

# Détecter les bad buzz grâce au Deep Learning



Octobre 03, 2023

Nasr-edine Draï, étudiant  
ingénieur IA

# Introduction

# Introduction

## Objectif:

- Développer un modèle de machine learning capable de prédire le sentiment associé à un tweet.

## Contexte:

- Air Paradis, une compagnie aérienne, souhaite pouvoir anticiper les bad buzz sur les réseaux sociaux.

# Introduction

## Méthodologie:

- Elaboration de 3 modèles de machine learning :
  - Modèle simple : régression logistique
  - 2 Modèle avancé : réseau de neurones profonds
- Comparaison des performances des modèles :
  - Mesure de la précision, de la perte
- Mise en production du modèle avancé :
- Utilisation d'un outil de MLOps pour la gestion des expérimentations et le déploiement continu

## Présentation du jeu de données:

- Le dataset Sentiment140 comprend 1 600 000 tweets collectés via l'API Twitter. Ces tweets ont été étiquetés en fonction de leur polarité sentimentale, où 0 représente un sentiment négatif et 4 un sentiment positif. Ce dataset est couramment utilisé pour des tâches d'analyse de sentiment.

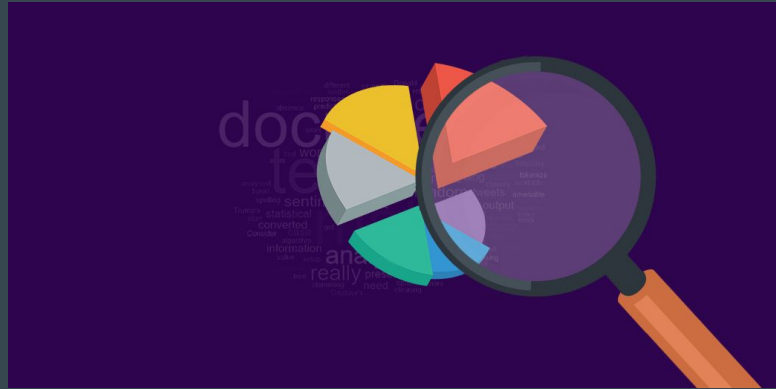
# Pré-traitement des données



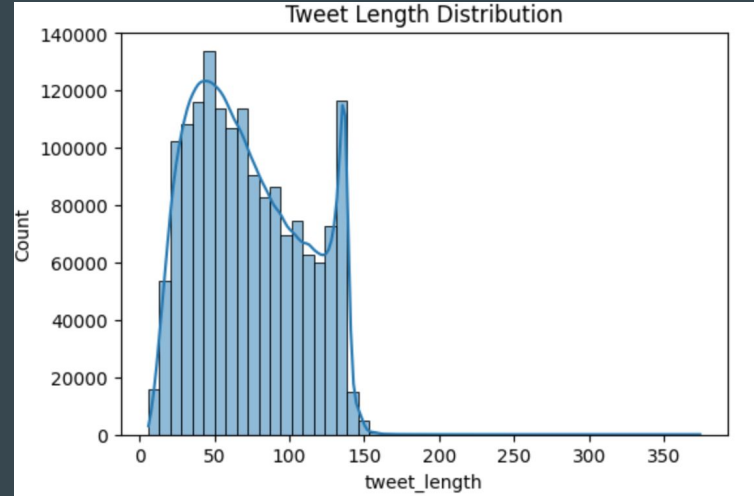
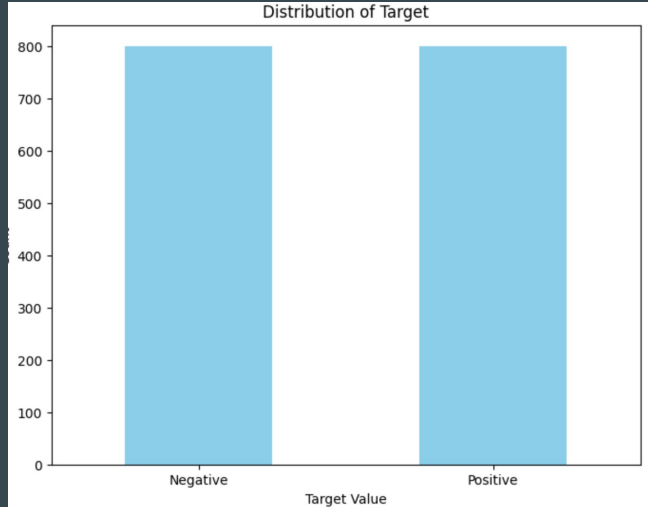
# Normalisation du texte 🧹 :

