Cahier des charges

Projet MLOps – RAKUTEN

1. Contexte

Ce projet s'inscrit dans le challenge Rakuten France Multimodal Product Data Classification : Il s'agit de prédire le code type de produits (tel que défini dans le catalogue Rakuten France) à partir d'une description texte et d'une image.

Challenge data : https://challengedata.ens.fr/participants/challenges/35/

Le chiffre d'affaires du commerce en ligne augmente de façon importante depuis plusieurs années (plus de 100% entre 2013 et 2019). L'arrivée de la COVID-19 a accéléré cette tendance qui va certainement s'inscrire dans le temps (étude Médiamétrie-FEVAD). Les progrès dans le domaine de classification « image plus texte » avaient été limités ces dernières années en raison du manque de données réelles provenant de catalogues commerciaux.

Ces techniques numériques d'automatisation des tâches sont fortement sollicitées par les entreprises de e-commerce, en particulier pour les raisons suivantes :

- Classification de produit à grande échelle pour mieux gérer l'offre d'un site
- Recommandation de produit (génère en moyenne 35% de revenu supplémentaire)
- Amélioration de l'expérience client

2. Base de données

Pour ce challenge Rakuten, fournit les fichiers suivants :

- X_train_update.csv : il s'agit d'un tableau contenant 84916 échantillons d'entraînement avec la description textuelle ainsi que le référencement du fichier image associé
- X_test_update.csv : ce fichier est un tableau de 13812 échantillons de test

  - **Id** : cet identifiant est utilisé pour associer le produit à son code correspondant.
  - **designation** : le titre du produit, un texte court le résumant.
  - **description** : un texte plus détaillé décrivant le produit. Tous les exposants n'utilisent pas ce champ, donc pour conserver l'originalité des données, le champ de description peut contenir une valeur NaN. productid : un identifiant unique pour le produit.
  - **imageid** : un identifiant unique pour l'image associée au produit.

- y_train_CVw08PX.csv : ce fichier correspond à un tableau contenant les 84916 codes produits (prdtypecode).
- images.zip : cela correspond à l'archivage des fichiers images, d'un nombre total de 84916 images dans le répertoire image_train et de 13812 images dans le répertoire image_test. Afin de pouvoir explorer, data visualiser et entraîner nos modèles dans des notebooks Jupyter, nous allons travailler dans un google collab / Drive afin d'y stocker les datasets et pouvoir entrainer sur une journée des modèles.

3. Modèle de référence et objectifs

Un modèle de référence est indiqué par le site du challenge : l'objectif du challenge est d'améliorer ce modèle. Cette référence est en réalité composée de deux modèles distincts : un pour les images, un pour le texte (les participants sont encouragés à utiliser ces deux sources lors de la conception d'un classificateur, car elles contiennent des informations complémentaires) :

1. Pour les données images, une version du modèle Residual Networks (ResNet), le ResNet50 pré-entraîné avec un jeu de données Imagenet; 27 couches différentes du haut sont dégelées, dont 8 couches de convolution pour l'entraînement.

2. Pour les données textes, un classificateur RNN simplifié est utilisé. Seuls les champs de désignation sont utilisés dans ce modèle de référence.

Les données appartiennent à 27 classes distinctes. Pour évaluer la qualité de la classification, il est demandé **d'utiliser la métrique weighted-F1-score**. Il s'agit de la moyenne des F1-scores de toutes les classes pondérées par le nombre de représentants dans ces classes.

Le F1-score de chaque classe est la moyenne harmonique de la précision et du rappel pour cette classe.

Le modèle de référence obtient les résultats suivants :

- 0.5534 pour le modèle image (ResNet)
- 0.8113 pour le modèle texte (RNN)

Optimiser les performances des modèles de référence selon un vision opérationnelle va être notre objectif. En effet dans le cadre d'un projet de MLops, nous n'allons pas tenter d'améliorer un modèle mais plutôt de mettre en production une configuration automatisée qui va tester tous les modèles qui seront disponibles et gérer le CI/CD ainsi que la surveillance des métriques et performances.

