Documentation API – Système de Détection de Fraude avec IA

1. Présentation du système

Cette phase vise à intégrer un **modèle de Machine Learning de détection de fraude**, entraîné lors de la Partie 1, dans une **API REST** afin de permettre une utilisation automatisée dans un système bancaire.

Le modèle est exposé via une **API Flask**, déployée sur une plateforme cloud (**Render**), permettant de :

- Recevoir les caractéristiques d'une transaction
- Prédire si la transaction est frauduleuse ou non
- Retourner un score de probabilité de fraude

2. Export du modèle de Machine Learning

Le modèle de détection de fraude utilisé est un **Random Forest Classifier**, entraîné sur des données transactionnelles comprenant :

- Montant de la transaction
- Catégorie du commerçant
- Heure de la transaction
- Jour de la semaine

2.1 Format d'export

Le modèle a été exporté au format **.pkl** à l'aide de la bibliothèque **joblib**, afin de pouvoir être chargé et utilisé dans une application Flask.

```
joblib.dump(rf_model, "random_forest_model.pkl")
joblib.dump(encoder, "onehot_encoder.pkl")
```

Ces fichiers sont ensuite chargés au démarrage de l'API.

3. Architecture de l'API

- **Framework** : Flask (Python)
- **Type d'API** : REST
- **Déploiement** : Render.com
- **Format des données** : JSON

- **Modèle chargé en mémoire** au démarrage de l'application

4. Endpoint de prédiction

4.1 Endpoint /predict

- Description : *Cet endpoint permet d'analyser une transaction et de déterminer si elle est potentiellement frauduleuse.*

- URL : *POST /predict*

- Headers requis : *Content-Type: application/json*

4.2 Format de la requête

La requête doit contenir les caractéristiques de la transaction sous forme de tableau JSON.


```
{
  "features": [  120.50,"retail",14,3]
}
```


Correspondance des features :

Position	Feature	Description
0	amount	Montant de la transaction
1	merchant_category	Catégorie du commerçant
2	hour_of_day	Heure de la transaction (0–23)
3	day_of_week	Jour de la semaine (0=lundi)

4.3 Traitement interne

1. La catégorie du commerçant est encodée via un **OneHotEncoder**
2. Les variables numériques sont concaténées avec les variables encodées
3. Le modèle prédit :
    a. Une **classe** (fraude / non fraude)
    b. Une **probabilité de fraude**