Étape de Normalisation	Description
Expansion des Contractions 📖	Étendre les contractions, par exemple, changer "it's" en "it is".
Tokenisation 📄	Diviser le texte en mots ou en tokens.
Suppression des Caractères Spéciaux 🚫	Éliminer les caractères spéciaux comme les symboles et les emojis.
Suppression de la Ponctuation 🗑️	Retirer les signes de ponctuation.
Suppression des Liens et des Noms d'Utilisateur 🌐	Cela permet de se concentrer sur le contenu du tweet lui-même, en éliminant les éléments extérieurs.
Suppression des Emojis 😊	Éliminer les émoticônes pour une analyse textuelle.
Suppression des Mots Vides 🚫	Éliminer les mots courants comme "et" ou "le".
Suppression des Tokens d'une Seule Lettre 📄	Retirer les mots composés d'une seule lettre.
Conversion en Minuscules ✍️	Transformer tous les mots en lettres minuscules.
Lemmatisation 📖	Réduire les mots à leur forme de base.

# Analyse exploratoire



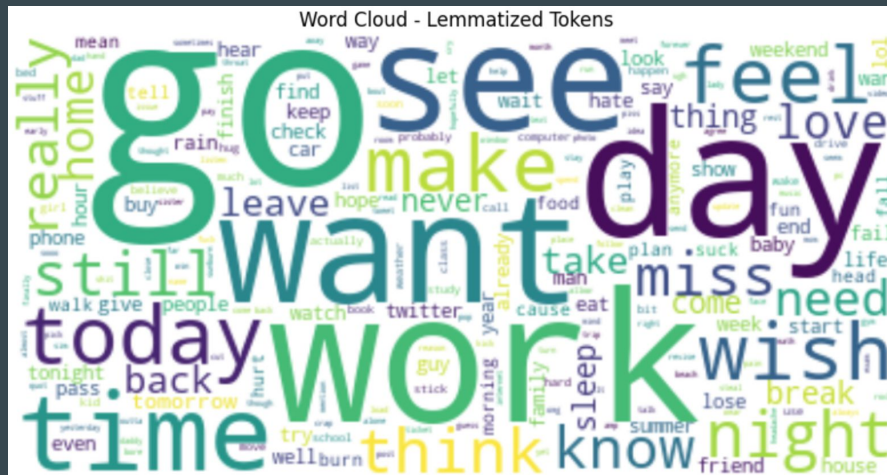
# Distribution des classes et longueur des tweets



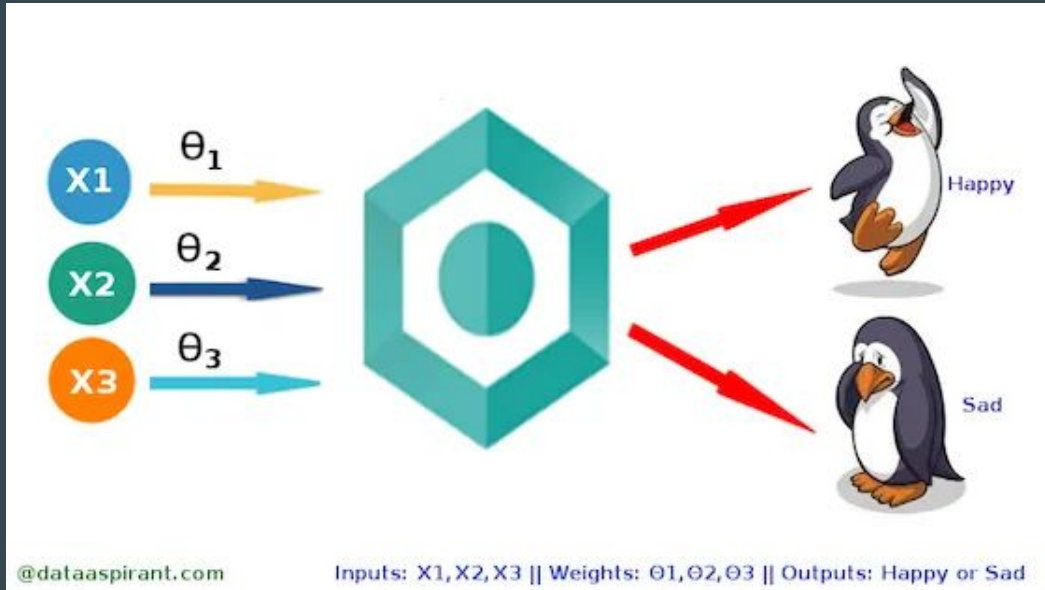
L'ensemble de données Sentiment140 est équilibré, ce qui est important pour éviter les biais du modèle  
La plupart des tweets ont entre 30 et 130 caractères de long.



