

Rapports	2
1. Introduction	3
2. EDA	6
3. Mise en production	11
4. Exploitation	21
5. Conclusion	23

Rapports

 Par Heuzef  déc. 23, 2024  Voir le nombre de vues • [Legacy editor](#)

👋 1. Introduction

👤 Par Heuzef • En cours 📄 Voir le nombre de vues

[Projet PariVision : prédiction des résultats des matchs de Basketball \(NBA\)](#)

[Introduction](#)

[Objectifs](#)

[Flux de travail](#)

[Hébergement du projet](#)

Projet PariVision : prédiction des résultats des matchs de Basketball (NBA) 🔗



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, fatigue du joueur, 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.


Ce projet illustre comment le MLOps peut transformer des données brutes en entrées exploitable, en alignant précision algorithmique et exigences opérationnelles du monde sportif.

Objectifs 🔗

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 des workflows : Conteneuriser chaque étape (génération de données, feature engineering, entraînement) avec Docker, et orchestrer les pipelines via GitHub Actions et Jenkins.
- Adaptabilité continue : Implémenter un système de ré-entraînement conditionnel, où un nouveau modèle (Random Forest, XGBoost) ne remplace l'ancien que si son accuracy dépasse celle de l'ancien modèle.
- Garantir la traçabilité et la reproductibilité avec des intégrations CI/CD (GitHub Actions), MLFlow et Jenkins.
- Surveillance en production grâce à Grafana et Prometheus, assurant un monitoring granulaire des requêtes API et de la base de données MongoDB.

Flux de travail [↗](#)

 WORKFLOW

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 2025](#)
- Accès au dépôt Git du projet :  https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/  Miroir :  [heuzef/DEC24_MLOPS_PARIS_SPORTIFS](https://github.com/heuzef/DEC24_MLOPS_PARIS_SPORTIFS)



2. EDA

 Par SHI  En cours  Voir le nombre de vues

Sources de données

VirtualBetsAPI 

Présentation de l'outil

VBA est une API développée 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 Virtual Bets API

- Appel : Demande de retour de 2 nodes

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

- Résultat

```
1  [
2      {
3          "Game ID": "0029700427",
4          "Game Event ID": "389",
5          "Player ID": "100",
6          "Player Name": "Tim Legler",
7          "Team ID": "1610612764",
8          "Team Name": "Washington Wizards",
9          "Period": "4",
10         "Minutes Remaining": "11",
11         "Seconds Remaining": "22",
12         "Action Type": "Jump Shot",
13         "Shot Type": "2PT Field Goal",
14         "Shot Zone Basic": "Mid-Range",
15         "Shot Zone Area": "Right Side(R)",
16         "Shot Zone Range": "8-16 ft.",
17         "Shot Distance": "15",
18         "X Location": "117",
19         "Y Location": "109",
20         "Shot Made Flag": "1",
21         "Game Date": "19980102",
22         "Home Team": "WAS",
23         "Away Team": "IND",
24         "Season Type": "Regular Season"
25     },
26     {
27         "Game ID": "0029700427",
28         "Game Event ID": "406",
29         "Player ID": "100",
30         "Player Name": "Tim Legler",
31         "Team ID": "1610612764",
```

```

32     "Team Name": "Washington Wizards",
33     "Period": "4",
34     "Minutes Remaining": "9",
35     "Seconds Remaining": "36",
36     "Action Type": "Jump Shot",
37     "Shot Type": "2PT Field Goal",
38     "Shot Zone Basic": "Mid-Range",
39     "Shot Zone Area": "Right Side(R)",
40     "Shot Zone Range": "8-16 ft.",
41     "Shot Distance": "14",
42     "X Location": "143",
43     "Y Location": "25",
44     "Shot Made Flag": "0",
45     "Game Date": "19980102",
46     "Home Team": "WAS",
47     "Away Team": "IND",
48     "Season Type": "Regular Season"
49 }
50 ]

```

• Limite

Le 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 appelé 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`

Ressources [↗](#)


 [Ajout nouvelle source de data \(VirtualBetsAPI\)](#) **TERMINÉS**

 [jja4/nba_mlops](#)

BetsAPI [↗](#)

Utilisation du service BetsAPI [↗](#)

BetsAPI est un service RESTful pour les données sur tous les sports. C'est un service payant. L'avantage le plus important est qu'il fournit des vrais données de qualité, en temps réels.

