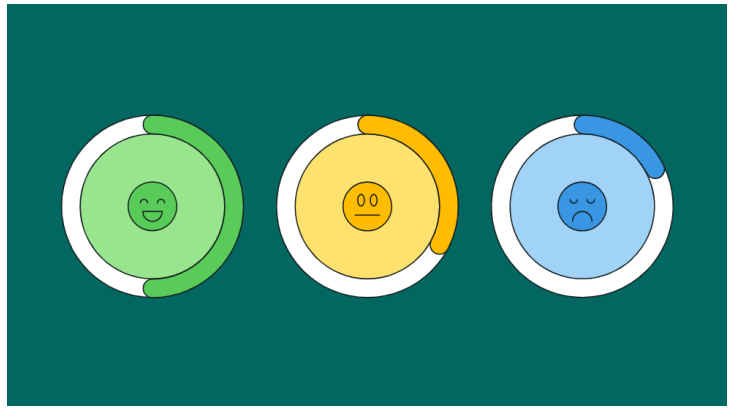
Détecter les Bad Buzz grâce au Deep Learning

Sommaire

PARTIE 1 – INTRODUCTION

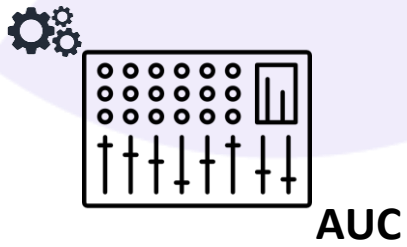
Le projet de l’entreprise

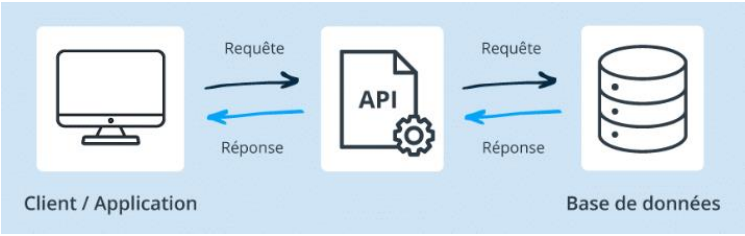
- Produit IA pour anticiper les bad buzz sur les réseaux sociaux
  - ➡️ enjeu court terme de communication en cas de crise
  - ➡️ enjeu long terme pour comprendre comment la marque est perçue en ligne
  - ➡️ prendre des décisions basées sur les données pour améliorer l’image de la marque



Notre réponse

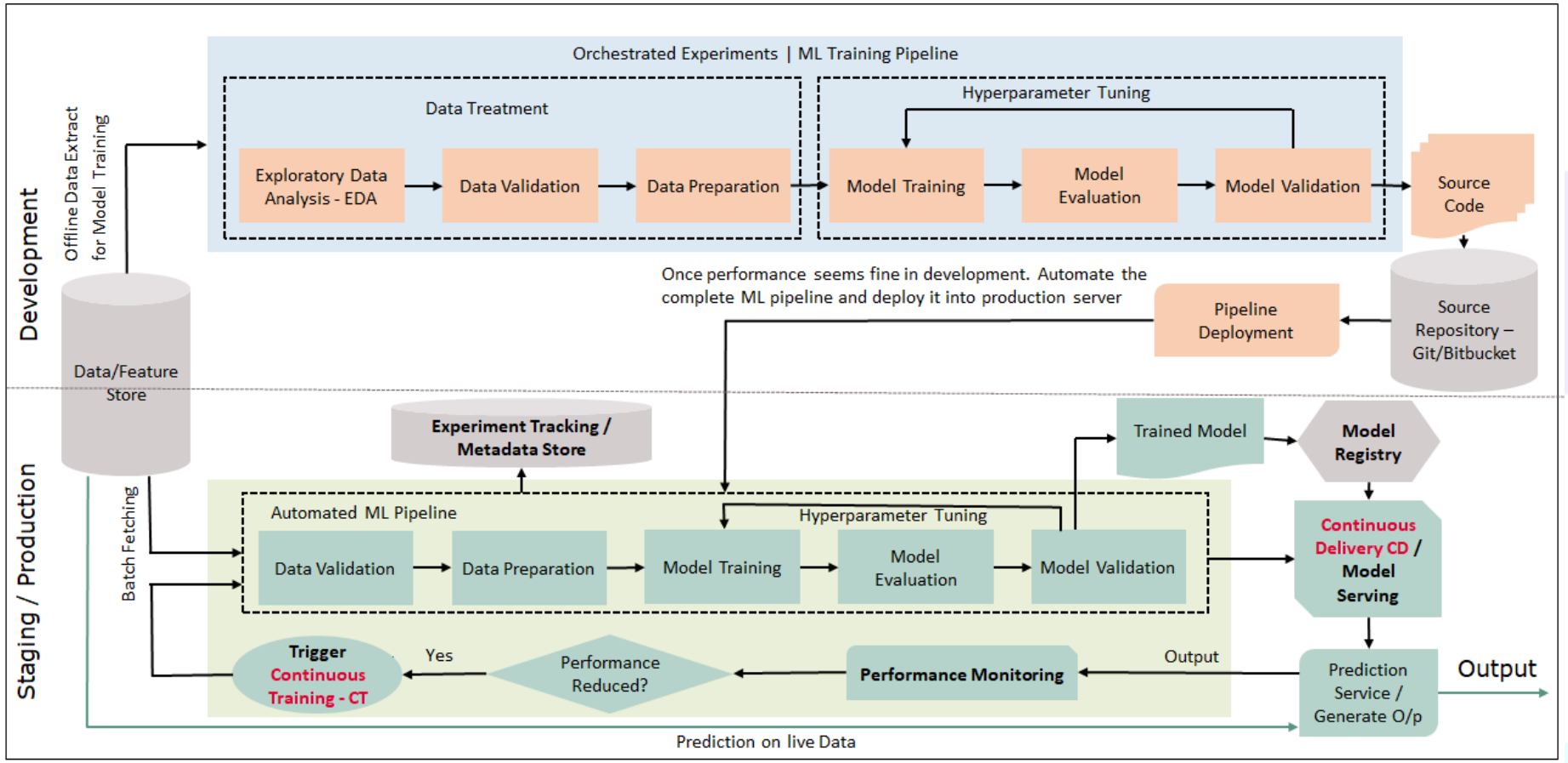
- Tests de plusieurs approches de modélisation :
  - Modèle sur mesure simple, régression logistique
  - Modèle sur mesure avancé, LSTM bidirectionnel
- Optimisation des modèles :
  - Méthodes de nettoyage
  - Méthodes de normalisation
  - Méthodes de plongement de mots (word embedding)
  - Hyperparamètres du modèle lui-même
- Création d'une API
- Création d'une UI





Notre réponse

- MLOps :



Notre réponse

- MLOps pour ce projet :
  - mlflow pour :
    - tracker les **hyperparamètres** de nos différents modèles
    - tracker les **métriques**
    - **enregistrer les modèles** que nous souhaitons conserver
    - **charger le modèle** gagnant et faire des prédictions
    - tester le service de *serving*
  - GitHub pour l'**intégration continue** du code : https://github.com/ClementPatin/OC-P7
  - docker pour créer des **conteneurs** exécutables pour notre **API** et notre **UI**
  - Azure WebApp Services pour le **déploiement**
  - GitHub Actions pour le déploiement **continue**