Fruit	Number of people
Apple	5
Banana	3
Orange	7
Grape	2



# Modèle simple avec la régression logistique



# TF-IDF Vectorization

Observations :

- La vectorisation TF-IDF est utile pour convertir le texte en données numériques.
- Les caractéristiques des tweets sont extraites à l'aide de la vectorisation TF-IDF.
- TF-IDF attribue des poids à chaque mot ou expression en fonction de sa fréquence dans le corpus et de sa rareté.

## Scores TF-IDF Premier Document

	TF-IDF Score	Word
Token ID		
0	0.276951	app
1	0.094559	back
2	0.228376	customer
3	0.506727	dmitris
4	0.196551	establishment
5	0.308943	fuck
6	0.169141	go
7	0.180685	half
8	0.167391	maybe
9	0.084476	never
10	0.196551	open
11	0.048927	order
12	0.054021	place
13	0.115477	really
14	0.088679	take
15	0.101794	tell
16	0.323257	term
17	0.441518	tipper

## Entraînement du modèle Logistic Regression avec TF-IDF

- Les caractéristiques des tweets sont extraites à l'aide de la vectorisation TF-IDF.
- Le modèle apprend à prédire la classe d'un tweet en fonction de ses caractéristiques.

## Évaluation du modèle avec la cohérence

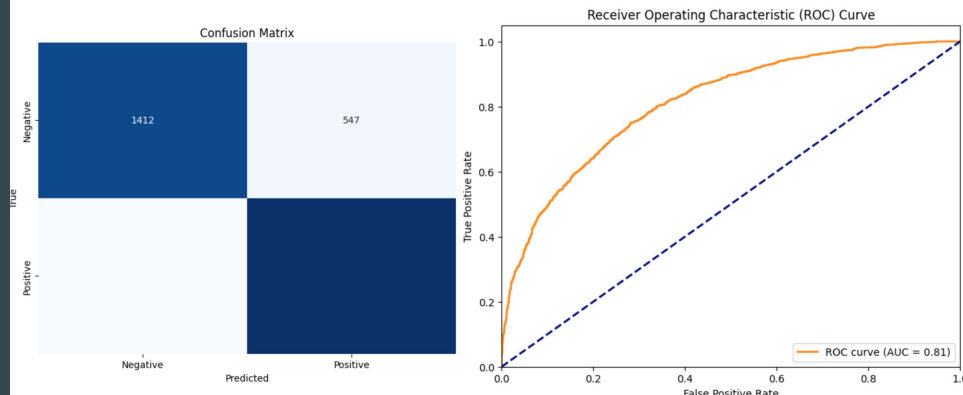
- Le modèle a atteint une précision de 73,18 %, classant correctement 73,18 % des tweets.

## Visualisation de la Performance du Modèle

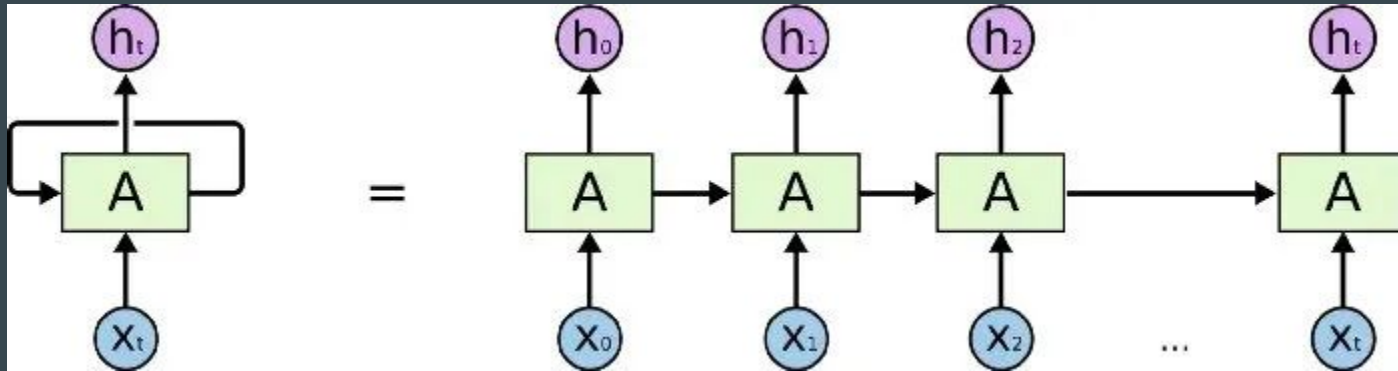
- La performance du modèle a été visualisée à l'aide d'une matrice de confusion et d'une courbe ROC.
- La matrice de confusion montre le nombre de tweets correctement et incorrectement classés.

