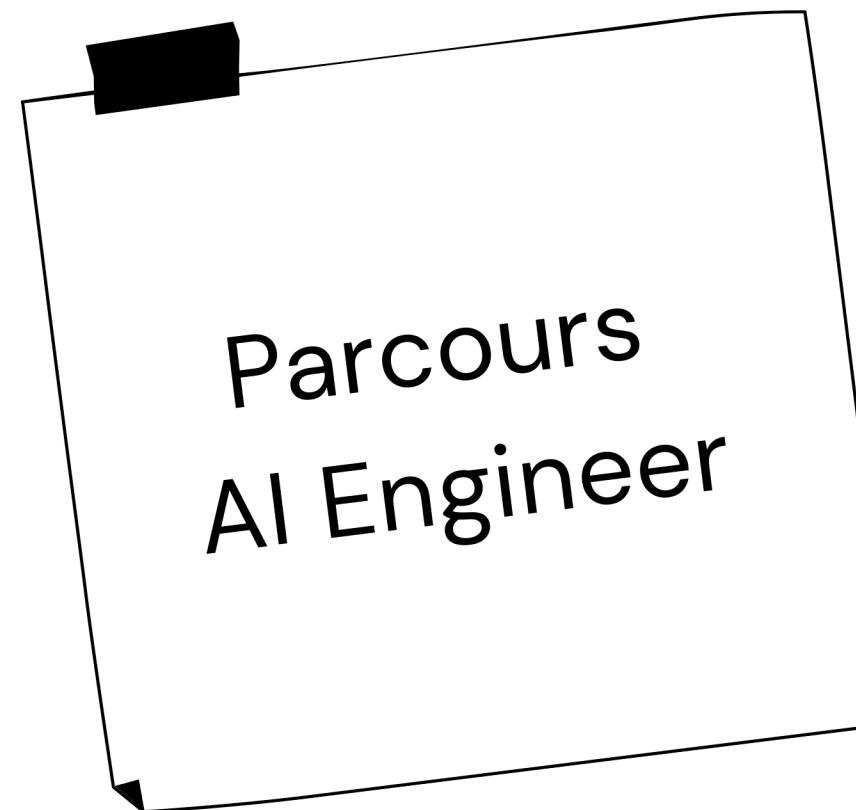
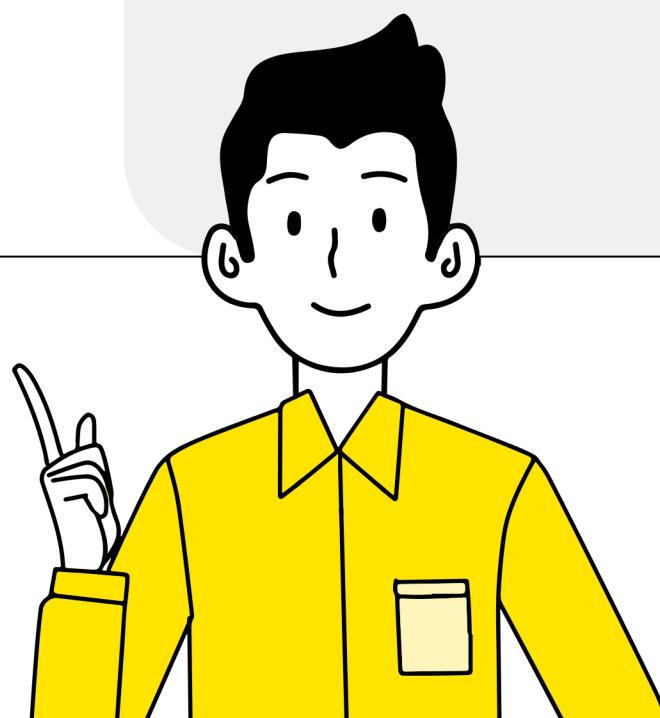


Réalisez une analyse de sentiments grâce au Deep Learning



Notre mission:

**Mettre en place un prototype de modèle,
pour l'utiliser et la tester via une Appli Web.**



Les étapes d'analyse

Partie 1

- Prise en main
- Présentation des outils

Partie 2

Mise en place des modèles

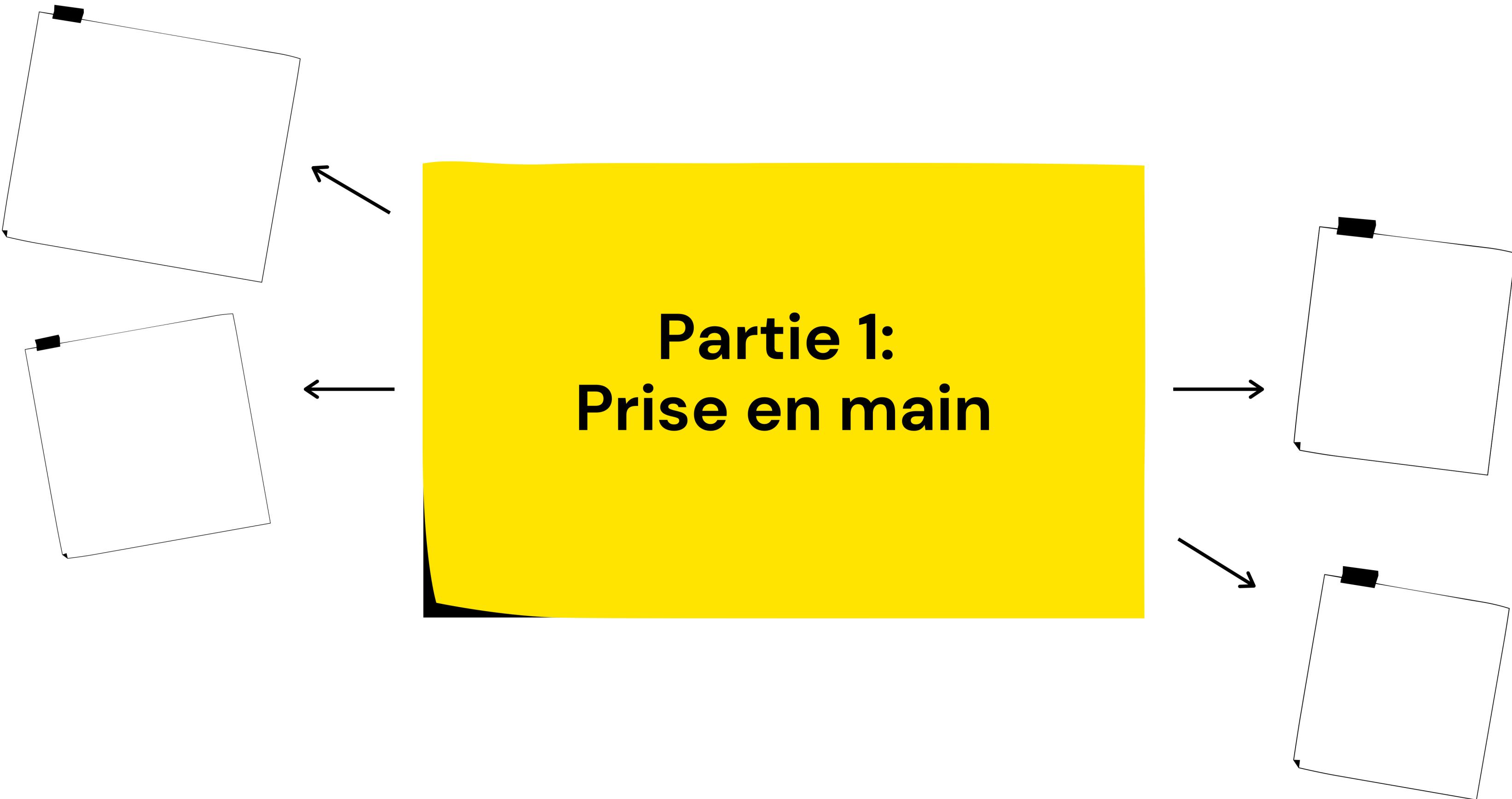
Partie 3

Déploiement et Test

Partie 4

Conclusion

Partie 1: Prise en main



Partie 1: Prise en main

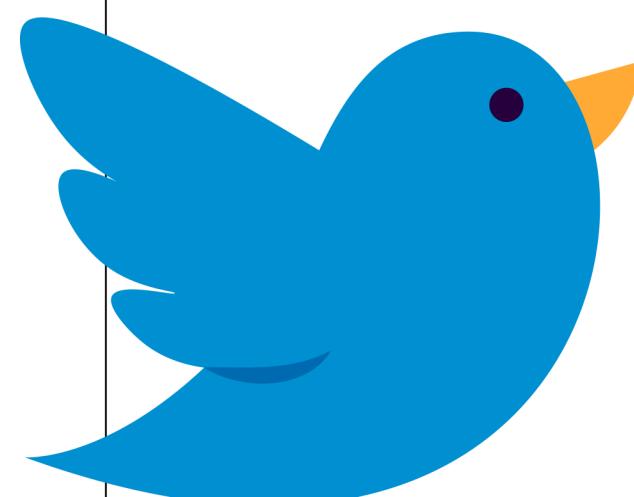
Ressources

- Jeu de données .csv

1.6 Millions de tweet

6 colonnes

Nettoyage



id	url	tweet
5768		
2478		
173		



Partie 1: Les outils

Ressources

- Entrainer les modèle sous Python
- Importer sur GitHub
- Déploiement via Azure
- Test avec PostMan



Partie 2: Mise en place des modèles



Partie 2: Mise en place des modèles

Modèle 1

Modèle simple :
Naïves Bayes

Modèle 2

Modèle avancé :
LSTM

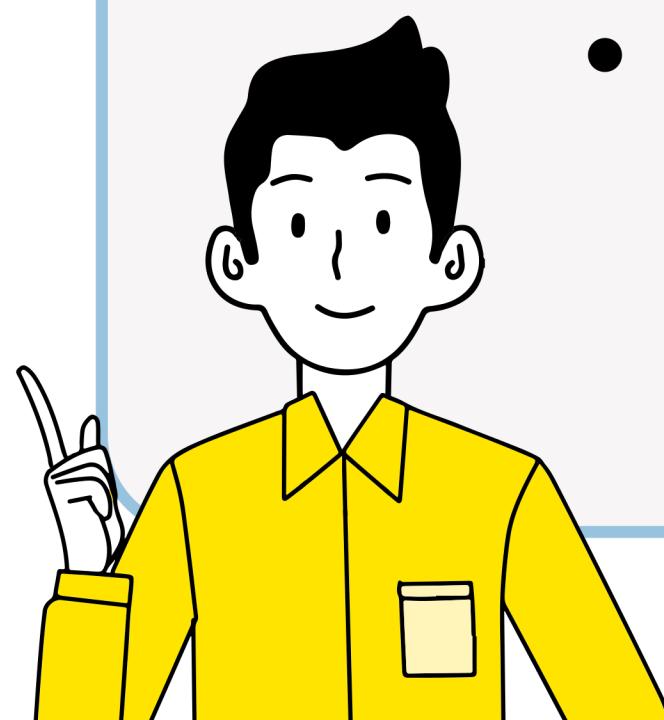
Modèle 3

BERT

Préparation Commune

De mise en place

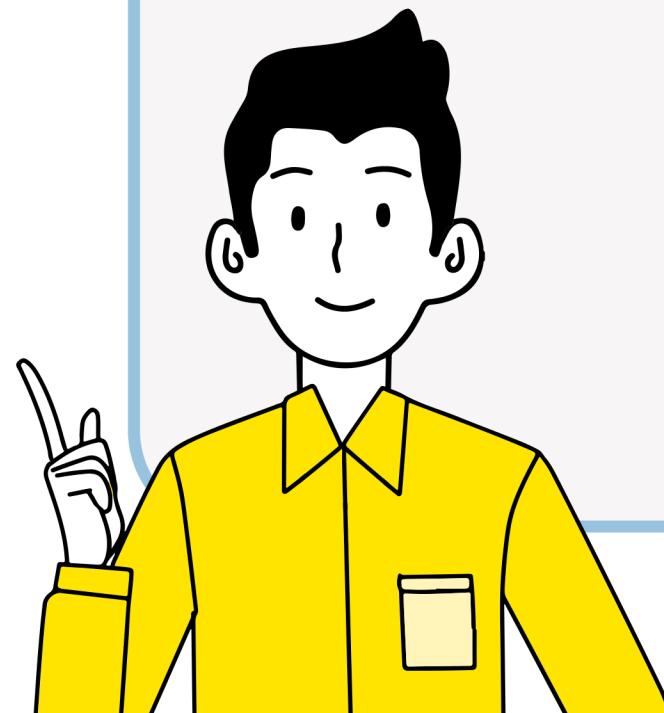
- **Nettoyage**
- **Possibilité de régler un pourcentage de donnée utilisé**
- **Tokenisation**
- **Lemmatisation**
- Chargement d'un modèle **Word2Vec**
- **Embedding** des textes avec **USE**
- **Séparation des données en:**
 - **Jeu d'entraînement (70%)**
 - **Jeu de validation (15%)**
 - **Jeu de test (15%)**



Préparation
Commune

De test

- Résultat d'accuracy
- Graphique accuracy et Loss par Epochs
(LSTM et BERT)
- Test d'une vingtaine de phrases



Partie 2: Mise en place des modèles

Modèle 1

Naïves Bayes

Force

- Approche simple, mais efficace pour le texte
- Rapidité et Facilité à prendre en main

Faiblesse

- Moins performant pour des données complexe
- Il fait difficilement le lien entre tout les éléments



Partie 2: Mise en place des modèles

Modèle 1

Naïves Bayes

On profite de la rapidité pour intégrer une GridSearch sur le paramètre Alpha

```
Best Parameters: {'alpha': 0.5}
Accuracy: 0.7020833333333333
Classification Report:
precision    recall   f1-score   support
          0       0.71      0.68      0.69      1178
          4       0.70      0.73      0.71      1222

accuracy                           0.70      2400
macro avg       0.70      0.70      0.70      2400
weighted avg    0.70      0.70      0.70      2400
```

```
# Exemple de prédiction
new_text = "I love this product!"
predicted_sentiment = predict_sentiment(new_text)
print(f"The predicted sentiment for the new text is: {predicted_sentiment}")

# Exemple de prédiction
new_text = "I hate this product!"
predicted_sentiment = predict_sentiment(new_text)
print(f"The predicted sentiment for the new text is: {predicted_sentiment}")

✓ 0.2s
The predicted sentiment for the new text is: Positif
The predicted sentiment for the new text is: Négatif
```