PARTIE 2 – EXPLORATION et PREPROCESSING

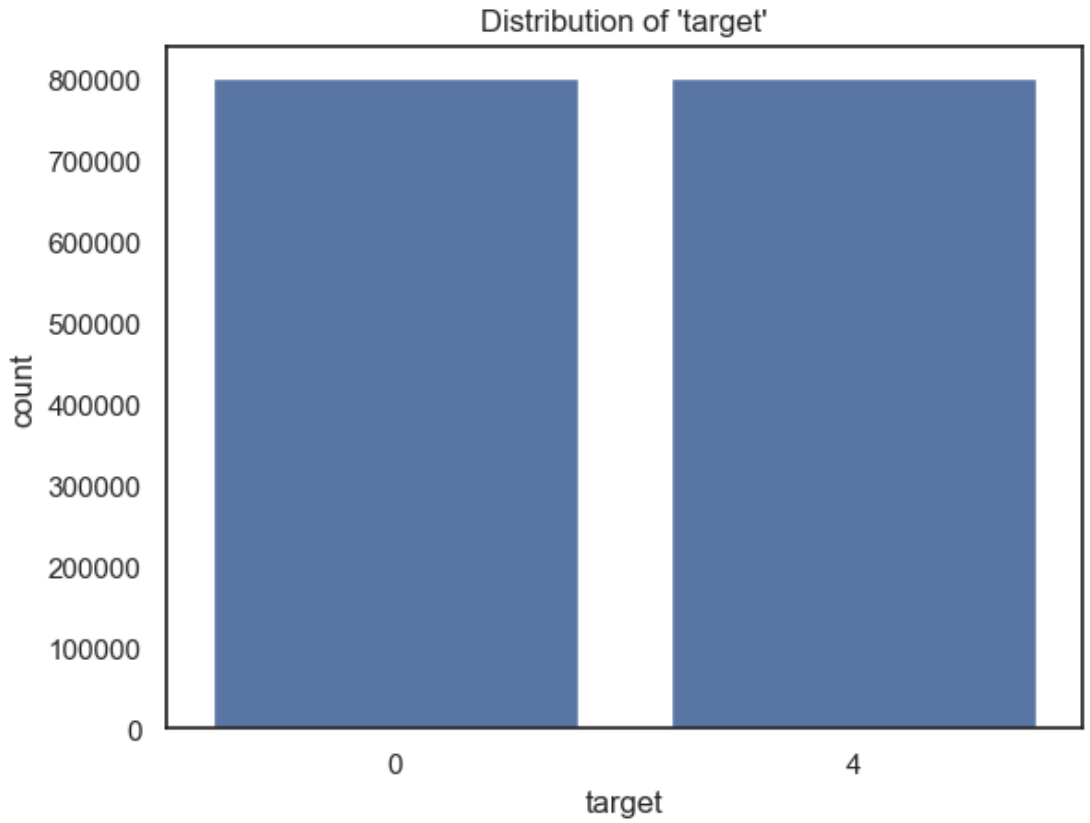
EDA & nettoyage

Les Doublons

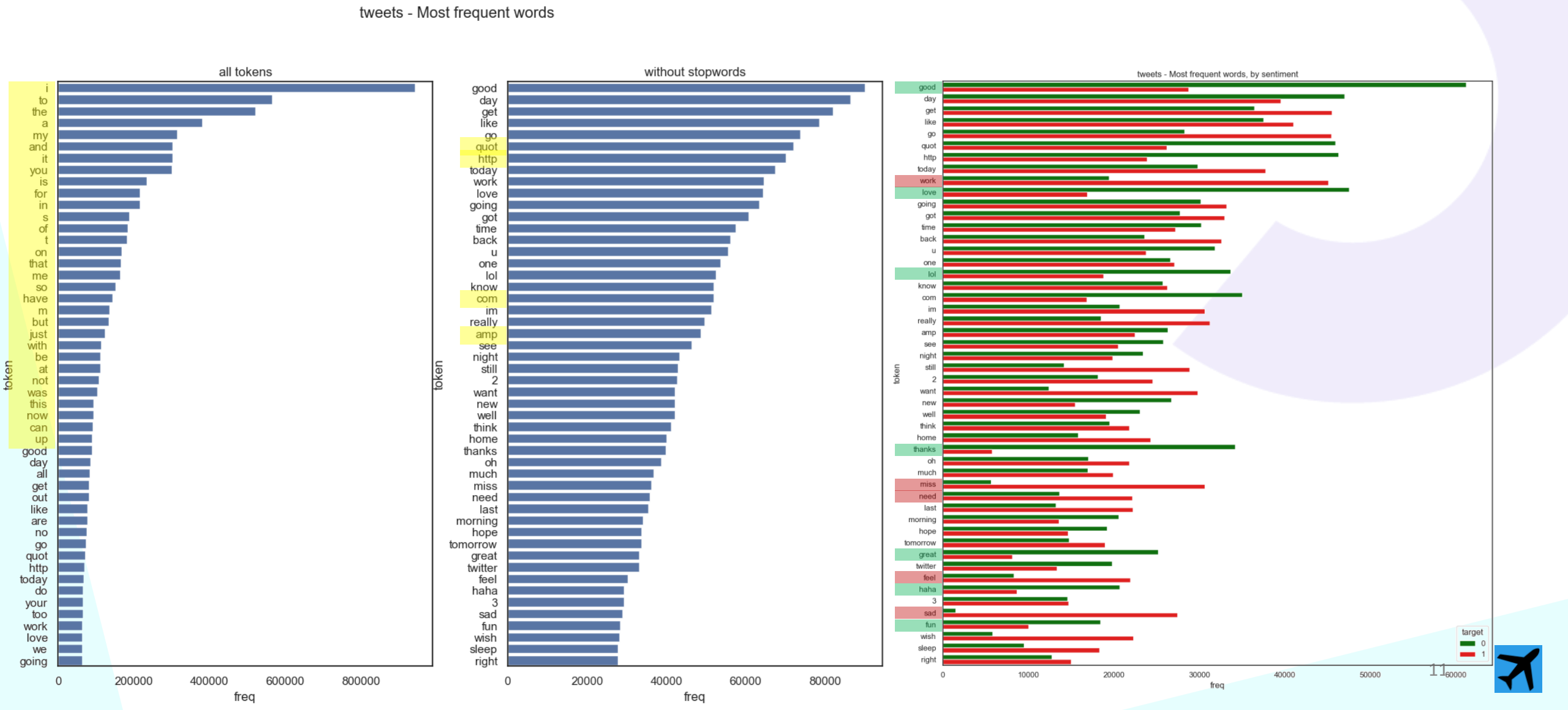
	target	text
271472	0	David must be hospitalized for five days end...
271505	0	David must be hospitalized for five days end...
1132076	4	bathroom is clean..... now on to more enjoya...
1132074	4	bathroom is clean..... now on to more enjoya...
540159	0	#IMISSCATH #IMISSCATH #IMISSCATH #IMISSCATH #...
...	...	...
458170	0	zofran is not making this work day any easier,...
464810	0	~ get SUICIDE !!
463926	0	~ get SUICIDE !!
1170868	4	~~~X0X0~~~!!!X0X0!!! I love you!!! &...
238478	0	~~~X0X0~~~!!!X0X0!!! I love you!!! &...

26968 rows × 2 columns

La target



EDA – les mots les plus fréquents



EDA & nettoyage

ASCII non « imprimables »

	target	text	NEW_len_text
1405400	0	shagÃ¼er	9
1459484	0	HallÃ¶le	9
638043	1	workÃ±ng	9
1211495	0	I â¥ you	11
1304341	0	Rock RoÃ§a	11
...	...	...	...
650945	1	human shield à®□à®©à¯?à®± à®□à¯□à®±à¯?à®¤à¯□à®...	348
380539	1	@neospace à¹□à¸£à¸·à¹□à¸à¸□à¸□à¸±à¹□à¸□à¸□à¸£...	359
1582941	0	5 days till new top gear î?□î?□î?□î?□î?□î?□î?...	359
1484882	0	@iannnnn à¹□à¸□à¸µà¹□à¸¢à¸§à¸à¸µà¸?à¸□à¸±à¸?à...	369
1295281	0	@catfish_ohm à¹□à¸□à¹□à¸²à¹□à¸«à¹□à¸?à¸²à¸£à¹□...	374

Les urls

EDA & nettoyage

Les adresses mail

Les caractères répétés

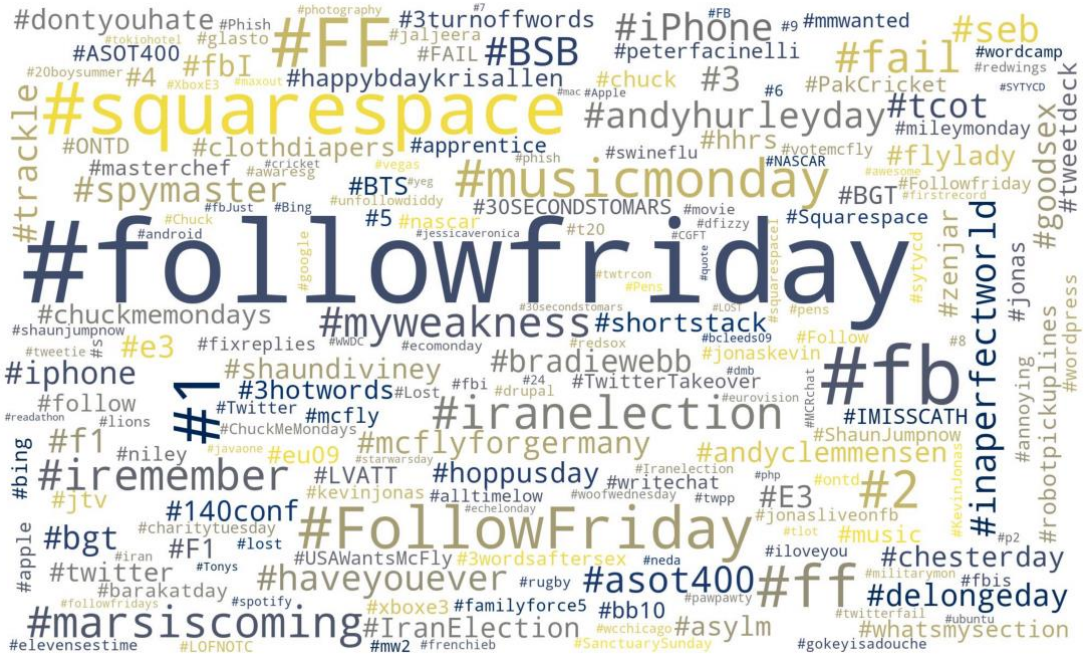
EDA & nettoyage

Les contractions

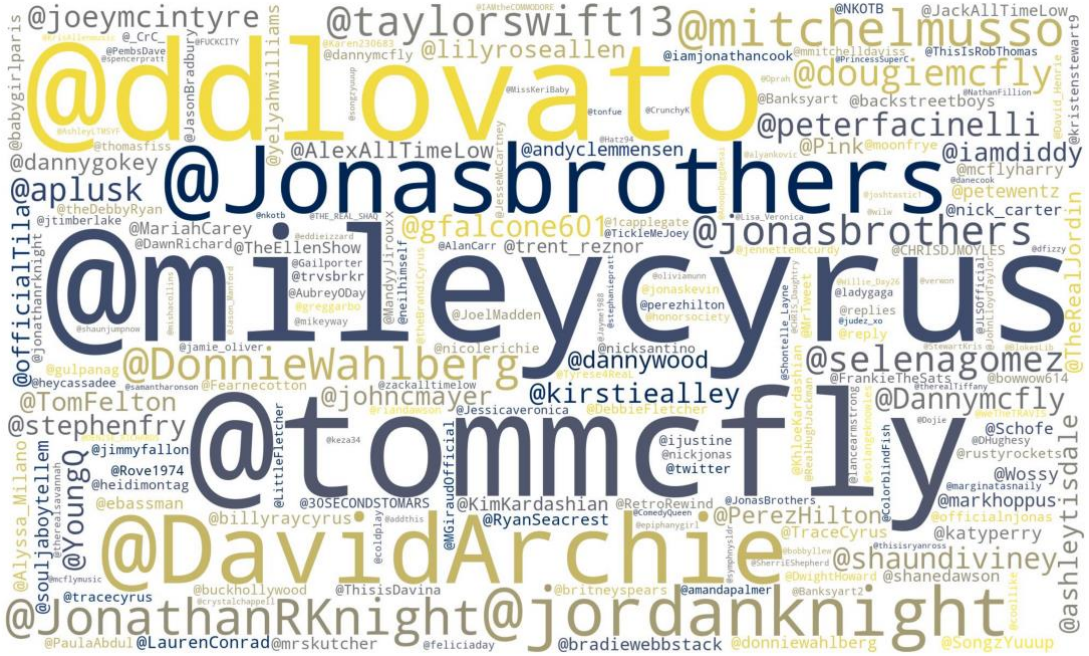
Les emoticons

EDA & nettoyage

Les hashtags



Les mentions



Nettoyage personnalisé

Modèle simple

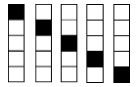
- Les étapes précédentes
- Stopwords
- Ponctuations
- Espaces
- Nombres
- POS
- Normalisation via stem ou lem