Documentation :  [Introduction · Live Sports, Betting Odds API - BetsAPI](#)

Pour enregistrer le TOKEN dans une variable environment :

```
1 export TOKEN="SECRET-TOKEN-123"
```

Afficher le TOKEN [↗](#)

```
1 env | grep TOKEN
```

Tester l'API [↗](#)

```
1 curl -s "https://api.b365api.com/v1/bet365/inplay?token=$TOKEN" | jq '.success'
2
3 curl -s "https://api.b365api.com/v1/bet365/inplay?token=$TOKEN" | jq '.[0] | .name'
4
5 curl -s "https://api.b365api.com/v1/bet365/inplay?token=$TOKEN" > test.json
```

Stockage des données [↗](#)

Le stockage des données brutes et pré-traités s'effectue au format JSON, dans une base de données 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.

Créée en 2009 par MongoDB Inc., cette base de données est devenue l'une des solutions les plus populaires pour les applications modernes, notamment les applications web, mobiles, et les projets nécessitant un traitement rapide de gros volumes de données.

Installation sur Docker [↗](#)

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

```
1 docker image
```

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

```
1 docker pull mongo:latest
```

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

```
1 docker image
```

Step 4 : Créer un répertoire "mongoDB_PariVision" qui nous servira de volume pour la persistance des données :

```
1 mkdir mongoDB_PariVision
2 cd mongoDB_PariVision
```

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

```
1 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 :

```
1 docker ps
```

Step 7 : Accéder au container :


```
1 docker exec -it mongodb bash
```

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

```
1 mongosh
```

Step 9 : Créer l'utilisateur

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

Utilisation de mongoDB [↗](#)

a) Afficher les différentes bases de données

```
1 > show dbs
2 admin    0.000GB
3 config  0.000GB
4 local    0.000GB
5 >
```

b) Création de la base de données pour PariVision

```
1 > use parivision
2 switched to db parivision
3 >
```


c) Insertion d'un objet

```
1 > db.user.insert({"name": "Curry"})
2 WriteResult({ nInserted: 1 })
```

d) Lister les objets

```
1 > db.user.find()
2 { "_id" : ObjectId("223d432656h765jo987d234a3"), "name": "Curry" }
```

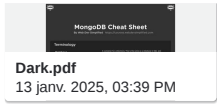
Utilisation de PyMongo [↗](#)

 Documentation de PyMongo : [Tutorial - PyMongo 4.12.0 documentation](#)

```
1 !pip3 install pymongo
2 from pymongo import MongoClient
3
4 client = MongoClient(
5     host = parivision.heuzef.com,
6     port = 27017,
7     username = *****,
8     password = *****
9 )
10
11 print(client.list_database_names())
```

Utile mongoDB [↗](#)

Sheet cheat **mongoDB** :



Analyse exploratoire [🔗](#)

Insérer le story telling de Shi ici.

3. Mise en production

 Par Heuzef  Voir le nombre de vues

[Déploiement de la VM](#)

[Connexion SSH](#)

[Installation](#)

[Initialisation](#)

[Configuration de la VM](#)

[GITHUB](#)

[Python](#)

[Docker](#)

[Homer](#)

[Jenkins](#)

[MLFlow](#)

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.

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

Installation

La VM est déployée sur un Hyperviseur Proxmox, l'installation d'Ubuntu Server est effectuée sans partitionnement LVM, avec les drivers des applications tierces, OpenSSH et sans ajout logiciels supplémentaire.

Un pare-feu et un reverse proxy est actuellement en place sur l'infra de Heuzef pour autoriser publiquement les ports suivants :

- 22 (SSH)
- 80 (HTTP)
- 443 (HTTPS)

Initialisation

```
1 # Mise à jour du système
2 sudo apt update && sudo apt upgrade
3
```

```

4 # Désactiver l'authentification par mot de passe
5 sudo vim /etc/ssh/sshd_config # PasswordAuthentication no
6 sudo systemctl restart ssh
7
8 # Ajouter les clefs SSH pour l'authentification
9 vim /home/parivision/.ssh/authorized_keys

```

 Une sauvegarde de bas niveau est effectuée à ce stade.

Configuration de la VM

