Optimisation des Modèles de Machine Learning pour l'analyse de sentiment des tweets

De l'approche classique au modèle BERT, avec une démarche orientée MLOps

Le domaine du **Machine Learning (ML)** a fait des avancées remarquables au fil des années, particulièrement dans la conception de modèles pour diverses applications comme la classification de texte, la reconnaissance d'image, et les recommandations personnalisées. Le choix de l'approche à adopter dépend non seulement de l'objectif à atteindre, mais aussi des ressources disponibles et des besoins spécifiques en termes de déploiement, de gestion et de suivi. Dans cet article, nous explorerons trois grandes approches de modèles de Machine Learning, allant des modèles simples comme la régression logistique aux modèles plus avancés comme **BERT**. Je détaillerai la démarche **MLOps** utilisée pour améliorer la gestion, le suivi et le déploiement de ces modèles en production.

1. Les Approches de Modèles de Machine Learning

1.1. Modèle sur mesure simple

Le modèle **sur mesure simple** est une approche classique dans la conception de modèles de Machine Learning. Ce type de modèle est souvent basé sur des algorithmes classiques comme les **régressions logistiques**, **k-NN**, ou encore les **arbres de décision**. Il est particulièrement adapté lorsque les données sont bien structurées et que le problème à résoudre est relativement simple.

Avantages :

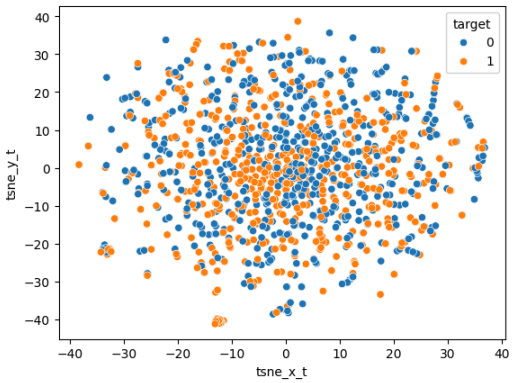
- **Facilité de mise en œuvre**
- **Bons résultats sur des données simples**
- **Interprétabilité** : Les modèles comme la régression logistique ou les arbres de décision sont interprétables, ce qui facilite la compréhension des résultats par les parties prenantes avec shap par exemple.

Limites :

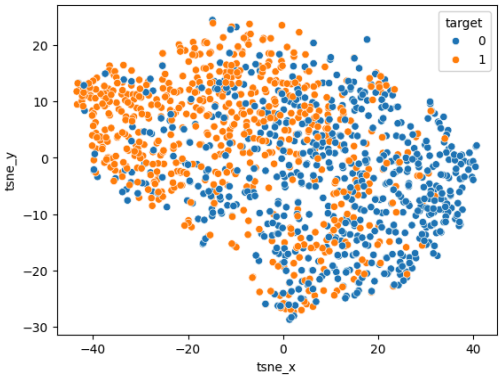
- **Performance limitée** : Sur des problèmes complexes ou avec des données non linéaires, ces modèles peuvent avoir des performances limitées.

- **Manque de flexibilité** : Ces modèles ne sont pas bien adaptés pour traiter des données non structurées, telles que le texte ou les images.

Pour l'analyse de sentiment, il est au préalable nécessaire d'utiliser un modèle d'embedding pour vectoriser le texte d'un tweet. Plusieurs techniques d'embedding existent. La vectorisation par TFIDF par exemple donnent des embedding simples, liés aux corpus analysés. Il ne comprend pas le sens et le point des mots et il est donc difficile de trouver des tendances statistiques à partir des features. En utilisant les embedding d'un modèle pré entraîné comme Bert ou Roberta par exemple, comprenant la sémantique des phrases et les relations entre les mots, il sera plus aisé de sortir des tendances statistique et permettre un apprentissage par machine learning comme la régression logistique, les arbres de décision ou encore les modèle bayésiens. Sur l'image ci-dessous on peut observer deux graphiques représentant la réduction de dimension des embeddings provenant de TFIDF et BERT. Pour TFIDF, aucune tendance ne se dessine en comparaison de Bert où les points ont tendance à former deux catégories.



Tsne d'embedding TFIDF



Tsne d'embedding Bert

1.2. Modèle sur mesure avancé

Le modèle **sur mesure avancé** repose sur des algorithmes plus puissants et plus complexes, comme les **réseaux de neurones artificiels (ANN)**, les **SVM (Support Vector Machines)** qui ne sont plus trop utilisés, **les forêts aléatoires** , **les transformers (encodage, attention, ANN, décodage)**. Ces modèles permettent de mieux capturer les relations complexes et les interactions non linéaires dans les données.

Avantages :

- **Meilleure performance sur des données complexes**
- **Polyvalence** : Ces modèles peuvent traiter divers types de données (structurées, semi-structurées, et non structurées comme les images ou le texte).

Limites :

- **Complexité et interprétabilité** : Ces modèles peuvent devenir difficiles à interpréter, particulièrement les réseaux de neurones profonds (deep learning), rendant l'analyse des résultats plus complexe.

- **Besoin en ressources**

1.3. Modèle avancé BERT

Le **modèle BERT** (Bidirectional Encoder Representations from Transformers) représente une approche de **modélisation du langage naturel** parmi les plus récentes et les plus puissantes. BERT, introduit par Google, utilise l'architecture **Transformer**, qui a révolutionné la façon de traiter les données textuelles en raison de sa capacité à gérer efficacement les dépendances contextuelles à long terme.