Modèle avancé

- Les étapes précédentes
- ~~Stopwords~~
- Ponctuations
- Espaces
- ~~Nombres~~
- ~~POS~~
- Normalisation via stem ou lem

Séparation des données

Modèle simple

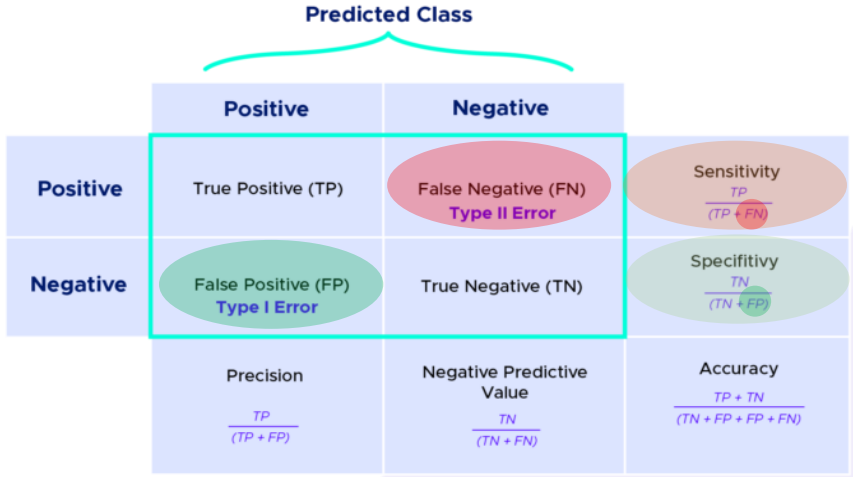
- Split train / test
  - Predictions
  - Interprétabilité
  - Hyperparamètres
  - Capacité à généraliser
  - Choix modèle
- Validation croisée 

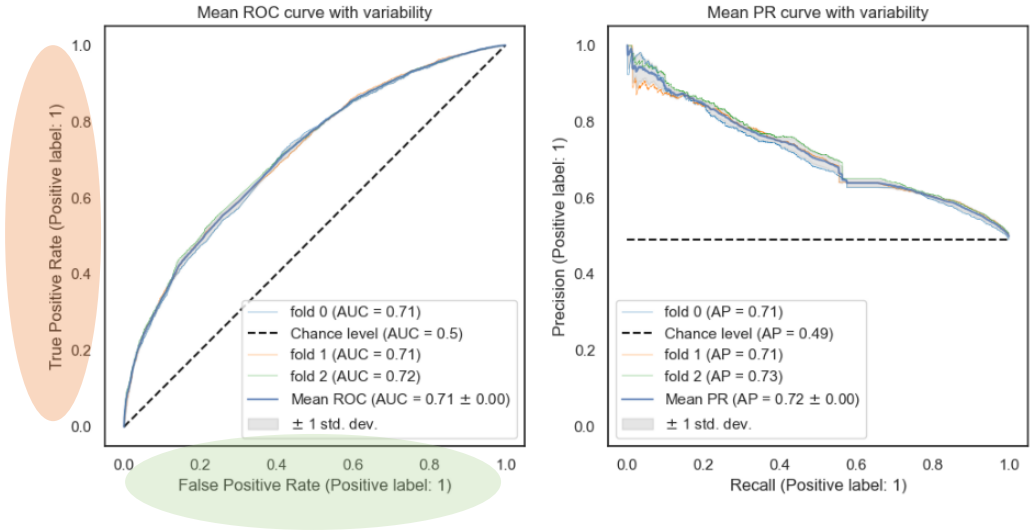
Modèle avancé

- Split train / test
  - Predictions
  - Interprétabilité
  - Hyperparamètres
  - Capacité à généraliser
  - Choix modèle
- Split train / validation

Métrique d'évaluation

- Dataset **équilibré**
- Minimiser les FN → Maximiser le Recall
- Minimiser les FP → Minimiser le FPR
- Choix : ROC AUC





ROC AUC (+accuracy + training time)

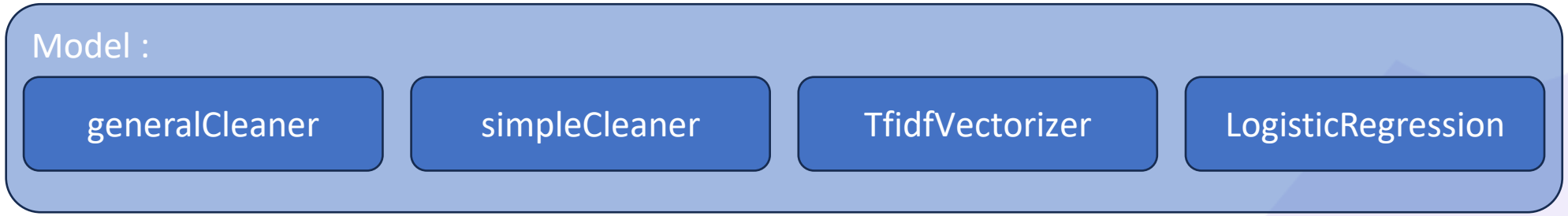
⬇️

mlflow

PARTIE 3 – MODÈLE SIMPLE

Modèle simple - pipeline

- Pipeline d'apprentissage dans une expérimentation mlflow :
  - Créer :



params ➡ mlflow

  - Validation croisée : 

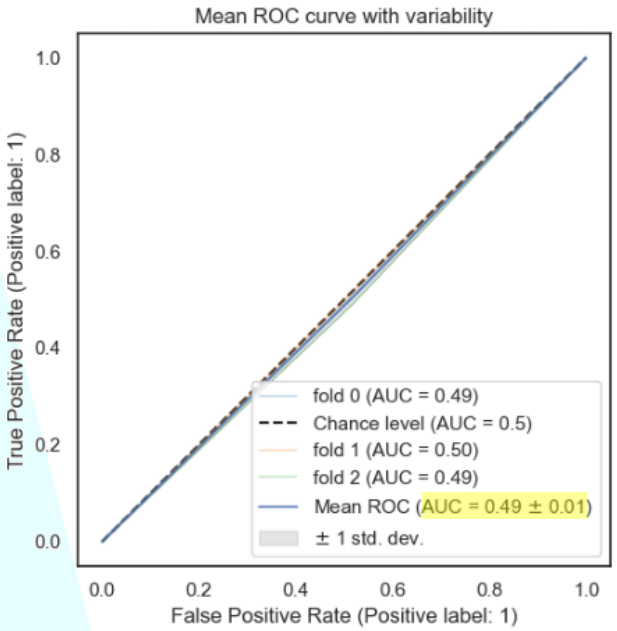
AUC, Acc, plots ➡ mlflow

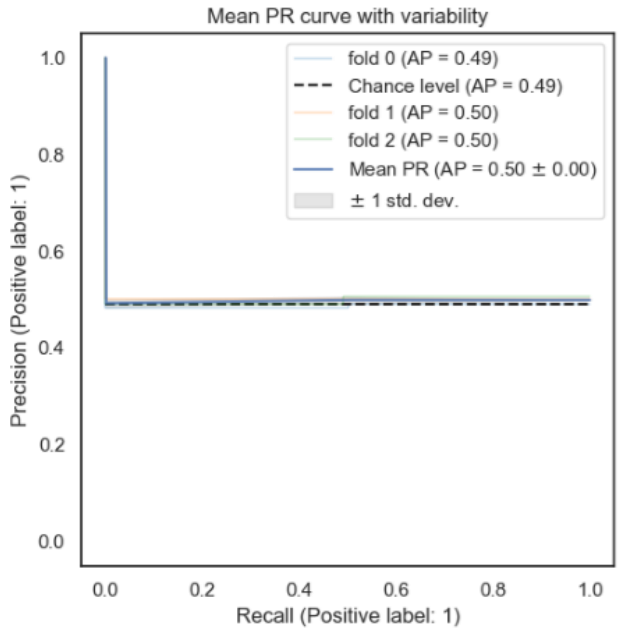
  - Ré-entraîner sur toutes les données :

