Cahier des charges projet MLOps

DataScientest • com

SOMMAIRE

1) **Contexte et Objectifs**

La demande croissante de solutions capables de catégoriser automatiquement les produits, que ce soit par leur aspect visuel, par une description textuelle ou par une combinaison des deux, est un véritable défi pour le monde des affaires moderne. Pour une grande entreprise comme Rakuten, il est impératif non seulement de répondre à cette demande, mais également de le faire d'une manière qui garantit rapidité, précision et évolutivité. Dans un environnement commercial dynamique qui évolue constamment, les décisions doivent être prises rapidement, tout en assurant une performance optimale.

Points clés du projet:

Commanditaire de l'application :

Rakuten, une entreprise majeure du commerce électronique, est à la recherche d'une solution automatisée, évolutive et en temps réel d'analyse et de prédiction de données. Cette API aidera Rakuten à améliorer sa prise de décision, à optimiser ses opérations et à fournir une expérience utilisateur encore plus enrichissante à sa vaste clientèle.

***API (Application Programming Interface)**

Utilisateur de l'application :

Les professionnels et partenaires de Rakuten, ainsi que les utilisateurs finaux qui souhaitent classer et catégoriser des produits à l'aide de l'application. Ces utilisateurs accéderont à l'API via une interface intuitive, soumettront leurs données (photos, descriptions ou les deux) et obtiendront des classifications précises en réponse.

Administrateur de l'application :

Les équipes techniques et les responsables IT au sein de Rakuten, qui auront la responsabilité de gérer, déployer, mettre à jour et surveiller l'API pour assurer son bon fonctionnement et sa performance optimale.

- Contexte d'intégration de l'application :

L'API sera hébergée sur une infrastructure cloud, assurant ainsi flexibilité, haute disponibilité et adaptabilité en fonction de la demande. Cette configuration cloud favorise également une harmonisation aisée avec les systèmes et services déjà en place chez Rakuten.

Support d'utilisation de l'application :

Pour une interaction directe, les utilisateurs finaux pourront accéder à l'API via une plateforme web dotée d'une interface utilisateur conviviale. Par ailleurs, pour une intégration plus poussée et une automatisation avancée, les développeurs et les administrateurs bénéficieront d'un accès direct à l’API, permettant une synergie fluide avec d'autres outils et processus au sein de Rakuten.

Solution Proposée :

Pour adresser cette problématique, nous envisageons la mise en place d'une API spécifiquement conçue pour la classification des produits. En exploitant les technologies de pointe et en l'hébergeant sur une infrastructure cloud, nous assurons une solution flexible et hautement disponible. Cette API offrira la capacité de réaliser des prédictions en temps réel en se basant sur des images, du texte descriptif, ou une fusion des deux, soumis par les utilisateurs.

2) **Modèle**

Type de modèle employé et son fonctionnement :

Pour répondre à la nécessité de classer les produits soit par une photo, une description, ou les deux, nous employons une fusion de modèles : BERT pour le traitement du texte et RESNET pour le traitement des images. La combinaison de ces deux approches permet d'exploiter la richesse des informations textuelles et visuelles pour aboutir à des prédictions précises.

BERT (Bidirectional Encoder Representations from Transformers) : C'est un modèle de traitement du langage naturel conçu pour comprendre le contexte des mots dans une phrase. Il a démontré des performances remarquables dans diverses tâches de NLP.

RESNET (Residual Network) : C'est une architecture de réseau neuronal convolutionnel pour la classification d'images. Elle est conçue pour former des réseaux profonds en introduisant des connexions dites "résiduelles". Ces connexions aident à prévenir le problème de disparition de gradient, permettant ainsi d'entraîner de manière efficace des réseaux très profonds.

1. Prétraitement du texte avec BERT :

Afin de pouvoir traiter le texte avec efficacité, nous nettoyons le texte de toutes adresses email, caractères spéciaux, ponctuation et toutes autres caractères qui pourrait être problématique dans la tokenisation des mots.

Nous utilisons le tokenizer de BERT pour convertir chaque description de produit en tokens.

Chaque texte est transformé en une séquence de tokens d'une longueur maximale de 128.

Ces tokens sont ensuite utilisés comme entrée pour le modèle BERT.

2. Prétraitement des images avec ResNet50 :

Le modèle vérifie ensuite que le format de l’image est bien en 224*224, si cela n’est pas le cas, elle est redimensionnée à ce format.

Les images sont transformées sous forme d'arrays numpy.

Ces images sont ensuite traitées avec le modèle ResNet50 pré-entraîné. Seules les caractéristiques (et non les prédictions de classification) sont extraites, car nous utilisons ResNet50 sans ses couches supérieures.

