

PariVision

Mise en place d’un pipeline MLOps  
pour un modèle de prédiction de tir NBA

P. COHEN, F. HEUZE, S. JIANYING | DEC24\_MLOPS\_PARIS\_SPORTIFS | 19/05/2025

[Section 1 – Présentation 4](#_Toc198138254)

[Introduction 4](#_Toc198138255)

[Objectif 5](#_Toc198138256)

[Contribution à l’industrie 5](#_Toc198138257)

[Principaux défis identifiés 6](#_Toc198138258)

[Flux de travail 7](#_Toc198138259)

[Hébergement du projet 8](#_Toc198138260)

[Section 2 – Exploration des données 9](#_Toc198138261)

[Sources de données 9](#_Toc198138262)

[VirtualBetsAPI 9](#_Toc198138263)

[Utilisation de VirtualBetsAPI 9](#_Toc198138264)

[Limite de VirtualBetsAPI 10](#_Toc198138265)

[Analyse exploratoire des données brutes 11](#_Toc198138266)

[Matrice de corrélation entre variables explicatives et cible 13](#_Toc198138267)

[Taux de réussite par type de tir 13](#_Toc198138268)

[Évolution du taux de réussite en fonction du temps écoulé 14](#_Toc198138269)

[Relation entre position du tir et distance 14](#_Toc198138270)

[Densité de la distance de tir selon le type de tir (encodé) 15](#_Toc198138271)

[Pré-traitement des données 16](#_Toc198138272)

[Variables finales 16](#_Toc198138273)

[Tableau comparatif avant/après pré-traitement (Bronze -> Silver) 17](#_Toc198138274)

[Nettoyage des données 18](#_Toc198138275)

[Ingénierie des fonctionnalités (features engineering) 18](#_Toc198138276)

[Fusion des variables temporelles en une caractéristique continue 18](#_Toc198138277)

[Encodage numérique du tir 19](#_Toc198138278)

[Transformation des variables de zone de tir lors du feature engineering 19](#_Toc198138279)

[Synthèse 19](#_Toc198138280)

[Stockage des données 20](#_Toc198138281)

[Introduction 20](#_Toc198138282)

[Installation sur Docker 20](#_Toc198138283)

[Utilisation de MongoDB 21](#_Toc198138284)

[Utilisation de PyMongo 22](#_Toc198138285)

[Modélisation 22](#_Toc198138286)

[Section 3 – Mise en production 23](#_Toc198138287)

[Déploiement de la VM 23](#_Toc198138288)

[Connexion SSH 23](#_Toc198138289)

[Initialisation 23](#_Toc198138290)

[Configuration de la VM 23](#_Toc198138291)

[Github 24](#_Toc198138292)

[Cloner le dépôt 24](#_Toc198138293)

[Actualiser le dépôt 24](#_Toc198138294)

[Mise en place d’un workflow Github Actions 25](#_Toc198138295)

[Mise en place d’un workflow de tests unitaires 26](#_Toc198138296)

[Livraison continue (CD) 26](#_Toc198138297)

[Python 27](#_Toc198138298)

[Mise à jour et installation des dépendances 27](#_Toc198138299)

[Création de l'environnement virtuel 27](#_Toc198138300)

[Installation des librairies 27](#_Toc198138301)

[Fichier d’environnement 27](#_Toc198138302)

[Docker 28](#_Toc198138303)

[Installation de Docker sur la VM 28](#_Toc198138304)

[Configuration de Docker 28](#_Toc198138305)

[Le fichier Docker-Compose 28](#_Toc198138306)

[Homer 29](#_Toc198138307)

[Jenkins 30](#_Toc198138308)

[Installation 30](#_Toc198138309)

[Configuration 30](#_Toc198138310)

[Plugins 30](#_Toc198138311)

[Créer un agent PariVision 31](#_Toc198138312)

[Pipeline 32](#_Toc198138313)

[MLFlow 33](#_Toc198138314)

[Objectif 33](#_Toc198138315)

[Initialisation 33](#_Toc198138316)

[Utilisation 33](#_Toc198138317)

[Configurer l’expérience PariVision 34](#_Toc198138318)

[Mise en œuvre 35](#_Toc198138319)

[Enregistrement des expériences 35](#_Toc198138320)

[Bénéfices 35](#_Toc198138321)

[Conclusion 35](#_Toc198138322)

[MLFlow-Champion 36](#_Toc198138323)

[Section 4 – Exploitation 37](#_Toc198138324)

[Pipeline ETL 37](#_Toc198138325)

[Pipeline ML 37](#_Toc198138326)

[Supervision 38](#_Toc198138327)

[Prometheus et Grafana 38](#_Toc198138328)

[Slack 38](#_Toc198138329)

[Evidently 39](#_Toc198138330)

[PariVision-Exporter 39](#_Toc198138331)

[Section 5 – Conclusion 40](#_Toc198138332)

[Les principaux points forts du projet 40](#_Toc198138333)

[Bilan 40](#_Toc198138334)

[Perspectives d’évolution 41](#_Toc198138335)

[Synthèse 42](#_Toc198138336)

Section 1 – Présentation

# Introduction

Le basketball, en particulier la NBA (National Basketball Association), figure parmi les sports les plus prisés au monde. Il attire des millions de spectateurs et génère un marché économique colossal, notamment dans le domaine des paris sportifs. Avec une saison régulière comprenant 82 matchs par équipe et des playoffs intenses, la NBA offre un volume conséquent de données exploitables pour l’analyse prédictive. Dans ce contexte, où chaque tir peut influencer l’issue d’un match et générer des enjeux stratégiques, économiques (paris sportifs, droits TV) ou émotionnels (engagement des fans), la capacité à prédire la réussite d’un tir spécifique devient un atout précieux.

