité des données fournies par l'utilisateur est vérifiée. Cela garantit que toutes les informations nécessaires sont présentes. Si des features sont manquantes, l'API retourne une réponse avec une erreur appropriée, indiquant les features manquantes.
5. **Prédiction et Réponse** : Après la validation des données, le modèle est utilisé pour effectuer une prédiction. La réponse de l'API inclut non seulement la classe prédite (durée de carrière) mais aussi la probabilité associée et une interprétation de la prédiction, facilitant ainsi la compréhension pour les utilisateurs. Par exemple, une réponse pourrait indiquer que le joueur est "probable de durer 5+ ans en NBA" ou "probable de durer <5 ans en NBA", en fonction de la probabilité calculée par le modèle.

Pour maximiser l'accessibilité de ce service et permettre à un plus grand nombre de personnes de l'utiliser, le déploiement de l'API sur une plateforme cloud est recommandé. Par exemple, Heroku ou AWS peuvent être utilisé comme solution pour héberger l'API, facilitant ainsi son accès et son utilisation par des utilisateurs du monde entier.

### IV.1.1. Serveur Flask

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

### IV.1.2. Requête cURL

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

Description générée automatiquement

## IV.1. Sur le cloud (PythonAnywhere)

Cette partie détaille la mise en place d'une application web utilisant Flask pour prédire la longévité de carrière des joueurs NBA débutants. L'application est hébergée sur PythonAnywhere, une plateforme cloud pour les applications Python.

Le lien de la webapp est : [**https://isdinval.pythonanywhere.com/**](https://isdinval.pythonanywhere.com/)

### IV.1.1. Structure du Projet

Le projet se compose de plusieurs éléments clés :

1. Une application Flask (Flask\_app\_WITH\_FE.py)
2. Un modèle de machine learning pré-entraîné
3. Des fichiers HTML pour l'interface utilisateur (index.html + result.html)
4. Des fichiers auxiliaires pour le prétraitement des données

### IV.1.2. Détails de l'Implémentation

**1. Application Flask (Flask\_app\_WITH\_FE.py)**L'application Flask sert de backend pour le service web. Voici ses principales fonctionnalités :

* **Chargement du Modèle** : Au démarrage, l'application charge le modèle pré-entraîné, les noms des features, le scaler, et le seuil de décision.
* **Routes** : L'application définit deux routes principales :
  + / : Affiche le formulaire d'entrée (render\_template('index\_WITH\_FE.html'))
  + /predict : Traite les données soumises et renvoie une prédiction
* **Traitement des Données** : L'application reçoit les statistiques de base du joueur, calcule des métriques avancées (feature engineering), et prépare les données pour la prédiction.
* **Prédiction** : Utilise le modèle chargé pour faire une prédiction basée sur les données fournies.

**2. Interface Utilisateur**L'interface utilisateur est composée de deux pages HTML :

* **index\_WITH\_FE.html** : Un formulaire permettant aux utilisateurs de saisir les statistiques du joueur.
* **result.html** : Affiche le résultat de la prédiction, incluant la probabilité et l'interprétation.

**3. Modèle de Machine Learning**Le modèle utilisé est un classificateur binaire qui prédit si un joueur aura une carrière de 5 ans ou plus en NBA. Il est pré-entraîné et chargé au démarrage de l'application.

**4. Prétraitement des Données**

L'application effectue plusieurs étapes de prétraitement :

* Vérification de la présence de toutes les features requises
* Calcul de métriques avancées à partir des statistiques de base
* Normalisation des données à l'aide d'un scaler pré-entraîné

### IV.1.3. Déploiement sur PythonAnywhere

Pour déployer l'application sur PythonAnywhere, les étapes suivantes ont été suivies :

1. Création d'un compte PythonAnywhere
2. Upload des fichiers du projet (Flask\_app\_WITH\_FE.py, templates HTML, fichiers du modèle)
3. Configuration d'une nouvelle application web dans l'interface PythonAnywhere
4. Spécification du fichier WSGI pour pointer vers l'application Flask
5. Démarrage de l'application

### IV.1.4. Quelques captures d’écran

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

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Conclusion : Développement d'un Modèle de classification binaire de la longévité des carrières des joueurs de NBA.

Ce projet de classification binaire visant à prédire la longévité des carrières des joueurs débutants en NBA a permis de développer un modèle robuste et performant, tout en abordant les défis inhérents à ce type de problématique.

L'approche méthodologique adoptée, comprenant une préparation des données, l'utilisation de techniques pour gérer le déséquilibre des classes (SMOTE), et l'application de feature engineering, a contribué à l'amélioration significative des performances du modèle.

Le choix final du modèle Random Forest, avec un recall de 0,77 et une AUC ROC de 0,83 sur le dataset d'entraînement ainsi qu’un recall de 0,74 et une AUC ROC de 0,77 sur le dataset de test, démontre sa capacité à identifier efficacement les joueurs susceptibles d'avoir une carrière longue, tout en maintenant un bon équilibre global de performance.

L'ajustement du seuil de décision à 0,48, basé sur l'analyse de la courbe Precision-Recall, a permis d'optimiser le compromis entre la détection des talents prometteurs et la minimisation des investissements risqués. Cette approche offre aux investisseurs un outil précieux pour guider leurs décisions.

L'analyse de l'importance des features a révélé que des facteurs tels que le nombre de matchs joués (GP), les lancers francs réussis (FTM), et des métriques composites comme le ratio points/utilisation et la polyvalence offensive, sont cruciaux dans la prédiction de la longévité d'une carrière NBA.

L'intégration du modèle dans une application web, déployée à la fois localement et sur le cloud (PythonAnywhere), démontre la volonté de rendre cet outil accessible et utilisable dans un contexte réel. L'interface utilisateur permet aux utilisateurs de saisir facilement les statistiques des joueurs et d'obtenir des prédictions instantanées.

Pour l'avenir, l'introduction d'une métrique d'évaluation basée sur le Coût Métier Total (CMT) pourrait encore affiner l'utilité du modèle dans un contexte d'investissement, en quantifiant directement l'impact financier des décisions basées sur les prédictions.

En conclusion, ce projet fournit non seulement un outil prédictif pour évaluer le potentiel à long terme des joueurs NBA débutants, mais aussi un cadre méthodologique pour aborder des problèmes similaires de classification binaire dans d'autres domaines. La combinaison d'une modélisation rigoureuse et d'une mise en œuvre pratique via une application web démontre la valeur ajoutée que l'apprentissage automatique peut apporter dans le domaine du sport professionnel et de l'investissement associé.