DataScientest • com

Rendu Projet Append Engineer

Mourad MECHERI

Promotion mai 2022

Déploiement du projet :

Rakuten France Multimodal Product Data Classification

Réalisé par : Mourad MECHERI

Supervisé par : Anthony JAILLET

Sommaire

1 INTRODUCTION ET ÉTAPES DU PROJET

Ce projet porte sur le déploiement d'un projet de Machine Learning dans le cadre de ma formation de ML Engineer au sein de l'organisme de formation DataScientest.

L'objectif étant de déployer en API et de conteneuriser un modèle de prédiction en Machine ou Deep Learning.

Dans ce projet , je reprends mon projet fil rouge : ***Rakuten France Multimodal Product Data Classification*** réalisé au cours de ma précédente formation de Data Scientist au sein du même organisme de formation.

Contexte et solution retenue lors du projet de formation Data Scientist

Dans un contexte de classification des produits « e-commerce », l'objectif était de prédire le code type (**prdtypecode**) de chaque produit en utilisant des données **textuelles** (désignation et description du produit) ainsi que des données **images** (image du produit) tel qu'il est défini dans le catalogue de Rakuten France.

À l'issu du projet , nous avons proposé une solution qui permet de réaliser des prédictions avec un modèle basé sur les données Texte ou Images ou les deux combines(Bimodal).

Voici les modèles et les combinations que nous avons retenus:

- Une classification **basée** sur le **Texte**: Conv1D et Simple DNN
- Une classification **basée** sur les **Images**: Xception et InceptionV3
- Une classification **Bimodal** - **Texte et Images**:
    - Conv1D, Simple DNN et Xception
    - Conv1D, Simple DNN et InceptionV3

Étapes du projet

- Reprendre les modèles de classification de produits e-commerce Rakuten France et les déployer sur une API: créer des Endpoints pour réaliser des prédictions
- Créer une base de données en Backend pour l'API avec la gestion et l'authentification des utilisateurs
- Conteneuriser avec Docker et déployer sur GitHub
- Réaliser des tests d'Authentification, d'Autorisation et de prédictions via des containers distincts

2 L'API - RAKUTEN FRANCE MULTIMODAL PRODUCT DATA CLASSIFICATION

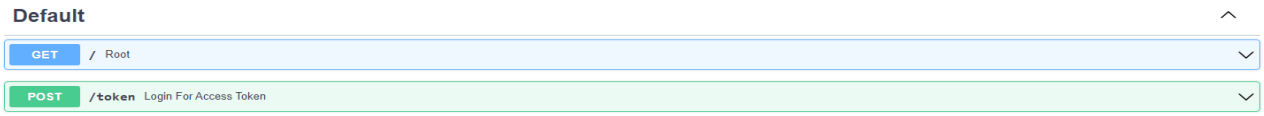
L'API a été développée en utilisant le Framework FastAPI. Elle permet de gérer et d'authentifier les utilisateurs. Cela est réalisé avec le protocole OAuth2 (avec un username et un mot de passe hashé), les JWT tokens et une base de donnée MongoDB.

Note : La durée de validité pour les access token est fixée à 30 minutes

L'API propose 17 Endpoints, qui sont répartis comme suit:

