**Déploiement**

Membre :

1. Abate
2. Bassim
3. Daoud

Le déploiement vise à rendre le modèle utilisable dans un environnement réel ou de production afin que les utilisateurs puissent obtenir des prédictions de manière simple, rapide et fiable.

**6.1 Sérialisation du modèle**

Le modèle final a été **sérialisé** afin de le sauvegarder dans un format réutilisable (.joblib).

* Ce processus permet de **charger le modèle rapidement** sans devoir le réentraîner.
* Le format choisi garantit une **taille compacte** et une compatibilité avec l’environnement Python utilisé.
* Si nécessaire, le modèle peut également être converti dans un format standard **ONNX** pour une compatibilité multi-plateformes.

**6.2 Mise en service du modèle**

Après sérialisation, le modèle a été **servi** pour exécuter des prédictions via :

* Une **application Python** locale pour les tests.
* Une **version déployée** dans un environnement cloud ou local (par exemple avec **Flask** ou **FastAPI**).
* Cette mise en service permet d’envoyer des données en entrée et de recevoir les prédictions instantanément.

**6.3 Intégration avec une API**

Pour rendre l’utilisation simple :

* Une **API REST** a été créée avec des endpoints spécifiques (/predict) acceptant des données en **JSON**.
* Les réponses sont également renvoyées en **JSON**, ce qui facilite l’intégration avec d’autres systèmes (site web, application mobile, etc.).
* Exemple de requête :

{

"age": 20,

"study\_hours\_per\_day": 4.5,

"attendance\_percentage": 95,

"sleep\_hours": 7

}

* Exemple de réponse :

{

"predicted\_exam\_score": 82.3

}

**6.4 Sécurité**

Des mesures de sécurité sont appliquées :

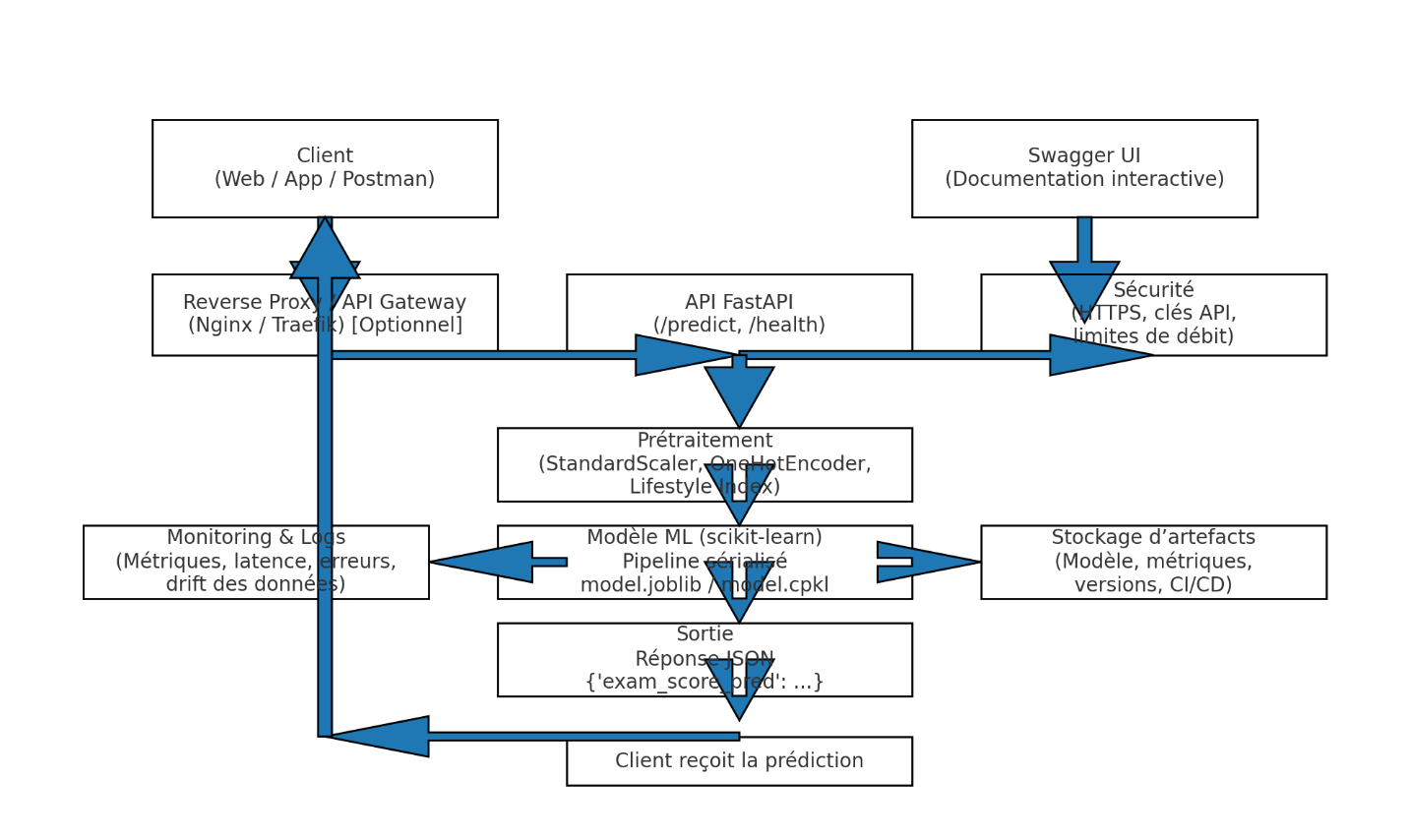
* **Authentification API** via clé secrète pour limiter l’accès.
* **Chiffrement HTTPS** pour protéger les échanges de données.
* **Filtrage d’entrées** pour éviter les injections ou données malveillantes.

**6.5 Surveillance et maintenance**

Après déploiement, le modèle est **monitoré** pour garantir des performances stables :

* **Temps de réponse** suivi pour détecter les ralentissements.
* **Taux d’erreurs** enregistré pour identifier les anomalies.
* Possibilité de **réentraîner** le modèle périodiquement si de nouvelles données sont disponibles.

**En résumé : le modèle est désormais prêt à être utilisé via une API sécurisée, hébergée dans un environnement stable, avec des mécanismes de surveillance et des possibilités d’évolution future.**



**Lecture du schéma (très bref) :**

* **Client / Swagger UI** → envoie une requête à l’**API FastAPI** (/predict).
* L’API applique le **prétraitement** (StandardScaler, OneHotEncoder, **Lifestyle Index**).
* Le **modèle sérialisé** (model.joblib ou model.cpkl) produit la prédiction.
* La **réponse JSON** revient au **client**.
* En parallèle : **Sécurité** (HTTPS/clé API), **Monitoring & Logs** (latence, erreurs, drift), **Stockage d’artefacts** (versions du modèle, métriques, CI/CD).