

# Titre : Développer et Déployer un Prototype d'IA pour la Prédiction de Sentiments sur les Tweets : Comparaison de Modèles et Intégration des Pratiques MLOps

---

## Introduction

Dans cet exercice, nous travaillons sur une commande d'Air Paradis. Notre mission est d'analyser les sentiments des tweets afin d'anticiper les bad buzz sur les réseaux sociaux. Nous allons préparer un prototype fonctionnel du modèle. Le modèle est exposé via une API déployée sur le Cloud, appelée par une interface locale qui envoie un tweet à l'API et récupère la prédiction de sentiment.

Nous allons comparer trois approches de modélisation différentes et intégrer des pratiques MLOps pour une gestion efficace. Puis nous ferons le déploiement sur Azure via GitHub et utiliserons les outils mis à disposition d'Azur WebApp pour suivre la stabilité de notre modèle.

---

## Présentation des Modèles

### 1. Modèle sur mesure simple : Naïve Bayes

Le modèle Naïve Bayes est une approche probabiliste simple mais efficace pour la classification de texte. Il repose sur le théorème de Bayes et l'hypothèse d'indépendance des caractéristiques. Ce modèle est particulièrement utile pour les tâches de classification où la rapidité et la simplicité sont cruciales.

**Principe de Fonctionnement** : Le théorème de Bayes utilise la probabilité conditionnelle pour calculer la probabilité qu'une donnée appartienne à une classe spécifique. Pour un texte donné, le modèle calcule la probabilité de chaque classe (par exemple, positif ou négatif) et choisit la classe avec la probabilité la plus élevée. L'hypothèse d'indépendance des caractéristiques signifie que chaque mot du texte est considéré comme indépendant des autres, ce qui simplifie considérablement les calculs.

**Exemple d'Application** : Supposons que nous ayons un tweet : "Je suis très heureux aujourd'hui". Le modèle Naïve Bayes calculera la probabilité que ce tweet soit positif ou négatif en se basant sur les fréquences des mots "je", "suis", "très", "heureux", et "aujourd'hui" dans les tweets positifs et négatifs du jeu de données d'entraînement.

- **Forces** :
  - Rapidité et efficacité en termes de calcul.
  - Facile à comprendre et à mettre en œuvre.
- **Faiblesses** :

- Hypothèse d'indépendance souvent irréaliste.
- Moins performant pour des données textuelles complexes.

## 2. Modèle sur mesure avancé : LSTM (Long Short-Term Memory)

LSTM est un type de réseau de neurones récurrents conçu pour traiter les séquences et capturer les dépendances à long terme dans les données. Il est particulièrement adapté aux tâches où le contexte historique est crucial, comme l'analyse de sentiment sur des séquences de texte.

**Principe de Fonctionnement** : Les LSTM sont capables de mémoriser des informations sur de longues séquences grâce à leurs cellules de mémoire spéciales. Contrairement aux réseaux neuronaux traditionnels, les LSTM peuvent conserver et oublier des informations en fonction des nécessités, ce qui les rend très efficaces pour capturer des dépendances temporelles complexes. Ils utilisent trois portes (input, output et forget) pour réguler le flux d'information.

**Exemple d'Application** : Pour le tweet "Je suis très heureux aujourd'hui", un modèle LSTM prendra en compte l'ordre des mots et la relation entre eux. Par exemple, il pourra comprendre que "très" modifie "heureux" et que l'ensemble de la phrase a une connotation positive.

- **Forces** :
  - Capacité à mémoriser des informations sur de longues séquences.
  - Efficace pour les tâches de classification de séquences telles que l'analyse de sentiment.
- **Faiblesses** :
  - Plus complexe et coûteux en termes de calcul.
  - Nécessite plus de données pour un bon entraînement.

## 3. Modèle avancé BERT (Bidirectional Encoder Representations from Transformers)

BERT est un modèle de langage basé sur des transformateurs qui peut comprendre le contexte des mots dans les deux directions (avant et arrière). Il représente une avancée significative dans le traitement du langage naturel grâce à sa capacité à capturer les nuances contextuelles complexes.

**Principe de Fonctionnement** : BERT utilise des mécanismes de transformateurs pour analyser le contexte complet d'un mot en regardant à la fois les mots qui le précèdent et ceux qui le suivent. Cette bidirectionnalité permet de mieux comprendre le sens des mots dans différents contextes. BERT est pré-entraîné sur une vaste quantité de données textuelles et peut être affiné (fine-tuned) pour des tâches spécifiques comme l'analyse de sentiment.

