UNE DÉMARCHE MLOPS POUR L'ANALYSE DE SENTIMENT DES TWEETS

Dans un monde où les réseaux sociaux jouent un rôle essentiel dans l'image des entreprises, anticiper les bad buzz est devenu une nécessité. Ces bad buzz peuvent rapidement affecter la réputation et la crédibilité d'une marque. L'analyse de sentiment permet de détecter rapidement les messages négatifs en ligne, offrant ainsi aux entreprises la possibilité de réagir avant que les problèmes ne s'amplifient.
Le projet présenté dans cet article explore trois approches majeures de modélisation : Machine Learning (ML), Deep Learning (DL) et BERT. L'objectif est d'identifier la solution la plus efficace pour analyser les sentiments des tweets. Parallèlement, une démarche rigoureuse orientée MLOps a été mise en place pour assurer la scalabilité, la traçabilité et le suivi des performances des modèles en production.

Analyse Exploratoire des Données (EDA)

Présentation du Dataset

Le dataset utilisé, est un jeu de données populaire pour les projets d'analyse de sentiment. Il contient 1,6 million de tweets étiquetés avec deux classes :

- POSITIF (représenté par la valeur 4).
- NÉGATIF (représenté par la valeur 0).

Ce dataset présente les caractéristiques suivantes :

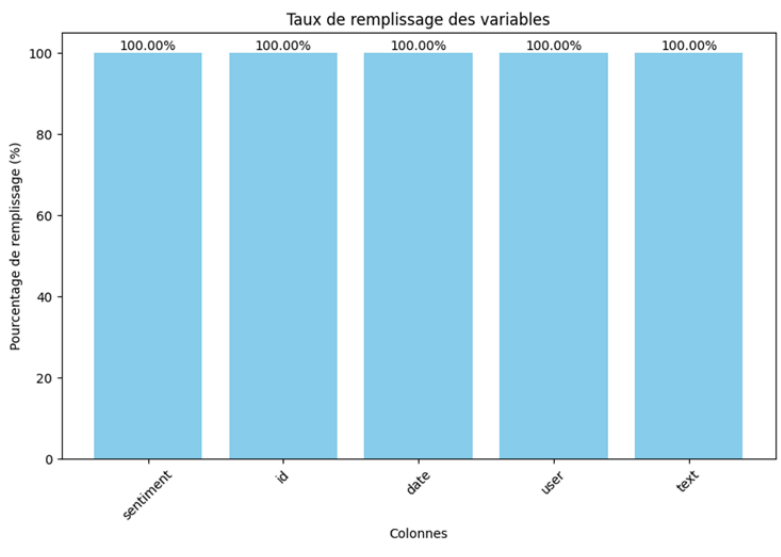
- Les tweets ont été collectés via Twitter, capturant un éventail de sentiments réels exprimés par les utilisateurs.

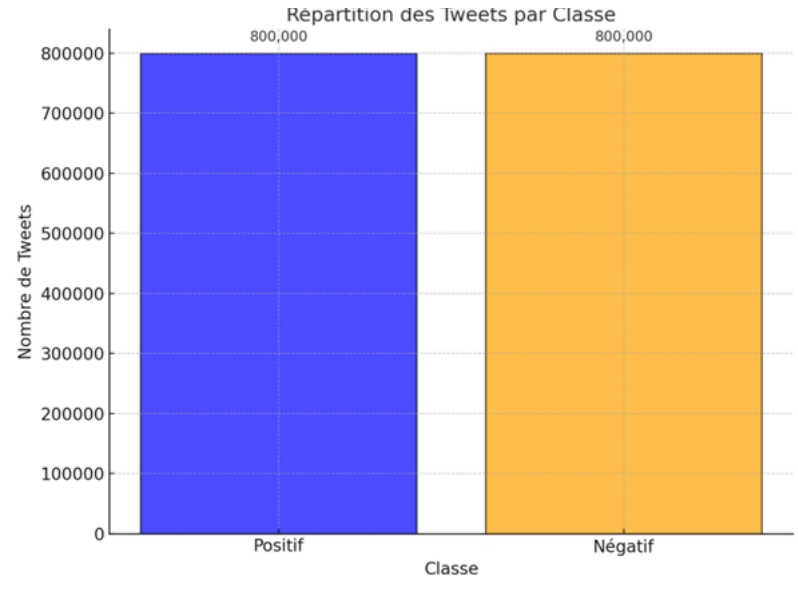
Les données incluent des emojis, des hashtags, et des mentions, rendant l'analyse plus proche d'un contexte d'utilisation réel.

Analyse Descriptive

Une exploration approfondie du dataset a permis de mieux comprendre les données :

- Longueur des tweets : En moyenne, les tweets négatifs sont légèrement plus longs que les positifs, reflétant potentiellement une tendance à exprimer davantage de détails ou de griefs dans les messages négatifs.
- Distribution des classes : Les classes positives et négatives sont équilibrées, ce qui facilite l'entraînement des modèles sans nécessiter de techniques de rééquilibrage.
- Nettoyage initial : Les données brutes contiennent des éléments spécifiques à Twitter (ex. : mentions @, URL). Ces éléments ont été nettoyés pour éviter les biais dans la modélisation.