Model, training_time ➡ mlflow

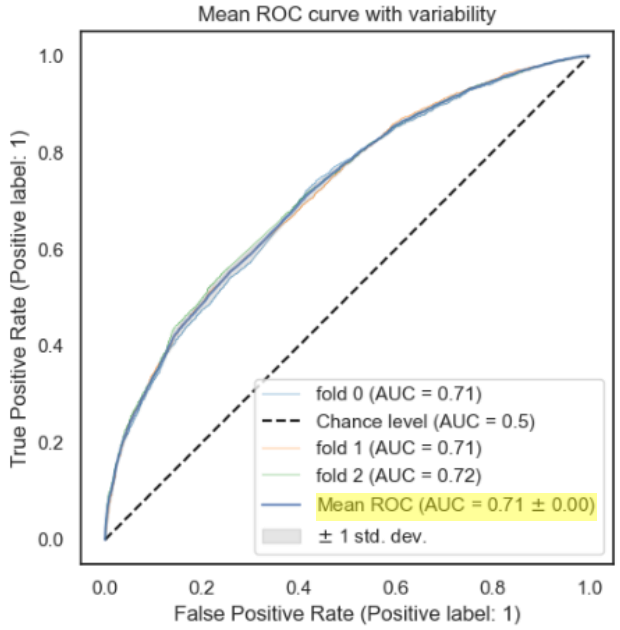
Modèle simple – premiers essais

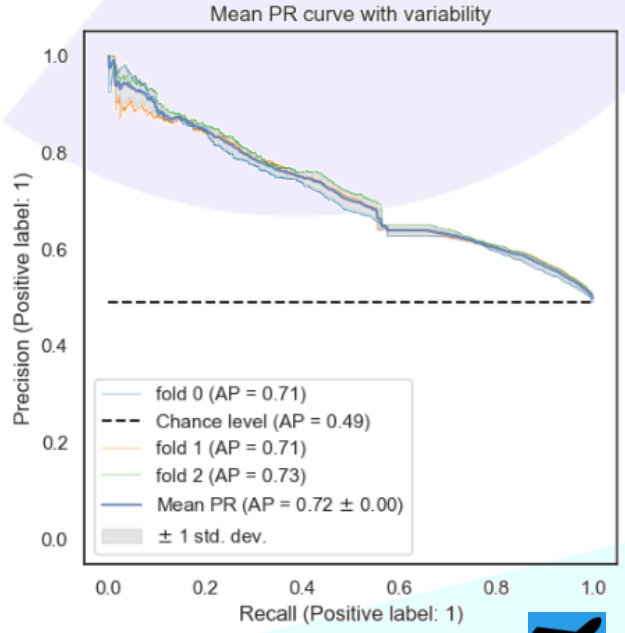
tf_idf_dummy
ROC and Precision-Recall curves
from Cross Validation
(using the mean of folds curves)





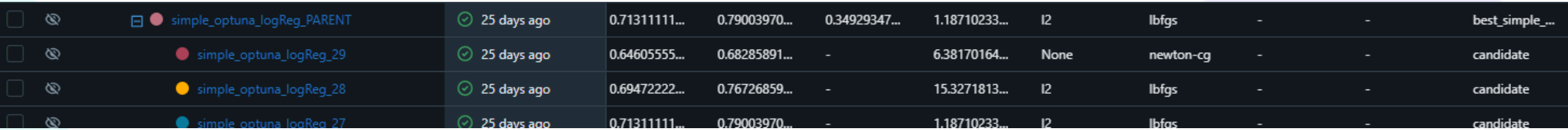
tf_idf_lr
ROC and Precision-Recall curves
from Cross Validation
(using the mean of folds curves)





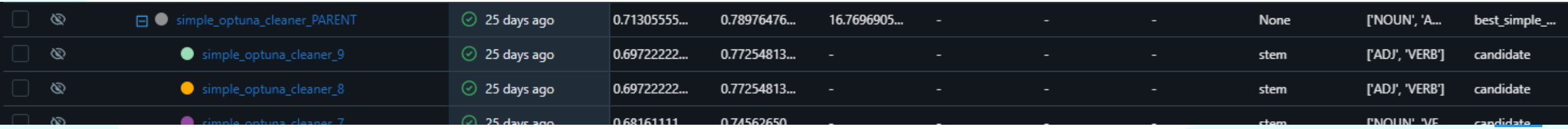
Modèle simple – recherche hyperparamètres

- Utiliser un run mlflow PARENT :
  - Créer une fonction objectif OPTUNA :
    - utilise le pipeline précédent
    - au sein d'un run mlflow ENFANT
  - Lancer une étude OPTUNA



simple_optuna_logReg_PARENT	25 days ago	0.71311111...	0.79003970...	0.34929347...	1.18710233...	l2	lbfgs	-	-	best_simple_...
simple_optuna_logReg_29	25 days ago	0.64605555...	0.68285891...	-	6.38170164...	None	newton-cg	-	-	candidate
simple_optuna_logReg_28	25 days ago	0.69472222...	0.76726859...	-	15.3271813...	l2	lbfgs	-	-	candidate
simple_optuna_logReg_27	25 days ago	0.71311111...	0.79003970...	-	1.18710233...	l2	lbfgs	-	-	candidate

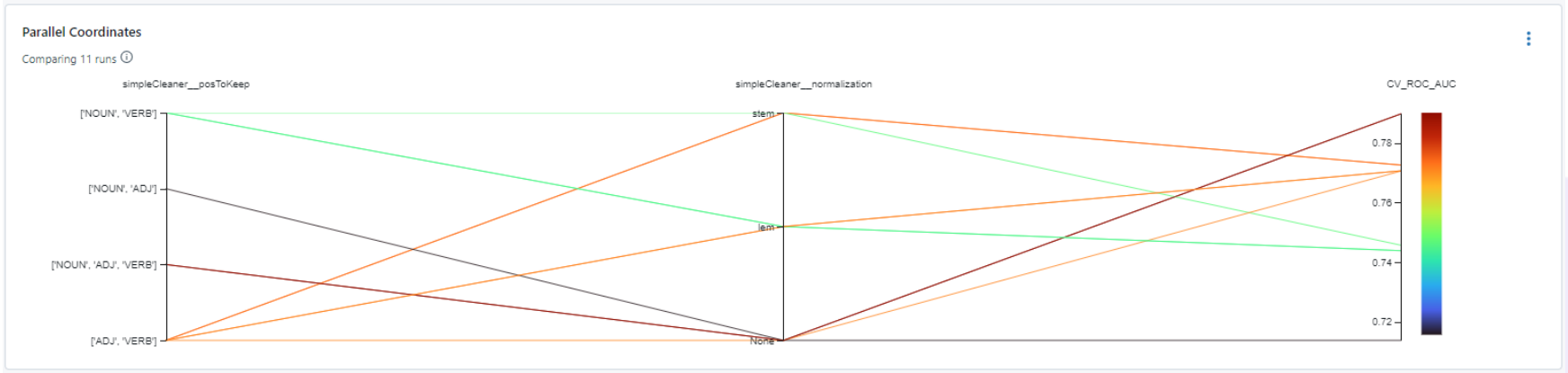
...



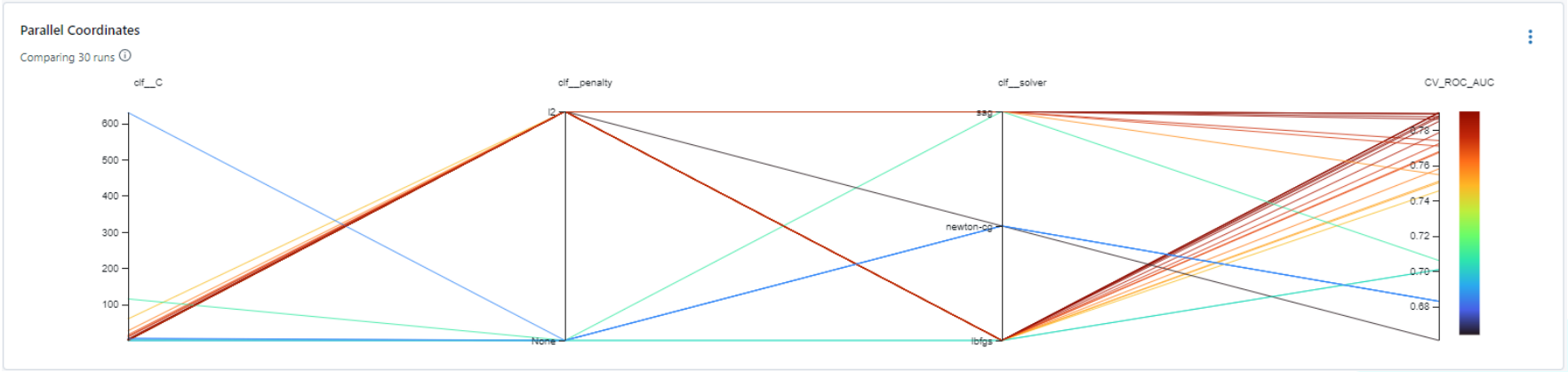
simple_optuna_cleaner_PARENT	25 days ago	0.71305555...	0.78976476...	16.7696905...	-	-	-	None	['NOUN', 'A...	best_simple_...
simple_optuna_cleaner_9	25 days ago	0.69722222...	0.77254813...	-	-	-	-	stem	['ADJ', 'VERB']	candidate
simple_optuna_cleaner_8	25 days ago	0.69722222...	0.77254813...	-	-	-	-	stem	['ADJ', 'VERB']	candidate
simple_optuna_cleaner_7	25 days ago	0.68161111...	0.74562650...	-	-	-	-	stem	['NOUN', 'VE...	candidate

Modèle simple – recherche hyperparamètres

- Grille preprocessing / cleaner :