Notre projet **PariVision** se concentre sur cette problématique en combinant machine learning et ingénierie MLOps pour concevoir un système prédictif robuste et automatisé, capable d’estimer les résultats des matchs NBA. La complexité réside dans la modélisation des multiples facteurs influençant un tir : position sur le terrain, pression défensive, ou même le contexte du match (dernières secondes, enjeux du score). Les approches statiques peinent à s’adapter à la variabilité des données sportives, nécessitant une infrastructure capable d’évoluer avec les saisons et les métriques émergentes.

La prédiction des tirs réussis au basketball est un enjeu central pour l’analyse de la performance des joueurs, l’optimisation des stratégies d’équipe et l’engagement des fans. Dans ce contexte, le projet « Prédiction des tirs réussis des joueurs NBA » vise à concevoir, entraîner et déployer un modèle de machine learning capable de prédire, à partir des caractéristiques du tir et de son contexte, la probabilité de réussite d’un tir tenté lors d’un match NBA.

Ce projet s’inscrit donc dans une démarche complète de science des données, de la préparation des données brutes jusqu’au déploiement et à la surveillance du modèle. Il offre un cadre réutilisable et évolutif pour l’analyse prédictive dans le domaine sportif et au-delà et permet d’illustrer comment le MLOps peut transformer des données brutes en entrées exploitable, en alignant précision algorithmique et exigences opérationnelles.

**En résumé**, ce projet répond à des besoins concrets de l’industrie du sport, s’appuie sur les technologies les plus récentes en data science et MLOps et anticipe les défis techniques, organisationnels et éthiques liées à l’analyse avancée de la performance sportive.

# Objectif

En ciblant cette problématique, nous élaborons les objectifs suivants :

* **Prédiction contextuelle :** déterminer la probabilité de réussite d’un tir en temps réel, en s’appuyant sur des données synthétiques ou historiques.
* **Automatisation du flux de travail :** conteneuriser chaque étape et orchestrer les pipelines.
* **Adaptabilité continue :** implémenter un système de réentrainement conditionnel, où un nouveau modèle ne remplace l’ancien que si son exactitude dépasse celle de l’ancien modèle.
* **Garantir la traçabilité et la reproductibilité :** des intégrations CI/CD, un suivi des performances, la reproductibilité et la sélection optimale des modèles.
* **Surveillance en production : assurer** une supervision granulaire des requêtes API, de la base de données ainsi qu’une toute dérive des données ou baisse de performance en production pour ainsi maintenir la fiabilité du modèle dans le temps.

Les principaux résultats attendus sont :

* **Un modèle de machine learning** validé par des métriques robustes.
* **Un service de prédiction déployé** sous forme d’API REST.
* **Une application** d’aide à la décision.
* **Des rapports et panneaux de supervision** détectant la dérive des données ou de la performance, garantissant la fiabilité continue du modèle.
* **Une documentation complète** du pipeline, des choix de modélisation et des résultats, facilitant la réutilisation et l’évolution du projet.

# Contribution à l’industrie

* **Amélioration de la performance sportive** : Les analyses issues du modèle permettent d’identifier les zones et situations à haut potentiel de réussite, d’ajuster les stratégies de tir et d’individualiser l’entraînement des joueurs.
* **Appui à la prise de décision en temps réel** : Les équipes NBA utilisent de plus en plus ces outils pour adapter les tactiques pendant les matchs, optimiser les rotations ou anticiper les faiblesses adverses.
* **Valorisation de la data dans l’écosystème sportif** : Le projet illustre comment la collecte, l’analyse et l’exploitation intelligente des données deviennent des leviers majeurs de compétitivité et d’innovation dans le sport professionnel.
* **Transférabilité** : Ce type de pipeline peut être adapté à d’autres sports ou contextes (recrutement, prévention des blessures, expérience fan, etc, ...).

# Principaux défis identifiés

* **Qualité et hétérogénéité des données** : Les données sportives sont souvent bruitées, incomplètes ou issues de sources multiples, ce qui nécessite un pré-traitement rigoureux.
* **Évolution des stratégies et du jeu** : Les modèles doivent s’adapter à des changements de règles, de tactiques ou de comportements des joueurs, rendant la supervision essentielle.
* **Dérive des données et des performances** : Sans surveillance, un modèle performant peut devenir obsolète si la distribution des données évolue (ex : nouveaux schémas de jeu, évolution du profil des joueurs).
* **Défis éthiques et réglementaires** : Respect de la vie privée des joueurs, sécurité des données, équité dans l’accès aux technologies avancées.
* **Intégration et adoption** : Faire accepter et intégrer ces outils dans les routines des équipes et du staff technique peut nécessiter un accompagnement au changement.

# Flux de travail

Dans une logique MLOps moderne, le projet intègre une pipeline ML Ci/CD :

* **Le suivi d’expériences avec MLflow**, pour garantir la traçabilité, la reproductibilité et la sélection optimale des modèles.
* **Le déploiement du modèle avec FastAPI**, permettant de rendre la prédiction accessible via une API REST.
* **La surveillance continue avec Evidently**, pour détecter toute dérive des données et ainsi maintenir la fiabilité du modèle sur la durée.

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

Description générée automatiquement

# Hébergement du projet

Le projet est entièrement hébergé sur un serveur unique en mode monolithique. Bien que cette approche ne soit pas la plus optimale en termes de performance et de scalabilité, elle a été choisie en raison de contraintes budgétaires et de ressources disponibles.

Cette configuration permet de centraliser les opérations et de simplifier la gestion des infrastructures, tout en respectant les limitations financières et techniques.