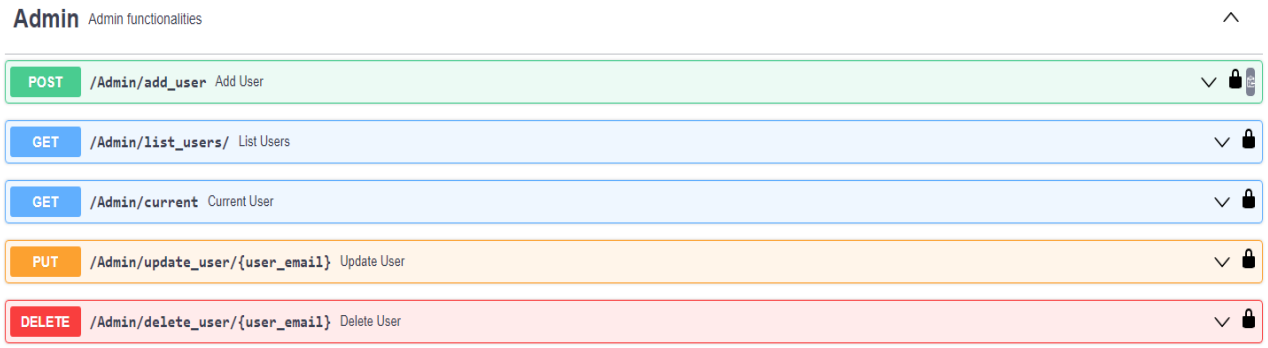
2.1 DEFAULT

Endpoint	Description
/Root	Permet de vérifier que l'API est opérationnelle
/token	Permet à l'utilisateur d'obtenir un access token à l'API en utilisant un username(adresse email) et un password



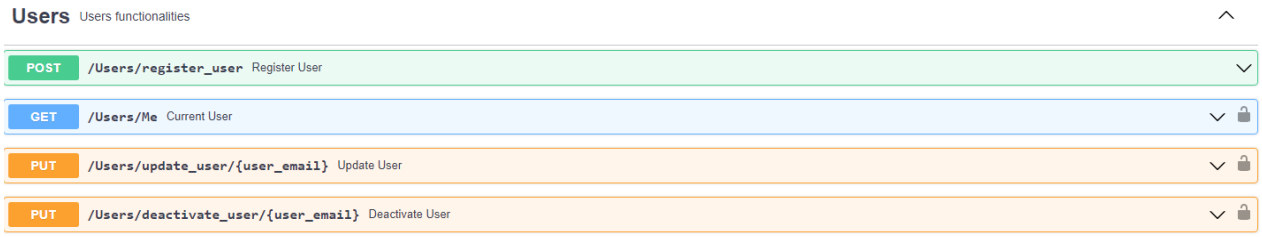
2.2 ADMIN FUNCTIONALITIES

Endpoint	Description
/Admin/add_user	Permet l'admin de créer un utilisateur et de lui assigner un rôle (admin , dev ou user)
/Admin/list_users	Permet d'obtenir la liste des utilisateurs enregistrés dans la base de données. L'utilisateur(Admin) peut choisir le nombre et le statut(actif ou non actif) des utilisateurs à afficher. Si les choix ne sont pas spécifiés , le Endpoint affiche 100 utilisateurs au maximum avec statut actif ou non actif
/Admin/current	Renvoie l'utilisateur actuellement authentifié
/Admin/update_user/{user_email}	Permet de mettre à jour les informations d'un utilisateur en utilisant son adresse email
/Admin/delete_user/{user_email}	Permet de supprimer un utilisateur en utilisant son adresse email



2.3 USERS FUNCTIONALITIES

Endpoint	Description
/Users/register_user	Permet à un utilisateur de s'enregistrer. Un rôle « user » lui est assigné automatiquement
/Users /Me	Renvoie l'utilisateur actuellement authentifié
/Users /update_user/{user_email}	Permet à l'utilisateur de mettre à jour ses informations
/Users/ deactivate_user/{user_email}	Permet à l'utilisateur de désactiver son compte. L'utilisateur est réactivé la prochaine fois qu'il se reconnecte