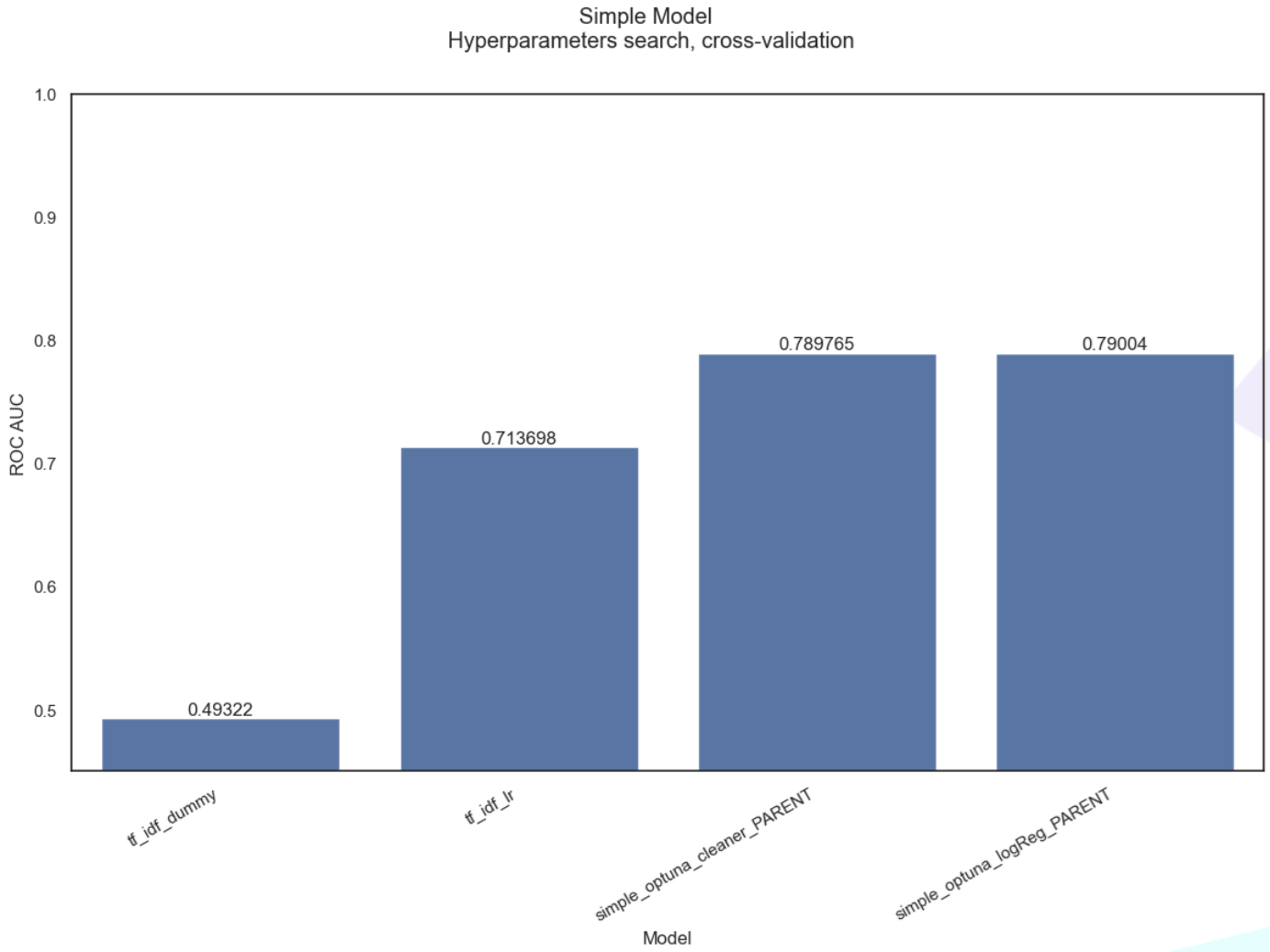


- Grille régression logistique :



Modèle simple – recherche hyperparamètres

- Bilan :

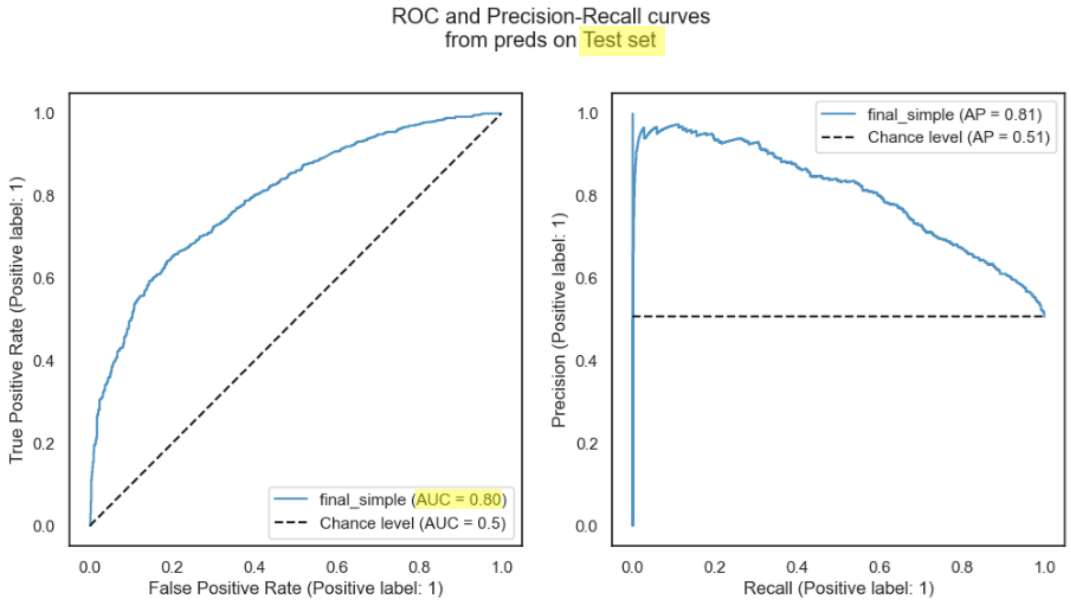


Modèle simple – modèle final

- Modèle final avec hyperparamètres optimisés :

```python
# create a pipeline with our simpleCleaner, a TFIDF and a Logistic Regression
completLrPipe = Pipeline(
    [
        ("generalCleaner", mf.generalCleaner(url = "url", email = "email", hashtag = "hashtag", mention = "mention")),
        ("simpleCleaner", mf.simpleModelCleaner()),
        ("vect", TfidfVectorizer()),
        ("clf", LogisticRegression())
    ]
)
```

```
{'simpleCleaner__normalization': 'None',
 'simpleCleaner__posToKeep': "['NOUN', 'ADJ', 'VERB']",
 'clf__C': 1.1871023326363952,
 'clf__solver': 'lbfgs',
 'clf__penalty': 'l2'}
```



PARTIE 4 – MODÈLE AVANCÉ

Modèle avancé

Text_vectorization Keras :

- standardize :
  - generalClean
  - advancedModelClean
- Taille vocabulaire
- Taille des séquences

Sequential model Keras :

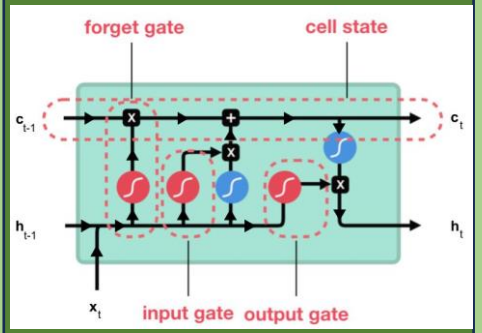
Embedding :

- Avec ou sans poids pré-entraînés

GloVe

fastText

Bidirectionnal LSTM :



Dense :

- 16 unités
- Activation : *reLu*

Dense :

- 1 unité
- Activation : *sigmoïde*

Modèle avancé - pipeline

- Pipeline d'apprentissage dans une expérimentation mlflow :
  - Créer :

Text_vectorization Keras :
- standardize :
  - generalClean
  - advancedModelClean
- Taille vocabulaire
- Taille des séquences

params ➡ mlflow

  - Créer la matrice d'embedding

params ➡ mlflow

  - Créer :

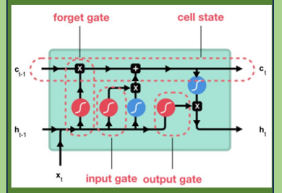
Sequential model Keras :

Embedding :
- Avec ou sans poids pré-entraînés

GloVe

fastText

Bidirectionnal LSTM :



Dense :
- 16 unités
- Activation : *reLu*

Dense :
- 1 unité
- Activation : *sigmoïde*

params ➡ mlflow

  - Entraîner le modèle

AUC, Acc, plots, Model, training_time ➡ mlflow

Modèle avancé - baseline

Text_vectorization Keras :
- standardize :
  - generalClean
  - advancedModelClean
- Taille vocabulaire
- Taille des séquences

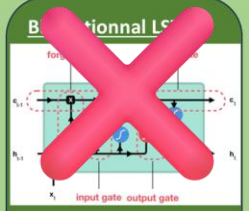
Sequential model Keras :

