

# **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

The screenshot shows a POST request in Postman to `http://127.0.0.1:5000/predict`. The request body is a JSON object:

```
1 {  
2   "features": [100, "retail", 14, 2]  
3 }  
4
```

The response is a 200 OK status with the following JSON data:

```
1 {  
2   "fraud_score": 0.0957863958466249,  
3   "is_fraud": false  
4 }
```

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

## deployment

The screenshot shows the Render dashboard for a project named "digitalbank-no-code-monitoring-1". At the top, there's a message: "Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more." with a "Upgrade now" button. Below that, a log entry from January 21, 2026, at 3:25 PM is shown as "Live". The log details the deployment process, including the addition of a requirements.txt file and the execution of setup steps. It also shows the service starting on port 10000 and provides documentation links for port binding.

```
03:36:30 PM    ==>
03:36:31 PM    ==> ///////////////////////////////////////////////////
03:36:31 PM    ==>
03:36:31 PM    ==> Available at your primary URL https://digitalbank-no-code-monitoring-1.onrender.com
03:36:31 PM    ==>
03:36:31 PM    ==> ///////////////////////////////////////////////////
03:36:33 PM [nv2tc] 127.0.0.1 - - [21/Jan/2026 14:36:33] "GET / HTTP/1.1" 404 -
03:40:50 PM [nv2tc] /opt/render/project/src/.venv/lib/python3.13/site-packages/sklearn/base.py:493: UserWarning: X does not have valid feature nam...
▲ 03:40:50 PM [nv2tc] warnings.warn(
03:40:50 PM [nv2tc] /opt/render/project/src/.venv/lib/python3.13/site-packages/sklearn/base.py:493: UserWarning: X does not have valid feature nam...
▲ 03:40:50 PM [nv2tc] warnings.warn(
03:40:50 PM [nv2tc] /opt/render/project/src/.venv/lib/python3.13/site-packages/sklearn/base.py:493: UserWarning: X does not have valid feature nam...
▲ 03:40:50 PM [nv2tc] warnings.warn(
03:40:50 PM [nv2tc] 127.0.0.1 - - [21/Jan/2026 14:40:50] "POST /predict HTTP/1.1" 200 -
03:41:38 PM    ==> Detected service running on port 10000
03:41:38 PM    ==> Docs on specifying a port: https://render.com/docs/web-services#port-binding
```

## Test api

The screenshot shows a Postman collection titled "My Collection / ml fraude post". It contains a single POST request to "http://127.0.0.1:5000/predict". The request body is defined as follows:

```
1 {
2   "features": [100, "retail", 14, 2]
3 }
```

The response status is 200 OK with a response time of 33 ms and a size of 217 B. The response body is a JSON object:

```
1 {
2   "fraud_score": 0.0957863958466249,
3   "is_fraud": false
4 }
```

## 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.