Training Time: 1.30 seconds  
Prediction Time: 0.00 seconds  
Accuracy: 0.7317747611865258  
Classification Report:

	precision	recall	f1-score	support
0	0.73	0.72	0.73	1959
1	0.73	0.74	0.74	2019
accuracy			0.73	3978
macro avg	0.73	0.73	0.73	3978
weighted avg	0.73	0.73	0.73	3978



## Modèle LSTM avec des Incrustations GloVe



# Modèle LSTM avec des Incrustations GloVe

## Incrustations GloVe

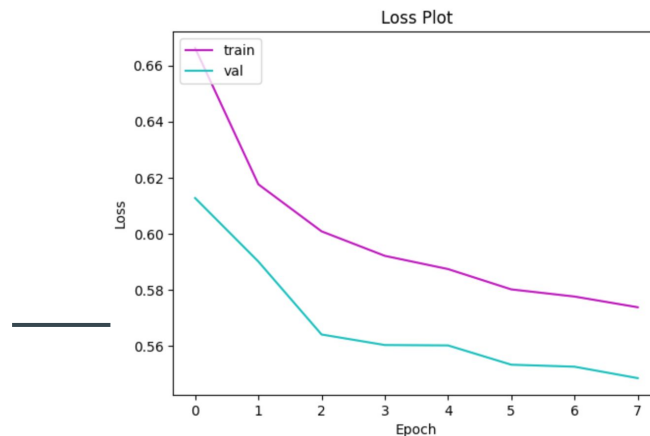
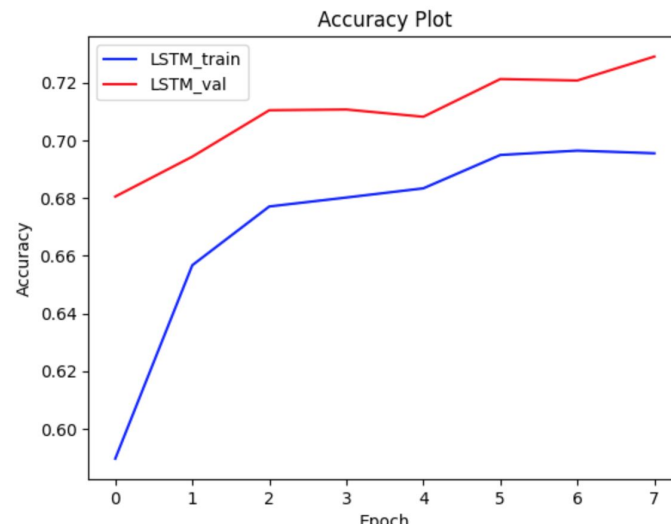
- Les incrustations GloVe pré-entraînées sont utilisées pour représenter les mots sous forme de vecteurs numériques.
- Les vecteurs GloVe capturent les relations sémantiques entre les mots.

## Architecture du Modèle

- Modèle LSTM (Long Short-Term Memory) est utilisé pour l'apprentissage du sentiment des tweets.
- Une couche Dense avec une activation softmax est utilisée pour la classification binaire (positif/négatif).

## Entraînement du Modèle

- Le modèle est entraîné sur les données d'entraînement avec 8 époques et une taille de lot de 1024.
- Les performances du modèle sont évaluées sur l'ensemble de test, atteignant une précision de 72,90%.