* Accès à la page d’accueil du projet : [PariVision](https://parivision.heuzef.com/)
* Accès au dépôt Git du projet : <https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/>
* Miroir : <https://github.com/heuzef/DEC24_MLOPS_PARIS_SPORTIFS>

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Section 2 – Exploration des données

Ce projet s’inscrit dans une démarche complète de science des données, de la préparation des données brutes jusqu’au déploiement et à la surveillance du modèle, selon les meilleures pratiques de l’industrie. Il offre un cadre réutilisable et évolutif pour l’analyse prédictive dans le domaine sportif et au-delà.

# Sources de données

## VirtualBetsAPI

VBA est une API developper sur-mesure dans le cadre du projet PariVision, nous permettant d’appeler des données fictives aléatoirement. Cet outil exploite FastAPI et se base sur un fichier d’origine CSV, transformé en JSON.

## Utilisation de VirtualBetsAPI

* **Appel : Demande de retour de 2 nodes**

http://parivision.heuzef.com:8800/getdata/2

* **Résultat**

[

{

"Game ID": "0029700427",

"Game Event ID": "389",

"Player ID": "100",

"Player Name": "Tim Legler",

"Team ID": "1610612764",

"Team Name": "Washington Wizards",

"Period": "4",

"Minutes Remaining": "11",

"Seconds Remaining": "22",

"Action Type": "Jump Shot",

"Shot Type": "2PT Field Goal",

"Shot Zone Basic": "Mid-Range",

"Shot Zone Area": "Right Side(R)",

"Shot Zone Range": "8-16 ft.",

"Shot Distance": "15",

"X Location": "117",

"Y Location": "109",

"Shot Made Flag": "1",

"Game Date": "19980102",

"Home Team": "WAS",

"Away Team": "IND",

"Season Type": "Regular Season"

},{...}

]

## Limite de VirtualBetsAPI

La limite est fixée à 50000 nodes contenus dans le fichier dat\_NBA.   
L’URL http://parivision.heuzef.com:8800/getdata/2 renvoie les deux premiers nodes du fichier. Pour renvoyer les 3 premiers nodes, il faut remplacer 2 par 3 dans l’URL

Si l’url est appelée deux fois, les deux noeuds suivants sont renvoyés. Un compteur est implémenté afin de ne pas fournir deux fois les mêmes nodes.

Pour réinitialiser le compteur à 0, il faut appeler l’URL suivante :   
http://parivision.heuzef.com:8800/reset

# Analyse exploratoire des données brutes

Les données brutes comportent 22 variables, couvrant à la fois des informations contextuelles (identifiants de match, joueur, équipe, période, date, etc, ...), des caractéristiques du tir (type de tir, zone du terrain, distance, coordonnées, etc.) et la cible à prédire (Shot Made Flag, indiquant si le tir a été réussi ou non).

Une image contenant texte, affichage, capture d’écran, Appareil de présentation

Description générée automatiquement

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

Description générée automatiquement

## Matrice de corrélation entre variables explicatives et cible

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

Description générée automatiquement

## Taux de réussite par type de tir

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

Description générée automatiquement

## Évolution du taux de réussite en fonction du temps écoulé

Une image contenant Tracé, diagramme, ligne, pente

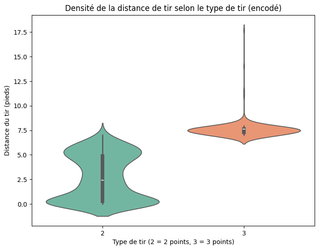
Description générée automatiquement

## Relation entre position du tir et distance

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

Description générée automatiquement

## Densité de la distance de tir selon le type de tir (encodé)



# Pré-traitement des données

Le pipeline de traitement des données mis en place comprend plusieurs étapes structurantes :

* **Pré-traitement et nettoyage** : suppression des colonnes non pertinentes, gestion des doublons, vérification et traitement des valeurs manquantes.
* **Ingénierie des fonctionnalités** : création de nouvelles variables plus informatives, telles que le temps restant dans la période, l’encodage numérique du type de tir, conversion des unités de mesure, etc …
* **Préparation à la modélisation** : sélection et transformation des variables explicatives, encodage des variables catégorielles, séparation des jeux d’entraînement et de test.

## Variables finales

|  |  |  |
| --- | --- | --- |
| **Variable** | **Description** | **Type** |
| Player ID | Identifiant du joueur | Variable nettoyée |
| X Location | Position X sur le terrain | Variable nettoyée |
| Y Location | Position Y sur le terrain | Variable nettoyée |
| Total\_Seconds\_Remaining | Temps total restant | Nouvelle variables |
| Shot\_Type\_Encoded | Type de tir (2 points ou 3 points) | Nouvelle variables |
| Shot\_Distance\_Meters | Distance en mètres | Nouvelle variables |
| Shot\_Zone\_Combined | Zone de tir (7 zones) | Nouvelle variables |
| Shot Made Flag | Indicateur de réussite du tir | Variable cible |