Partie 2: Mise en place des modèles

Modèle 1

Naïves Bayes

Visualisation du MLFlow

	Run Name	Create	Duration	Source	Models	best_alpha	vectorizer_max_i	vectorizer_max_d	vectorizer_min_c	vectorizer_ngram
	colorful-calf-408	✓	11.7s	c:\Users\...	sklearn	0.5	0.5	1000	5	(1, 2)
	amusing-loon-262	✓	10.1s	c:\Users\...	sklearn	0.5	0.5	1000	5	(1, 2)
	unique-bee-915	✓	10.2s	c:\Users\...	sklearn	0.5	0.5	1000	5	(1, 2)
	delightful-ray-449	✓	10.9s	c:\Users\...	sklearn	0.5	0.5	1000	5	(1, 2)
	puzzled-shrike-848	✓	8.1s	c:\Users\...	sklearn	0.01	0.5	1000	5	(1, 2)

Ici, on modifie essentiellement la quantité de données et pas les paramètres



Partie 2: Mise en place des modèles

Modèle 2

LSTM

Force

- Mémorise les informations
- Efficace pour analyser des sentiments (ce que l'on veut)

Faiblesse

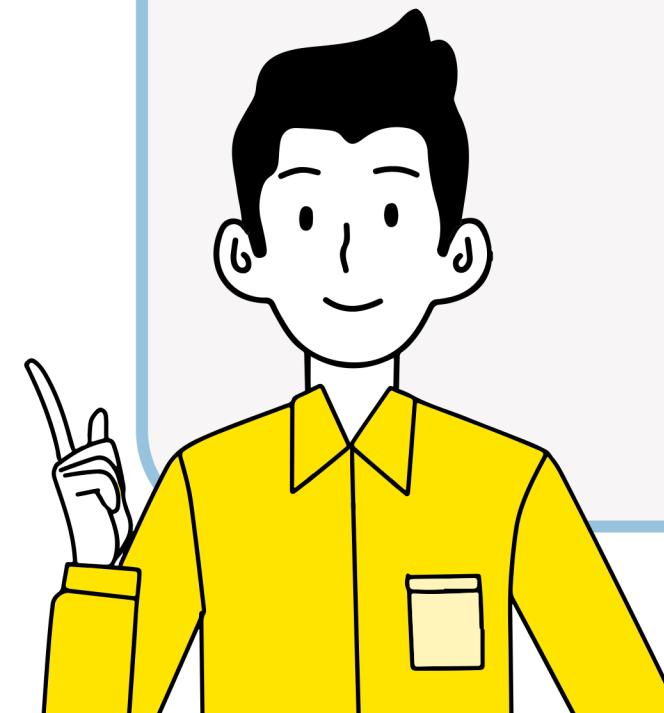
- Plus complexe et donc plus couteux
- Nécessite beaucoup de données (On en a)



**Manipulation
Spécifique**

Ajout de paramètres

- Optimiser Adam
- Régularisation L2
- Early Stopping



Partie 2: Mise en place des modèles

Modèle 2

LSTM

Visualisation du MLFlow

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

- Modification de la quantité de données
- Modification des paramètres



Partie 2: Mise en place des modèles

Modèle 2

LSTM

Pretty good overall,
just a few minor issues
here and there.

Resultat- useful-kit-393

Score: 0.0000,
Sentiment: Négatif

Resultat- adventurous- ox-22

Score: 0.0001,
Sentiment: Négatif

Resultat- salty-calf-151

Score: 0.4159,
Sentiment: Négatif

Resultat- bedecked- fox-705

Score: 0.6937,
Sentiment: Positif

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

Partie 2: Mise en place des modèles

Modèle 3

BERT

Force

- Excellente performance sur la compréhension du langage naturel
- Capable d'interpréter les nuances

Faiblesse

- Très couteux en calcul
- Complexité accrue dans le réglage et déploiement
(Notre situation)

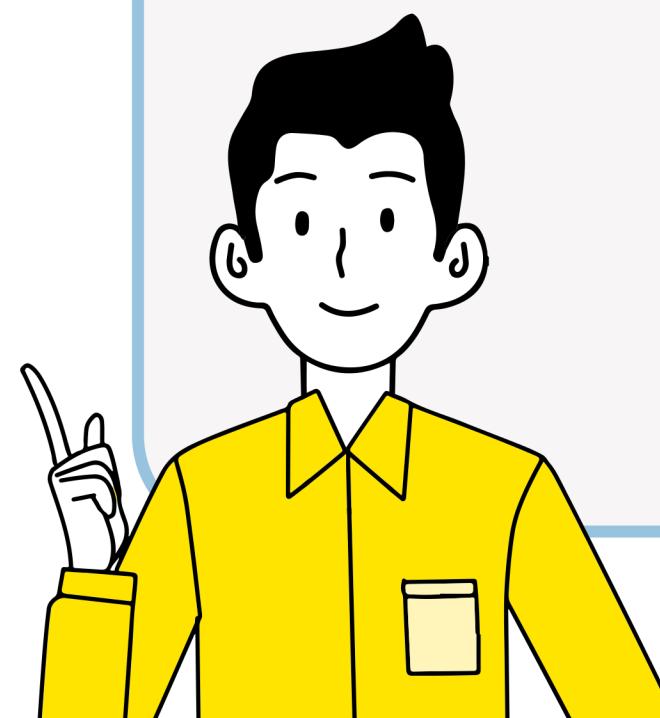


Ajout de paramètres

- modèle pré-entraîné BERT pour la classification :

Bert For Sequence Classification

- K-Fold
- input_ids
- attention_masks



Partie 2: Mise en place des modèles

Modèle 3

BERT

Visualisation du MLFlow

	Run Name	Create	Duration	Source	Models
	intelligent-eel-228	⌚		⚡ c:\Users\...	-
	capable-auk-759	🕒	7.0h	⚡ c:\Users\...	pytorch

Pour comparaison : le % du jeu de données utilisé

Ici pour BERT : 0,05% des données disponibles

Pour LSTM : 2% des données disponibles



Partie 2: Mise en place des modèles

Modèle 3

BERT

```
# Exemple d'utilisation
tweet = "I Love using BERT for natural language processing!"
prediction = predict_sentiment(tweet)
print(f"Sentiment: {prediction}")

# Exemple d'utilisation 2
tweet = "I hate using BERT for natural language processing!"
prediction = predict_sentiment(tweet)
print(f"Sentiment: {prediction}")
```

Sentiment: Positive
Sentiment: Negative

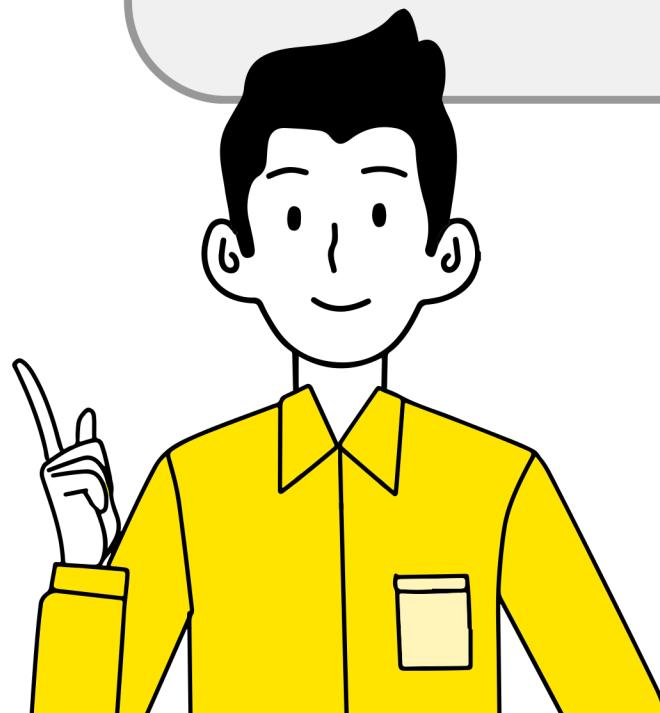


Modèle Paramètre	Naïves Bayes	LSTM	BERT
Temps d'entraînement	149 min de préparation 1min30 d'entraînement	2,9h	3h30min
Pourcentage de données utilisées	20%	2%	0,025%
Accuracy	TA : 0.74 VA : 0.73 alpha: 0.01	TA : 0.80 VA : 0.75	TA : 0.68 VA : 0.30
<u>Efficacité sur test:</u> Pretty good overall, just 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



Partie 3: Déploiement et Test.