```

1 # Configurer sur l'heure de Paris
2 sudo timedatectl set-timezone "Europe/Paris"
3
4 # Ajout de quelques outils
5 sudo apt install -y curl wget git tmux htop vim nano tree unzip smartmontools bmon

```

GITHUB

Le dépôt GIT du projet est hébergé ici : https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS **CONTENU RESTREINT**

Cloner le dépôt

```

1 # Création d'une nouvelle clef SSH
2 ssh-keygen -t ed25519
3
4 # La clef doit être ajoutée sur le repo github :
5 # https://github.com/DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS/settings/keys
6 cat .ssh/id_ed25519.pub
7
8 # Cloner le repo GIT :
9 cd
10 git clone git@github.com:DataScientest-Studio/DEC24_MLOPS_PARIS_SPORTIFS.git
11 tree DEC24_MLOPS_PARIS_SPORTIFS/

```

Actualiser le dépôt

```

1 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.yml](#)

```

1 name: Workflow GitHub Action
2
3 # Run Workflow when push on main
4 on:
5   push:
6     branches:
7       - master
8 jobs:
9   execute:
10    name: Update the production VM

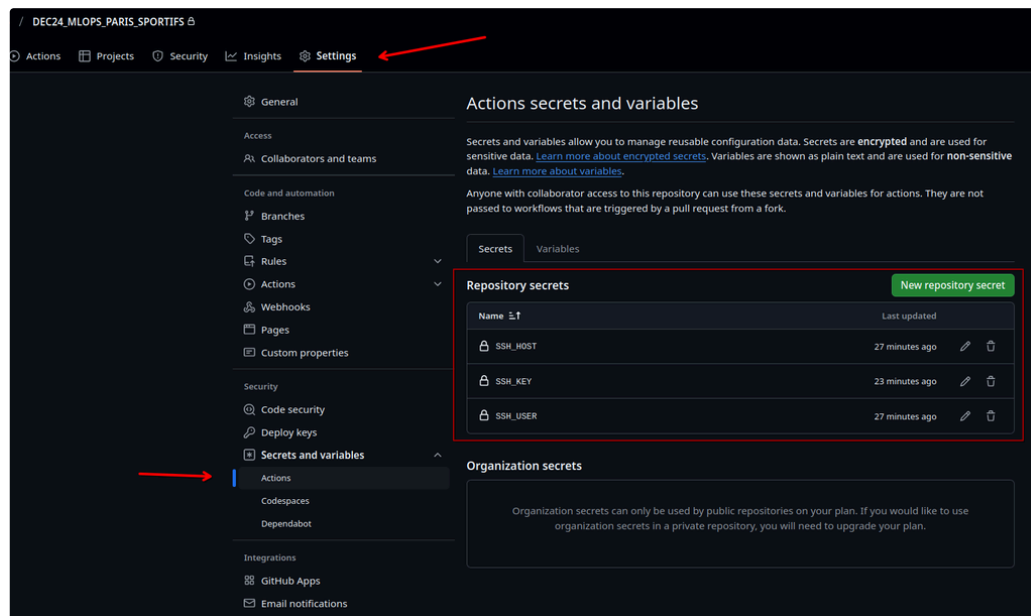
```

```

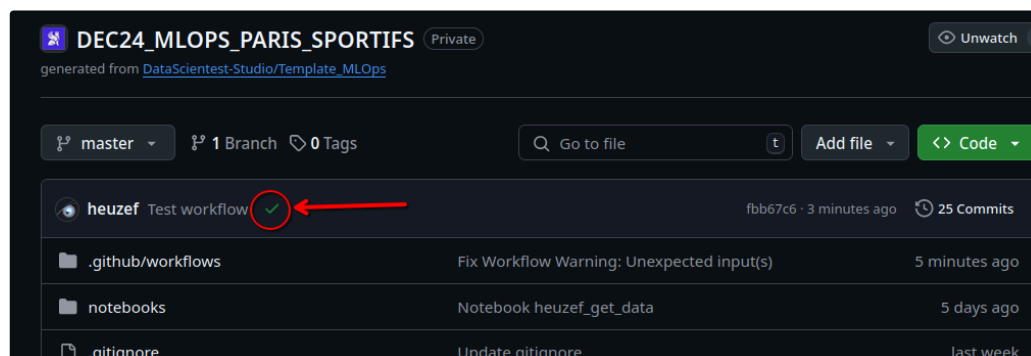
11 runs-on: ubuntu-latest
12 steps:
13 - name: Connecting on remote VM
14   uses: appleboy/ssh-action@master
15   with:
16     host: ${ secrets.SSH_HOST }
17     username: ${ secrets.SSH_USER }
18     key: ${ secrets.SSH_KEY }
19     script: |
20       cd /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/
21       git fetch
22       git pull