Protocole d'Évaluation

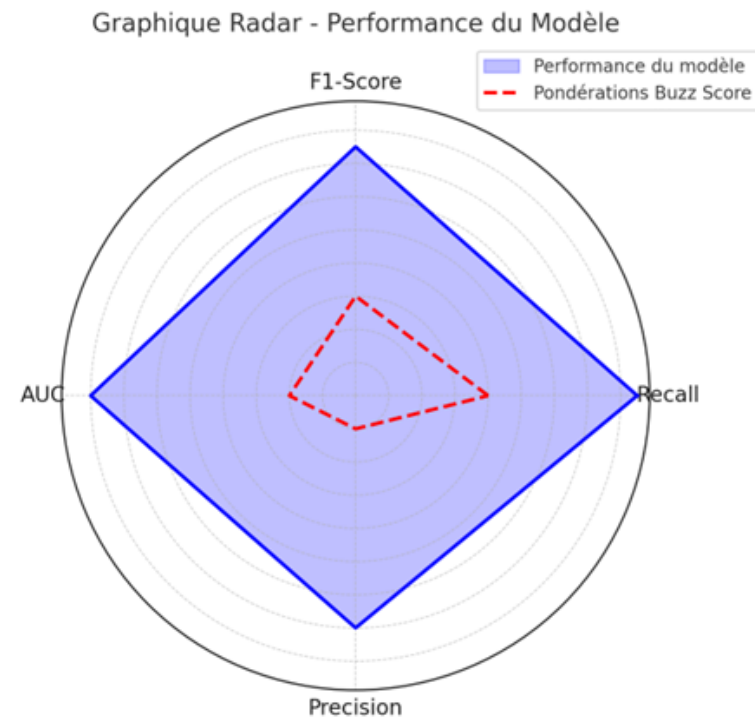
Métriques Utilisées

Pour comparer les performances des modèles, plusieurs métriques standard ont été utilisées :

- Accuracy : Proportion de prédictions correctes parmi l'ensemble des prédictions.
- Precision : Mesure de la qualité des prédictions positives, indiquant combien parmi elles sont effectivement correctes.
- Recall : Taux de récupération des véritables positives. Indique la proportion de cas positifs correctement identifiés.
- F1-Score : Moyenne harmonique entre Precision et Recall, utile lorsque les classes ne sont pas parfaitement équilibrées.
- ROC AUC : Aire sous la courbe ROC, mesurant la capacité du modèle à distinguer les classes positives et négatives.
- Buzz Score : Une métrique personnalisée qui accorde une importance particulière aux faux négatifs, car ignorer un message négatif critique pourrait avoir de graves conséquences. calcul :

$$0.4 \times \text{Recall} + 0.3 \times \text{F1-Score} + 0.2 \times \text{AUC} + 0.1 \times \text{Precision}$$

Ces métriques ont permis d'évaluer les modèles sous plusieurs angles, garantissant une compréhension approfondie de leurs forces et limites.

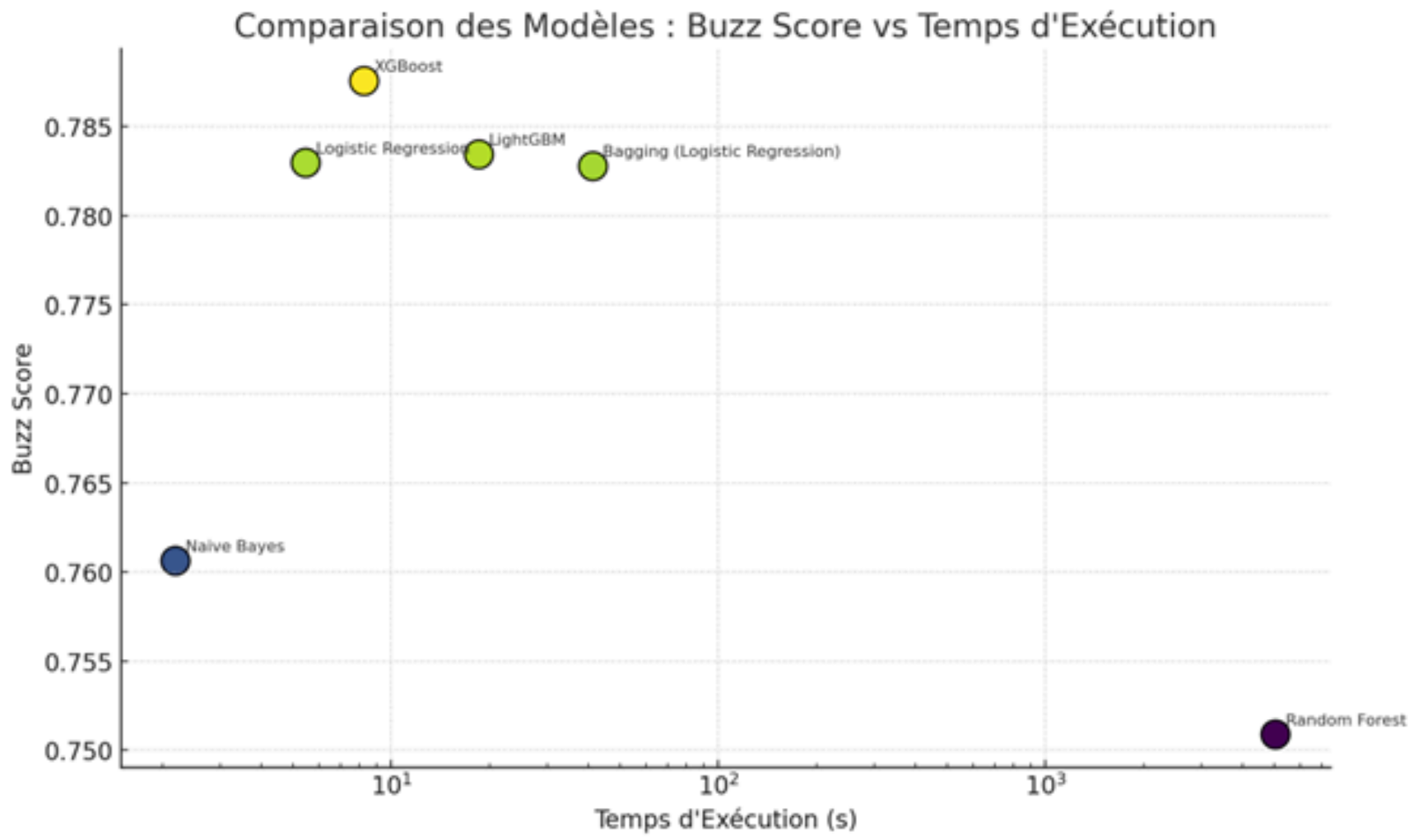


Étude et Comparaison des Modèles

Modèles de Machine Learning (ML)