# Analyse de sentiment des tweets avec BERT



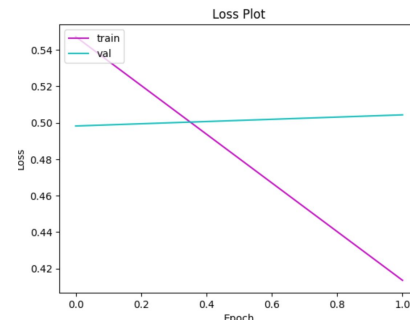
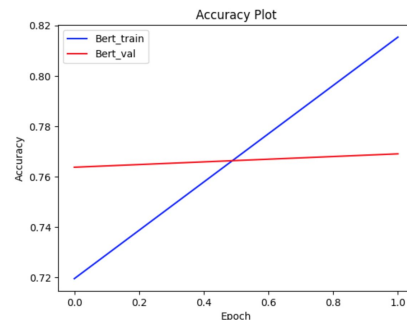
## Utilisation de bert

- Il s'agit d'un type de modèle de langage qui a été développé par Google AI en 2018.
- Les modèles BERT sont entraînés sur un ensemble de données massif de texte et de code, et ils peuvent être utilisés pour une variété de tâches.

## Entraînement du Modèle

- Sur l'ensemble de test retenu, le modèle BERT a atteint une précision de 76.91 %. C'est significativement plus élevé que la précision des deux modèles précédents.

100/100 [=====] - 41s 357ms/step - loss: 0.5044 - accuracy: 0.7691  
Accuracy: 76.91%  
Loss: 50.44%





## Comparaison des 3 modeles

- Le modèle BERT est le meilleur choix pour une précision élevée. Cependant, il est plus lent à entraîner et à prédire que les modèles de régression logistique ou GloVe avec LSTM.
- Si vous avez besoin d'un modèle rapide à entraîner et à prédire, choisissez la régression logistique ou GloVe avec LSTM.
- Si vous avez besoin d'un modèle avec la précision la plus élevée possible, choisissez BERT.

Modèle	Précision
BERT	76,91%
Régression logistique	73,17%
GloVe avec LSTM	72,90%

Modèle	Avantages	Inconvénients
BERT	Haute précision, peut apprendre les dépendances contextuelles	Long temps d'entraînement, long temps de prédiction
Régression logistique	Simple à entraîner et à interpréter, temps de prédiction rapide	Précision inférieure, ne peut pas apprendre les dépendances contextuelles
GloVe avec LSTM	Bonne précision, temps d'entraînement plus rapide que BERT	Temps de prédiction plus lent que la régression logistique, ne peut pas apprendre les dépendances contextuelles aussi bien que BERT

# API en Flask pour la prédiction



# Flask

web development,  
one drop at a time

## API flask

- L'API est utilisée pour prédire le sentiment d'un tweet, qui peut être positif ou négatif.
- La route de l'API pour la prédiction est /predict. Elle accepte une requête POST avec le corps JSON suivant :

JSON

```
{  
  "tweet": "Ceci est un tweet !"  
}
```

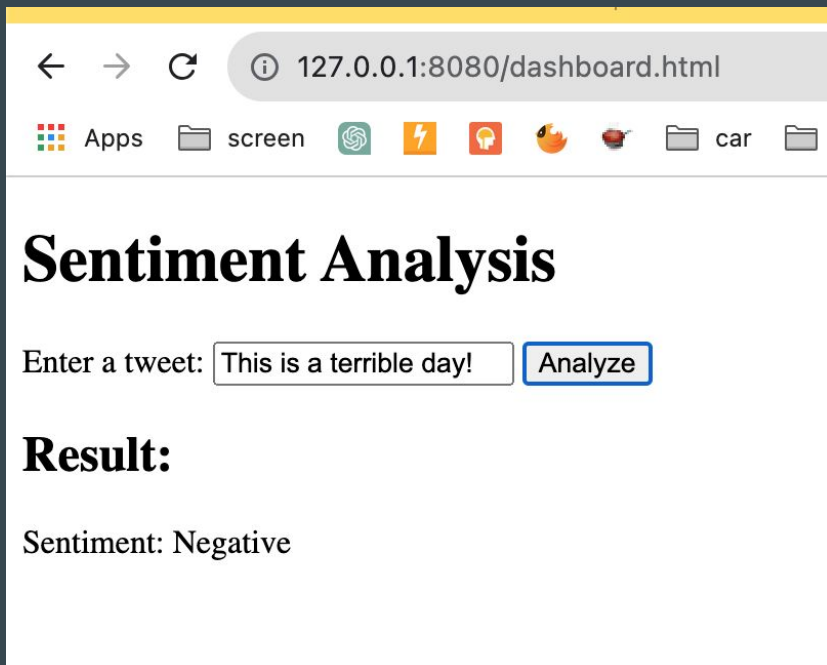
- L'API renverra une réponse JSON avec le corps suivant :