```

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



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



Python [🔗](#)

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

```

1 sudo apt upgrade python3

```

```
2 sudo apt install python3-pip python3-venv
```

Création de l'environnement virtuel [↗](#)

```
1 cd /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS
2 python3 -m venv .venv
3 source .venv/bin/activate
```

Installation des librairies [↗](#)

```
1 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.

```
1 /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/.env
```

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

```
1 import os
2 from dotenv import load_dotenv
3 load_dotenv(".env")
4
5 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 : 

```
1 # Add Docker's official GPG key:
2 sudo apt update
3 sudo apt install ca-certificates curl
4 sudo install -m 0755 -d /etc/apt/keyrings
5 sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
6 sudo chmod a+r /etc/apt/keyrings/docker.asc
7
8 # Add the repository to Apt sources:
9 echo \
10 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu
11 $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
12 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
13 sudo apt update
14
15 # Install the latest version:
```

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

Configuration [↗](#)

 Réf: [🔗 Post-installation steps](#)

```
1 # Post-install:
2 sudo usermod -aG docker parivision
3 newgrp docker
4 docker run hello-world
5
6 # Auto-start:
7 sudo systemctl enable docker.service
8 sudo systemctl enable containerd.service
```

Docker-Compose [↗](#)

La configuration complète de Docker repose sur Docker-Compose.

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


Pour tester, exécuter manuellement le lancement de l'infrastructure Docker ainsi :

```
1 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>

 Une sauvegarde de bas niveau est effectuée à ce stade.

Homer [↗](#)

Description [↗](#)

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  [bastienwartz/homer](https://github.com/bastienwartz/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 **CONTENU RESTREINT**

Le portail est configuré via un simple fichier YAML.

Jenkins [↗](#)

Installation [↗](#)

L'installation de Jenkins ne sera pas containerisée car le projet est sur une VM unique et le nœud maître doit avoir la possibilité de gérer les instances docker du projet.

```
1 sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
2 echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" https://pkg.jenkins.io/debian-stable binary/ | sudo tee /e
3
4 sudo apt-get update
5 sudo apt install -y fontconfig openjdk-17-jre jenkins
6
7 sudo usermod -aG docker jenkins
8 sudo systemctl restart docker
9
10
11 sudo systemctl enable jenkins
12 sudo systemctl stop jenkins
13 sudo systemctl start jenkins
14 sudo systemctl status jenkins
15
16 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 remédier à cela en créant un agent "parivision" qui se connectera sur l'hôte local avec l'utilisateur parivision.

Tableau de bord > Nœuds > parivision > Configurer

☒ Déconnexion
☒ Open Blue Ocean

État du lanceur de compilations 0/1 ▾

Number of executors ?
1

Répertoire de travail du système distant ?
/home/parivision/

Étiquettes ?
parivision

Utilisation ?
Utiliser ce nœud autant que possible

Méthode de lancement ?
Launch agents via SSH

Host ?
localhost

Credentials ?
parivision/***** (parivision)
+ Ajouter

Host Key Verification Strategy ?
Non verifying Verification Strategy

Avancé ▾ Edited

Disponibilité ?
Maintenir l'agent activé le plus longtemps possible

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

Tableau de bord > Nœuds > contrôleur > Configurer

☒ Statut
☒ Configurer

Nombre d'exécuteurs ?
0

Finalement, les différents Jobs du projets 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 **CONTENU RESTREINT**

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

Pipeline
Define your Pipeline using Groovy directly or pull it from source control.

Definition
Pipeline script from SCM

SCM ?
Git

Repositories ?

Repository URL ?
/home/parivision/DEC24_MLOPS_PARIS_SPORTIFS

Credentials ?
parivision/***** (parivision)

+ Ajouter

Avancé ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?
*/master

Add Branch

Navigateur de la base de code ?
(Auto)

Additional Behaviours
Ajouter ▾

Script Path ?
jenkinsfile

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

```
1 su jenkins
2 git config --global --add safe.directory /home/parivision/DEC24_MLOPS_PARIS_SPORTIFS/.git
```

i Une sauvegarde de bas niveau est effectuée à ce stade.

