Travail dirigé : Tests orientés data

© Pierre-Antoine Guillaume

2024-12-09

TP : le test appliqué au data

Visée

- Ce TP est le TP final
- La consigne vise à vous faire écrire la partie « test » et la partie « monitoring » d'un pipeline de MLOps
- ainsi qu'à écrire une stratégie de test pour ce système
- Dans le dossier, beaucoup de classes sont utilisées, cela facilite la testabilité
- Si vous n'etes pas à l'aise avec les classes, regardez les exemples de test_cleaner.py et test_pipeline.py
- Lorsque vous modifierez le code, si vous ne savez pas faire avec des classes, vous pouvez utiliser ce que vous préférez

1 Rendu

Le format attendu du rendu est un fichier ainsi nommé :
methodologie-du-test-tp3-m2-dataeng-<nom>-<prenom>.<ipynb|pdf>
Il sera rendu en fonction des possibilités :

1. soit via mail à pierreantoine.guillaume@ynov.com en **pièce jointe**
2. soit via teams

Détails techniques

Il vous est fourni dans le repertoire `https://github.com/PierreAntoineGuillaume/methodologie-du-test/tree/main/session_4`

Vous est fourni dans ce dépôt un programme qui nettoie des données, et entraine un modèle.

L'enjeux de ce programme est de faire de l'estimation de bien immobiliers depuis trois paramètres, une surface, un type de bien, et un nombre de pièces.

Mettre ce genre de programme dans un pipeline de MLOps a pour but de proposer un modèle se basant sur des données rafraichies automatiquement ; sans craindre un appauvrissement du modèle soudain dû à des changements de code non testés, ni sans craindre un arrivage de données de qualité médiocres.

Première Partie : Compréhension du code

Familiarisez vous avec les différents composants du systèmes.

Voilà les commandes qui permettent d'interagir avec le système :

```
# ouvrez votre IDE dans le dossier session_4
python3.12 -m venv .venv

# sur une distribution basee sur unix
. .venv/bin/activate

# sur windows
# Il est possible que vous deviez taper la commande suivante pour pouvoir activer le venv
Set-ExecutionPolicy Unrestricted -Scope Process
.venv/scripts/activate

# partout
python -m pip install -r requirements.txt

python src/main.py train # pour lancer le nettoyage de donnees et l'entrainnement
python src/main.py predict # pour faire des predictions
```

Le programme est pourvu de plusieurs composants sous forme de classes.

- une classe Pipeline, qui orchestre tous les autres composants et est le point d'entrée de l'entrainnement comme de la prédiction
- un fichier data, qui donne les formes des données utilisées
- une classe Provider, qui asbtrait la récupération des données.
- JsonProvider.get implémente Provider retourne une liste de tuples à nettoyer qu'il récupère d'un fichier json
- une classe Cleaner, qui fait le nettoyage des données en transformant des TupleSale en TuplePropre
- une classe Trainer qui abstrait l'entrainnement d'un modèle
- une classe ScikitLearnTrainer qui implémente Trainer et qui utilise ScikitLearn pour produire un modèle
- une classe Model qui encapsule l'utilisation de ScikitLearn, produite par ScikitLearnTrainer (ici, le coût d'une abstraction aurait trop amoindri la lisibilité du TP)
- une classes ModelSaver qui abstrait la sauvegarde d'un modèle pour réutilisation ultérieure
- une classe ModelComparator qui abstrait la sauvegarde et la comparaison des performances d'un modèle

Les fichiers src/main.py et tests/test_pipeline.py montrent plusieurs facons d'utiliser la classe Pipeline

Par ailleurs, des Mocks adaptés aux abstractions ont été fournis, et la fonction make_pipeline dans tests/test_pipeline.py permet d'instancier un pipeline en mockant tout ce qui touche réèlement au système.

```
def test_pipeline() -> None:
    make_pipeline(provider=MockProvider([TupleSale(prix="100000", surface="100", type_bien="
        maison", pieces="4")]))
```

Note : Si vous lancez le programme, vous verrez que les modèles sont nommés à partir d'identifiants dits « human readable ».

```
from hrid import HRID
HRID().generate() # walkon-seal-turn-rightfully
HRID().generate() # frantic-lynx-carried-promptly
```

Ici, le seul objectif est de pouvoir communiquer à propos de ces modèles. Ces identifiants sont aléatoires et parfois, des mots surprenants peuvent en sortir.

Seconde Partie : Description d'une stratégie de test

Voilà les suppositions concernant le produit :
- Vous pouvez facturer une estimation entre vingt centimes et cinq euros à un consommateur, en fonction de la négociation et de ce que vous espérez tirer d'un client
- Vous commercialisez votre moteur d'estimation à des agents immobiliers
- une estimation de bonne qualité attire les clients finaux des agents immobiliers, vous influez donc sur leur image et en ce sens vous êtes apporteur d'affaires
- dans ce cas, il est acceptable de calculer l'estimation et l'envoyer au client par mail et non pas au retour de l'API, ce qui permet également une prise de contact à l'agent immobilier voire une inscription du client sur son site
- Lorsqu'un utilisateur fait une demande d'estimation sur le site de l'agent immobilier, celui ci vous envoie une requête via média non représenté dans le code du dépôt comme suit :
  - un serveur http attend des requêtes
  - lorsqu'une requête arrive, elle contient une ID de réponse qui identifie la requête auprès du demandeur de service, des identifiants qui authentifient le client, et les données de l'estimation
  - cette requête est mise en queue pour être traitée de manière asynchrone
  - lorsque la prediction est terminée, le callback est effectué vers le site du client avec le resultat de l'estimation

Voilà la consigne :
- Écrivez une stratégie de test concernant le produit :
- Expliquez quels sont les risques du produit
- Expliquez quel est son service rendu
- Détaillez les différents composants du système
- Expliquer quel typologie de test (unitaire, intégration, contrat, fonctionnel, End-to-end, Securité) permet de les tester automatiquement
- Expliquer quel monitoring utiliser pour garantir une continuité de service

Troisième Partie : Application

- Écrivez des tests unitaires pour le composant Cleaner
- Faites en sorte que seul le meilleur modèle soit utilisé par le pipeline
- Rajoutez des logs sur les différents composants pour garder une trace explicite des flux de données, les taux de transformations
- Vérifiez que la distribution de donnée que vous récupérez est homogène avec un snapshot préalable (Pour garantir une stabilité des données dans le temps)