## Tableau comparatif avant/après pré-traitement (Bronze -> Silver)

|  |  |  |
| --- | --- | --- |
| **Variable** | **Bronze** | **Silver** |
| Nombre de colonnes | 22 | 12 |
| Colonnes présentes | Game ID, Game Event ID, Player ID, Player Name, Team ID, Team Name, Period, Minutes Remaining, Seconds Remaining, Action Type, Shot Type, Shot Zone Basic, Shot Zone Area, Shot Zone Range, Shot Distance, X Location, Y Location, Shot Made Flag, Game Date, Home Team, Away Team, Season Type | Player ID, Action Type, Shot Type, Shot Zone Basic, Shot Zone Area, Shot Zone Range, Shot Distance, X Location, Y Location, Shot Made Flag, Total\_Seconds\_Remaining, Shot\_Type\_Encoded |
| Colonnes supprimées | - | Game ID, Game Event ID, Player Name, Team ID, Team Name, Period, Game Date, Home Team, Away Team, Season Type |
| Colonnes créées | - | Total\_Seconds\_Remaining (Minutes Remaining × 60 + Seconds Remaining), Shot\_Type\_Encoded (2 ou 3 selon Shot Type) |
| Doublons | Non vérifié initialement | Vérifiés et supprimés (aucun doublon trouvé) |
| Valeurs manquantes | Non connues initialement | Vérifiées et remplacées par la médiane pour les numériques (aucune valeur manquante détectée) |
| Encodage | Variables catégorielles non encodées | Shot\_Type\_Encoded (2 = 2 points, 3 = 3 points, 0 = autre) |
| Colonnes temporelles | Minutes Remaining, Seconds Remaining | Fusionnées en Total\_Seconds\_Remaining |
| Colonnes de type de tir | Shot Type (texte) | Shot\_Type\_Encoded (numérique) |
| Cible | Shot Made Flag (0/1) | Shot Made Flag (0/1) |
| Format des variables | 12 numériques, 10 catégorielles | Majoritairement numériques ou prêtes à l’encodage |
| Prêt pour l’apprentissage | Non | Oui |

Ce tableau montre comment le pré-traitement améliore la structure et la qualité des données, facilitant ainsi la modélisation et la reproductibilité du projet.

BRONZE : Les données sont brutes, avec des colonnes redondantes, des variables catégorielles non encodées et des informations temporelles séparées.

SILVER : Le dataset est épuré, enrichi (feature engineering) et prêt pour l’appr

entissage automatique, sans doublons ni valeurs manquantes.

# Nettoyage des données

* Suppression des variables non pertinentes 'Game ID', 'Game Event ID', 'Player Name', 'Team ID', 'Team Name', 'Period', 'Season Type', 'Game Date', 'Home Team', 'Away Team'.
* Vérification des doublons : Aucun doublons
* Vérification des valeurs manquantes : Aucune valeurs manquantes

# Ingénierie des fonctionnalités (features engineering)

Pour améliorer la qualité des données, plusieurs transformations ont été effectuées lors de la phase de Feature Engineering.

## Fusion des variables temporelles en une caractéristique continue

Les variables Minutes Remaining et Seconds Remaining ont été fusionnées en **Total\_Seconds\_Remaining** = (Minutes Remaining \* 60) + Seconds Remaining.

Cela permet de représenter plus simplement le temps restant pour chaque tir, facilitant l’analyse par les modèles de machine learning.

Les colonnes d’origine ont été supprimées pour éviter la redondance.

## Encodage numérique du tir

La variable catégorielle Shot Type a été convertie en numérique : 2 pour les tirs à deux points, 3 pour les tirs à trois points, et 0 pour les cas rares.

Cette transformation facilite l'utilisation de cette information dans les modèles numériques via une fonction d'encodage dédiée.

## Transformation des variables de zone de tir lors du feature engineering

Le jeu de données original décrit la localisation du tir avec des variables catégorielles distinctes :

* **Shot Zone Basic** : zone principale (ex : Mid-Range, Restricted Area)
* **Shot Zone Area** : zone latérale ou centrale (ex : Left Side, Center, Right Side)
* **Shot Zone Range** : distance (ex : Less Than 8 ft., 8-16 ft.)

Les variables de zone de tir, telles que 'Less Than 8 ft.', '8-16 ft.', '16-24 ft.', '24+ ft.' et 'Back Court Shot', sont informatives mais redondantes. Elles ont été combinées en une seule variable, nommée **Shot\_Zone\_Combined**, pour simplifier la modélisation.

## Synthèse

En résumé, ces transformations ont permis d’obtenir un jeu de données plus compact, plus informatif et directement exploitable par les algorithmes de machine learning, tout en conservant l’essentiel de l’information sportive et contextuelle utile à la prédiction du succès des tirs NBA.

# Stockage des données

Le stockage des données brutes et pré-traités s’effectue au format JSON, dans un base de donné NoSQL.

## Introduction

MongoDB est une base de données NoSQL orientée documents, qui se distingue par sa flexibilité, sa scalabilité, et sa capacité à gérer des données semi-structurées ou non structurées. Contrairement aux bases de données relationnelles traditionnelles (comme MySQL ou PostgreSQL), qui utilisent des tables pour stocker des données, MongoDB utilise des documents au format **JSON-like** (BSON en interne). Cette structure permet de modéliser les données de manière plus naturelle et intuitive pour les développeurs.

## Installation sur Docker

Step 1 : Sur la machine host, vérifier que l’image de **mongoDB** n’est pas présente :

docker image

Step 2 : Récupérer la dernière image de **mongoDB** disponible :

docker pull mongo:latest

Step 3 : Vérifier que l’image est bien téléchargée :

docker image

Step 4 : Créer un répertoire “mongoDB\_PariVision” qui nous servira de volume pour la persistance des données :

mkdir mongoDB\_PariVision

cd mongoDB\_PariVision

Step 5 : Lancer Docker avec l’image **mongoDB** en ligne de commande :

docker run -d -p 27017:27017 -v ~/mongoDB\_PariVision/:/data/db --name parivision\_mongodb\_container mongo:latest

-d : dash mode, tourne en background

-p : association des ports de la machine host et du port sur lequel tourne **mongoDB** (27017)