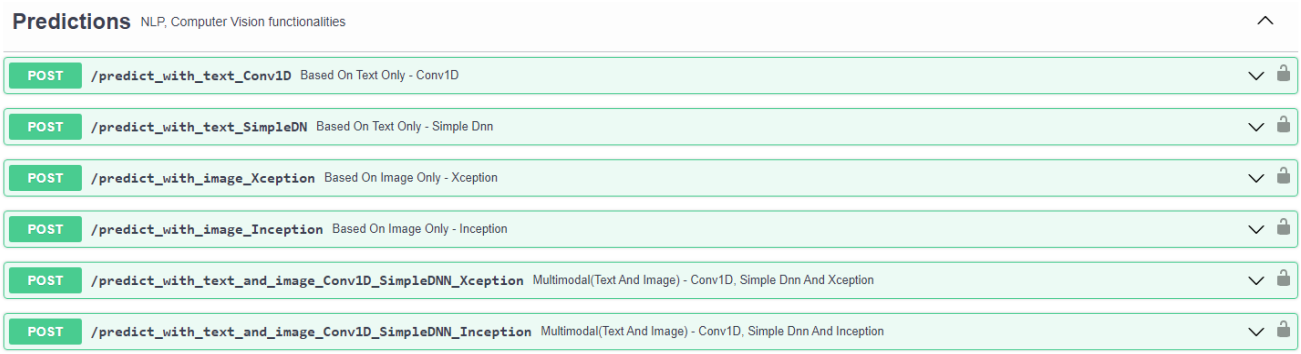
2.4 PREDICTIONS (NLP, COMPUTER VISION FUNCTIONALITIES)

Comprend les Endpoints permettant de réaliser des prédictions. Celles-ci peuvent être effectuées avec un modèle basé sur les données Texte ou Images ou les deux combinés.

Modèles de prédictions basés sur des données Texte	Description
/predict_with_text_Conv1D	Permet de réaliser une prédiction avec le modèle Conv1D en se basant sur des données texte uniquement. **Paramètres :** • **designation** (obligatoire): le titre du produit Rakuten: un texte court résumant le produit • **description**(optionnel): un texte plus détaillé décrivant le produit
/predict_with_text_SimpleDN	Permet de réaliser une prédiction avec le modèle Simple DNN en se basant sur des données texte uniquement. **Paramètres :** • **designation** (obligatoire): le titre du produit Rakuten: un texte court résumant le produit • **description**(optionnel): un texte plus détaillé décrivant le produit

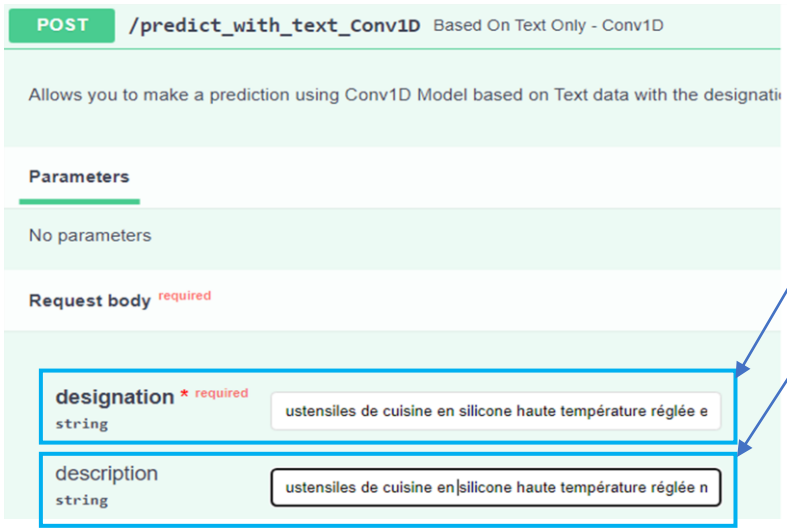
Modèles de prédictions basés sur des données Image	Description
/predict_with_image_Xception	Permet de réaliser une prédiction avec le modèle Xception en se basant sur une donnée image uniquement. **Paramètres** : Un fichier **Image**(obligatoire) du produit Rakuten
/predict_with_image_Inception	Permet de réaliser une prédiction avec le modèle Inception en se basant sur une donnée image uniquement. **Paramètres** : Un fichier **Image (obligatoire)** du produit Rakuten
Modèles de prédictions basés sur des données Texte et Image	Description
/predict_with_text_and_image_Conv1D_SimpleDNN_Xception	Permet de réaliser une prédiction en combinant les deux modèles Texte Conv1D et Simple DNN et le modèle Image Xception. Une prédiction finale est renvoyée ainsi que la prédiction effectuée par chaque modèle séparément. **Paramètres** : • **designation** (obligatoire): le titre du produit Rakuten: un texte court résumant le produit • **description**(optionnel): un texte plus détaillé décrivant le produit • Un fichier **image(**obligatoire**)** du produit Rakuten
/predict_with_text_and_image_Conv1D_SimpleDNN_Inception	Permet de réaliser une prédiction en combinant les deux modèles Texte Conv1D et Simple DNN et le modèle Image Inception. Une prédiction finale est renvoyée ainsi que la prédiction effectuée par chaque modèle séparément. **Paramètres** : • **designation** (obligatoire): le titre du produit Rakuten: un texte court résumant le produit