Différents algorithmes de ML ont été testés pour évaluer leurs performances :

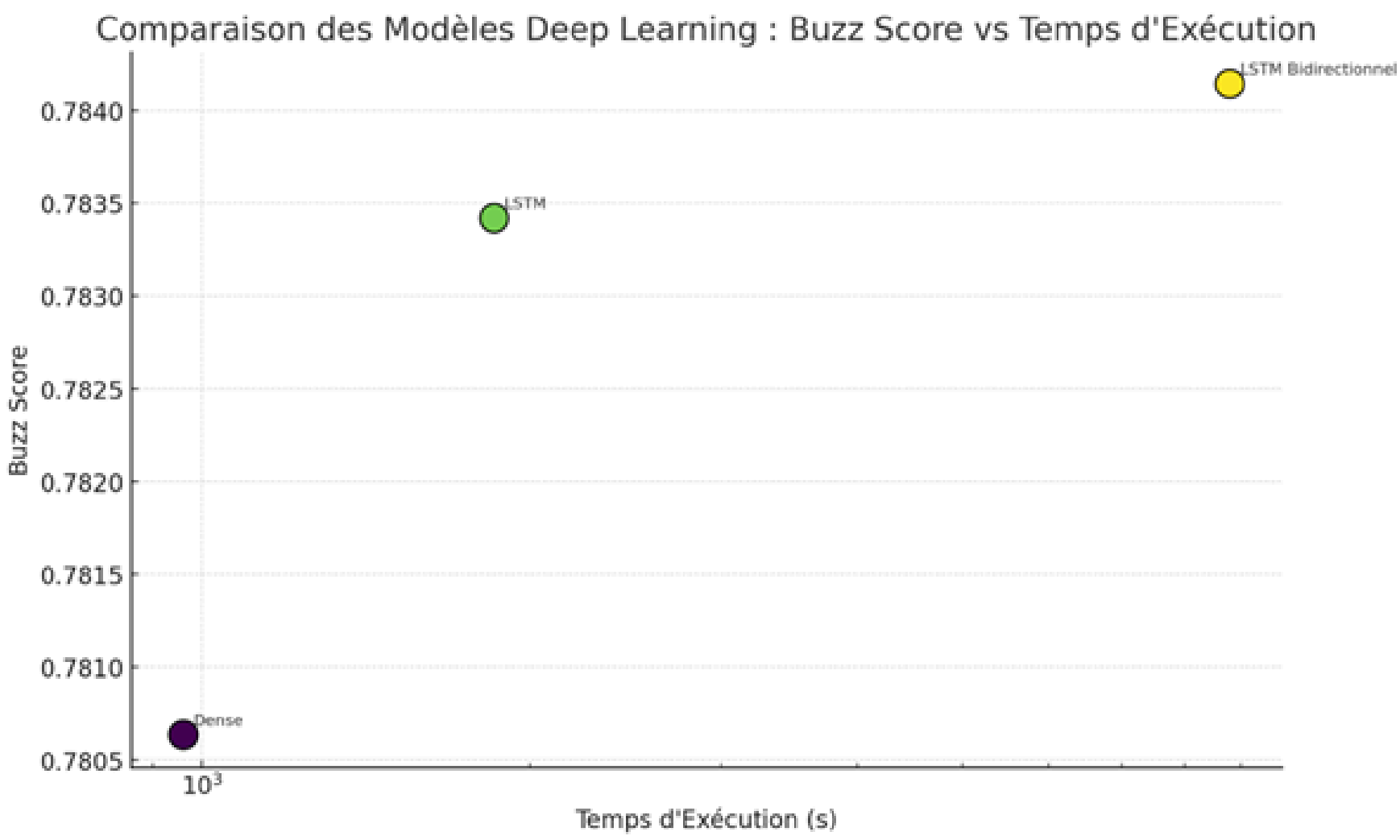
- Naive Bayes :
  - Méthode simple et rapide.
  - Performances limitées en raison de l'hypothèse d'indépendance entre les caractéristiques.
- Régression Logistique :
  - Modèle linéaire efficace pour des données de texte vectorisées.
  - Offrant un bon compromis entre complexité et précision.
- Random Forest :
  - Combinaison de multiples arbres de décision pour améliorer la robustesse.
  - Plus lent, mais plus précis que les méthodes linéaires.
- LightGBM :
  - Conçu pour traiter efficacement de grandes volumétries de données.
  - Offre un bon équilibre entre vitesse et performance.
- XGBoost :
  - Modèle ML le plus performant dans ce projet. (voir graphique)
  - Ses techniques d'optimisation avancées lui ont permis d'atteindre des scores élevés sur les données de test.