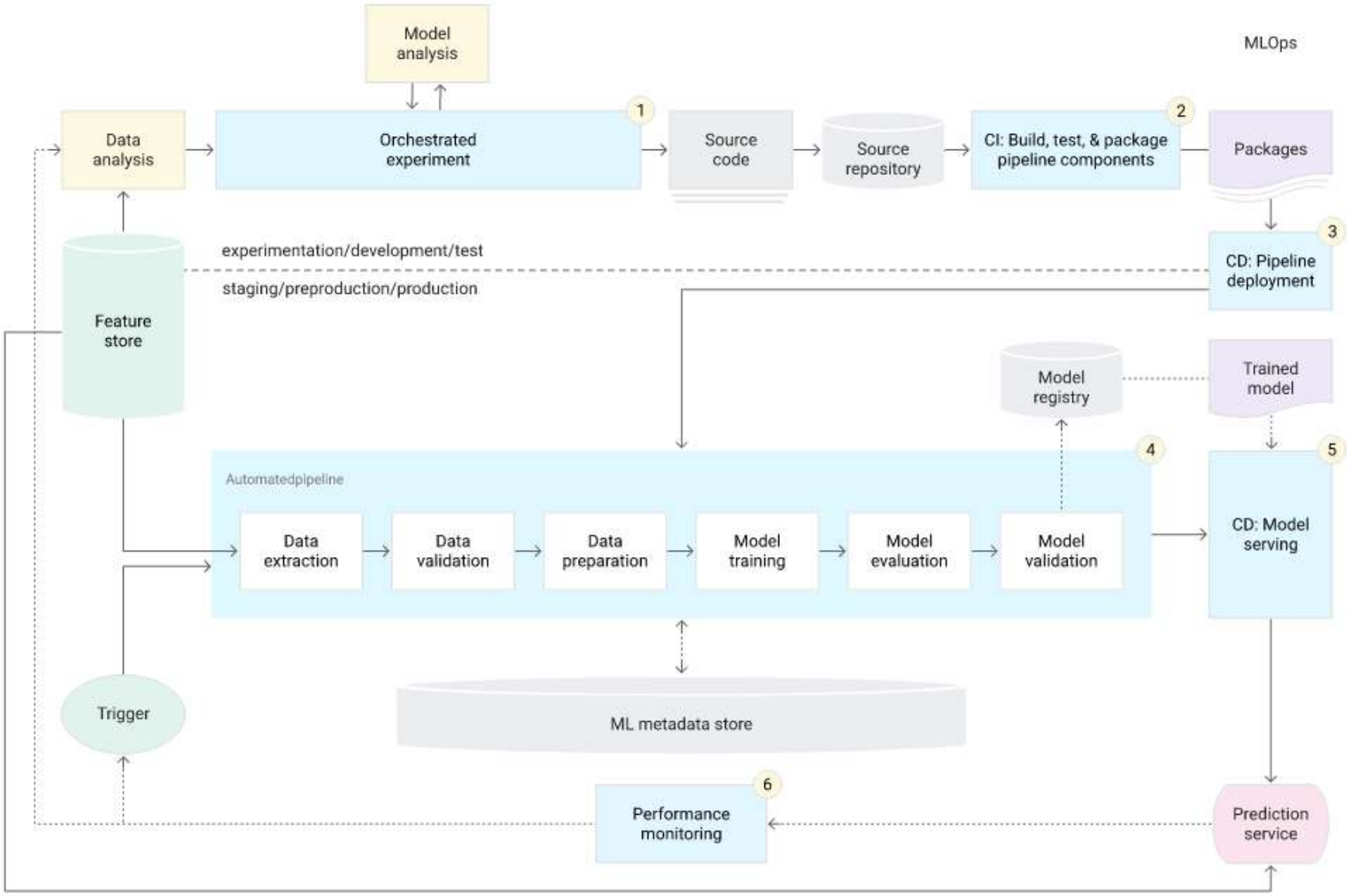
Pour cela nous allons partir des travaux des équipes précédentes qui ont développés ce projet sous la vision « Datascientist ».