	• **description**(optionnel): un texte plus détaillé décrivant le produit • Un fichier **Image(**obligatoire**)** du produit Rakuten



2.4.1 Exemples de prédictions

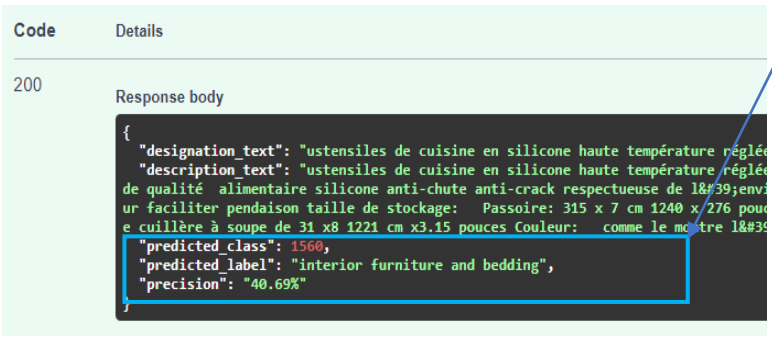
2.4.1.1 /predict_with_text_Conv1D - Modèle basé sur des données Texte



Champ **designation** (obligatoire): Indiquant le titre du produit Rakuten: un texte court résumant le produit

Champ **description** (optionnel): un texte plus détaillé décrivant le produit

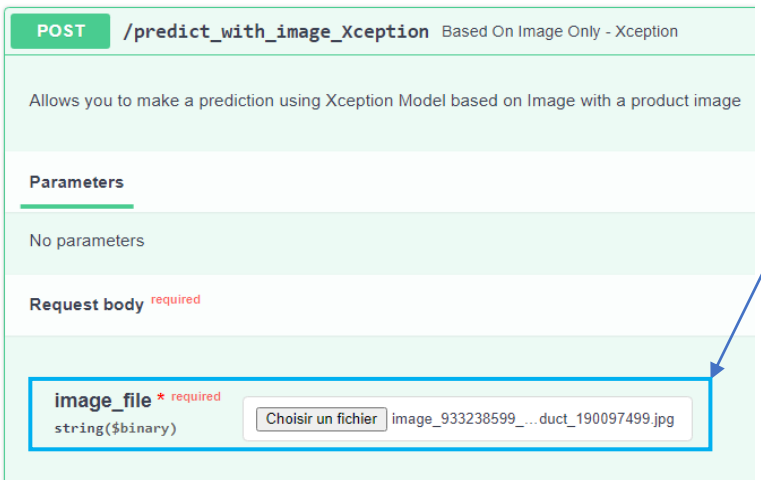
Réponse:



La réponse renvoyée par le Endpoint correspondant à notre modèle de prédiction Conv1D comprend:

- la classe du produit,
- le label (catégorie) du produit,
- et la précision de la prédiction