-v : ajout du volume pour la persistance des données

–-name : nom du container **mongoDB**

Step 6 : Vérifier que le container est bien en RUN :

docker ps

Step 7 : Accéder au container :

docker exec -it mongodb bash

Step 8 : Une fois à l’intérieur du container, on démarre **mongoDB :**

mongosh

Step 9 : Créer l’utilisateur

db.createUser({user: "parivision", pwd: "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", roles: [ { role: "readWrite", db: "parivision" }]})

## Utilisation de MongoDB

a) Afficher les différentes bases de données

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

>

b) Création de la base de données pour PariVision

> use parivision

switched to db parivision

>

c) Insertion d’un objet

> db.user.insert({"name":"Curry"})

WriteResult({ nInserted: 1 })

d) Lister les objets

> db.user.find()

{ "\_id" : ObjectId("223d432G56h765jo987d234a3"), "name":"Curry"}

## Utilisation de PyMongo

Documentation de PyMongo : <https://pymongo.readthedocs.io/en/stable/tutorial.html>

!pip3 install pymongo

from pymongo import MongoClient

client = MongoClient(

host = parivision.heuzef.com,

port = 27017,

username = \*\*\*\*\*\*\*\*\*\*,

password = \*\*\*\*\*\*\*\*\*\*

)

print(client.list\_database\_names())

# Modélisation

La phase de modélisation s’appuie sur une sélection de modèles de classification robustes et complémentaires : régression logistique, Random Forest, Gradient Boosting, XGBoost et LightGBM. Chaque modèle est entraîné et évalué, avec un suivi rigoureux des paramètres et des métriques de performance grâce à MLflow.

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

Description générée automatiquement

Section 3 – Mise en production

# Déploiement de la VM

La machine virtuelle utilisée pour la mise en production du projet dispose des ressources suivantes :

* RAM : 4G
* CPU : 4 (2 Sockets, 2 Cores)
* Stockage : SSD 128G
* OS : Ubuntu Server 24.04
* Nom de domaine : **parivision.heuzef.com**

## Connexion SSH

La connexion s’effectue en SSH avec votre clé privée.

ssh -i .ssh/cle.priv parivision@parivision.heuzef.com

## Initialisation

# Mise à jour du système

sudo apt update && sudo apt upgrade

# Désactiver l'authentification par mot de passe

sudo vim /etc/ssh/sshd\_config # PasswordAuthentication no

sudo systemctl restart ssh

# Ajouter les clefs SSH pour l'authentification

vim /home/parivision/.ssh/authorized\_keys

Une sauvegarde de bas niveau est effectuée à ce stade.

## Configuration de la VM

# Configurer sur l'heure de Paris

sudo timedatectl set-timezone "Europe/Paris"

# Ajout de quelques outils

sudo apt install -y curl wget git tmux htop vim nano tree unzip smartmontools bmon

# Github

Le dépôt GIT du projet est hebergé ici : <https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS>

# Cloner le dépôt

# Création d'une nouvelle clef SSH

ssh-keygen -t ed25519

# La clef doit être ajouté sur le repo github :

# https://github.com/DataScientest-Studio/DEC24\_MLOPS\_PARIS\_SPORTIFS/settings/keys

cat .ssh/id\_ed25519.pub

# Cloner le repo GIT :

cd

git clone git@github.com:DataScientest-Studio/DEC24\_MLOPS\_PARIS\_SPORTIFS.git

tree DEC24\_MLOPS\_PARIS\_SPORTIFS/

# Actualiser le dépôt

cd /home/parivision/DEC24\_MLOPS\_PARIS\_SPORTIFS/ ; git pull

# Mise en place d’un workflow Github Actions

Le workflow suivant est ajouté dans notre dépôt GIT dans [**.github/workflows/workflows.yaml**](https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/.github/workflows/workflows.yaml)

Une image contenant texte, Appareils électroniques, capture d’écran, multimédia

Description générée automatiquement

Une clé SSH est créée pour l’occasion, dédiée à Github et autorisée sur notre VM de production et ajoutée sur Github.com.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

# Mise en place d’un workflow de tests unitaires

Le workflow suivant est ajouté dans notre dépôt GIT dans [**.github/workflows/pytest.yaml**](https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/.github/workflows/pytest.yaml)

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

Description générée automatiquement

Ce dernier va effectuer une série de test unitaires, avec le module Pytest, l’ensemble des scripts tests/\*.py qui seront ajoutée au fur et à mesure du développement, offrant une évolution TDD possible.

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

Description générée automatiquement

## Livraison continue (CD)

L’actualisation du dépôt est maintenant effectuée automatiquement à chaque push sur Github

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Python

## Mise à jour et installation des dépendances

sudo apt upgrade python3

sudo apt install python3-pip python3-venv

## Création de l'environnement virtuel

cd /home/parivision/DEC24\_MLOPS\_PARIS\_SPORTIFS

python3 -m venv .venv

source .venv/bin/activate

## Installation des librairies

pip install -r requirements.txt

## Fichier d’environnement

Un fichier d’environnement présent sur la VM de production est ignoré à la racine du dépôt GIT, ce dernier contient les différents secrets nécessaire au projet.

/home/parivision/DEC24\_MLOPS\_PARIS\_SPORTIFS/.env

Les scripts python du projet utilisent la librairie **dotenv** pour charger les secrets :

import os

from dotenv import load\_dotenv

load\_dotenv(".env")

secret = os.getenv('secret')

Une sauvegarde de bas niveau est effectuée à ce stade.

# Docker

Docker sert de composante principale pour l’infrastructure du projet.