Modèles de Deep Learning (DL)

Le Deep Learning a été exploré avec trois architectures principales :

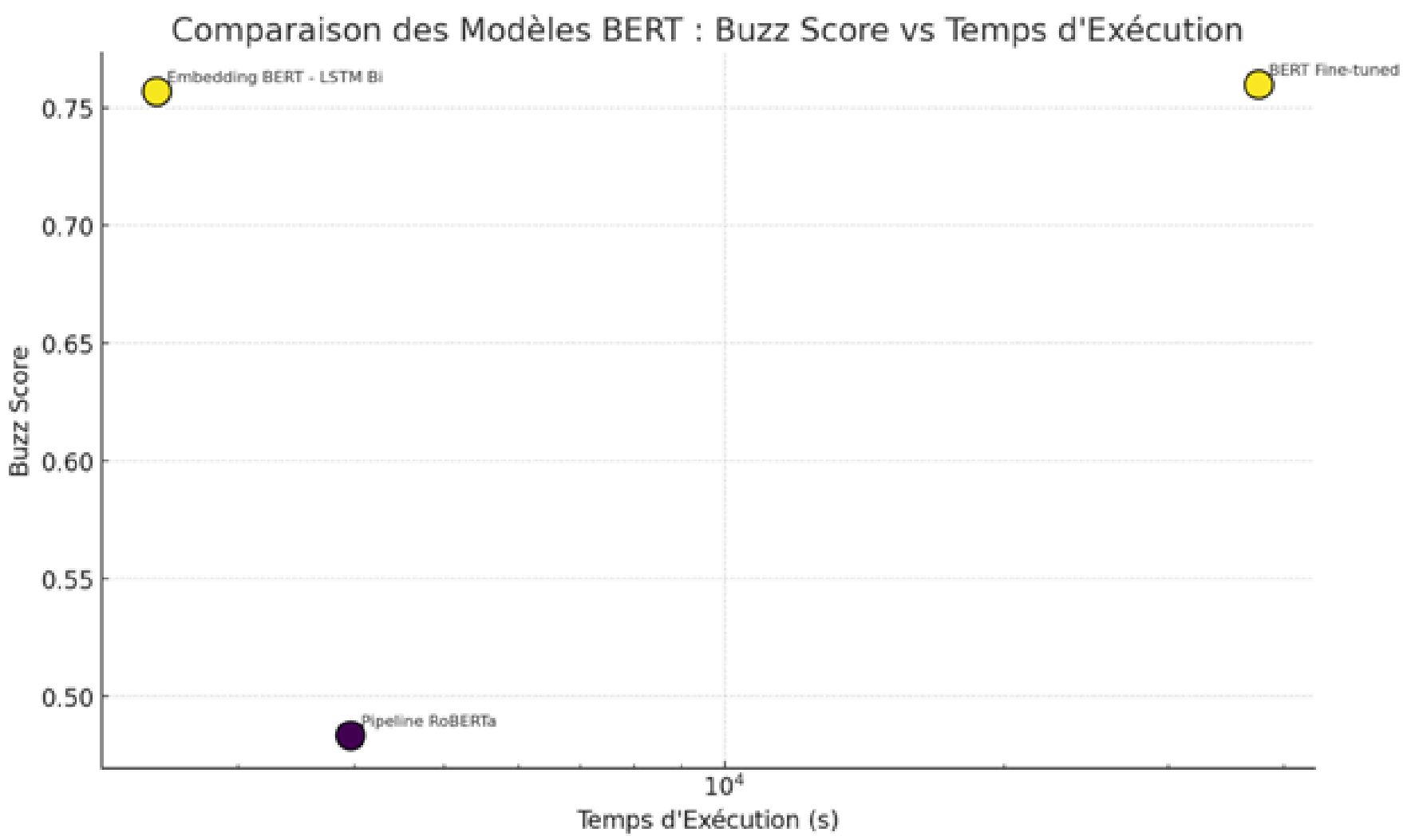
- Dense (Keras) :
  - Architecture de base utilisant uniquement des couches denses.
  - Performances correctes, mais insuffisantes face aux architectures avancées.
- LSTM :
  - Capable de capturer les relations séquentielles dans les textes.
  - Efficace pour traiter les tweets avec une structure temporelle marquée.
- LSTM Bidirectionnel :
  - Ajoute une couche contextuelle en analysant les données dans les deux directions (avant et arrière).
  - Modèle DL le plus performant du projet grâce à sa compréhension contextuelle accrue.



Approche BERT

Trois approches basées sur BERT ont été comparées :

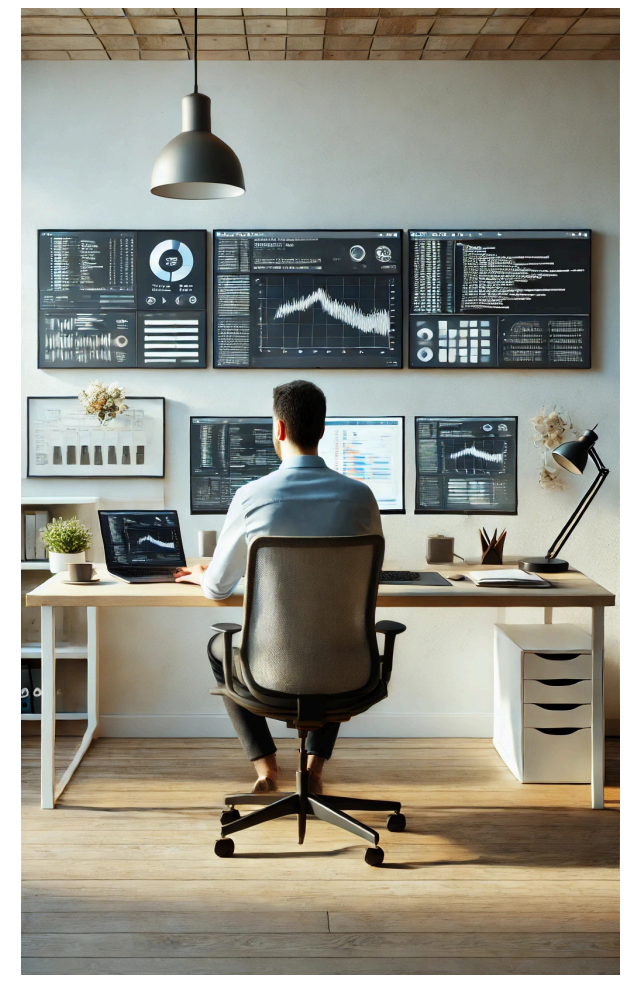
- Pipeline Hugging Face (RoBERTa) :
  - Solution rapide, idéale pour un usage sans entraînement spécifique.
  - Performances limitées comparées aux autres approches.
- Embedding BERT + LSTM Bidirectionnel :
  - Combine la richesse des embeddings BERT et la puissance des LSTM.
  - Résultats solides, mais légèrement inférieurs à BERT fine-tuned.
- BERT Fine-Tuned :
  - Approche où le modèle BERT est ajusté sur le dataset spécifique.
  - Meilleure performance globale, bien qu'exigeant en termes de ressources.