**Exemple d'Application** : Pour le tweet "Je suis très heureux aujourd'hui", BERT analysera chaque mot en tenant compte de son contexte global. Par exemple, il saura que "heureux" est un indicateur de sentiment positif en raison de sa relation avec les autres mots de la phrase.

- **Forces :**
  - Excellente performance sur les tâches de compréhension du langage naturel.
  - Capable de capturer des nuances contextuelles complexes.
- **Faiblesses :**
  - Très coûteux en calcul et nécessite des ressources substantielles.
  - Complexité accrue dans le réglage et le déploiement.

---

## Comparaison des Modèles

Pour comparer les performances des modèles, nous avons évalué leur précision, leur temps d'entraînement, et leur complexité de mise en œuvre sur un jeu de données de tweets annotés.

Modèle Paramètre	Naïves Bayes	LSTM	BERT
Temps d'entraînement	149 min de préparation 1min30 d'entraînement	2,9h	3h30min
Pourcentage de données utilisées	20%	2%	0,025%
Accuracy	TA : 0.74 VA : 0.73 alpha: 0.01	TA : 0.80 VA : 0.75	TA : 0.68 VA : 0.30
<u>Efficacité sur test:</u> Pretty good overall, just a few minor issues here and there.	Score: 0.7314, Sentiment: Positif	Score: 0.6937, Sentiment: Positif	Score: 0.9942, Sentiment: Positif
Adapté au projet	Non	Oui	Non

C'est donc le modèle LSTM que nous utiliserons pour déployer notre application. C'est lui qui obtient les meilleurs résultats avec un ratio temps d'entraînement par quantité de données utilisé intéressant. Il a également une tendance à pouvoir être précis sur des phrases aux sentiments plus neutres.

---

## Étapes Mises en Œuvre

<input type="checkbox"/>	Run Name	Created	Duration	Models	dropout_rate	embedding_dim	learning_rate	lstm_units	max_len	max_words
<input type="checkbox"/>	bedecked-fox-705		2.9h	keras	0.4	200	0.0003	256	60	15000
<input type="checkbox"/>	salty-calf-151		1.4h	keras	0.3	200	0.0005	128	50	10000
<input type="checkbox"/>	adventurous-ox-22		3.5h	keras	0.3	200	0.0005	128	60	20000
<input type="checkbox"/>	useful-kit-393		59.1min	keras	0.4	100	0.001	64	500	50000
<input type="checkbox"/>	casual-croc-946		9.0h	keras	0.4	100	0.001	64	500	50000

*exemple de MLFlow*

### 1. Suivi (Tracking)

- L'utilisation d'outils comme MLflow permet de suivre les expériences, les hyperparamètres, et les résultats de chaque itération du modèle. Cela aide à comprendre quelles configurations fonctionnent le mieux et pourquoi.
- Chaque expérimentation est enregistrée avec des détails complets pour faciliter la comparaison et l'analyse.

### 2. Stockage du Modèle

- Les modèles sont initialement stockés en local pour des tests rapides et des ajustements. Une fois validés, ils sont placés dans un dépôt comme GitHub pour faciliter le partage et le déploiement.
- Chaque itération du modèle est versionnée, ce qui permet de revenir facilement à une version précédente si nécessaire et de comparer les performances entre différentes versions.

### 3. Gestion des Versions

- L'utilisation de systèmes de contrôle de version comme Git permet de gérer les modifications apportées au code et aux données de formation. Cela inclut les modifications des scripts de prétraitement des données, les architectures de modèles, et les configurations d'entraînement.
- La gestion des versions assure que chaque modification est tracée et documentée, facilitant la collaboration entre les membres de l'équipe et la reproductibilité des résultats.

### 4. Tests Unitaires

- Les tests unitaires sont essentiels pour valider le comportement des modèles. Ils garantissent que les modèles se comportent comme prévu avant le déploiement.
- Ces tests incluent des vérifications sur les entrées et sorties du modèle, des tests de performance pour s'assurer que le modèle fonctionne efficacement, et des tests de robustesse pour garantir que le modèle peut gérer des cas extrêmes ou inattendus.