MLFlow [↗](#)

i Accès à l'instance déployé via Docker : <http://parivision.heuzef.com:5000>

[Initialisation](#)

[Utilisation](#)

[Configurer l'expérience Parivision](#)

Initialisation [↗](#)

Une fois les dépendance 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 appropriés :

```
1 docker exec -it mlflow bash
2 export MLFLOW_TRACKING_USERNAME=admin
3 export MLFLOW_TRACKING_PASSWORD=password
4 python3
5 >>> import mlflow
6 >>> from mlflow.server.auth.client import AuthServiceClient
7 >>> client = AuthServiceClient("http://localhost:5000")
8 >>> client.create_user(username="parivision", password="*****")
9 >>> client.update_user_admin(username="parivision", is_admin=True)
```

```

10 >>> client.get_user("parivision").is_admin
11 >>> mlflow.MlflowClient(tracking_uri="http://localhost:5000").create_experiment(name="parivision")
12 >>> client.update_user_password("admin", "*****")
13 >>> exit()
14 exit

```

Utilisation [↗](#)

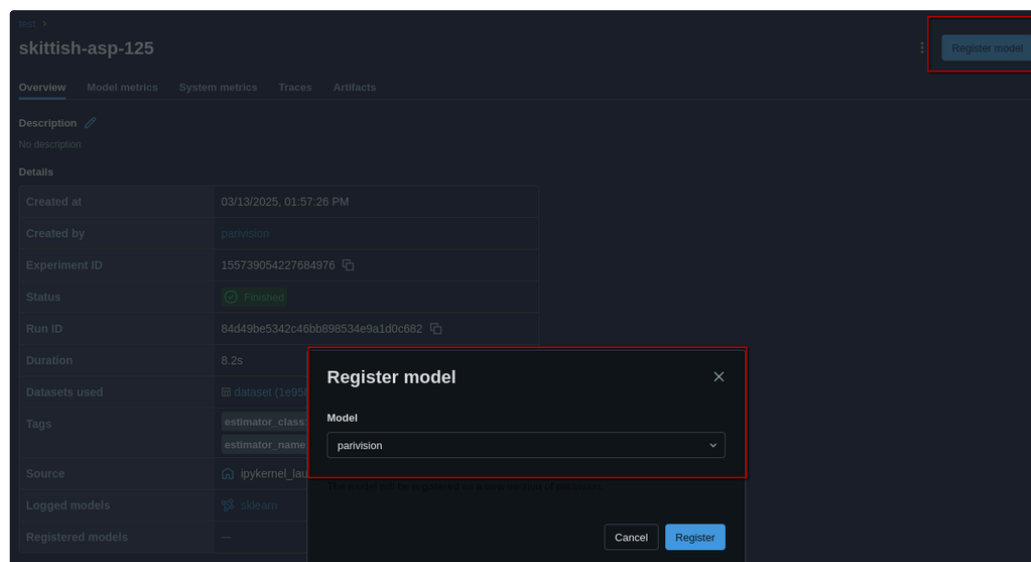
Se référer au Notebook tutoriel :