Nous devons suivre plusieurs étapes une fois notre modèle et paramètres sélectionnés.



Partie 3: Déploiement et Test.

Etape 1

Finaliser le code

Intégration de Flask

```
app = Flask(__name__)

# Variables globales pour le modèle et un indicateur de disponibilité
model = None
model_ready = False

# Fonction pour charger le modèle
def load_model():
    global model, model_ready
    try:
        print("Starting model load in a separate thread...")
        # Chargez votre modèle ici
        # ...
        model = ...  # Remplacez par votre modèle
        model_ready = True
    except Exception as e:
        print(f"Error loading model: {e}")

# Route pour obtenir l'état du modèle
@app.route('/status')
def status():
    return {"model_ready": model_ready}
```

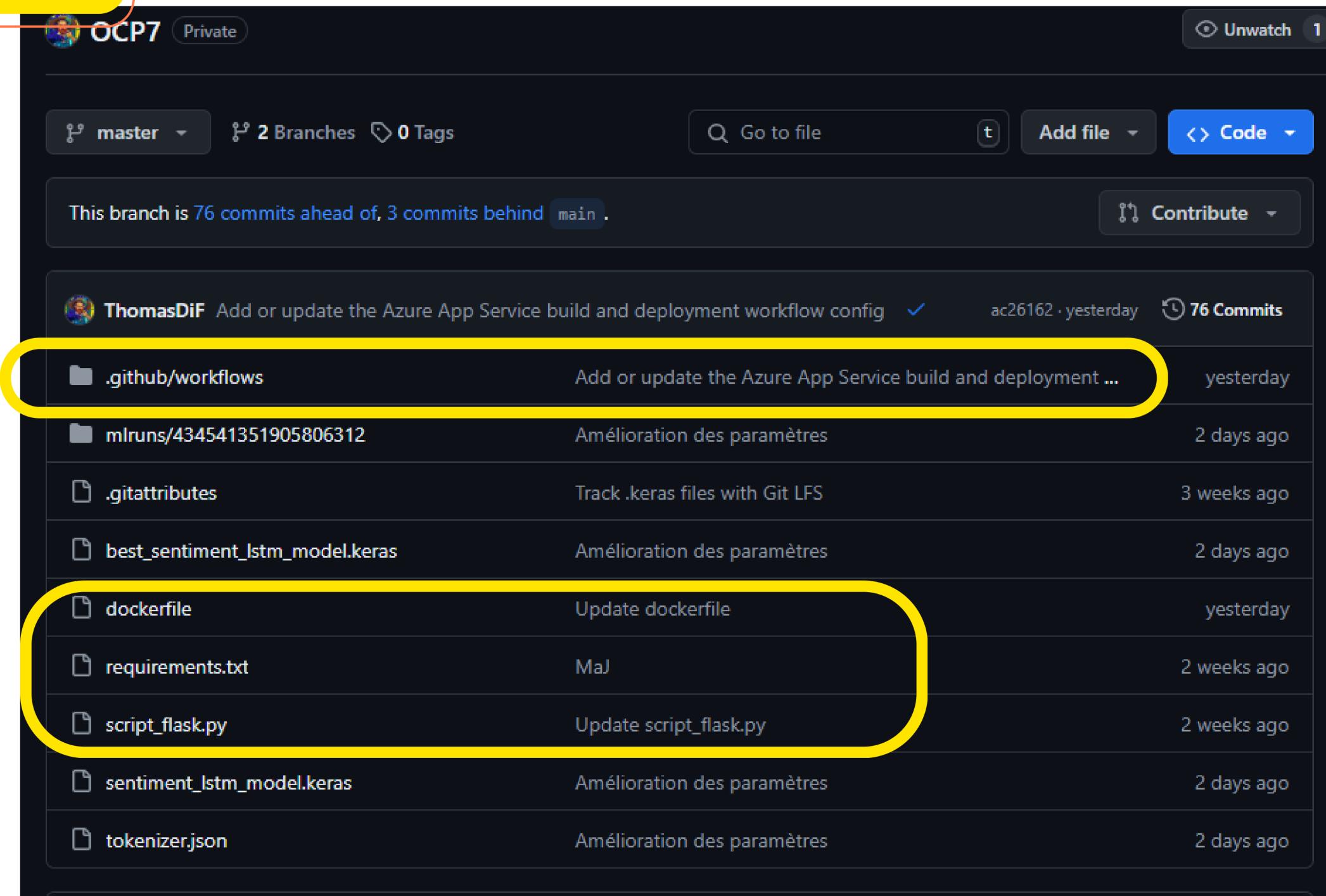
Flask est un outil puissant pour développer des applications web en Python tout en offrant une grande flexibilité et simplicité.



Partie 3: Déploiement et Test.

Etape 2

On “Push” le code vers GitHub



The screenshot shows a GitHub repository named "OCP7" with a "Private" status. The "master" branch is selected, showing 2 branches and 0 tags. A message indicates the branch is 76 commits ahead of "main". The commit list includes:

- .github/workflows**: Add or update the Azure App Service build and deployment workflow config (yesterday)
- mlruns/434541351905806312**: Amélioration des paramètres (2 days ago)
- .gitattributes**: Track .keras files with Git LFS (3 weeks ago)
- best_sentiment_lstm_model.keras**: Amélioration des paramètres (2 days ago)
- dockerfile**: Update dockerfile (yesterday)
- requirements.txt**: Maj (2 weeks ago)
- script_flask.py**: Update script_flask.py (2 weeks ago)
- sentiment_lstm_model.keras**: Amélioration des paramètres (2 days ago)
- tokenizer.json**: Amélioration des paramètres (2 days ago)



Partie 3: Déploiement et Test.