Avantages :

- **Performance exceptionnelle pour le traitement du langage naturel** : BERT est particulièrement efficace pour des tâches comme la classification de texte, l'analyse de sentiment, la traduction automatique et bien plus encore. Il a obtenu des résultats de pointe sur de nombreuses tâches de NLP (Natural Language Processing).

- **Pré-entrainement et fine-tuning** : BERT permet de pré-entrainer un modèle sur de grandes quantités de données textuelles générales, puis de le "fine-tuner" (adapter) à des tâches spécifiques avec un petit jeu de données. Cela rend l'utilisation de BERT très flexible et adaptable.

Limites :

- **Complexité** : Le modèle BERT est très complexe, avec des millions de paramètres. Cela nécessite une puissance de calcul significative pour l'entraînement et l'inférence.

- **Consommation mémoire** : En raison de sa taille, BERT peut consommer beaucoup de mémoire, ce qui peut poser des problèmes dans un environnement de production avec des ressources limitées.

2. Démarche Orientée MLOps

Le **MLOps** (Machine Learning Operations) est un domaine qui se concentre sur l'automatisation, le suivi, et la gestion des modèles de Machine Learning dans le cycle de vie de développement et de production. MLOps vise à combler le fossé entre les équipes de data

science et les équipes d'opérations pour assurer un déploiement fluide, une gestion robuste des versions des modèles, et un suivi continu des performances. Les principes de **MLOps** sont similaires à ceux du DevOps pour le développement logiciel, mais adaptés aux défis du Machine Learning.

- **Automatisation du pipeline** : Automatiser les étapes du cycle de vie du ML, de l'entraînement à l'inférence, et jusqu'au déploiement.
- **Collaboration entre équipes** : Faciliter la communication entre les équipes de data science, les ingénieurs DevOps, et les responsables de la production.
- **Suivi et gestion des performances** : Mettre en place des mécanismes pour suivre la performance des modèles en production et détecter tout problème (comme un **drift** des données ou une dégradation des performances).

Pour ce faire, j'utilise MLFlow, un outil web permettant l'enregistrement des différentes expériences, entraînement de modèles avec différents paramètres et recherche d'hyper paramètres optimales pour chaque modèle étudié. L'utilisation de plusieurs GPU et parallèle avec des optimisations bayésienne ou avec la librairie optuna permettent des cycles courts avec des résultats intéressants. Le **tracking** est essentiel pour suivre les versions des modèles, les hyperparamètres, et les résultats d'entraînement. **MLflow** stocke toutes les informations liées à l'entraînement du modèle et de les rendre facilement accessibles pour une analyse future. J'enregistre souvent le temps d'entraînement, l'accuracy, le f1-score. Mlflow permet de visualiser des graphiques et de comparer les modèles pour sélectionner les meilleurs.

Une fois que le modèle a été entraîné, il doit être stocké de manière sécurisée et versionnée pour une utilisation future. Le **modèle enregistré** peut être stocké dans des **stockages cloud** comme **Amazon S3**, **Azure Blob Storage**, ou des bases de données dédiées pour les modèles. Des outils comme **DVC** (Data Version Control) permettent de versionner non seulement les données, mais aussi les modèles eux-mêmes..

Les **tests** unitaires et d'intégration permettent le maintien d'un code de qualité et safe. Une fois les tests et le versionning effectué**,** le modèle est prêt à être déployé en production. Le déploiement des modèles peut se faire de plusieurs façons : **serveur d'API**, **conteneurs Docker**, ou **services de machine learning dans le cloud**. J'ai l'habitude d'utiliser un serveur fastapi avec worker, mais streamlit ou encore gradio permet un prototypage très rapide.

Le suivi des performances en production est essentiel pour détecter les dérives de données (ou **data drift**) et toute dégradation des performances des modèles. Des outils comme **Azure Application Insights** ou **Prometheus** permettent de collecter des **traces** et des **alertes** pour surveiller l'état des modèles en production. J'aime utiliser Prometheus pour les metrics et loki pour les logs, le tout connecter à une plateforme grafana pour la visualisation et la gestion des alertes. En cas de détection d'une anomalie ou d'une dégradation des performances, des alertes peuvent être envoyées aux ingénieurs pour une analyse plus approfondie.

Afin d’améliorer le modèle dans le temps, plusieurs démarches peuvent être mises en place. Le retraining automatique avec les nouvelles données collectées par l’utilisation du modèle en production.

En conclusion

Les approches de modélisation en Machine Learning varient en fonction de la complexité du problème et des données. Les **modèles sur mesure simples** et **avancés** sont utilisés pour des tâches plus classiques, tandis que des modèles comme **BERT** sont utilisés pour des problèmes plus complexes, notamment l’analyse de sentiment ou la compréhension du sens et du context sont importants. Le cycle de vie des modèles en production est automatisé par des outils MLOps, incluant le suivi des performances, le stockage des modèles, la gestion des versions, et l’automatisation des déploiements. L’amélioration continue des modèles en production grâce au monitoring et le ré-apprentissage garantissent des modèles performants et adaptés à l'évolution des données au fil du temps.