4.4 Format de la réponse


```
{
  "fraud_score": 0.0957863958466249,
```

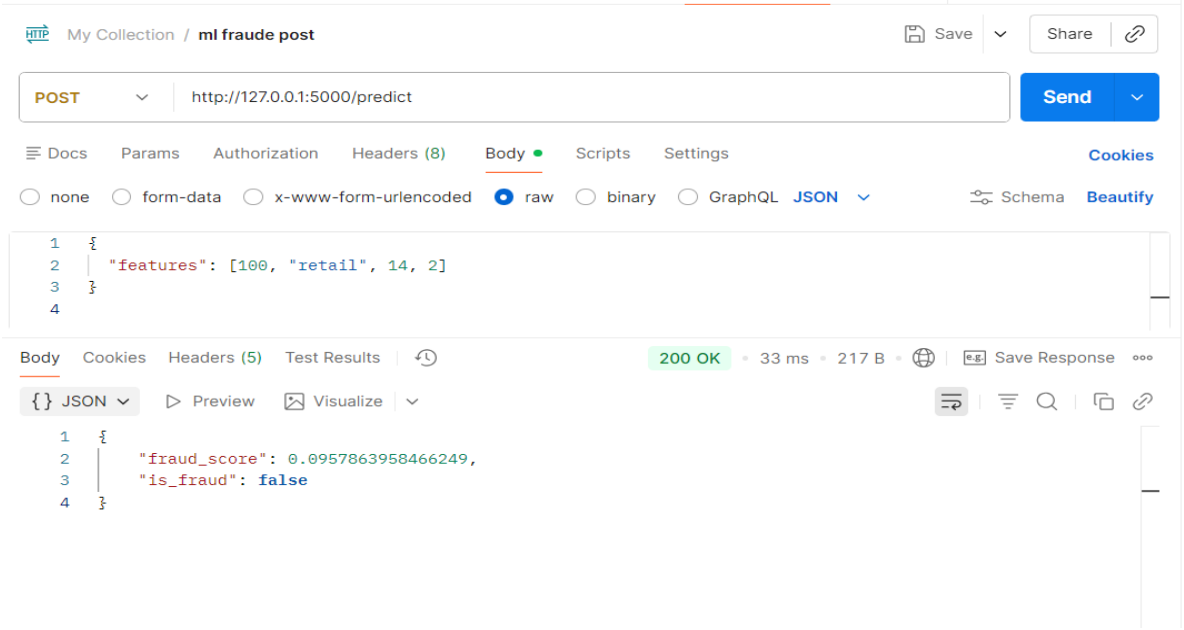
"is_fraud": false
}

Interprétation :

- is_fraud :
  - true → transaction suspecte
  - false → transaction normale
- fraud_score : probabilité (entre 0 et 1) que la transaction soit frauduleuse

5. Exemple d'utilisation

Requête HTTP



Cela indique que la transaction analysée présente un **faible risque de fraude**.

6. Déploiement de l'API

L'API a été déployée sur **Render.com** en tant que **Web Service Python**.

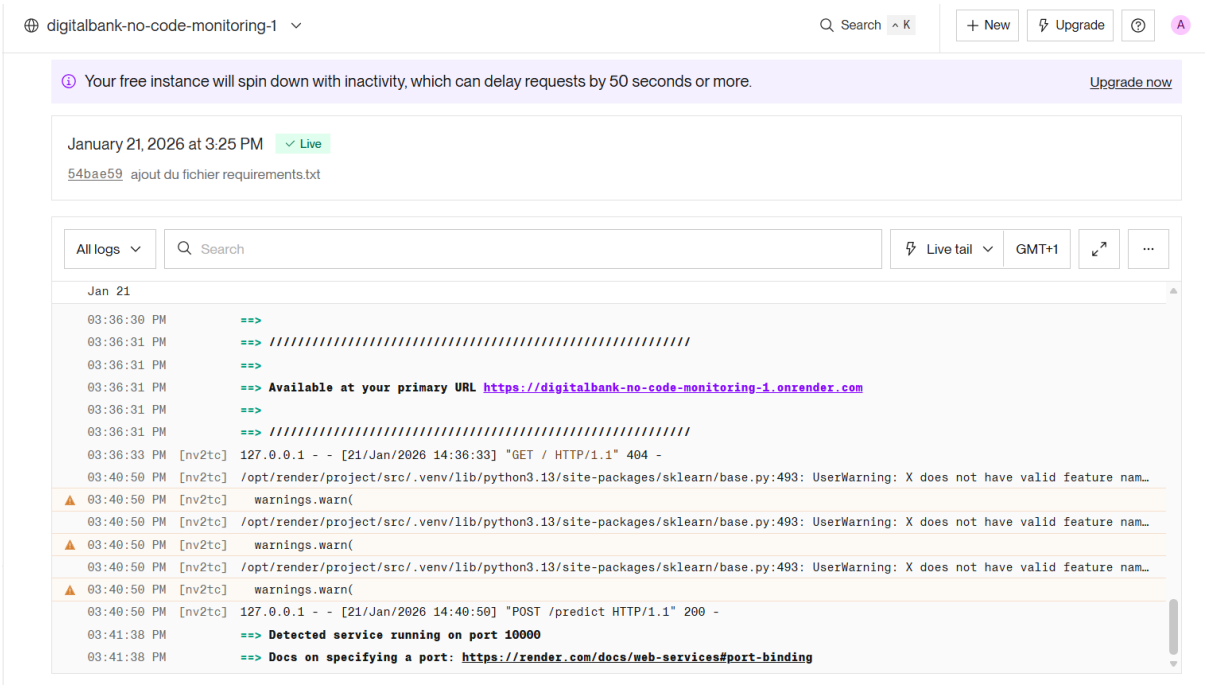
Caractéristiques du déploiement :