## Installation de Docker sur la VM

Réf : <https://docs.docker.com/engine/install/ubuntu/>

# Add Docker's official GPG key:

sudo apt update

sudo apt install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update

# Install the latest version:

sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-compose

## Configuration de Docker

Réf: <https://docs.docker.com/engine/install/linux-postinstall/>

# Post-install:

sudo usermod -aG docker parivision

newgrp docker

docker run hello-world

# Auto-start:

sudo systemctl enable docker.service

sudo systemctl enable containerd.service

## Le fichier Docker-Compose

La configuration complète de Docker repose sur Docker-Compose.

Pour cela, le fichier **docker-compose.yml** qui définie la configuration de PariVision est intégré au processus de démarrage. : <https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/docker-compose.yml>

Pour tester, executer manuellement le lancement de l’infrastructure Docker ainsi :

docker-compose up /home/parivision/DEC24\_MLOPS\_PARIS\_SPORTIFS/docker-compose.yml

L’ensemble des services devraient être accessibles.

Le service PORTAINER permet d’administrer sur une interface web toute la configuration Docker, accessible sur le port 9443. Par exemple, pour lister tous les containers :

[https://parivision.heuzef.com:9443/#!/2/docker/containers](https://192.168.0.130:9443/#!/2/docker/containers)

Une sauvegarde de bas niveau est effectuée à ce stade.

# Homer

Homer est un portail statique personnalisable servant de point d’entrée du projet pour lister et accéder rapidement aux différents services du projet.

Dépôt du projet :homer: <https://github.com/bastienwirtz/homer>

L’adresse d’accès est le domaine principale du projet : <https://parivision.heuzef.com>

La configuration du Dashboard de Homer s’effectue directement sur le dépôt GIT, dossier **“homer”** : <https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/tree/master/homer>

Le portail est configuré via un simple fichier YAML.

# Jenkins

## Installation

L’installation de Jenkins ne sera pas containérisé car le projet est sur une VM unique et le nœud maître doit avoir la possibilité de gérer les instances docker du projet. La méthode DooD n’est pas favorisée ici.

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt install -y fontconfig openjdk-17-jre jenkins

sudo usermod -aG docker jenkins

sudo systemctl restart docker

sudo systemctl enable jenkins

sudo systemctl stop jenkins

sudo systemctl start jenkins

sudo systemctl status jenkins

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Se rendre sur l’instance, port **8080**, pour déverrouiller l’installation avec le initialAdminPassword.

## Configuration

* Démarrer l'installation des plugins communautaires recommandés.
* Créer le compte administrateur.

## Plugins

Installer les plugins suivants :

* GitHub Integration
* Blue Ocean
* AnsiColor

## Créer un agent PariVision

L’utilisateur Jenkins ne dispose pas des mêmes permissions que l’utilisateur parivision.   
Nous pouvons remerdier à cela en créant un agent “parivision” qui se connectera sur l’hôte local avec l’utilisateur parivision.

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

Description générée automatiquement

Finalement, le nœud maître principal doit être désactivé, en ajustant son nombre d'exécuteurs à zéro.

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

Description générée automatiquement

Finalement, les différents Jobs du projet peuvent être mis en place.

## Pipeline

Concernant la Pipeline ML du projet PariVision, celle-ci sera synchronisé sur le JenkinsFile disponible dans le dépôt GIT : <https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/Jenkinsfile>

La création de l’objet Pipeline s’effectue avec les paramètres suivants :

Une image contenant texte, capture d’écran, nombre

Description générée automatiquement

Afin d’autoriser l’accès au repo à Jenkins, il faut désactiver la vérification de propriété depuis le compte jenkins :

su jenkins

git config --global --add safe.directory /home/parivision/DEC24\_MLOPS\_PARIS\_SPORTIFS/.git

Une sauvegarde de bas niveau est effectuée à ce stade.

# MLFlow

Accès à l’instance déployé via Docker : [http://parivision.heuzef.com:5000](http://parivision.heuzef.com:5000/)

## Objectif

La mise en place de MLflow a pour but de tracer, comparer et reproduire toutes les expériences de modélisation menées dans le projet de prédiction des tirs réussis NBA. MLflow permet de centraliser les informations relatives aux modèles testés, d’optimiser le processus de sélection et d’assurer la transparence du pipeline data science.

## Initialisation

Une fois les dépendances pour l’authentification MLFlow déployés, il faut s’authentifier dans le container pour modifier le mot de passe administrateur par défaut puis créer un nouvel utilisateur. Enfin, créer l’expérience avec les permissions appropriées :

docker exec -it mlflow bash

export MLFLOW\_TRACKING\_USERNAME=admin

export MLFLOW\_TRACKING\_PASSWORD=password

python3

>>> import mlflow

>>> from mlflow.server.auth.client import AuthServiceClient

>>> client = AuthServiceClient("http://localhost:5000")

>>> client.create\_user(username="parivision", password="\*\*\*\*\*\*\*\*\*\*")

>>> client.update\_user\_admin(username="parivision", is\_admin=True)

>>> client.get\_user("parivision").is\_admin

>>> mlflow.MlflowClient(tracking\_uri="http://localhost:5000").create\_experiment(name="parivision")

>>> client.update\_user\_password("admin", "\*\*\*\*\*\*\*\*\*\*")

>>> exit()

Exit

## Utilisation

Se référer au Notebook tutoriel pour obtenir un exemple d’utilisation :

<https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/blob/master/notebooks/heuzef_mlflow.ipynb>

## Configurer l’expérience PariVision

Afin de permettre le bon fonctionnement du Workflow, une expérience nommé ***“parivision”*** contenant au moins une run avec un alias ***“champion”*** est requis.