Embedding :
- Avec ou sans poids pré-entraînés

GloVe

fastText

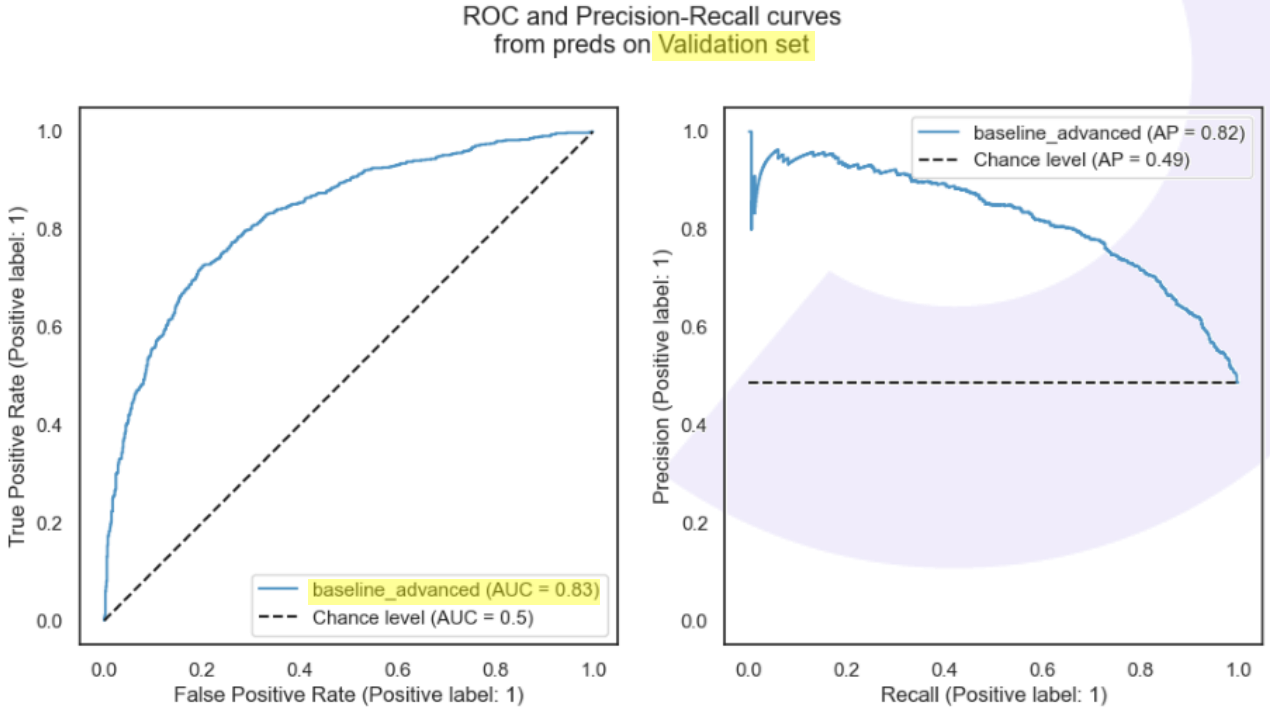
Bidirectionnal LSTM



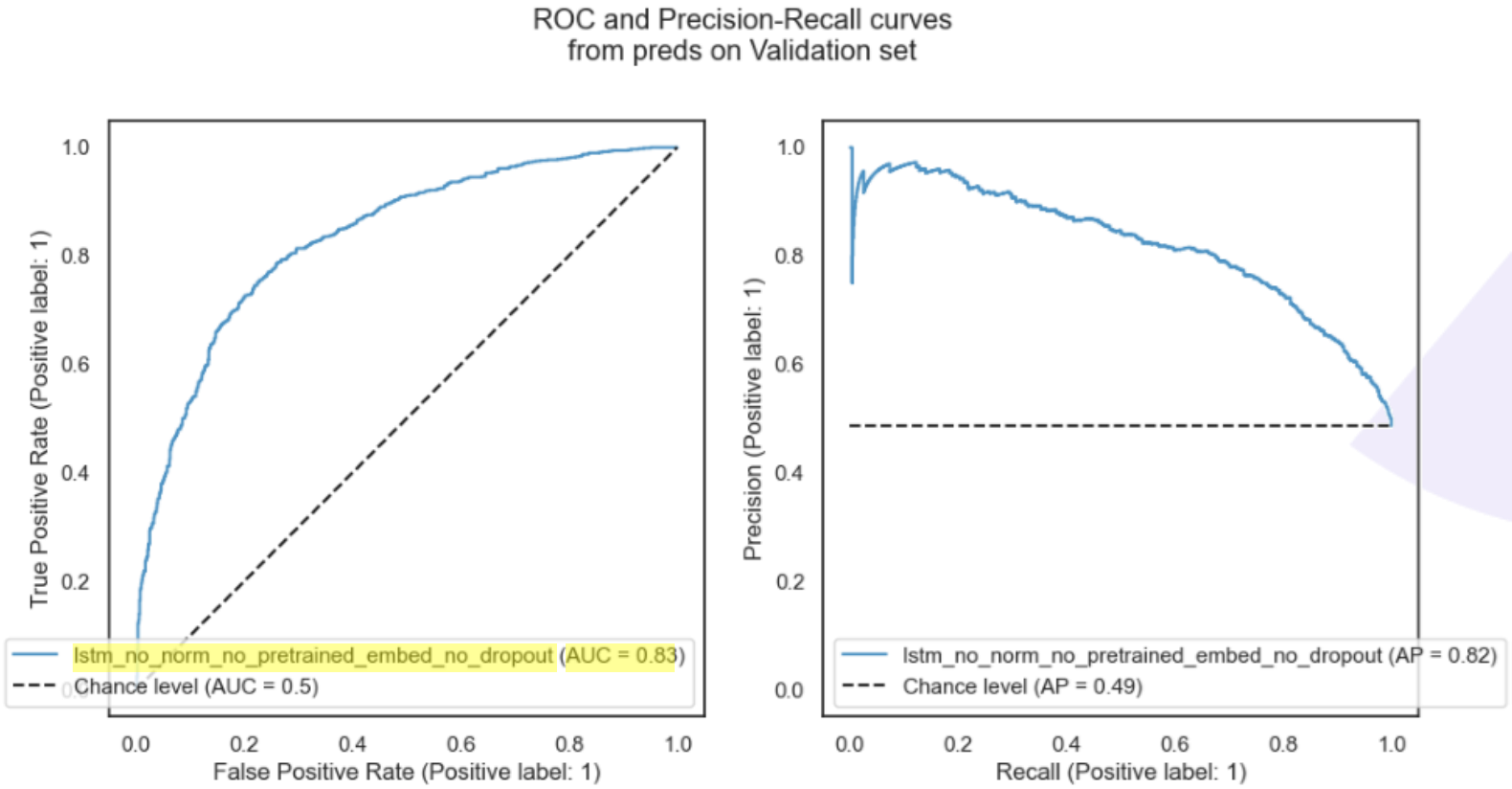
Flatten()

Dense :
- 16 unités
- Activation : *reLu*

Dense :
- 1 unité
- Activation : *sigmoïde*

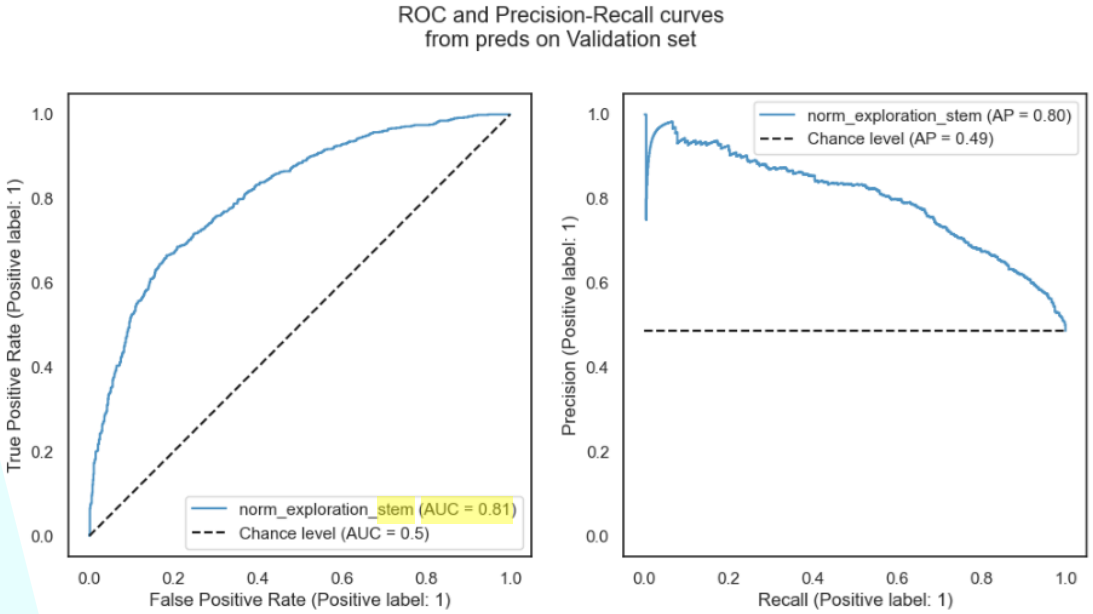


Modèle avancé – premier essai



Modèle avancé – recherche hyperparamètres

- `normalization = "stem"` ou `normalization = "lem"` ?



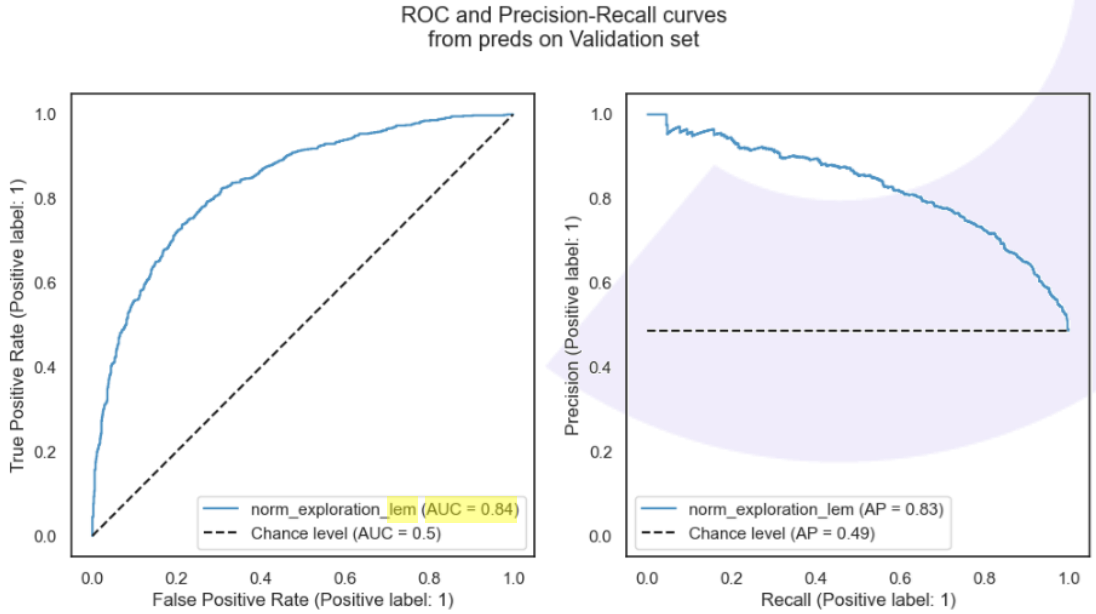


Table | Chart | Evaluation | Experimental

			Metrics			Parameters	Tags
Run Name	Created	Duration	training_time	valid_Accuracy	valid_ROC_AUC	normalization	status
norm_exploration_lem	25 days ago	2.3min	138.859679...	0.75833332...	0.83659863...	lem	candidate
norm_exploration_stem	25 days ago	1.9min	115.127860...	0.73388886...	0.81029224...	stem	candidate
lstm_no_norm_no_pretrained_embed_no_dr...	25 days ago	1.6min	96.6895365...	0.76055556...	0.83386272...	None	candidate

Modèle avancé – recherche hyperparamètres

- GloVe ou fastText ? `trainable = True` ou `trainable = False` ?