2.4.1.2 /predict_with_image_Xception - Modèle basé sur des données Image



Fichier **Image** du produit Rakuten

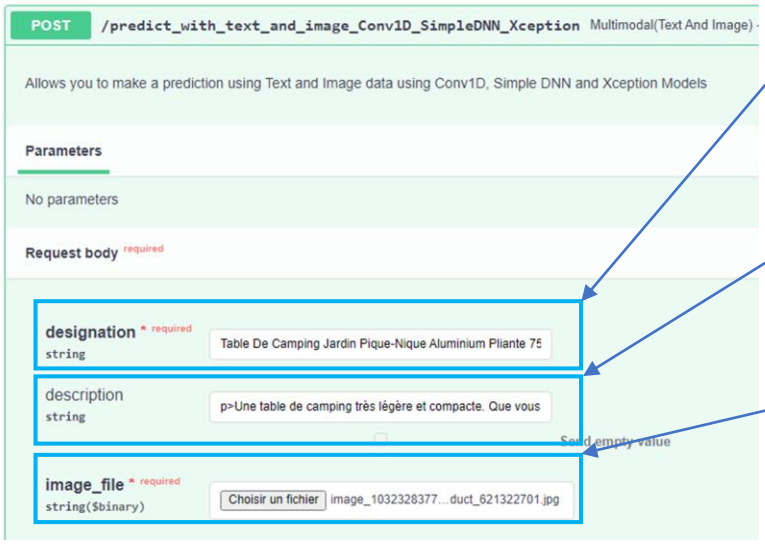
image_933238599_product_190097499.jpg

Réponse:

La réponse renvoyée par le Endpoint correspondant à notre modèle de prédiction Xception comprend :

- la classe du produit,
- le label (catégorie) du produit,
- et la précision de la prédiction

2.4.1.3 /predict_with_text_and_image_Conv1D_SimpleDNN_Xception Multimodal(Text And Image)



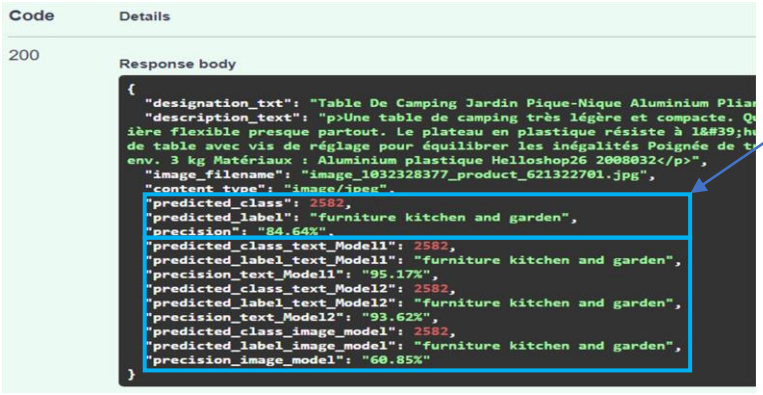
Champ **designation** (obligatoire): Indiquant le titre du produit Rakuten: un texte court résumant le produit

Champ **description** (optionnel): un texte plus détaillé décrivant le produit

Fichier **Image** du produit Rakuten

image_1032328377_product_621322701.jpg

Réponse:



La réponse renvoyée comprend la prédiction finale ainsi que la prédiction effectuée par chaque modèle séparément

2.5 LISTE DES UTILISATEURS PRÉENREGISTRÉS DANS LA BASE DE DONNÉES

Utilisateur	Mot de passe	Rôle
admin_account1@example.com	adminsecret1	Admin
alicewonderson@example.com	secret1	User
johndoe@example.com	secret2	User
clementinemandarine@example.com	secret3	User

3 CONTENEURISATION DES DIFFÉRENTS SERVICES

3.1 CONTENEURISATION DE L'API

Une image Docker de l'API a été construite et téléversée dans DockerHub sous le répertoire :

mmecheri/mle_project_api:1.0.0

(https://hub.docker.com/repository/docker/mmecheri/mle_project_api)

3.2 CONTENEURISATION DE LA BASE DE DONNÉE

Un service de base de donnée MongoDB isolé dans un conteneur est utilisé pour la gestion et l'authentification des utilisateurs.

3.3 TESTS

3.3.1 Authentifications

Dans ce service, des tests de vérification d'identification ont été mis en place. Pour cela, des requêtes de type POST sur le Endpoint /token avec des utilisateurs préalablement enregistrés dans la base de donnée ont été effectuées .