Une fois que la première run est remontée dans l’expérience ***“parivision”***, il suffit de consulter cette dernière sur l’UI d'MLFlow, et d'effectuer ce premier enregistrement :

Une image contenant texte, logiciel, Logiciel multimédia, Logiciel de graphisme

Description générée automatiquement

Finalement, lui attribuer l’alias ***“champion”*** :

Une image contenant texte, logiciel, Logiciel multimédia, Logiciel de graphisme

Description générée automatiquement

Nous avons défini un premier modèle Champion, qui est maintenant prêt à être challengé par les nouvelles runs.

## Mise en œuvre

## Enregistrement des expériences

À chaque entraînement d’un modèle (Régression Logistique, Random Forest, Gradient Boosting, XGBoost, LightGBM), un nouveau run MLflow est lancé. Pour chaque run, MLflow enregistre automatiquement :

* Les hyperparamètres du modèle (ex : nombre d’estimateurs, learning\_rate, max\_depth…)
* Les métriques de performance (accuracy, F1-score, ROC-AUC, etc.)
* Les artefacts produits (matrices de confusion, graphiques d’importance des variables, rapports HTML)
* Le modèle entraîné, sérialisé et versionné
* Comparaison et sélection :  
  L’interface web de MLflow permet de visualiser et comparer les runs sur la base des métriques enregistrées. Cette fonctionnalité facilite l’identification du meilleur modèle et l’analyse de l’impact des hyperparamètres.

## Bénéfices

* Reproductibilité : Chaque expérience peut être rejouée à l’identique.
* Transparence : Toutes les étapes et résultats sont tracés et disponibles pour l’équipe.
* Collaboration : L’interface facilite le partage et la revue des expériences entre data scientists

## Conclusion

MLflow s’est avéré un outil essentiel pour la gestion du cycle de vie des modèles dans ce projet. Il a permis de structurer le processus d’expérimentation, d’optimiser la sélection du modèle final, et de garantir la robustesse et la traçabilité du pipeline de modélisation.

# MLFlow-Champion

Dans le cadre de PariVision, un script sur-mesure est développé afin de permettre une interaction avec MLFlow. Ce script Python est conçu pour interagir avec un serveur MLFlow dans le cadre d'une pipeline automatisée de gestion de modèles de machine learning.

Voici une synthèse de ses actions :

1. **Configuration du Client MLFlow :** Initialise un client MLFlow pour interagir avec le serveur.
2. **Récupération des Runs :** Récupère les runs (valides) terminées du modèle spécifique PariVision.
3. **Identification du Champion actuel :** Charge le modèle actuel marqué comme "Tenant au titre" pour obtenir son identifiant de run.
4. **Détermination du nouveau Champion :** Compare les performances des runs terminées pour identifier si un nouveau "champion" (meilleur modèle) est présent.
5. **Mise à Jour du Champion :** Si un nouveau champion est identifié, il enregistre (incrémente) une nouvelle version du modèle sur MLFlow et lui attribue l'alias "champion".

En résumé, ce script automatise le processus de comparaison des performances des modèles, identifie le meilleur modèle (champion), puis met à jour le serveur MLFlow en conséquence.

Cela permet de maintenir un modèle optimal en production de manière automatisée.

Section 4 – Exploitation

Ce chapitre explique comment la solution PariVision est exploité. Cela sert également de notice d’utilisation.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Pipeline ETL

La récupération et le traitement des données est ordonnées via Jenkins. Chaque requête permet de collecter un certain nombre documents pour alimenter un Lac de données afin d’avoir un contrôle granulaire sur la quantité de données ingéré par le modèle.

Finalement, est effectué la transformation et le stockage des données dans un entrepôt d’exploitation.

# Pipeline ML

Toujours ordonnés via Jenkins, la Pipeline complète automatisé entraîne les actions suivantes :

1. Génère un rapport de surveillance de dérive des données
2. Entraîne un nouveau modèle sur les données fraîches et injecte toutes les métriques et artefacts sur un serveur de tracking MLFlow
3. Effectuer une comparaison entre tous les modèles entraînés afin de déterminer un modèle “champion”, en se focalisant sur les scores de performance
4. Déploie un service conteneurisé qui charge le modèle champion et le met à disposition pour effectuer des prédictions via un service API
5. Déploie un service conteneurisé mettant à disposition une application pour l’utilisateur final

# Supervision

## Prometheus et Grafana

La supervision du projet PariVision est effectuée via Prometheus et Grafana.

Le service d’Alert Manager de Prometheus est configuré pour déclencher les alertes sur un salon SLACK dédié au projet.

Les détails de ces implémentations sont détaillés dans l’architecture ci-dessous :

monitoring  
├── alertmanager  
│   └── alertmanager.yml -> Configuration de Alter Manager (Slack)  
├── evidently  
│   └── main.py -> Génère un rapport de surveillance sur la dérive des données  
├── grafana  
│   └── provisioning -> La configuration de Grafana est entièrement provisionné via GIT  
│       ├── dashboards -> Les différents Dashbord (json) utile au projet  
│       └── datasources -> Les sources de données (Prometheus)  
└── prometheus   
   ├── prometheus.yml -> Configuration de Prometheus (Maintenances appliqués via Jenkins)  
   └── rules -> Les règles de Prometheus  
       ├── alerting.rules -> Règles d'alerte  
       └── recording.rules -> Règles d'enregistrement des métriques

## Slack

Une application sur Slack nous a été fourni par notre mentor, cette dernière est configuré avec AlertManager nous permet de recevoir les notifications d’alerte de Prometheus.