Résultats:

Tests effectués sur un échantillon de 500k tweets avec une stemmatisation du texte nettoyé.

- Meilleur modèle ML : XGBoost
  - Buzz Score: 0.788
  - Vectorisation Bag Of Word.
- Meilleur modèle DL : LSTM Bidirectionnel
  - Buzz Score: 0.784
  - Embedding : Word2Vec.
- Meilleur approche BERT : Bert fine-tune.
  - Buzz Score: 0.783

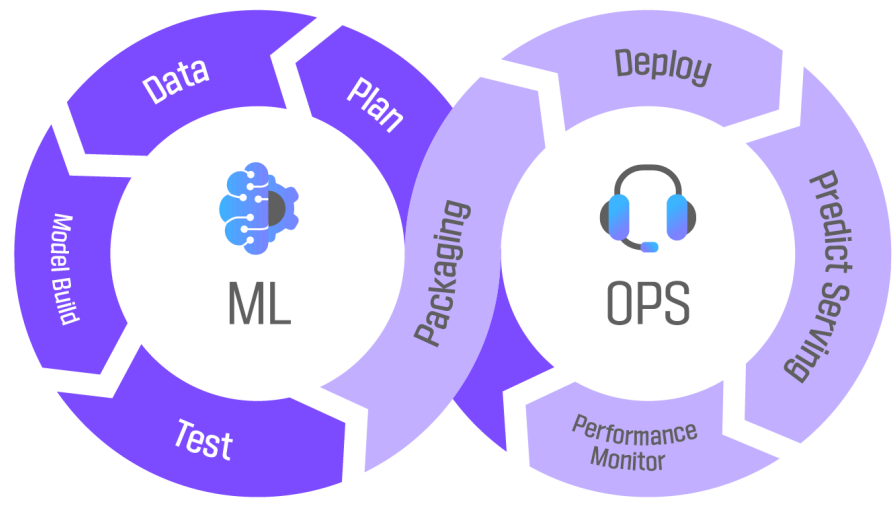


Suivi et Déploiement MLOps

Qu'est-ce que MLOps ?

MLOps, ou Machine Learning Operations, est une méthodologie inspirée des pratiques DevOps, appliquée au cycle de vie des modèles de Machine Learning. Elle vise à :

- Automatiser : Simplifier les étapes répétitives comme le déploiement et la surveillance des modèles.
- Standardiser : Assurer une approche cohérente pour la gestion des pipelines ML.
- Suivre et optimiser : Tracer les versions des modèles et surveiller leurs performances en production.



MLOps réunit des outils et des pratiques pour garantir que les modèles ML sont fiables, reproductibles et scalables dans des environnements de production. Cette approche a été cruciale pour structurer ton projet, du suivi des expériences au déploiement et monitoring via MLFlow, Docker et Azure Insights.

Suivi avec MLFlow

Qu'est-ce que MLFlow ?

MLFlow est une plateforme open-source qui facilite la gestion du cycle de vie des modèles de Machine Learning. Elle permet :