- Un premier test avec un compte admin (username: "admin_account1@example.com") avec un mot de passe correct ('adminsecret1') afin de vérifier que la requête renvoi bien un code d'erreur 200
- Un deuxième test avec un compte user (username: " alicewonderson@example.com") avec un mot de passe correct ('secret1') afin de vérifier que la requête renvoi bien un code d'erreur 200
- Un troisième tests avec un compte user (username :"johndoe@example.com") et un mot de passé erroné afin de vérifier que la requête renvoi bien un code d'erreur 403

3.3.2 Autorisations

Dans ce deuxième service, des tests de gestion des droits d'utilisateurs sont vérifiés.

- Un premier test avec un compte admin (username: "admin_account1@example.com") afin de vérifier les droits de suppression d'utilisateurs accordés aux comptes admin.
    - Un code d'erreur 204 est renvoyé lorsque la suppression est bien effectuée

- Un code d'erreur 404 est renvoyé lorsque l'utilisateur que l'on souhaite supprimer n'existe pas dans la base de donnée

- Un deuxième test avec un compte user (username: " alicewonderson@example.com") avec un mot de passe correct ('secret1') afin de vérifier que la requête renvoi bien un code d'erreur 403

3.3.3 Prédictions

Dans ce dernier test, je vérifie que l'API fonctionne comme elle doit fonctionner. Un test est effectué pour chaque Endpoint/modèle de prédiction:

- Test #1 : en utilisant le modèle basé sur des données texte uniquement : **Conv1D** avec un champ texte *designation*(obligatoire) et un champ *description*(optionnel)

- Test #2 : Un test en utilisant le modèle basé sur des données texte uniquement : **Simple DNN** avec un champ texte *designation* uniquement

- Test #3 : Un test en utilisant le modèle basé sur des données image uniquement : **Xception** en utilisant une image d'un produit

- Test #4 : Un test en utilisant le modèle basé sur des données image uniquement : **Inception** en utilisant une image d'un produit

- Test #5 : Un test en combinant les deux modèles Texte **Conv1D** et **Simple DNN** et le modèle Image **Xception**

- Test # 6 : Un test en combinant les deux modèles Texte **Conv1D** et **Simple DNN** et le modèle Image **Inception**

Les tests ont été effectués en utilisant le compte user : username=johndoe@example.com, password="secret2".

Pour chacun des modèles implémentés , un résultat de prédictions est renvoyé contenant: la classe du produit , le label (catégorie) du produit et la précision de la prédiction . Le statut du test est qualifié en « SUCCESS » si la classe de produit prédite correspond à celle attendue.

3.4 DOCKER COMPOSE

La composition des différents services est réalisée via fichier « *docker-compose.yml*» permettant de :

- De déclarer le container l'API FastAPI , en spécifiant l'image préalablement téléversée dans DockerHub
- De déclarer un service de base de donnée , en spécifiant une image MongoDB
- Déclarer 3 container de tests construits séparément à partir de fichiers Dockerfile
    - Un container pour les tests d'authentification
    - Un container pour les tests d'autorisation
    - Un container permettant de réaliser des prédictions pour chacun des modèles de l'API

    (À la fin de l'exécution des différents tests, un fichier api_tests.log contenant les résultats de ces tests est généré)

- Un fichier appelé « *setup.sh »* contenant les commandes utilisées pour construire les images de test et lancer le docker-compose est créé

4 RÉPERTOIRE GITHUB

Le lien vers le repository de GitHub est:
https://github.com/mmecheri/Append_Engineer_Project

Le contenu du répertoire:

	Description
app	Dossier contenant les fichier Python de l'API(Fast API)
authentication_image	Dossier contenant le Dockerfile utilisé pour construire l'image authentication_image et le code Python utilisé pour réaliser les tests
authorization_image	Dossier contenant le Dockerfile utilisé pour construire l'image authorization_image et le code Python utilisé pour réaliser les tests
prediction_image	Dossier contenant le Dockerfile utilisé pour construire l'image prediction_image et le code Python utilisé pour réaliser les tests de prédictions
README.md	Comprend les instructions d'installation
Rendu_Projet_Append_Engineer.pdf	Contient le présent document
api_tests.log	Contient les résultats des différents tests
docker-compose.yml	Le docker-compose qui contient les services construits
setup.sh	contient les commandes utilisées pour construire les images et lancer le docker-compose