Configuration de l’application : <https://dst-dec.slack.com/marketplace/A0F7XDUAZ-webhooks-entrants>

Appel Webhooks :

curl -X POST --data-urlencode "payload={\"channel\": \"#dec24cmlops\_paris\_sportif\", \"username\": \"PariVision\", \"text\": \"TEST ALERT PROMETHEUS\", \"icon\_emoji\": \":trophy:\"}"

<https://hooks.slack.com/services/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>

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

Description générée automatiquement

## Evidently

La surveillance de la qualité des données utilisée pour l’entraînement est effectuée via Evidently afin de surveiller la dérive des données d’entrées du modèle.

## PariVision-Exporter

Un exporter Prometheus sur-mesure pour le modèle de Machine Learning a été développé, afin de surveiller les métriques du modèle utilisés, comme le temps de prédiction, le nombre de prédiction effectués, etc …

Section 5 – Conclusion

Ce projet a démontré le potentiel du machine learning et des pratiques MLOps dans le domaine complexe mais passionnant de la prédiction sportive appliquée à la NBA. Il constitue une illustration complète et professionnelle de la démarche data science appliquée au sport de haut niveau. À partir d’un jeu de données riche, nous avons conçu un pipeline robuste, reproductible et industrialisable, en respectant les meilleures pratiques du domaine.

En combinant rigueur scientifique et innovation technologique, il a permis de développer un outil robuste capable d’analyser efficacement les données techniques et contextuelles pour fournir des prédictions précises sur les résultats des matchs NBA.

# Les principaux points forts du projet

* Une infrastructure MLOps évolutive permettant l’automatisation complète du pipeline (collecte → entraînement → prédiction).
* Un code source organisé et claire entièrement orientée déclaratif et accompagnée d’une documentation exhaustive.
* Un stockage des données NoSQL pour maximiser les performances.
* Une flexibilité concernant l'évolution des performances des modèles grâce à un serveur de suivi des expériences.
* Une application utilisateur intuitive offrant une visualisation claire des performances algorithmiques.
* Un système d’orchestration confortable et automatisé.
* Une supervision complète entièrement gérée de façon déclarative pour assurer une maintenance efficace.

# Bilan

Ce projet fournit non seulement un outil puissant pour améliorer la précision des paris sportifs sur la NBA mais aussi une base solide pour explorer d’autres opportunités dans le domaine de l’analyse prédictive.

Grâce à son approche centrée sur les données et sa structure MLOps robuste, ce projet illustre comment la technologie peut transformer notre compréhension et notre exploitation stratégique du sport moderne.

L’intégration d’outils et de plateformes MLOps dans ce projet NBA a profondément transformé la gestion et la valorisation des données tout au long du cycle de vie du modèle.

* MLflow a permis de centraliser l’historique des entraînements, de comparer objectivement les modèles et d’assurer la sélection transparente des meilleures configurations.
* MongoDB, en tant que base NoSQL, a facilité la gestion de grands volumes de données hétérogènes et l’évolution du schéma sans rupture, ce qui est essentiel dans un contexte sportif où les données évoluent rapidement.
* Evidently, Pormetheus et Grafana ont apporté une surveillance continue et visuelle de la stabilité des données et des performances du modèle, permettant d’anticiper toute dérive et d’assurer la robustesse du système en production.
* Pour finir, les briques FastAPI et Streamlit permettent une exploitation claire et intuitive de la solution.

Au-delà de la simple automatisation, l’approche MLOps adoptée a permis de professionnaliser le déploiement, la supervision et la maintenance des modèles, tout en favorisant la collaboration. Cette démarche garantit non seulement la qualité et la fiabilité des prédictions, mais aussi la pérennité et la capacité d’adaptation du projet face aux évolutions futures du domaine sportif.

# Perspectives d’évolution

Pour aller plus loin, plusieurs axes d’amélioration et d’extension sont envisageables :

* **Exploiter la source de donnée réelle BetsAPI :** Cette API maintenue par un tiers en temps réel représente une opportunité d’exploiter des nouvelles données quotidienne fraîche.
* **Enrichissement des données** : intégrer des informations contextuelles supplémentaires (position des défenseurs, séquences de jeu, fatigue du joueur, météo, etc.), ou exploiter les données issues du tracking vidéo pour affiner la prédiction.
* **Modèles avancés** : explorer des architectures de deep learning (réseaux de neurones, modèles séquentiels, computer vision) pour traiter des données plus complexes ou non structurées.
* **Analyse en temps réel** : adapter le projet pour le traitement en streaming, permettant des prédictions instantanées pendant les matchs.
* **Personnalisation** : développer des modèles spécifiques par joueur ou par équipe, ou intégrer des modules de recommandation pour l’entraînement individualisé.
* **Expérience utilisateur** : concevoir des dashboards interactifs pour les coachs, analystes et fans, facilitant l’exploration des résultats et la prise de décision.
* **Automatisation sur alerte** : mettre en place des workflows automatiques en cas d’alerte détectée, pour garantir la pertinence continue du service.
* **Orchestration avancée avec Kubernetes** : apporter robustesse, flexibilité et performance à la solution en offrant une gestion déclarative en livraison continue, un système de mise à l’échelle en haute-disponibilité et une gestion des versions (canary)

Cette architecture permettrait de répondre efficacement aux besoins de production, tout en offrant un cadre évolutif pour les futures extensions du projet.

# Synthèse

Ce projet pose les bases d’une démarche data-driven ambitieuse au service de la performance sportive, tout en respectant les standards industriels de qualité, de robustesse et d’évolutivité.

Il constitue un modèle réplicable pour d’autres problématiques d’analyse prédictive dans le sport et au-delà et ouvre la voie à de nombreuses innovations à venir dans le domaine de l’intelligence artificielle appliquée.