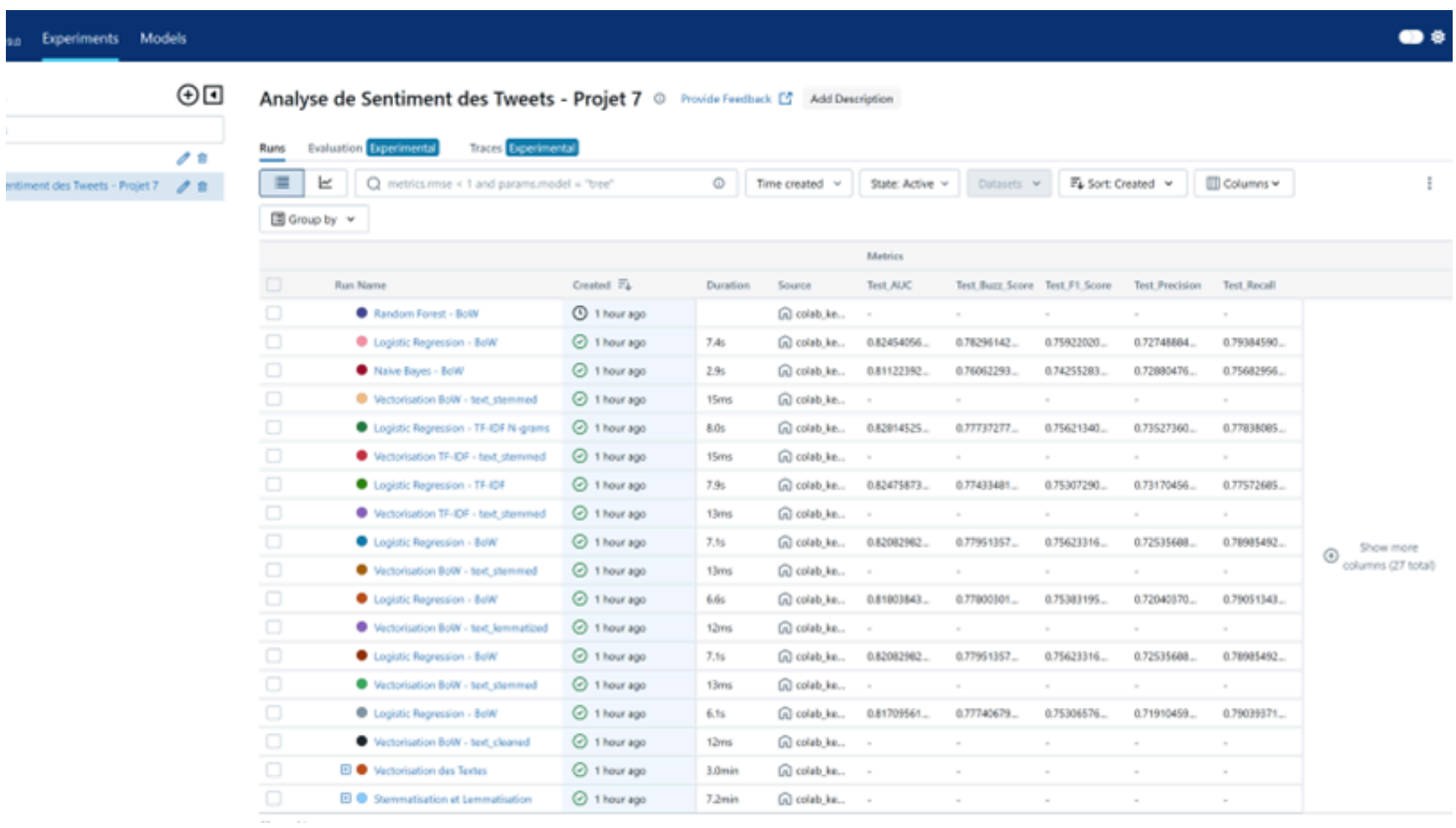
- Le suivi des expériences : Enregistrer les hyperparamètres, métriques et résultats pour chaque essai.
- Le versioning des modèles : Sauvegarder et comparer les versions de modèles.
- Le déploiement simplifié : Intégrer les modèles sous forme d'API ou dans divers environnements.

Pourquoi l'avons-nous utilisé ?

Dans ce projet, MLFlow a servi à :

- Suivre les performances des modèles (XGBoost, LSTM, BERT Fine-Tuned).
- Comparer les résultats pour sélectionner les configurations optimales.
- Sauvegarder les modèles finaux en vue de leur déploiement via l'API et le monitoring.

MLFlow a ainsi assuré traçabilité, reproductibilité et facilité de déploiement.



* *Environnement MLFlow*

Création d'une API et Interface Utilisateur

Deux composants principaux ont été développés pour rendre le modèle accessible et interactif :

API d'inférence

L'API a été implémentée avec FastAPI, un framework moderne et rapide, parfaitement adapté à la création d'applications basées sur Python. Voici ses principales fonctionnalités :

- Prédictions en temps réel : L'API permet d'envoyer un tweet en entrée et de recevoir immédiatement le résultat de l'analyse de sentiment.
- Résultats au format JSON : Les prédictions sont retournées dans un format structuré, facilitant leur intégration dans d'autres systèmes ou applications.
- Simplicité d'utilisation : Grâce à la documentation interactive générée automatiquement par FastAPI, les utilisateurs peuvent explorer et tester facilement les différents endpoints.

Interface utilisateur

L'interface utilisateur a été développée avec Streamlit, un outil léger et intuitif conçu pour créer des applications interactives de data science. Elle inclut les fonctionnalités suivantes :

- Saisie et analyse de tweets : Les utilisateurs peuvent entrer manuellement des tweets ou charger un fichier pour analyser plusieurs tweets en une seule fois.
- Visualisation des résultats : Les prédictions sont affichées de manière claire, avec des graphiques et des tableaux interactifs pour une meilleure compréhension.

Accessibilité : Aucune connaissance technique n'est requise pour utiliser l'interface, ce qui la rend idéale pour des démonstrations ou des tests rapides.

En combinant FastAPI pour la gestion des prédictions et Streamlit pour l'interactivité, ces deux outils ont permis de transformer le modèle en une solution fonctionnelle, prête à être utilisée par des non-développeurs.

* *Interface Utilisateur*

Containerisation avec Docker

Pour garantir une portabilité et une reproductibilité optimales, les deux composants principaux de l'application ont été containerisés à l'aide de Docker. Cette approche permet d'isoler les environnements et de faciliter le déploiement sur n'importe quelle infrastructure.

Backend : Conteneur pour FastAPI

Le backend, basé sur FastAPI, a été encapsulé dans un conteneur Docker indépendant. Cela a permis :

- Portabilité totale : Le backend peut être exécuté de manière identique sur différents systèmes (local, cloud, ou serveurs).
- Configuration simplifiée : Toutes les dépendances nécessaires (Python, bibliothèques, et modèle de prédiction) sont définies dans un fichier Dockerfile.
- Facilité de scaling : Le conteneur peut être dupliqué pour gérer des charges plus importantes en production.