Les caractéristiques extraites sont ensuite aplaties pour être utilisées comme entrées pour notre modèle combiné.

3. Fusion des modèles et définition du modèle combiné :

Les caractéristiques extraites des images et des textes sont ensuite combinées en une seule couche.

Cette couche combinée est ensuite connectée à une couche dense suivie d'une couche de sortie pour la classification.

Le modèle résultant est compilé avec une fonction de perte de crossentropy catégorielle et un optimiseur Adam.

4. Préparation des étiquettes pour l'entraînement :

Les étiquettes de produit sont d'abord encodées en entier.
Elles sont ensuite converties en format one-hot pour correspondre aux 27 classes de produits possibles.

Métriques d'évaluation du modèle :

FUSION (pour le traitement du texte / image) :
-Accuracy (Précision) :
-Robustesse :
-Temps d'entraînement : 2h30 par Epoche / Nombres d’Epoches : 7 *2h30 = 17h30
-Temps de prédiction : _____

3) **Base de données**

Dans le cadre de ce projet, la structure de la base de données sera tripartite, comme détaillé ci-dessous :

BDD Utilisateurs :
Cette base de données aura pour objectif de conserver les détails pertinents des utilisateurs interagissant avec l'API.

BDD Entraînement :
Elle est conçue pour héberger les informations nécessaires à l'entraînement des modèles. Elle se divisera en plusieurs conteneurs représentant les phases d'entraînement et de validation.

BDD Logs :
Cette base aura la responsabilité d'archiver tous les logs résultant des interactions, un élément crucial pour assurer un débogage efficace, la surveillance de l'API et l'optimisation de ses performances.

Données Statiques :
Au sein de ce projet, ces données se réfèrent principalement à un jeu de données initial, composé d'images et de descriptions de produits, destiné à l'entraînement et à la validation du modèle. Ces éléments demeureront constants tout au long du projet.

Dans un Contexte d'Entreprise :
En envisageant une mise en application au sein d'une structure telle que Rakuten, il est clair que la base de données serait soumise à une dynamique plus intense. Elle recevrait constamment de nouvelles informations en raison de l'introduction de nouveaux produits, des actualisations de descriptions, ou encore des feedbacks clients. Par conséquent, une gestion rigoureuse des versions des données serait impérative pour assurer une traçabilité optimale.

Gestion de l'Ingestion des Nouvelles Données :
Il est capital de garder le modèle à jour et en phase avec les tendances actuelles. Pour ce faire, un processus d'ingestion des nouvelles données, régulier et bien huilé, est de mise. Cette procédure pourrait être orchestrée par des scripts automatiques chargés d'examiner et de valider la qualité des nouvelles données. Un mécanisme d'alertes pourra aussi être instauré pour informer l'équipe technique en cas d'anomalies ou d'incohérences.

Afin d'assurer la pertinence continue du modèle face aux nouvelles données, un pipeline d'entraînement MLOps fiable et automatisé serait primordial. Il permettrait d'actualiser le modèle de manière périodique avec les données les plus récentes et de mettre en ligne sans accroc les mises à jour de l'API.

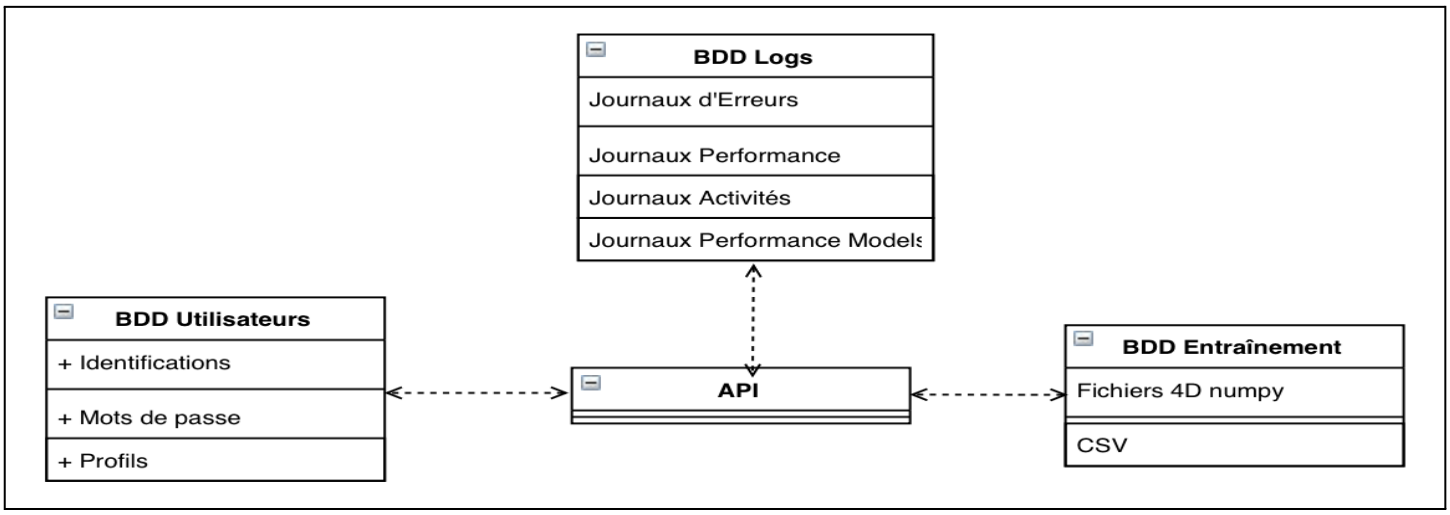
Recommandations Supplémentaires :