JSON

```
{  
  "sentiment": "Positif"  
}
```

## Utilisation via un navigateur ou par un outil de requete comme Postman

- L'API est utilisation via un navigateur et la page html effectuant la requete.
- L'API est egalment utilisable via une simple requete au format JSON



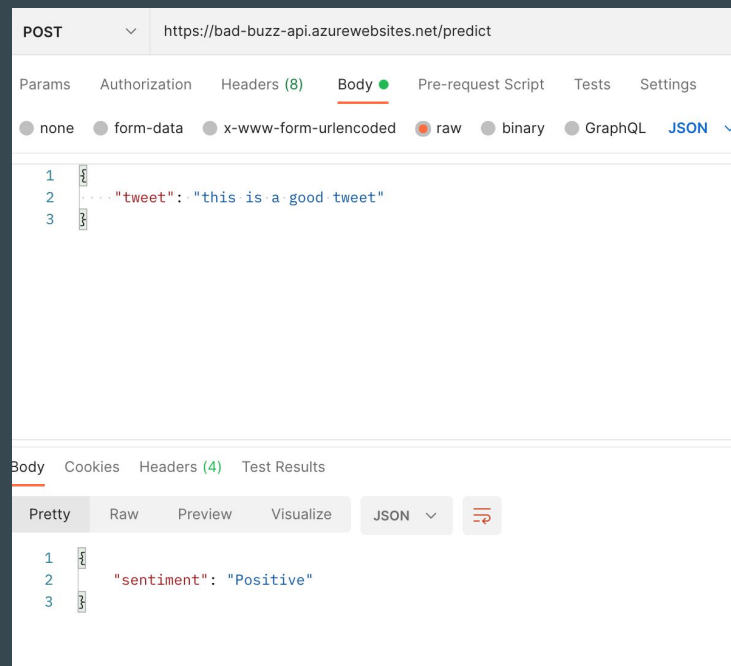
A screenshot of a web browser displaying a sentiment analysis dashboard. The address bar shows the URL `127.0.0.1:8080/dashboard.html`. The page has a title "Sentiment Analysis" and a form with the label "Enter a tweet:". The input field contains the text "This is a terrible day!". To the right of the input field is a blue button labeled "Analyze". Below the form, the word "Result:" is displayed, followed by the text "Sentiment: Negative".

# Sentiment Analysis

Enter a tweet:  Analyze

**Result:**


Sentiment: Negative







# Utilisation de Github et Github Action














# Github and github action


 **detect-badbuzz** Public


 **main**  **1 branch**  **0 tags**


 **nasr-edine** add medium article


 .github/workflows	Add unit tests
 deliverables	add medium article
 models	Enhance Model for Dual-Class Probabilities
 notebooks	Enhance Model for Dual-Class Probabilities
 tests	remove a test
 .gitignore	Add notebooks
 README.md	first commit
 app.py	Enhance Model for Dual-Class Probabilities
 dashboard.html	Add Sentiment Analysis Dashboard HTML
 requirements.txt	Use latest versions of dependencies
 text_processing_utils.py	add wordnet


21 workflow runs


 **add medium article**  
Build and deploy Python app to Azure Web App - bad-buzz-api #21: Commit 4709dc4 pushed by nasr-edine main

 **Use latest versions of dependencies**  
Build and deploy Python app to Azure Web App - bad-buzz-api #20: Commit ae994d8 pushed by nasr-edine main

 **Enhance Model for Dual-Class Probabilities**  
Build and deploy Python app to Azure Web App - bad-buzz-api #19: Commit a304434 pushed by nasr-edine main

 **remove a test**  
Build and deploy Python app to Azure Web App - bad-buzz-api #18: Commit cac93c0 pushed by nasr-edine main

 **Add tensorflow to requirements file**  
Build and deploy Python app to Azure Web App - bad-buzz-api #17: Commit 5c25e30 pushed by nasr-edine main

 **Add advanced model**  
Build and deploy Python app to Azure Web App - bad-buzz-api #16: Commit df1b52b pushed by nasr-edine main