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

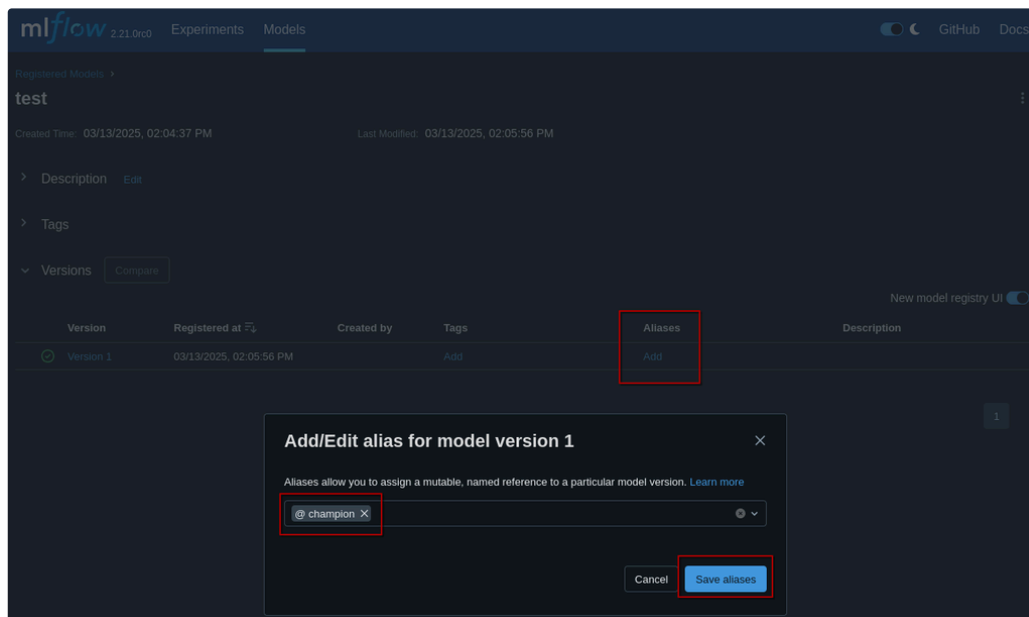
Configurer l'expérience PariVision [↗](#)

Afin de permettre le bon fonctionnement du Workflow, une expérience nommée ***"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 :



Finalement, lui attribuer l'alias ***"champion"*** :



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



4. Exploitation

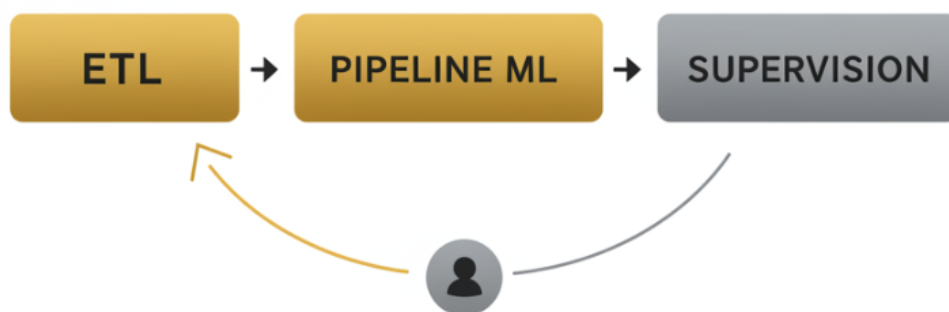
Par Heuzef [Voir le nombre de vues](#)

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

[Pipeline ETL](#)

[Pipeline ML](#)

[Supervision](#)



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 ce focalisant sur les scores de performance
4. Déploie un service conteneurisé qui charge le modèle champion et le met à disposition pour effectuer des predictions 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 à été fournit par notre mentor, cette dernière est configuré avec AlertManager nous permet de recevoir les notifications d'alert de Prometheus.

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

Appel Webhooks :

```
1 curl -X POST --data-urlencode "payload={\"channel\": \"#dec24cmlops_paris_sportif\", \"username\": \"PariVision\""
```



Preview unavailable

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 à été développé, afin de surveiller les métriques du modèle utilisés, comme le temps de prédiction, le nombre de prédiction effectués, etc ...

5. Conclusion

 Par Heuzef •  En cours  Voir le nombre de vues

Retours des expériences

Les défis à relever ...

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. 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. L'intégration d'un module de backtesting a également permis d'évaluer la rentabilité potentielle des stratégies basées sur ces prédictions, offrant ainsi aux utilisateurs un avantage concurrentiel dans le marché des paris sportifs.

Les principaux points forts du projet incluent :

- Une infrastructure MLOps évolutive permettant l'automatisation complète du pipeline (collecte → entraînement → prédiction).
- Des modèles performants comme XGBoost et LSTM capables de capturer les dynamiques spécifiques au basketball.
- Une interface utilisateur intuitive offrant une visualisation claire des performances algorithmiques et financières.

Cependant, plusieurs défis ont été rencontrés, notamment la gestion du bruit dans les données sociales (comme Twitter) et l'adaptation aux changements rapides dans les performances individuelles ou collectives en cours de saison. Ces obstacles ont été surmontés grâce à une approche méthodologique rigoureuse combinant traitement avancé des données (NLP pour l'analyse sociale) et monitoring continu en production.

En conclusion, ce projet marque une étape importante dans l'application du machine learning au domaine sportif. Il 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 sportive. 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.