Livrables

Mission : Déployez un modèle de machine learning

Ce document contient les livrables suivants :

1. Un dépôt Git
2. L'API fonctionnelle et déployée
3. Les scripts de tests unitaires et fonctionnels
4. La base de données PostgreSQL
5. La configuration du pipeline CI/CD

1. Dépôt Git

Le dépôt Git contient :

- L'ensemble du code source
- Le requirements.txt
- L'historique de commits, branches et utilisation de tags
- Le README.md complet

• Livrable : https://github.com/zsdata-git/employee-attrition-api.git

2. API fonctionnelle et déployée

L'API est déployée sur Hugging Face Spaces et accessible via les liens suivants :

• Interface Hugging Face : https://huggingface.co/spaces/Hayette/employee-attrition-api

• API en ligne : https://hayette-employee-attrition-api.hf.space

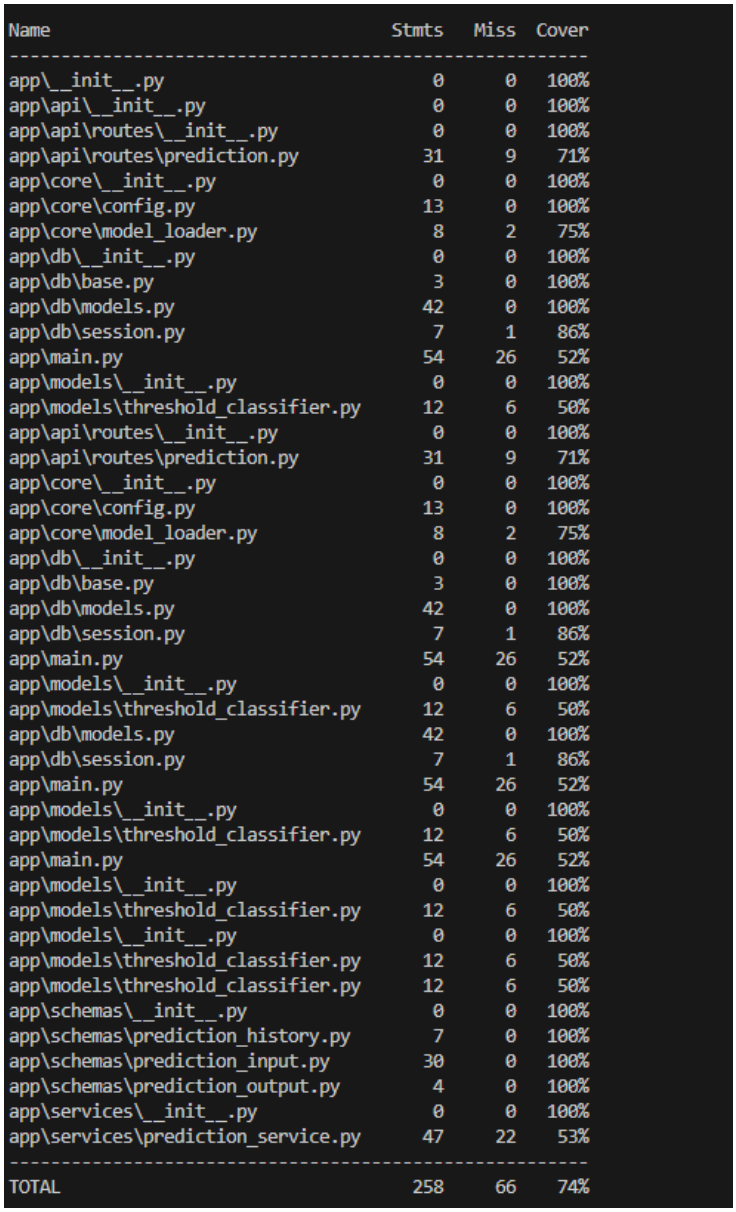
• Documentation interactive (Swagger) : https://hayette-employee-attrition-api.hf.space/docs

3. Scripts de tests

Les scripts des tests unitaires et fonctionnels se trouvent dans le repo GitHub

- tests/unit/test_prediction_service.py
- tests/functional/test_health.py
- tests/functional/test_predict.py

Les tests ont été exécutés avec Pytest. Le taux de couverture du code est de 74%, comme illustré ci-dessous :



