

Cahier des charges projet MLOps

**1) Contexte et Objectifs**

À l'heure de la transformation numérique, la classification des produits est devenue primordiale pour les entreprises. Face à ce défi, elles cherchent des solutions automatisées et évolutives pour gérer la complexité croissante des données et des demandes de prédiction. En utilisant des technologies avancées telles que Docker et Kubernetes, notre projet propose de développer une infrastructure robuste spécifiquement dédiée à la classification de produits, répondant ainsi aux exigences contemporaines.

Problématique à laquelle l'application doit répondre :

Face à une demande croissante d'outils capables de catégoriser des éléments soit par leur aspect visuel, soit par une description textuelle, ou les deux combinés, il est essentiel de disposer d'une **API** qui répond à cette nécessité. Cette API doit permettre d'effectuer des prédictions en temps réel basées sur des images, des descriptions textuelles ou une combinaison des deux fournies par les utilisateurs. L'objectif est de garantir des prédictions rapides, précises et évolutives. De plus, cette solution doit être facile à gérer, à déployer et adaptable aux besoins changeants des utilisateurs et de l'environnement technologique.

**\*API (Application Programming Interface)**

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.

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 API, permettant une synergie fluide avec d'autres outils et processus au sein de Rakuten.

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.

Métriques d'évaluation du modèle :

BERT (pour le traitement du texte) :

-Accuracy (Précision) : 0.8311

-Robustesse : Grâce à son architecture basée sur les transformers, BERT possède une capacité intrinsèque à gérer les variations contextuelles, rendant le modèle robuste face aux variations subtiles dans les descriptions de produits.

-Temps d'entraînement : 25 minutes par Epoche / Nombres d’Epoches : 5 / 2h05

-Temps de prédiction : \_\_\_\_\_\_\_\_

RESNET (pour le traitement des images) :

-Accuracy (Précision) : 0.5711

-Robustesse : L'architecture résiduelle de RESNET le rend particulièrement robuste face aux variations subtiles des images, grâce à ses connexions résiduelles qui permettent une meilleure propagation de l'information à travers le réseau.

-Temps d'entraînement : 30 minutes par Epoche / Nombres d’Epoches : 5 / 2h30

- 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 sécurité des bases de données doit être une priorité. Il est indispensable de mettre en place des mécanismes robustes pour garantir l'intégrité et la confidentialité des données et prévenir toute intrusion.

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 :

[Insérer le schéma ici]

**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/bert/predict` : Pour obtenir des prédictions du modèle pour le traitement de texte

4. `/model/resnet/predict` : Pour obtenir des prédictions du modèle pour le traitement des images

5. `/model/fusion/predict` : Pour obtenir des prédictions du modèle pour le traitement des combinaisons textes/images

6. `/logs/write` : Pour écrire dans les logs.

7. `/data/update` : Pour mettre à jour la base de données.

8. `/model/update` : Pour mettre à jour le modèle si nécessaire.

**5) Testing & Monitoring**

Pour garantir la fiabilité et la performance de la solution proposée, une stratégie de tests et de monitoring sera mise en place. Ci-dessous, vous trouverez les spécifications relatives à cette stratégie :

Tests Unitaires:

Modèle lors de l'entraînement :

Vérification de la convergence du modèle pendant l'entraînement.

Surveillance de la progression de la perte pour assurer une réduction continue.

Modèle lors de la prédiction :

Validation des prédictions sur un jeu de données test indépendant.

Endpoints de l'API :

Tests automatisés pour chaque endpoint pour garantir la fiabilité des retours et la gestion des erreurs.

Processus d'ingestion de nouvelles données :

Contrôles d'intégrité et de qualité des données à chaque nouvelle ingestion.

Monitoring :

Évaluation de la performance du modèle :

Utilisation d'un ensemble de données de test pour des évaluations périodiques.

Analyses sur des données récentes pour s'assurer de la pertinence continue du modèle.

Critères de ré-entraînement :

Révisions périodiques (e.g., mensuelles, trimestrielles).

Déclencheurs basés sur des seuils de performance.

Sélection des données pour le ré-entraînement :

Option de ré-entraînement sur l'intégralité du jeu de données.

Possibilité d'utiliser un échantillon des données les plus récentes.

Procédures en cas de performances insatisfaisantes :

Mise en place d'alertes automatisées pour les équipes concernées.

Procédures de contingence en cas de performances gravement compromises, incluant des directives pour la mise hors ligne temporaire de l'application.

6) Schéma d’implémentation