### 5. Déploiement

- Le déploiement de la version du modèle la plus efficace, dans ce cas, le modèle LSTM, se fait sur une plateforme cloud comme Azure. GitHub Actions ou d'autres outils d'automatisation sont utilisés pour automatiser le processus de déploiement.
- Ce processus automatisé inclut le transfert du modèle, la configuration de l'environnement d'exécution, et la mise en service du modèle pour qu'il soit accessible via des API ou d'autres interfaces.

Triggered via push yesterday

ThomasDiF pushed -o- ac26162 master

Status

Success

Total duration

19m 13s

Billable time

20m

Artifacts

1

master\_projet7oc.yml

on: push

✓ build

1m 40s

✓ deploy

17m 16s

https://projet7oc.azurewebsites.net


exemple d'un déploiement GitHub vers Azure

Build and deploy Python app to Azure Web App - Projet7OC

Filter workflow runs


15 workflow runs


Event Status Branch Actor

 Simplification


Build and deploy Python app to Azure Web App - Projet7OC #29: Commit 79ee937 pushed by ThomasDiF

modèle-rapide

 34 minutes ago


 In progress


...

 MaJ Requirements


Build and deploy Python app to Azure Web App - Projet7OC #28: Commit df0b33f pushed by ThomasDiF

modèle-rapide

 1 hour ago


 18m 55s


...

 Adaptation WebApp


Build and deploy Python app to Azure Web App - Projet7OC #27: Commit 3387ade pushed by ThomasDiF

modèle-rapide

 3 hours ago


 11m 42s


...

 correction logger


Build and deploy Python app to Azure Web App - Projet7OC #20: Commit f04e70f pushed by ThomasDiF

modèle-rapide

 3 days ago


 48m 34s


...

 MaJ Docker et Requi


Build and deploy Python app to Azure Web App - Projet7OC #19: Commit 1ead657 pushed by ThomasDiF

modèle-rapide

 3 days ago


 47m 21s


...

 Ajout des metriques


Build and deploy Python app to Azure Web App - Projet7OC #18: Commit 6db5706 pushed by ThomasDiF

modèle-rapide

 3 days ago


 12m 3s


...

 Changement de clef


Build and deploy Python app to Azure Web App - Projet7OC #17: Commit f2b0faa pushed by ThomasDiF

modèle-rapide

 last week


 13m 16s


...

 MaJ requirements


Build and deploy Python app to Azure Web App - Projet7OC #16: Commit b219285 pushed by ThomasDiF

modèle-rapide

 last week


 11m 34s


...

 Add or update the Azure App Service build and deployment workflow config


Build and deploy Python app to Azure Web App - Projet7OC #14: Commit 2cd0ef0 pushed by ThomasDiF

modèle-rapide

 last week


 11m 39s


...

 Simplified script for testing without Azure Insights

Build and deploy Python app to Azure Web App - Projet7OC #12: Commit d33b45d pushed by ThomasDiF

modèle-rapide

 last week

 11m 39s

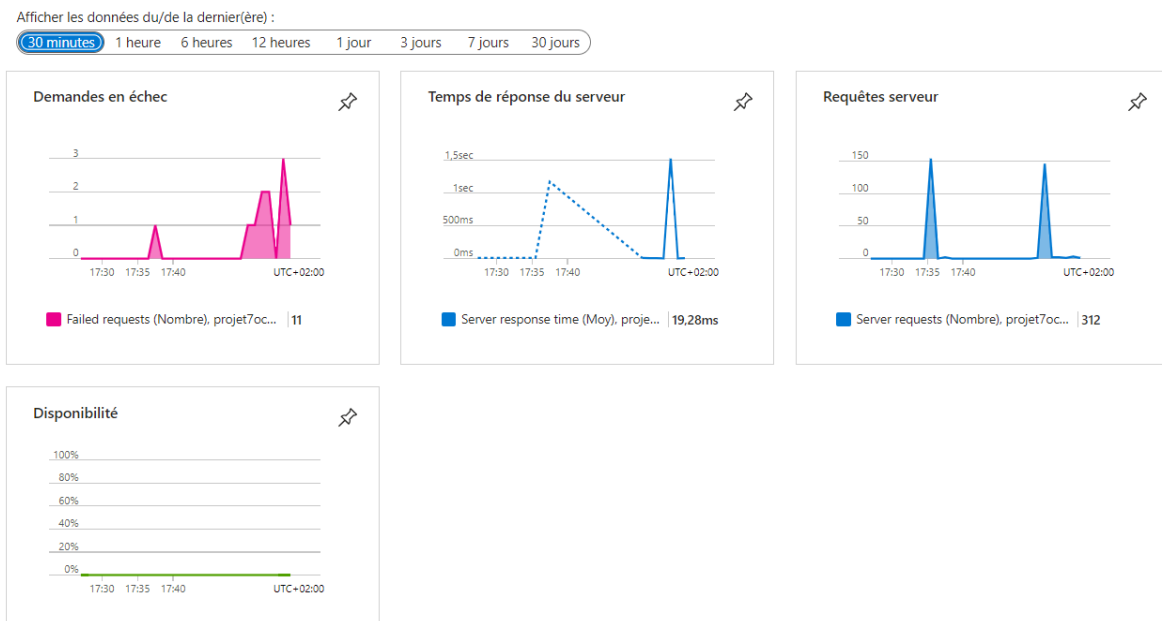
...