Etape 3

On construit et on déploie

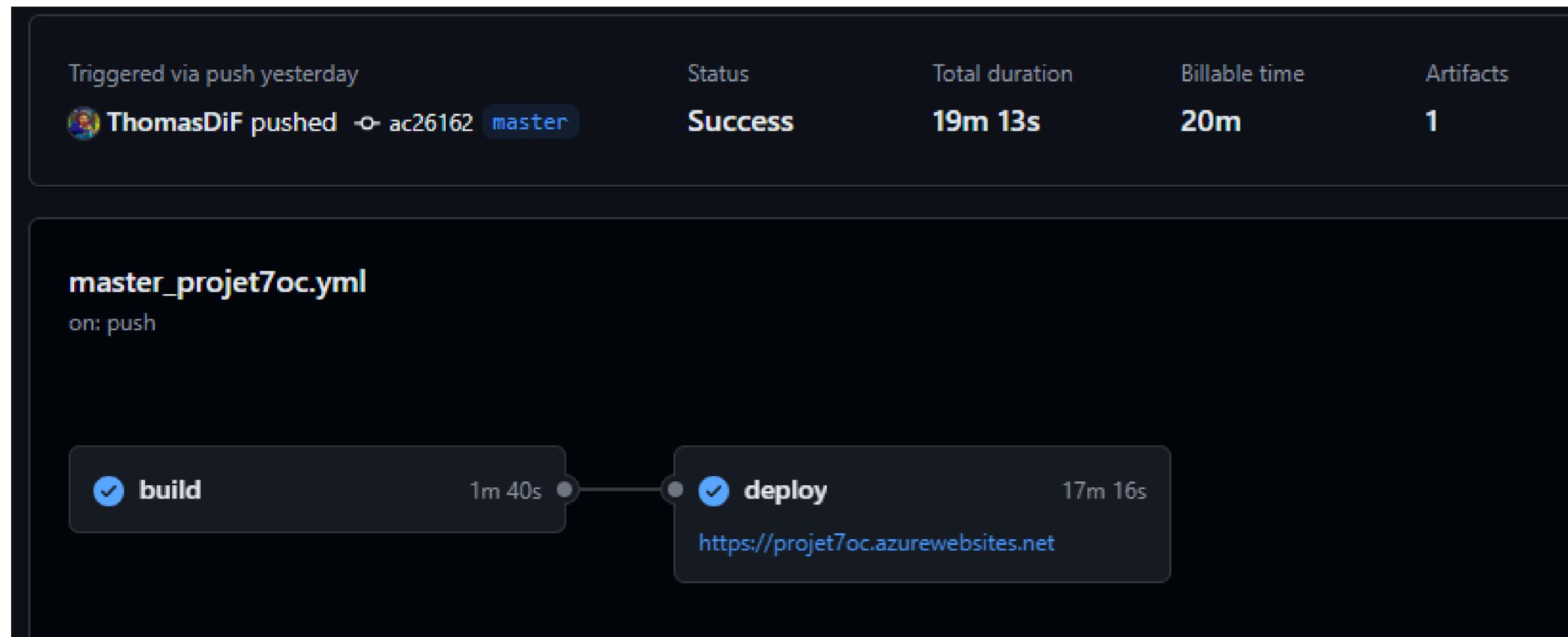
Triggered via push yesterday
ThomasDiF pushed -o ac26162 master

Status	Total duration	Billable time	Artifacts
Success	19m 13s	20m	1

master_projet7oc.yml
on: push

```
build --> deploy https://projet7oc.azurewebsites.net
```

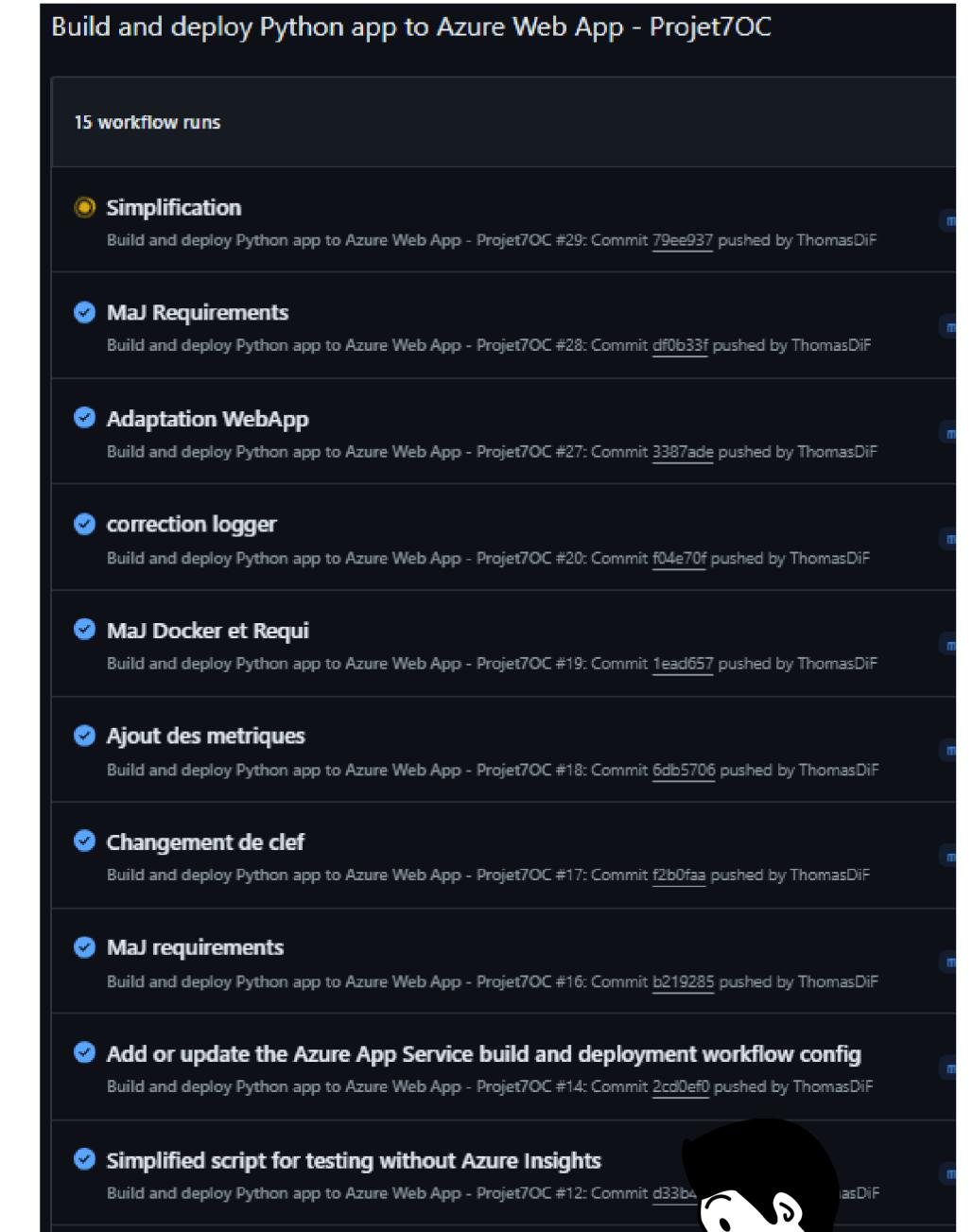
1m 40s



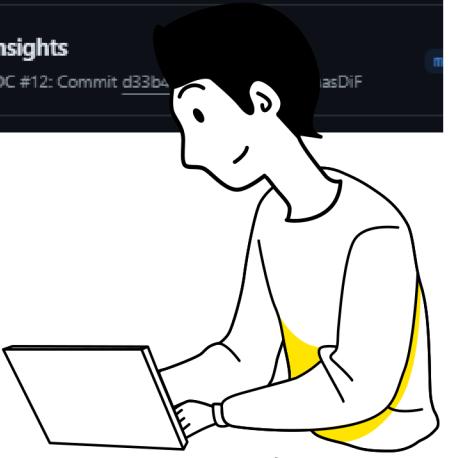
Build and deploy Python app to Azure Web App - Projet7OC

15 workflow runs

- Simplification**
Build and deploy Python app to Azure Web App - Projet7OC #29: Commit 79ee937 pushed by ThomasDiF
- MAJ Requirements**
Build and deploy Python app to Azure Web App - Projet7OC #28: Commit df0b33f pushed by ThomasDiF
- Adaptation WebApp**
Build and deploy Python app to Azure Web App - Projet7OC #27: Commit 3387ade pushed by ThomasDiF
- correction logger**
Build and deploy Python app to Azure Web App - Projet7OC #20: Commit f04e70f pushed by ThomasDiF
- MAJ Docker et Requi**
Build and deploy Python app to Azure Web App - Projet7OC #19: Commit 1ead657 pushed by ThomasDiF
- Ajout des metriques**
Build and deploy Python app to Azure Web App - Projet7OC #18: Commit 6db5706 pushed by ThomasDiF
- Changement de clef**
Build and deploy Python app to Azure Web App - Projet7OC #17: Commit f2b0faa pushed by ThomasDiF
- MAJ requirements**
Build and deploy Python app to Azure Web App - Projet7OC #16: Commit b219285 pushed by ThomasDiF
- 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
- Simplified script for testing without Azure Insights**
Build and deploy Python app to Azure Web App - Projet7OC #12: Commit d33b4...asDiF



Sur cette étape, plusieurs erreurs peuvent être mises en lumière et nécessitent des ajustements.