- Installation automatique des dépendances via requirements.txt
- Exposition du service sur un port public
- API accessible via une URL HTTPS

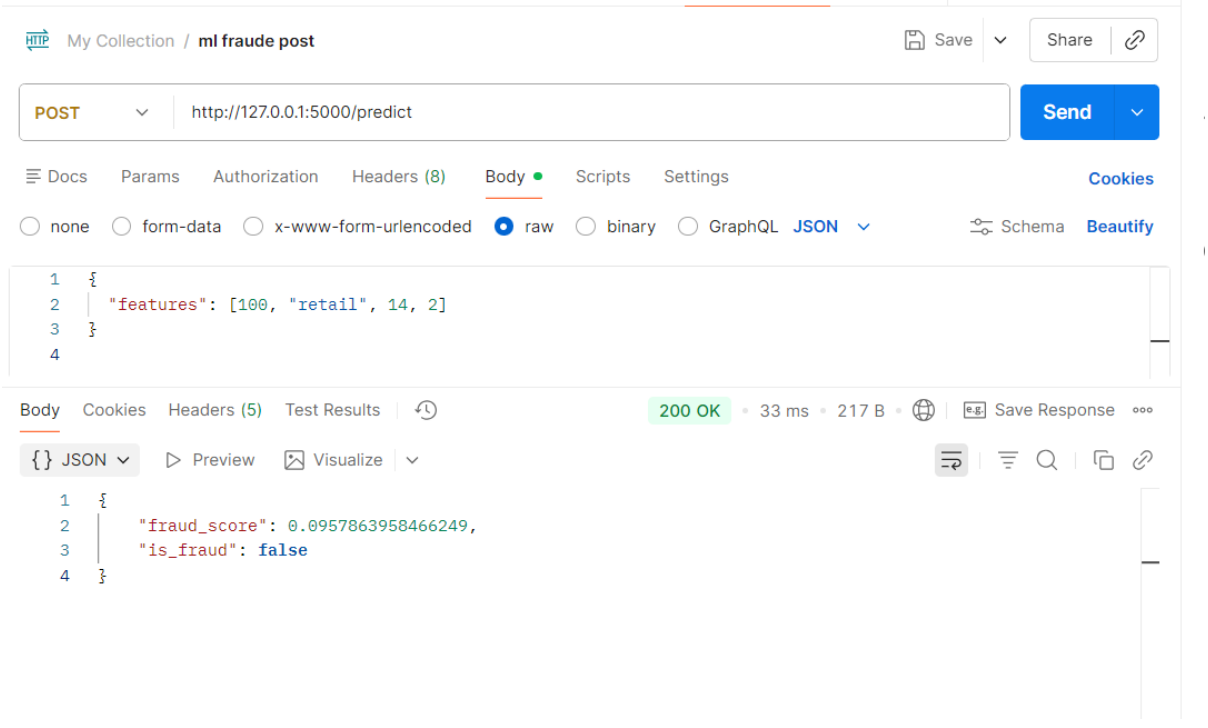
Le déploiement permet une intégration facile avec des outils externes tels que :

- Supabase
- n8n
- Make.com
- Postman

deployement



Test api



7. Limites actuelles

- Le workflow automatisé (n8n / Make.com) n'a pas encore été implémenté
- Les actions automatiques (email, dashboard, blocage de carte) ne sont pas actives

8. Conclusion

Cette API constitue une première brique fonctionnelle d'un **système de détection de fraude automatisé**.
Elle permet d'exposer un modèle de Machine Learning sous forme de service REST, prêt à être intégré dans des workflows no-code et des systèmes bancaires existants.