exemple de différents déploiements

## Suivi des Performances en Production

- **Monitoring intégré d'Azure WebApp**

- Une fois le modèle déployé, il est crucial de surveiller ses performances en production pour s'assurer qu'il continue de fonctionner de manière optimale.
- La section supervision est utilisée pour capturer les métriques de performance, telles que la latence des prédictions, le taux de réussite des requêtes, et la fréquence des erreurs.
- Les alertes peuvent être configurées pour notifier l'équipe en cas de dégradation des performances ou d'anomalies détectées.



*exemple de métriques récoltées via Azure Web App*

## Analyse et Amélioration Continue

- **Analyse des Statistiques**

- Les données collectées par les outils de monitoring sont analysées régulièrement pour identifier les modèles de dégradation de performance. Par exemple, une baisse de précision sur de nouvelles données peut indiquer que le modèle a besoin d'être réentraîné avec des données plus récentes ou diversifiées.
- Cette analyse permet de prendre des décisions éclairées sur les ajustements à apporter au modèle ou aux processus de traitement des données.

- **Métriques Sélectionnées**

Pour mesurer ces métriques, nous allons ajouter le paramètre 'true\_label' dans notre requête sur Postman. Le principe est d'indiquer quelle est la valeur que le modèle est censé prédire : 1 pour positif, 0 pour négatif.

Ensuite, le score du modèle sera comparé à notre valeur. Si elle est proche, notre modèle a fait une bonne prédiction ; sinon, une mauvaise. De cette comparaison ressortiront les résultats de nos métriques sur Azure.

- Les métriques spécifiques sélectionnées pour le suivi des performances peuvent inclure la précision, le rappel, la F1-score, la latence des prédictions, et l'utilisation des ressources.
- Ces métriques aident à évaluer non seulement l'efficacité du modèle, mais aussi son coût opérationnel et son impact sur l'infrastructure.

En intégrant ces pratiques MLOps, le projet assure une gestion fluide et efficace du cycle de vie du modèle, de la formation initiale à la surveillance en production, garantissant ainsi la robustesse et la fiabilité des prédictions de sentiment sur les tweets.

---

## Conclusion

Ce projet a démontré la puissance de différentes approches de modélisation pour la prédiction de sentiment sur les tweets. Nous avons conclu que LSTM était le modèle le plus adapté à notre projet.

L'intégration des pratiques MLOps a assuré une gestion fluide et efficace du cycle de vie du modèle, de la formation initiale à la surveillance en production. L'utilisation d'outils comme MLflow a facilité le suivi des expériences et la gestion des versions, permettant une comparaison et une amélioration continue des modèles.

La mise en œuvre sur Azure a offert une infrastructure robuste pour le déploiement et le suivi. L'analyse des journaux et métriques ont été cruciaux pour le monitoring des performances en production, en capturant des métriques essentielles telles que la latence des prédictions et la fréquence des erreurs.

En résumé, ce projet a illustré l'importance de choisir le bon modèle pour la tâche de prédiction de sentiment et l'efficacité des pratiques MLOps pour gérer le cycle de vie complet du modèle. La combinaison de modèles avancés comme LSTM et des outils de gestion et de déploiement comme Azure a permis de créer une solution performante et scalable, prête à répondre aux exigences de précision pour l'analyse de sentiments sur les tweets.

---

## Références

- Documentation LSTM
- [Documentation BERT](#)
- [Azure Machine Learning](#)
- [MLflow](#)
- [Docker](#)
- [Azure Application Insights](#)