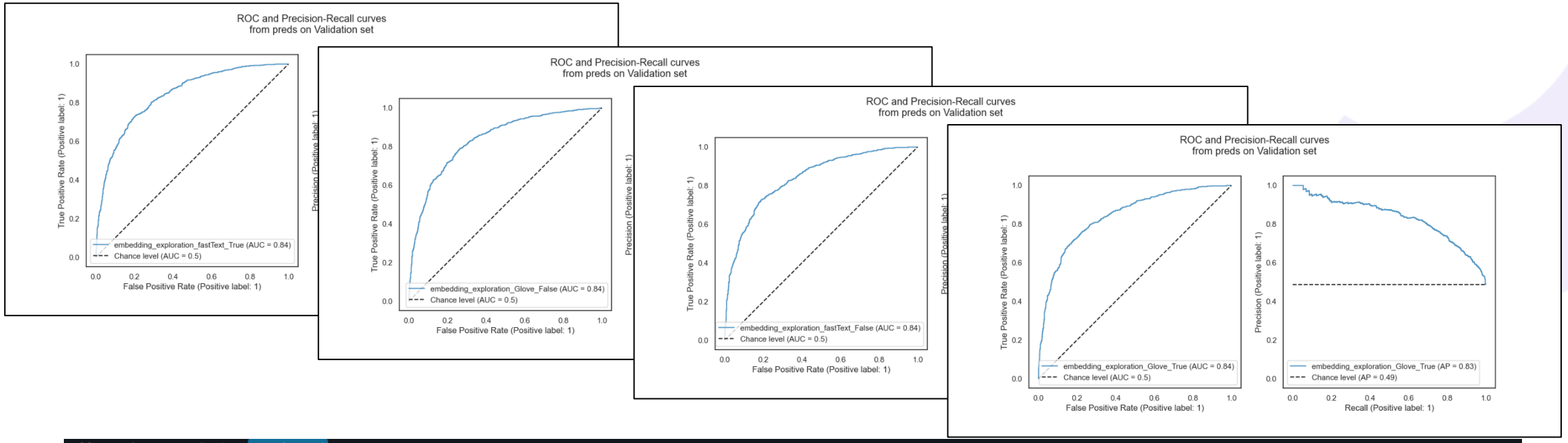
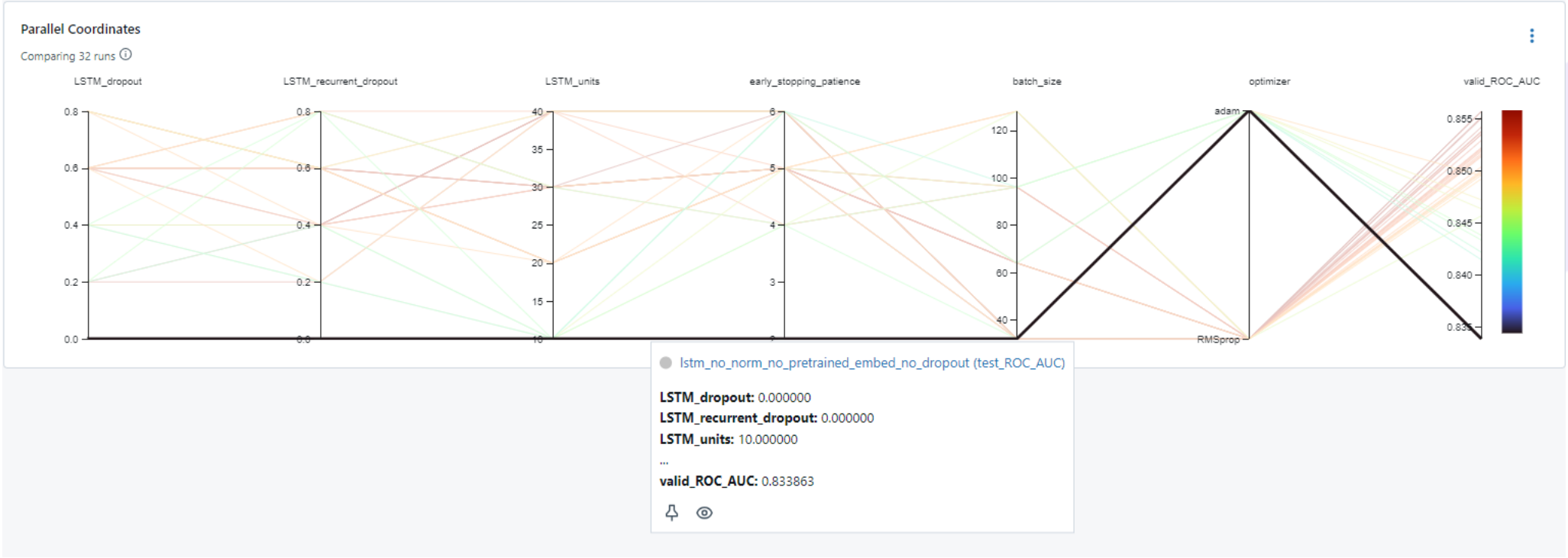


Table | Chart | Evaluation | Experimental

			Metrics				Parameters		Tags
Run Name	Created	Duration	training_time	valid_Accuracy	valid_ROC_AUC	embedding_file_path	trainable_embec	status
embedding_exploration_fastText_True	25 days ago	3.1min	183.450798...	0.75166666...	0.83910292...	pretrained_embeddings/fasttext-wiki-news-300d-1M.vec	True	candidate
embedding_exploration_Glove_True	25 days ago	2.4min	143.117421...	0.76333332...	0.84144091...	pretrained_embeddings/glove.6B.300d.txt	True	candidate
embedding_exploration_fastText_False	25 days ago	3.1min	182.859122...	0.76666665...	0.83901143...	pretrained_embeddings/fasttext-wiki-news-300d-1M.vec	False	candidate
embedding_exploration_Glove_False	25 days ago	2.2min	130.811452...	0.75944441...	0.83685868...	pretrained_embeddings/glove.6B.300d.txt	False	candidate