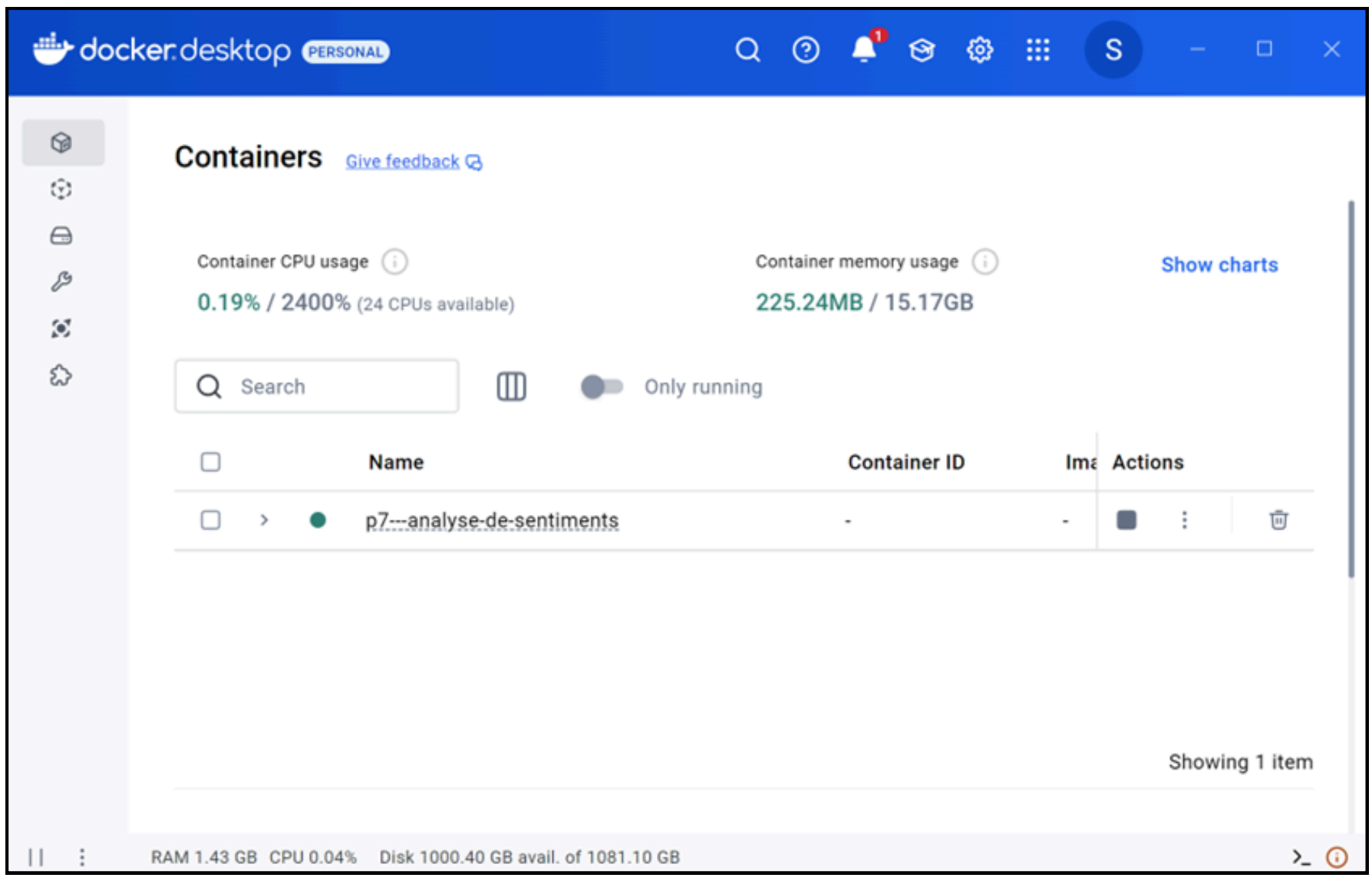
Frontend : Conteneur pour Streamlit

L'interface utilisateur, développée avec Streamlit, a également été mise dans un conteneur distinct. Ce choix offre plusieurs avantages :

- Maintenance facilitée : Les mises à jour de l'interface n'affectent pas le backend, et vice versa.
- Séparation des responsabilités : Chaque conteneur gère une tâche spécifique, garantissant une architecture modulaire.
- Environnement standardisé : Tous les outils et bibliothèques nécessaires pour exécuter Streamlit sont préinstallés, éliminant les problèmes de compatibilité.

Pourquoi Docker ?

Docker a permis de simplifier le processus de déploiement tout en assurant une cohérence entre les environnements de développement, de test et de production. En utilisant deux conteneurs distincts (un pour le backend et un pour le frontend), nous avons obtenu une solution robuste et flexible, prête à être déployée sur des plateformes comme Heroku



* Docker Desktop

Déploiement et Monitoring

Déploiement sur Heroku

Pour rendre l'application accessible en production, le déploiement a été effectué sur Heroku, une plateforme cloud connue pour sa simplicité et son coût abordable. Les principales étapes incluent :