4. Configuration MLops choisie

Une configuration standard complète MLOps comprend les composants suivants :

- Contrôle des sources
- Tests et création des services
- Services de déploiement
- Registre de modèles
- Magasin de caractéristiques
- Magasin de métadonnées de ML
- Orchestrateur de pipeline de ML

Le schéma suivant (page 5) montre la mise en œuvre du pipeline de ML avec intégration CI/CD, qui présente les caractéristiques de la configuration des pipelines de ML automatisés et les routines CI/CD automatisées.

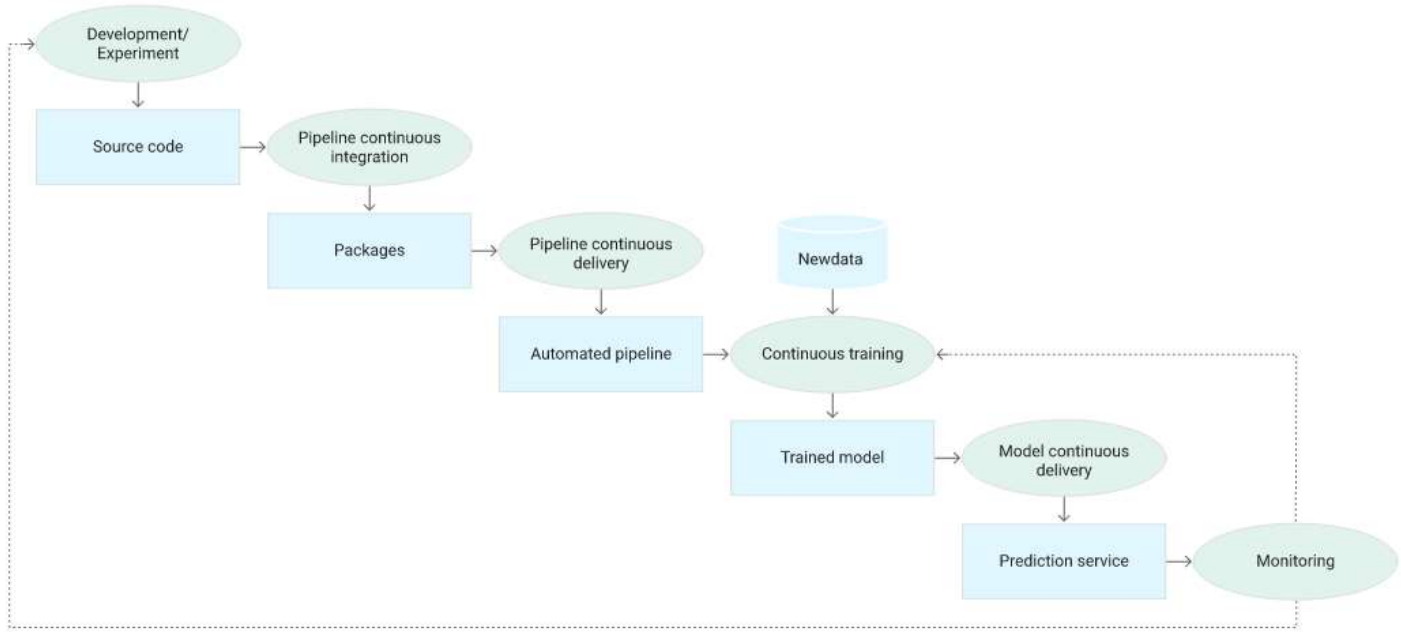

MLOps
Model analysis
Data analysis
Orchestrated experiment
1
Source code
Source repository
CI: Build, test, & package pipeline components
2
Packages
3
CD: Pipeline deployment
experimentation/development/test
staging/preproduction/production
Feature store
Model registry
Trained model
4
5
Automatedpipeline
Data extraction
Data validation
Data preparation
Model training
Model evaluation
Model validation
CD: Model serving
Trigger
ML metadata store
6
Performance monitoring
Prediction service

Dans le cadre du temps disponible, nous allons proposer un pipeline plus linéaire et plus simple pour notre projet.