Sécurité : La protection de nos bases de données est au cœur de nos préoccupations. La mise en œuvre de dispositifs solides est essentielle pour assurer l'intégrité, la confidentialité des données et se prémunir de toute tentative d'intrusion.

Nos données sont structurées dans deux conteneurs spécifiques. La base de données "Utilisateurs", en respectant rigoureusement les directives du RGPD, est dotée d'une sécurité renforcée par rapport à nos autres bases.

Optimisation : Afin d'assurer une expérience utilisateur sans heurt, il est recommandé d'optimiser la base de données, notamment pour les requêtes les plus fréquentes.

Schéma d'Architecture :



4) API

Endpoints de l'API :

1. `/auth/login` : Pour l'authentification des utilisateurs/administrateurs.
2. `/data/query` : Pour interroger la base de données.
3. `/model/fusion/predict` : Pour obtenir des prédictions du modèle pour le traitement des combinaisons textes/images
4. `/logs/write` : Pour écrire dans les logs.
5. `/data/update` : Pour mettre à jour la base de données.
6. `/model/update` : Pour mettre à jour le modèle si nécessaire.

Concernant les updates de la base de données, nous souhaitons la mise en place régulière de réentrainement du modèle avec bases de données qui varie et enrichit de nouvelles données qui ont été validées et certifiées par un administrateur.

En ce qui concerne la mise à jour du modèle, nous pouvons envisager d'améliorer ses performances en augmentant sa rapidité de réponse à l'utilisateur ou en optimisant les processus de réentrainement.

5) Testing & Monitoring

Pour assurer la robustesse, la fiabilité et la performance de notre solution, nous avons défini une stratégie rigoureuse d'essais et de surveillance.

Essais :
Durant l'entraînement du modèle :
Suivi de la convergence du modèle pour garantir sa stabilité.
Monitoring de la perte pour s'assurer d'une diminution régulière et éviter le surapprentissage.

Durant la phase de prédiction :
Validation des performances du modèle sur un ensemble de données test indépendant pour garantir son efficacité en situations réelles.

Endpoints de l'API :
Mise en place de tests automatisés pour chaque endpoint, garantissant ainsi la cohérence des retours et une gestion adéquate des erreurs.

Ingestion de nouvelles données :
Vérifications régulières de la qualité et de l'intégrité des données nouvellement intégrées.

Surveillance :

Performance du modèle :
Évaluations régulières avec un ensemble de données test pour s'assurer que le modèle reste performant face aux évolutions des données.

Analyses basées sur des données récentes pour confirmer la pertinence continue du modèle face aux tendances actuelles.

Critères de ré-entraînement :
Revues à des intervalles définis, par exemple mensuellement ou trimestriellement.
Mécanismes basés sur des seuils de performance pour déclencher le ré-entraînement.

Mécanismes basés sur des seuils de performance pour déclencher un ré-entraînement automatique. Si une dégradation est notée, les nouvelles données stockées dans la BDD "Logs" et le CSV sont utilisées pour ce réentraînement, garantissant ainsi la prise en compte des tendances récentes.

Données utilisées pour le ré-entraînement :
Option de ré-entraînement avec la totalité du dataset ou seulement avec un échantillon représentatif des données récentes, selon les besoins.

Gestion des performances suboptimales :
Instauration d'alertes automatiques à destination des équipes en cas de dégradation notable des performances.

Protocoles d'urgence en cas de défaillances majeures, incluant la possibilité de mettre l'application en mode maintenance.

Les temps de réaction face à ces alertes ont été optimisés pour garantir une intervention rapide et efficace.

Seuil pour le réentraînement :
Le seuil d'exactitude a été défini à 70%. Si les performances du modèle descendent en dessous de cette valeur lors des évaluations périodiques, cela initiera le processus de ré-entraînement.

6) Schéma d'implémentation