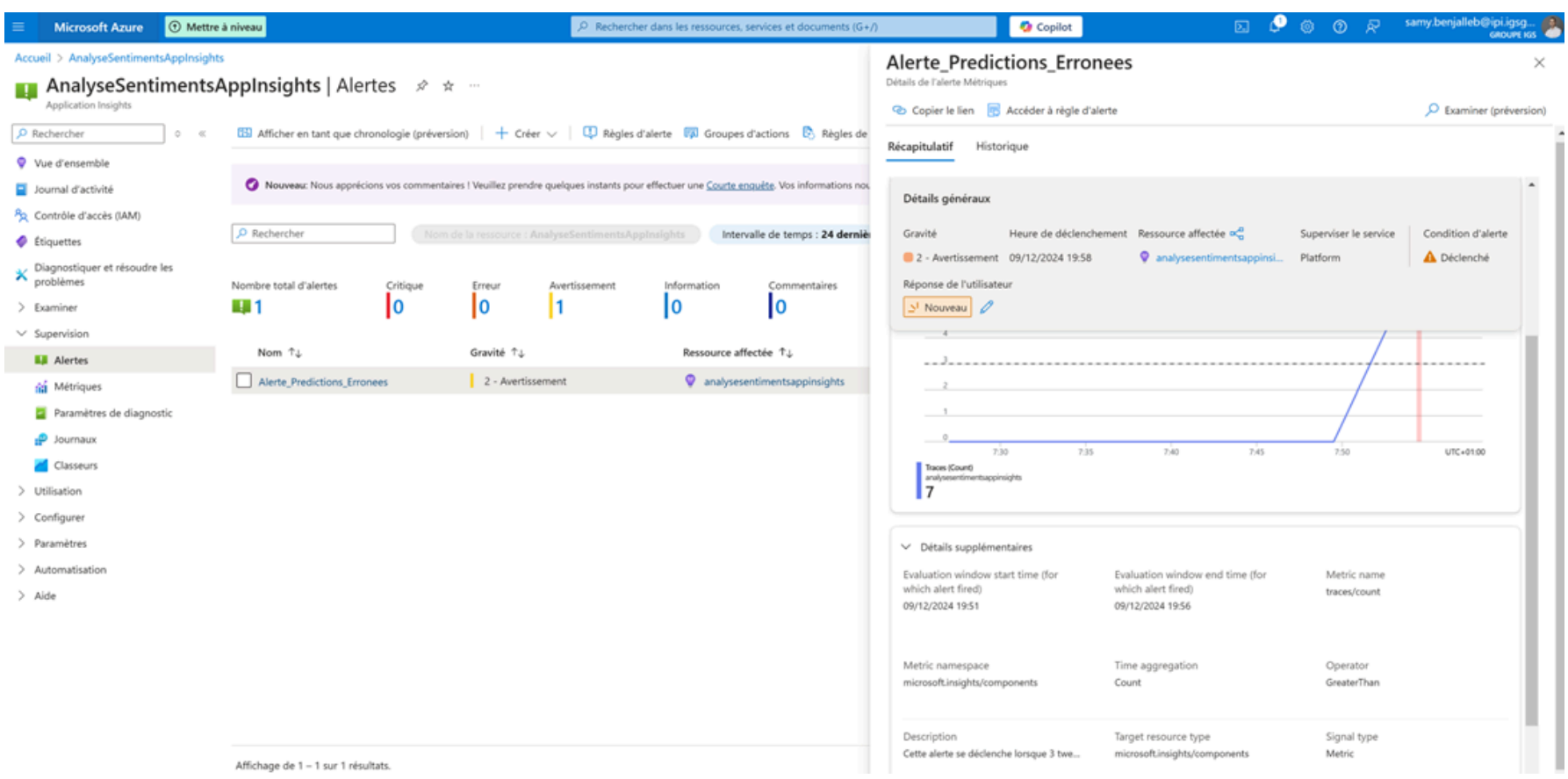
- Hébergement des conteneurs Docker : Les images Docker du backend (FastAPI) et du frontend (Streamlit) ont été déployées sur Heroku, permettant une exécution fluide des deux composants.
- Simplicité d'intégration : Grâce à la compatibilité native de Heroku avec Docker, le processus de déploiement a été rapide, avec un minimum de configuration supplémentaire.
- Accessibilité globale : L'application est accessible publiquement via une URL sécurisée, permettant à n'importe quel utilisateur de tester l'analyse de sentiment en temps réel.
-

Monitoring via Azure Insights

Une fois l'application déployée, il était essentiel de surveiller ses performances pour garantir une expérience utilisateur optimale. Azure Application Insights a été intégré pour :

- Suivi des performances : Les métriques clés comme la latence des requêtes, le taux de réussite des prédictions, et les erreurs d'API sont collectées en temps réel.
- Identification des problèmes : Les erreurs ou anomalies sont rapidement détectées, facilitant le diagnostic et la résolution.
- Création d'alertes : Des alertes personnalisées ont été configurées, par exemple :
  - Une notification est envoyée si le ratio de faux négatifs dépasse un seuil critique en moins de 5 minutes.
  - Une alerte est également déclenchée en cas d'augmentation soudaine de la latence des requêtes.
  -

Cette combinaison d'un déploiement efficace sur Heroku et d'un monitoring avancé via Azure Insights garantit non seulement la disponibilité de l'application mais aussi sa fiabilité à long terme. Ces outils permettent de réagir rapidement aux incidents et d'assurer un haut niveau de qualité pour les utilisateurs finaux.



* Azure Application Insights

Ce projet illustre la puissance des modèles ML, DL et BERT dans le cadre de l'analyse de sentiment. Bien que BERT fine-tuned offre les meilleures performances en termes de métriques, XGBoost s'est révélé être le modèle le plus équilibré pour une mise en production rapide et efficace.
Grâce à une démarche MLOps structurée incluant MLFlow, Docker et Azure Insights, une solution robuste et scalable a été développée. Les perspectives futures incluent l'exploration d'architectures plus avancées et l'amélioration continue basée sur les données collectées en production.