Name	Stmts	Miss	Cover
app__init__.py	0	0	100%
app\api__init__.py	0	0	100%
app\api\routes__init__.py	0	0	100%
app\api\routes\prediction.py	31	9	71%
app\core__init__.py	0	0	100%
app\core\config.py	13	0	100%
app\core\model_loader.py	8	2	75%
app\db__init__.py	0	0	100%
app\db\base.py	3	0	100%
app\db\models.py	42	0	100%
app\db\session.py	7	1	86%
app\main.py	54	26	52%
app\models__init__.py	0	0	100%
app\models\threshold_classifier.py	12	6	50%
app\api\routes__init__.py	0	0	100%
app\api\routes\prediction.py	31	9	71%
app\core__init__.py	0	0	100%
app\core\config.py	13	0	100%
app\core\model_loader.py	8	2	75%
app\db__init__.py	0	0	100%
app\db\base.py	3	0	100%
app\db\models.py	42	0	100%
app\db\session.py	7	1	86%
app\main.py	54	26	52%
app\models__init__.py	0	0	100%
app\models\threshold_classifier.py	12	6	50%
app\db\models.py	42	0	100%
app\db\session.py	7	1	86%
app\main.py	54	26	52%
app\models__init__.py	0	0	100%
app\models\threshold_classifier.py	12	6	50%
app\main.py	54	26	52%
app\models__init__.py	0	0	100%
app\models\threshold_classifier.py	12	6	50%
app\models__init__.py	0	0	100%
app\models\threshold_classifier.py	12	6	50%
app\models\threshold_classifier.py	12	6	50%
app\schemas__init__.py	0	0	100%
app\schemas\prediction_history.py	7	0	100%
app\schemas\prediction_input.py	30	0	100%
app\schemas\prediction_output.py	4	0	100%
app\services__init__.py	0	0	100%
app\services\prediction_service.py	47	22	53%
TOTAL	258	66	74%

4. Base de données PostgreSQL

La version locale du projet utilisé PostgreSQL comme base de données principale. Cette base permet :

- de stocker les données des employés
- d'enregistrer l'historique des prédictions effectuées par l'API

Le dépôt Git contient tous les éléments nécessaires à sa mise en place :

- Les modèles SQLAlchemy définis dans app/db/models.py
- La classe de base SQLAlchemy : app/db/base.py
- La configuration de connexion à la base : app/db/session.py
- Script de création des tables : scripts/create_db.py
- Script de chargement des données : scripts/load_dataset.py

Preuve de fonctionnement :

La base PostgreSQL a été testée localement avec succès (les tables ont été créées correctement, les données contenant 1470 employés ont été chargées, les prédictions sont enregistrées dans la base)

```
employee_attrition_db=# \c employee_attrition_db
Vous êtes maintenant connecté à la base de données « employee_attrition_db » en tant qu'utilisateur « postgres ».
employee_attrition_db=# \dt
             Liste des tables
 SchÚma |     Nom     | Type  | PropriÚtaire
--------+-------------+-------+--------------
 public | employees   | table | postgres
 public | predictions | table | postgres
(2 lignes)
```

```
employee_attrition_db=# SELECT COUNT(*) FROM employees;
 count
-------
  1470
(1 ligne)
```

```
employee_attrition_db=# SELECT id, prediction, probability FROM predictions;
 id | prediction |     probability
----+------------+---------------------
  1 |          0 | 0.43620411470795206
  2 |          0 | 0.10560790945251508
  3 |          0 |  0.5633724213107569
  4 |          0 | 0.43620411470795206
  5 |          0 |  0.38602146346348008
  6 |          0 |  0.3860214634634008
  7 |          0 | 0.43620411470795206
  8 |          1 |                 0.8
  9 |          1 |                 0.8
 10 |          0 | 0.43620411470795206
 11 |          0 | 0.26106294019765564
 12 |          1 |                 0.8
 13 |          1 |                 0.8
(13 lignes)
```

Remarque sur le déploiement :

Pour le déploiement sur Higgins Face, une base SQLite a été utilisée afin de s'adapter aux contraintes de l'environnement distant.

5. Configuration du pipeline CI/CD

Le projet contient des workflows GitHub Actions :

- ci.yml qui automatise l'exécution des tests (.github/workflows/ci.yml)
- déploy.yml qui automatise le déploiement vers Hugging Face (.github/workflows/deploy.yml)

La gestion des environnements et des secrets repose sur :

- Des variables de configuration définies dans app/core/config.py
- Un environnement local utilisant PostgreSQL et un environnement distant HF utilisant SQLite
- Un secret GitHub HF_TOKEN utilisé pour le déploiement sécurisé.

Auteur : Bouzouita Hayette