

Classification de texte avec pipeline CI/CD complet

Projet DevOps & MLOps

Akram Benhammou Oussama Khouya

ENSET / Université Hassan II

14 décembre 2025

Plan

- 1 Contexte et objectifs
- 2 Dataset et prétraitement
- 3 Modèle et suivi expérimental
- 4 API, Docker et CI/CD
- 5 Limites et perspectives
- 6 Conclusion

Contexte du projet

- Problème de classification de texte sur le jeu de données *20 Newsgroups*.
- Mise en place d'une chaîne MLOps de bout en bout.
- Intégration des bonnes pratiques DevOps : CI/CD, tests, containerisation.

Objectifs

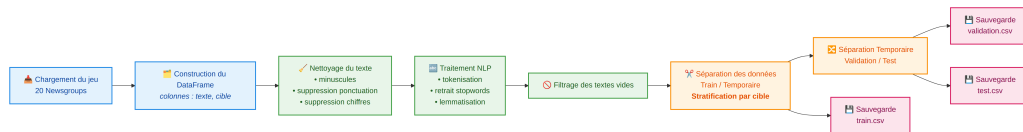
- Automatiser le cycle de vie du modèle (données \rightarrow modèle \rightarrow déploiement).
- Garantir la reproductibilité des expériences.
- Faciliter le déploiement et la maintenance du service de prédiction.

Dataset 20 Newsgroups

- 20 classes de newsgroups (informatique, sport, politique, religion, ...).
- Chargement via `sklearn.datasets.fetch_20newsgroups`.
- Séparation en train / validation / test (70% / 15% / 15%).

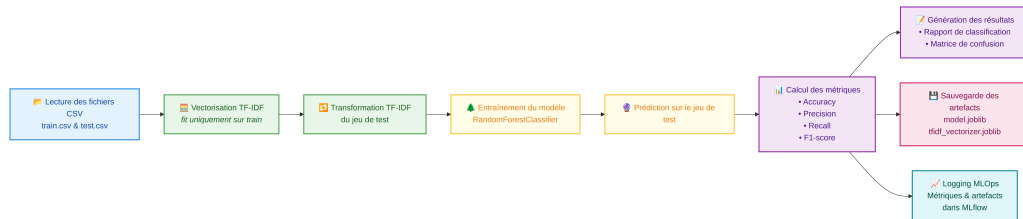
Pipeline de données

- Nettoyage simple du texte brut (minuscules, ponctuation, chiffres).
- Tokenisation, suppression des stopwords anglais, lemmatisation (NLTK).
- Vectorisation TF-IDF (5000 features max).



Modèle de classification

- RandomForestClassifier sur les vecteurs TF-IDF.
- Entraînement sur le jeu d'apprentissage, évaluation sur le test.
- Métriques : accuracy, précision, rappel, F1 pondérée, matrice de confusion.



Suivi avec MLflow

- Backend local `mlruns/`.
- Logging des hyperparamètres, métriques et artefacts (modèle, vectoriseur).
- Possibilité de comparer plusieurs expériences.

Service d'inférence FastAPI

- Endpoint `/health` pour la supervision.
- Endpoint `/predict` pour la prédiction sur un texte.
- Chargement des artefacts au démarrage de l'application.

The screenshot shows a REST client interface with a POST request to `http://localhost:8000/predict`. The request body is a JSON object containing a text snippet about basketball. The response is a JSON object with prediction details.

POST `http://localhost:8000/predict` Send

Query Headers ² Auth Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 { "text": "Basketball is more than just a sport; it is a global language  
  that brings people together across cultures, ages, and backgrounds.  
  Invented in 1891 by Dr. James Naismith, basketball was originally
```

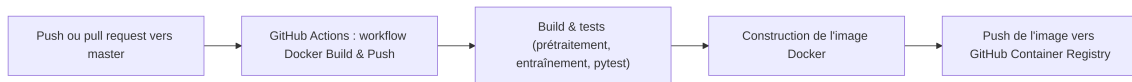
Status: 200 OK Size: 537 Bytes Time: 9.02 s

Response Headers ⁵ Cookies Results Docs {} ≡

```
2  "text": "Basketball is more than just a sport; it is a global language  
  that brings people together across cultures, ages, and backgrounds.  
  Invented in 1891 by Dr. James Naismith, basketball was originally  
  desi...",  
3  "prediction_class_id": 10,  
4  "category_name": "Sport",  
5  "confidence_scores": [  
6    {  
7      "name": "Sport",  
8      "value": 0.71  
9    },
```

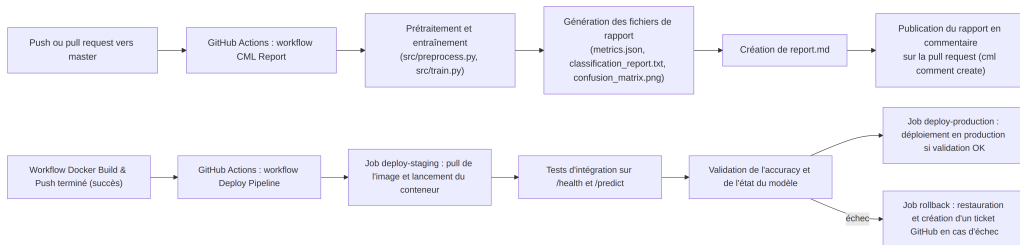
Containerisation Docker

- Image basée sur `python:3.9-slim`.
- Installation des dépendances et des ressources NLTK.
- Exposition du service via Uvicorn sur le port 8000.



CI/CD GitHub Actions

- Workflow de build et tests : prétraitement, entraînement, tests, build/push Docker.
- Workflow CML : génération automatique d'un rapport de performance.
- Workflow de déploiement : staging → production avec rollback.



Limites

- Modèle classique (Random Forest + TF-IDF) sans transformers.
- MLflow en mode local, sans model registry centralisé.
- Monitoring de la dérive de données et des performances en production non implémenté.

Pistes d'amélioration

- Expérimenter des modèles basés sur des embeddings ou des transformers.
- Mettre en place un registry de modèles et un tracking centralisé.
- Ajouter du monitoring, des alertes et des tests de robustesse avancés.

Conclusion

- Mise en œuvre d'un pipeline MLOps complet autour d'un cas réel de NLP.
- Automatisation du cycle de vie du modèle avec GitHub Actions, Docker et MLflow.
- Base solide pour des évolutions vers des architectures et modèles plus avancés.

Questions ?

Merci pour votre attention !