Partie 3: Déploiement et Test.

Etape 3

En parallèle sur Azure

Paramètres Journaux Informations d'identification FTPS

Déployez et générez du code à partir de votre fournisseur de build et source par défaut

Source GitHub [Se déconnecter](#)

GitHub

Connecté comme	ThomasDiF
Organisation	ThomasDiF
Dépot	OCP7
Branche	master master

Build

Fournisseur de build	GitHub Actions
Pile d'exécution	Python
Version	Python 3.12

master_projet7oc.yml

Commande de démarrage

```
gunicorn --bind 0.0.0.0:8000  
script_flaskapp --workers 3 --timeout  
1800 --graceful-timeout 1800 --threads  
4
```



Partie 3: Déploiement et Test.

Etape 4

Validation du déploiement et test de stabilités

Centre de déploiement

Journaux de déploiement Afficher les journaux

Dernier déploiement ✓ Réussi le jeudi 27 juin, 01:55:52 PM Actualiser

Fournisseur de déploiement GitHubAction

 POSTMAN

https://projet7oc.azurewebsites.net/predict

POST https://projet7oc.azurewebsites.net/predict



Projet7OC | Flux de journaux

Application web

Regles d'alerte

+ Créer Colonnes Actualiser Exporter au format CSV Ouvrir une requête Supprimer Activer

Rechercher Abonnement : Azure subscription 1 Type de ressource cible : tous Étendue cible

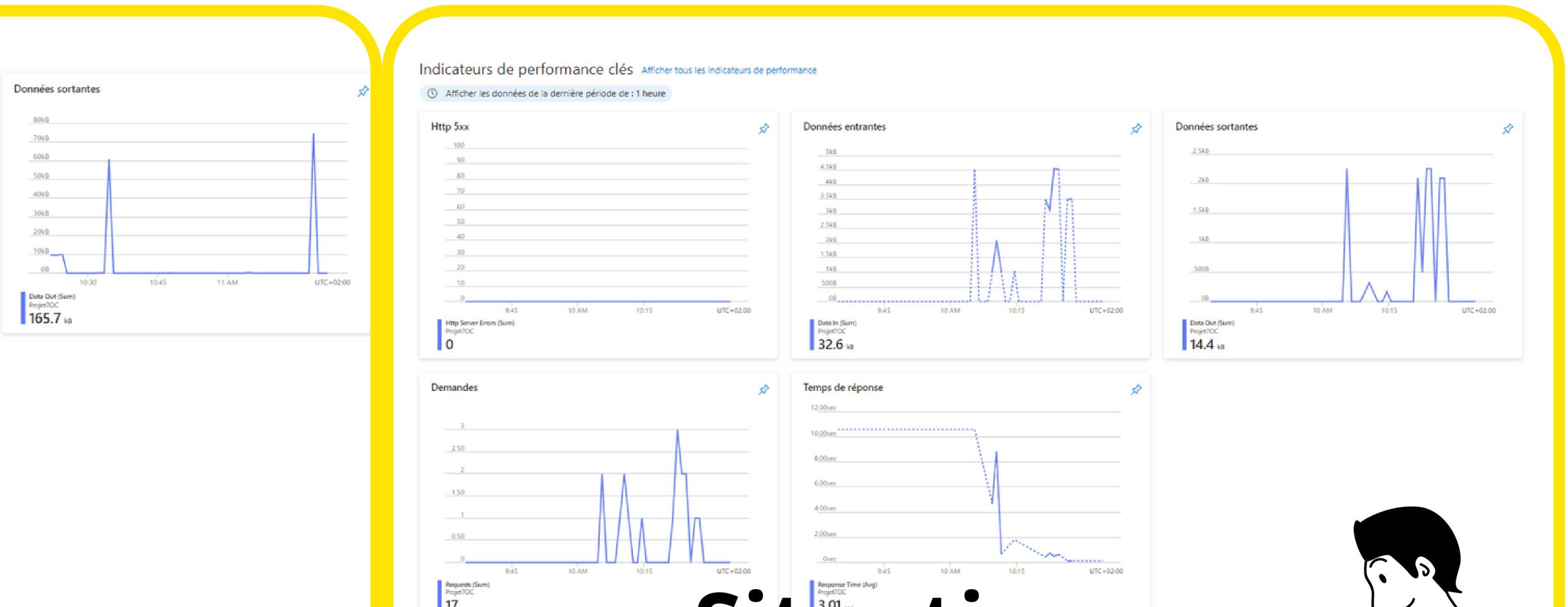
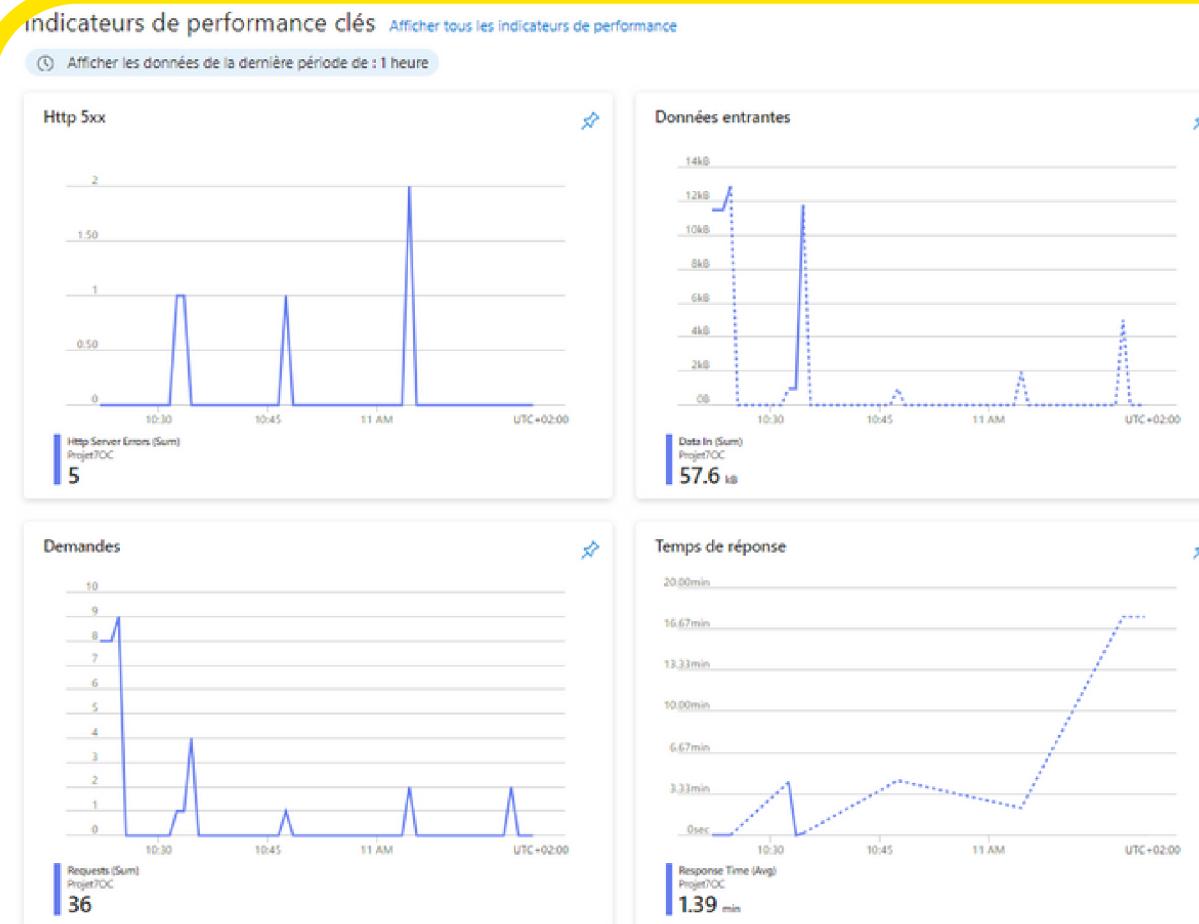
Nom ↑	Condition	Gravité ↑
<input type="checkbox"/> Erreur 4XX	Http4xx > 2	1 - Erreur
<input type="checkbox"/> Temps de réponse trop long	HttpResponseTime > seuil dynamique	2 - Avertissement