Le schéma suivant montre les étapes du pipeline de ML avec routines CI/CD automatisées :

Le pipeline comprend les étapes suivantes :

1. Développement et expérimentation : orchestrez les étapes de test et testez de manière itérative de nouveaux algorithmes de ML et une nouvelle modélisation. Le résultat de cette étape est le code source des étapes du pipeline de ML qui sont ensuite transférées vers un dépôt source. **Dans notre cas nous allons partir de travaux déjà réalisés par des équipes précédentes situé sur un repo Github**
2. Intégration continue du pipeline : créez le code source et exécutez divers tests. Les résultats de cette étape sont les composants du pipeline (packages, exécutables et artefacts) qui seront déployés ultérieurement.
3. Livraison continue du pipeline : déployez les artefacts produits par l'étape CI dans l'environnement cible. Le résultat de cette étape est un pipeline déployé avec la nouvelle mise en œuvre du modèle.
4. Déclenchement automatisé : le pipeline est automatiquement exécuté en production en fonction d'un programme ou en réponse à un déclencheur. Le résultat de cette étape est un modèle entraîné qui est transmis au registre de modèles.
5. Livraison continue du modèle : diffusez le modèle entraîné en tant que service pour les prédictions. Le résultat de cette étape est un service de prédiction de modèle déployé.
6. Surveillance : collectez des statistiques sur les performances du modèle en fonction des données en ligne. Le résultat de cette étape est un déclencheur permettant d'exécuter le pipeline ou d'exécuter un nouveau cycle de tests.



5. BACKLOG

Les actions prioritaires à mener pour créer le pipeline comprennent les étapes suivantes :

1 - Préparer trois modèles de prédictions :

- un modèle type RNN permettant une prédiction de classification du texte contenu dans la description et la désignation des produits
- un modèle type de « computer vision » basé sur l'algorithme « Xception » permettant une prédiction de classification sur l'image jointe aux descriptions
- un modèle dit « bi-modal » qui va mettre en œuvre les deux modèles précédents et livrer une prédiction de classification sur toutes les data d'un sujet.

2 - Prototypage :

- Explorer les données et finaliser la Data visualisation
- Reprendre les modèles des notebooks
- Réaliser un refactorisation des codes des notebooks (VS CODE) afin d'obtenir des codes pythons séparés et décomposés en fonctions distinctes (preprocessing, entraînement, inférence)

3 - Préparer l'industrialisation du code avec MLflow :

Outil permettant d'industrialiser de bout en bout les process de mise au point de projets Machine Learning.

4 - Mettre en place l'architecture d'archivage dans un repo Github

5 - Création de l'API : réalisation de l'API avec FastAPI

- Gestion des données a approfondir (ajout, suppression, autre dataset….)
- Point de terminaison pour vérifier la disponibilité de l'API
- Mettre en place les fonctions d'authentification et autorisation
- Mettre en place une fonction d'administration permettant l'ajout et la suppression/modification de comptes utilisateurs.
- Réaliser la classification selon un modèle déterminé (choix possibles)

6 - CI/CD : tests a réaliser en pipeline automatisé (docker compose/ jenkins / autres ?)

- Tests unitaire des fonctions distinctes des codes pythons
- Réaliser les tests pour authentification et autorisation
- Réaliser le monitoring des performances du modèle
- Réaliser une extraction des paramètres (journal log.txt)
- Réaliser des tests sur la validation des prédictions (classifications) des modèles

- Tester l’environnement, vérifier la compatibilité du modèle avec l'infrastructure cible avant de déployer le modèle. (Container avec requirements.txt automatisé et env virtuel)

7 - Monitoring

- Fonction pour collectez des statistiques sur les performances du modèle (sur le dataset de test)
- Rapport mensuel de validité des performance pouvant conditionner la relance automatique du pipeline pour réentraîner, CI/CD et validation.

8 - orchestrations : Déploiement automatisé (Docker / Kubernetes / …..)

- Dans un environnement de test
- En version finale sur VM

6. Schéma d'implémentation