## Test unitaires pour l'API



- J'ai utilisé la bibliothèque Python unittest pour écrire les tests unitaires de mon API.
- Les tests unitaires sont exécutés dans un dossier séparé de l'API, ce qui permet de garantir que les tests ne sont pas affectés par l'état de l'API.

```
python -m unittest
```

```
Ran 2 tests in 0.001s
```

```
OK
```

# Utilisation de la platform Azure





## Deploiement sur Azure

- Pour déployer l'API, il suffit de pousser le fichier de workflow sur GitHub. GitHub Actions lancera alors automatiquement le workflow.
- Une fois le workflow terminé, l'API est sur Azure Web App. Je peux ensuite accéder à l'API via l'URL: `bad-buzz-api.azurewebsites.net`
- Pour une démarche MLOPS, j'ai intégré les tests unitaires dans la phase de build

^ Bases

Groupe de ressources ( <a href="#">déplacer</a> )	: <a href="#">hello-flask-resource-group</a>	Domaine par défaut	: <a href="#">bad-buzz-api.azurewebsites.net</a>
Statut	: En cours d'exécution	Plan App Service	: ASP-badbuzzappgroup-89cc (F1: 1)
Emplacement ( <a href="#">déplacer</a> )	: West Europe	Système d'exploitation	: Linux
Abonnement ( <a href="#">déplacer</a> )	: <a href="#">Azure subscription 1</a>	Contrôle d'intégrité	: Non configuré
ID d'abonnement	: 5be951e7-c40e-4f06-91b9-5ffe2a181970	Projet GitHub	: <a href="https://github.com/nasr-edine/detect-badbuzz">https://github.com/nasr-edine/detect-badbuzz</a>
Étiquettes ( <a href="#">modifier</a> ) : <a href="#">Ajouter des étiquettes</a>			

add medium article #21

Summary

Jobs

- build
- deploy

Run details

Usage

Workflow file

**build**

succeeded 3 hours ago in 1m 22s

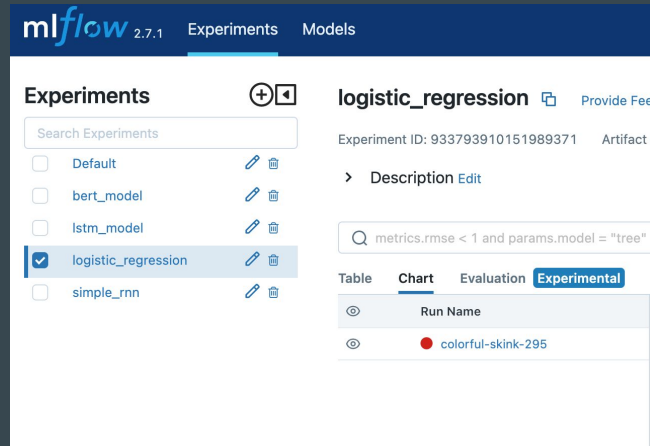
- Set up job
- Run actions/checkout@v2
- Set up Python version
- Create and start virtual environment
- Install dependencies
- Run unit tests
- Upload artifact for deployment jobs
- Post Run actions/checkout@v2
- Complete job

# Utilisation de MLflow



# Utilisation de mlflow

- MLflow est une plateforme open source de gestion du cycle de vie des modèles (MLOps). Il permet de suivre les expérimentations, de reproduire les résultats et de déployer des modèles de machine learning.
- Suivi des expérimentations: MLflow permet de suivre les expérimentations de machine learning en enregistrant les paramètres, les données et les résultats de chaque expérimentation. Cela permet de reproduire facilement les résultats et de comparer les différentes versions des modèles.



## Utilisation de Medium

The logo consists of a solid black circle followed by two vertical black bars of equal height, positioned to the left of the word "Medium".

**Medium**

## Medium pour la redaction de mon article

- J'ai utilise la plateforme Medium pour la redaction de mon article, a l'adresse suivante:

<https://medium.com/@nasredinedrai/d%C3%A9tecter-les-bad-buzz-gr%C3%A2ce-au-deep-learning-72599fd4c8c7>

# Détecter les bad buzz grâce au Deep Learning



Nasr-edine Draï

5 min read · 3 hours ago



# L'auteur



Nasr-edine Draï, étudiant  
ingénieur IA

---

**Merci de votre attention ! Je suis maintenant prêts à répondre à vos questions.**