Partie 3: Déploiement et Test.

Etape 4

Validation du déploiement et test de stabilités



CPU trop faible

Situation fonctionnelle



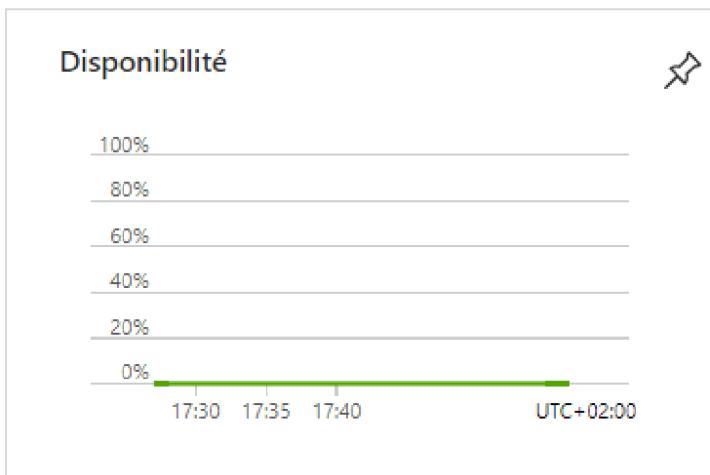
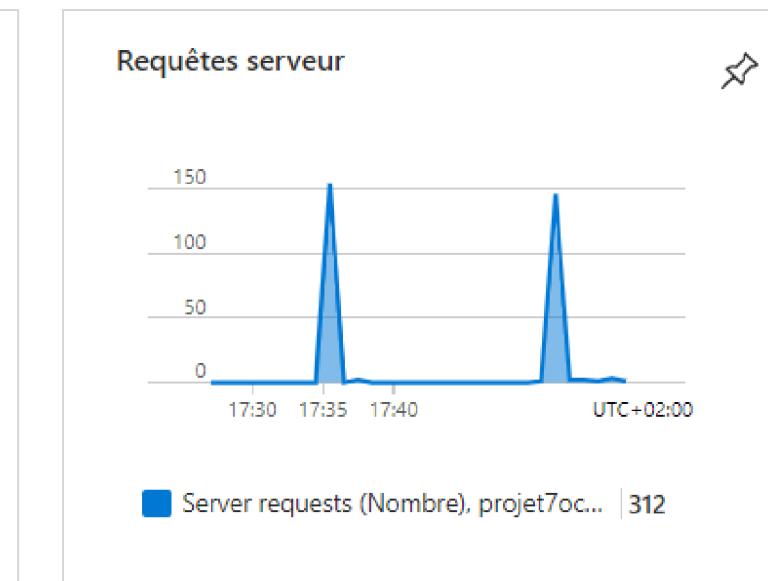
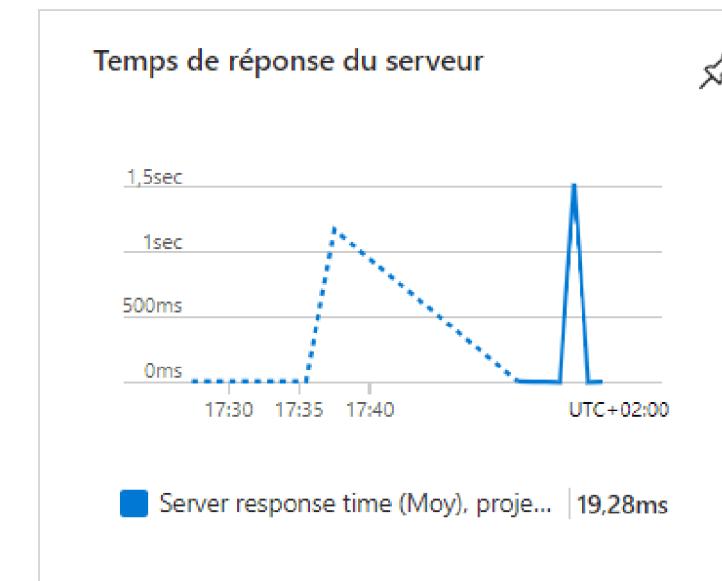
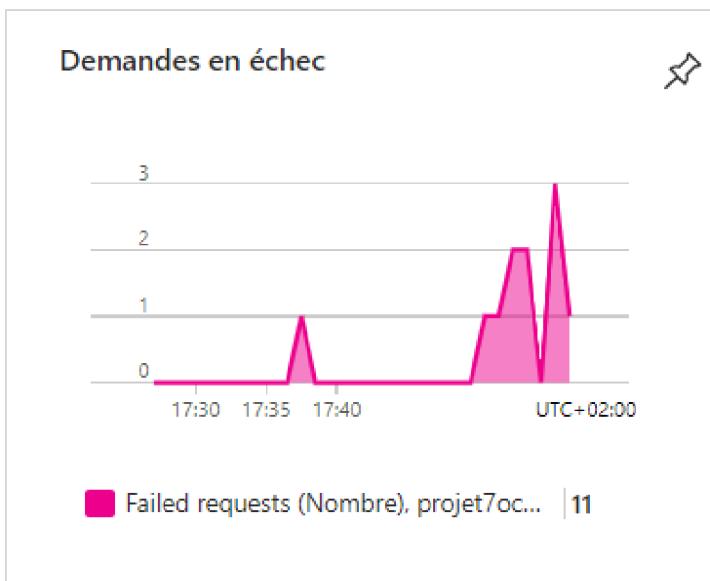
Partie 3: Déploiement et Test.

Etape 4

Validation du déploiement et test de stabilités

Afficher les données du/de la dernier(ère) :

(30 minutes) 1 heure 6 heures 12 heures 1 jour 3 jours 7 jours 30 jours



Partie 3: Déploiement et Test.

Etape 5

Observation sur PostMan

```
1  {
2    "text": "It's fine, nothing too special but not too bad either."
3  }
4
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON ↗

```
1  {
2    "score": 0.00010553881293162704,
3    "sentiment": "Négatif"
4  }
```



Partie 4: Conclusion

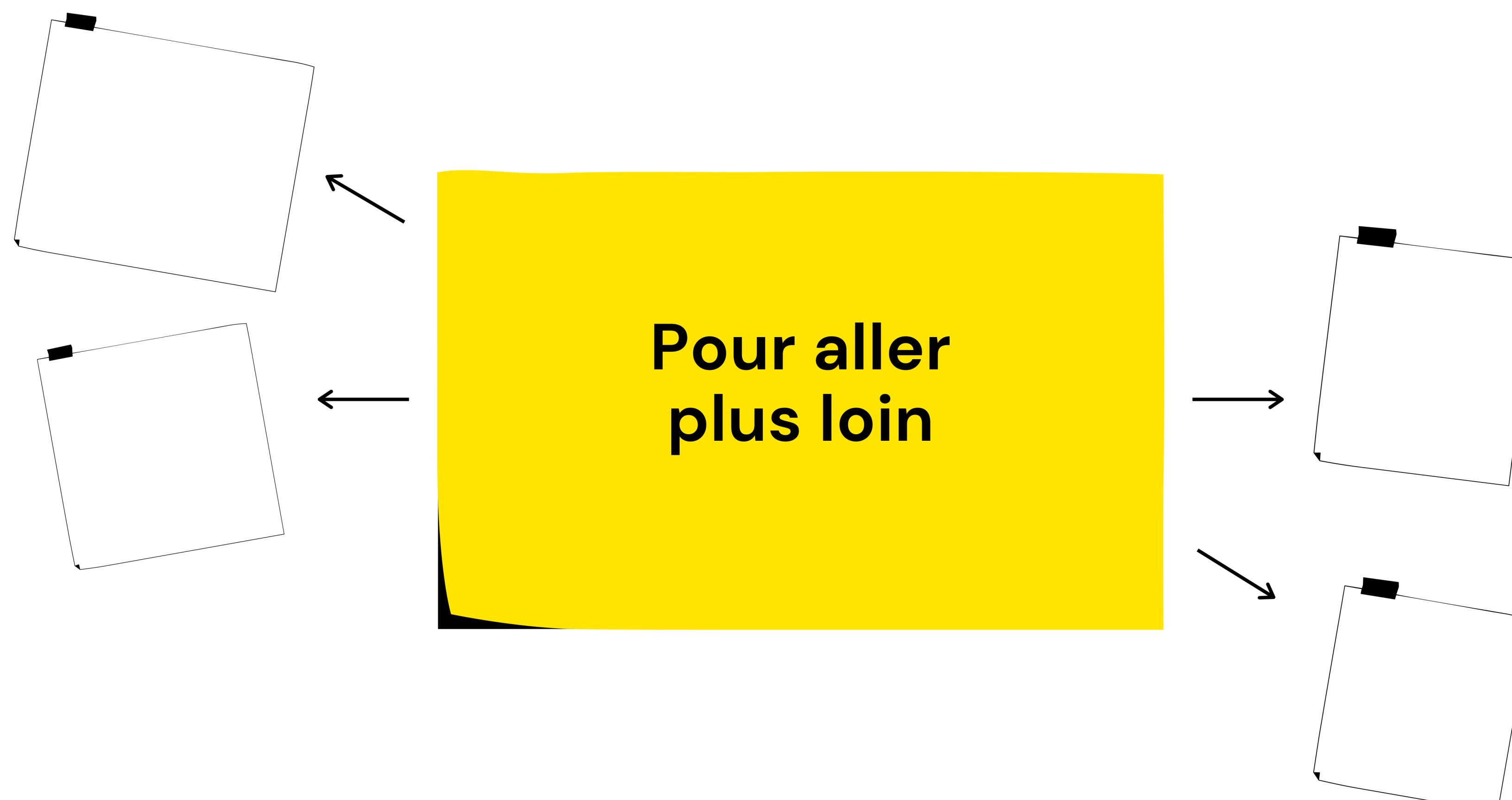
Partie 4 Conclusion

LSTM est un modèle adapté à notre projet

**Trouver un équilibre entre les paramètres et
la quantité de données**

Adapter les outils de suivis à nos besoins





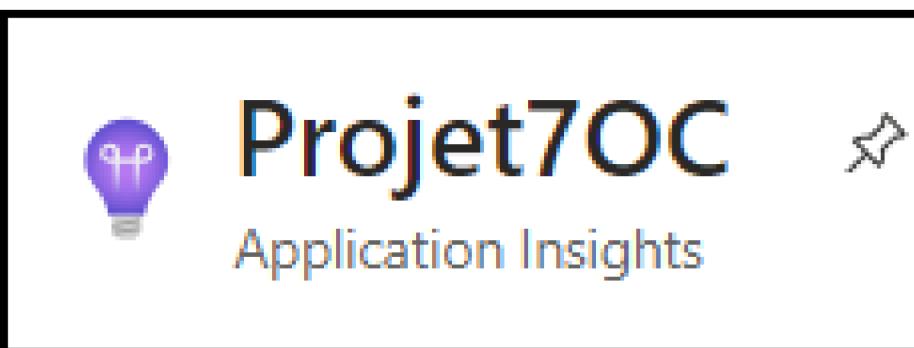
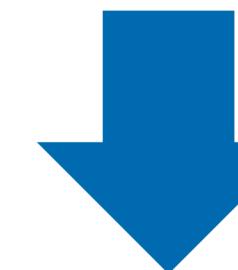
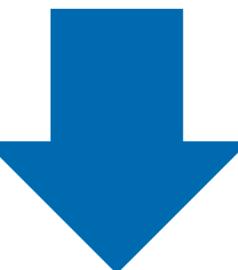
**Pour aller
plus loin**

Projection

Pour aller plus loin

```
1 {  
2   "text": "Really satisfied with this. It's just what I needed.",  
3   "true_label": 1  
4 }  
5
```

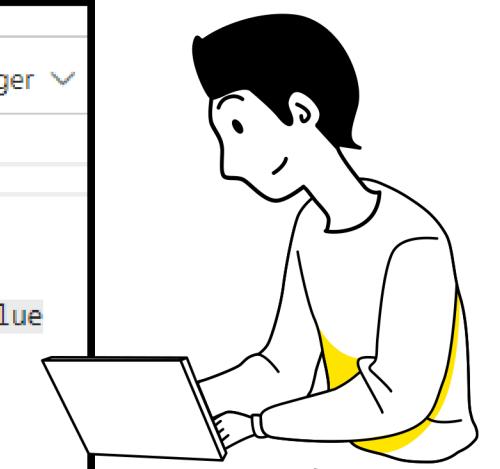
On ajoutera "true_label" pour calculer de nouvelles métriques



Exemple de code pouvant observer ces métriques



```
1 AppTraces  
2 | where customDimensions.metricName == "precision"  
3 | or customDimensions.metricName == "recall"  
4 | or customDimensions.metricName == "f1_score"  
5 | or customDimensions.metricName == "latency"  
6 | project timestamp, customDimensions.metricName, customDimensions.value  
7 | order by timestamp  
8
```



Merci!