Modèle avancé – recherche hyperparamètres

- Comme précédemment : mlflow et OPTUNA

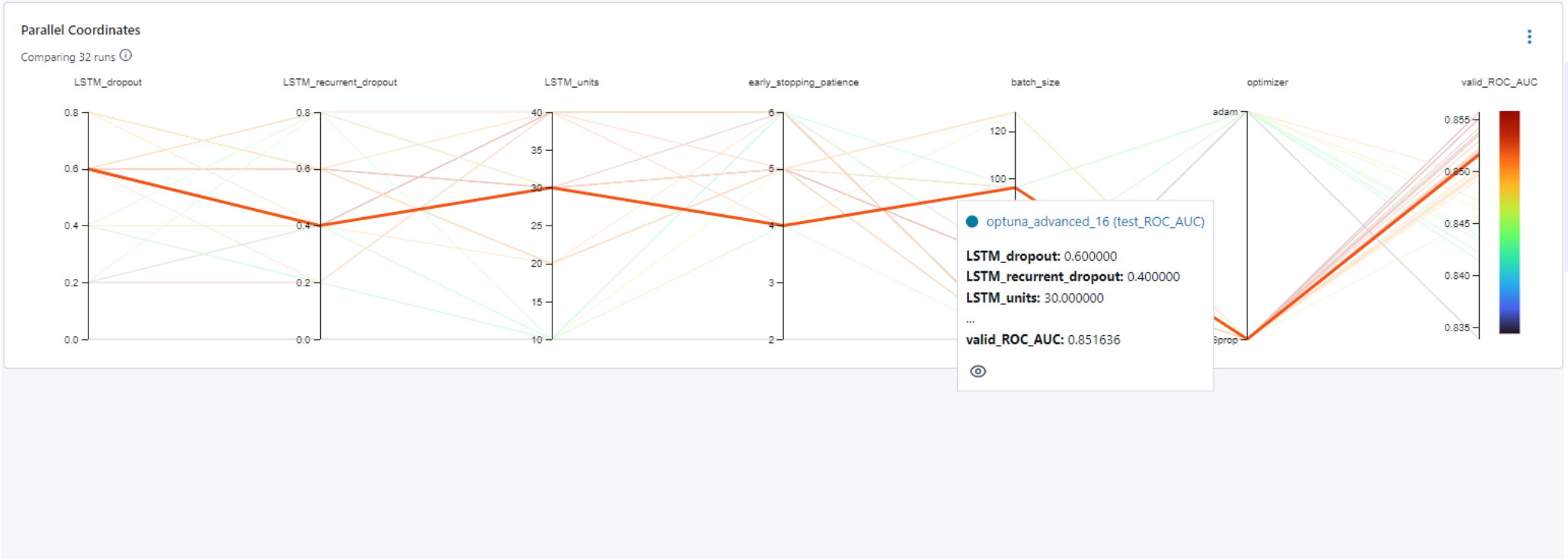


Modèle avancé – recherche hyperparamètres

- Comme précédemment : mlflow et OPTUNA

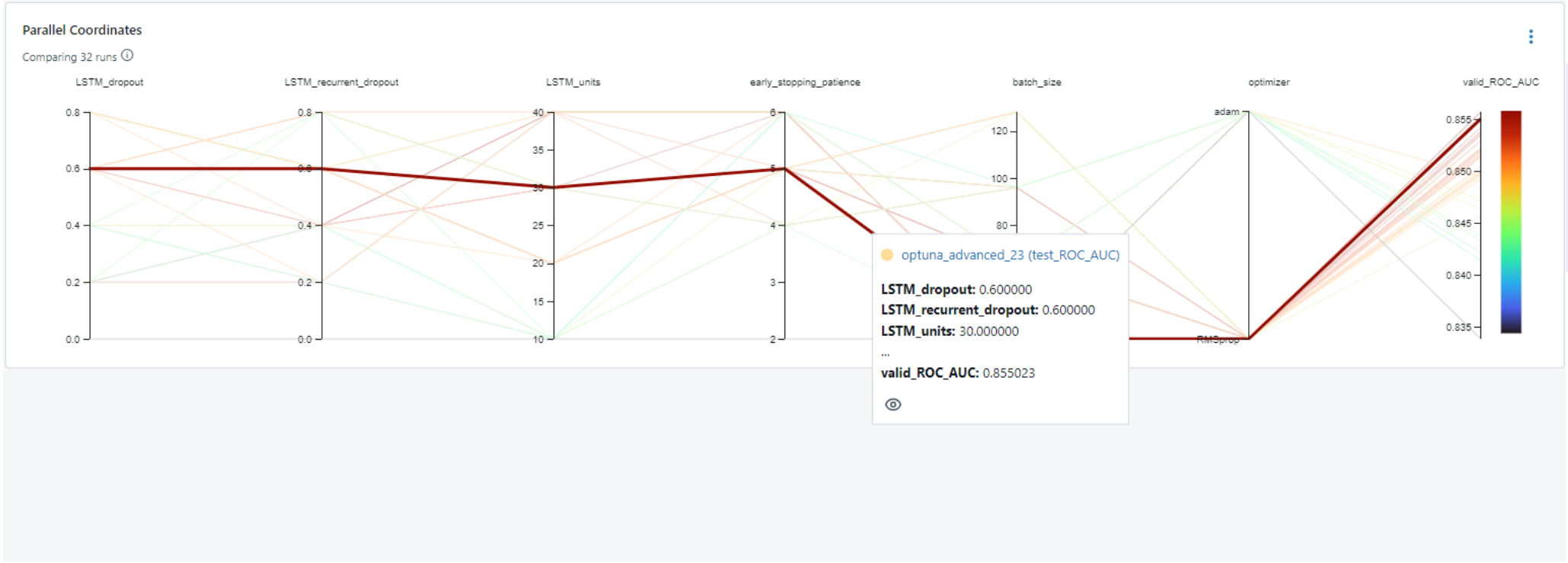
Modèle avancé – recherche hyperparamètres

- Comme précédemment : mlflow et OPTUNA



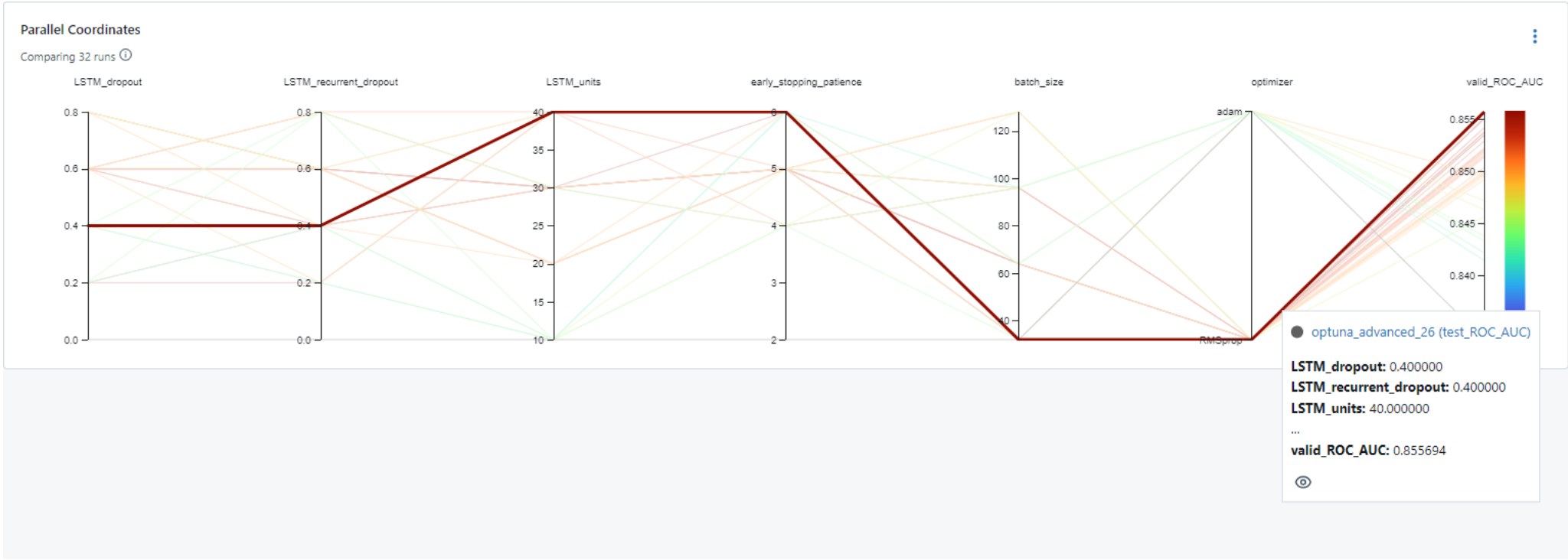
Modèle avancé – recherche hyperparamètres

- Comme précédemment : mlflow et OPTUNA



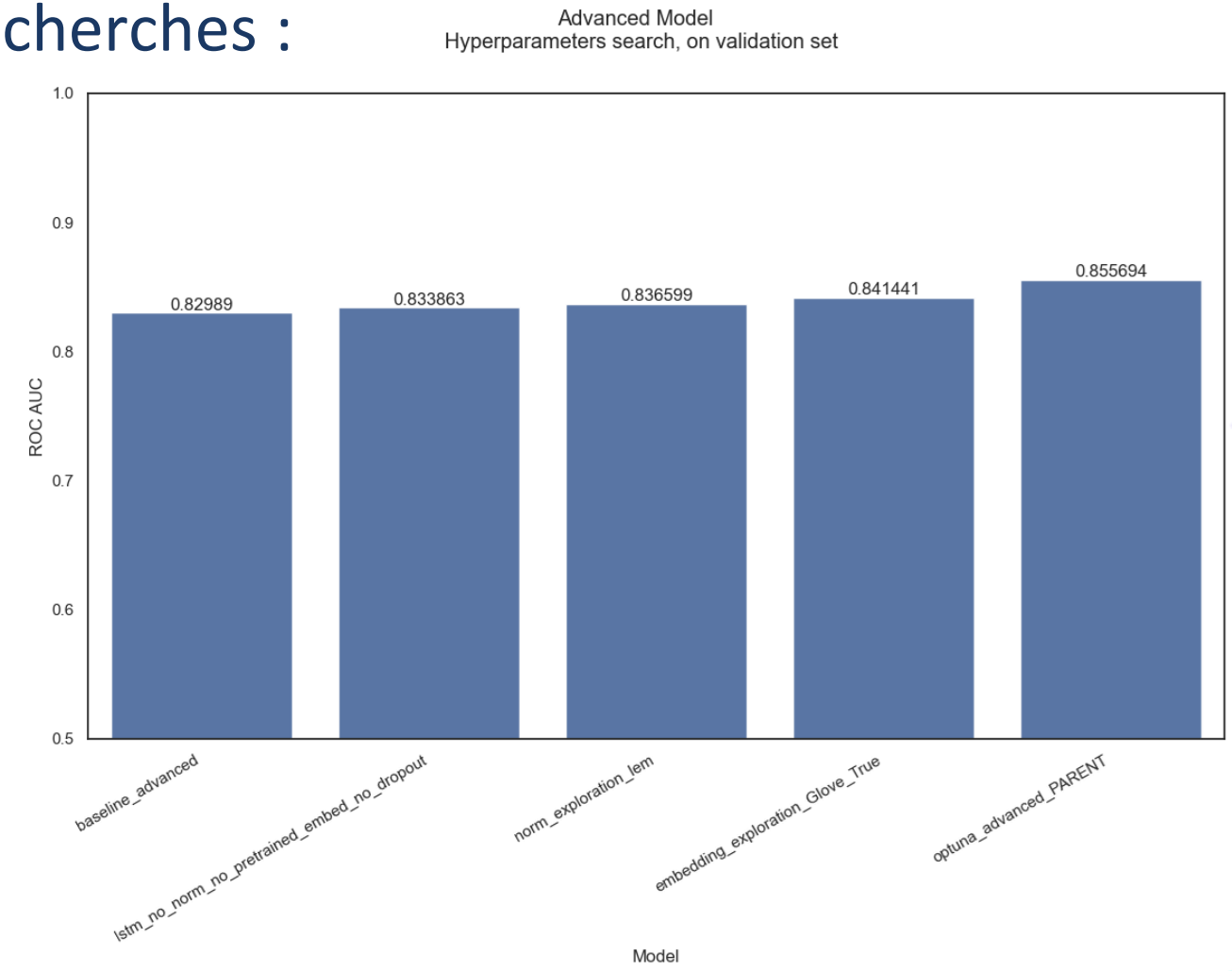
Modèle avancé – recherche hyperparamètres

- Comme précédemment : mlflow et OPTUNA



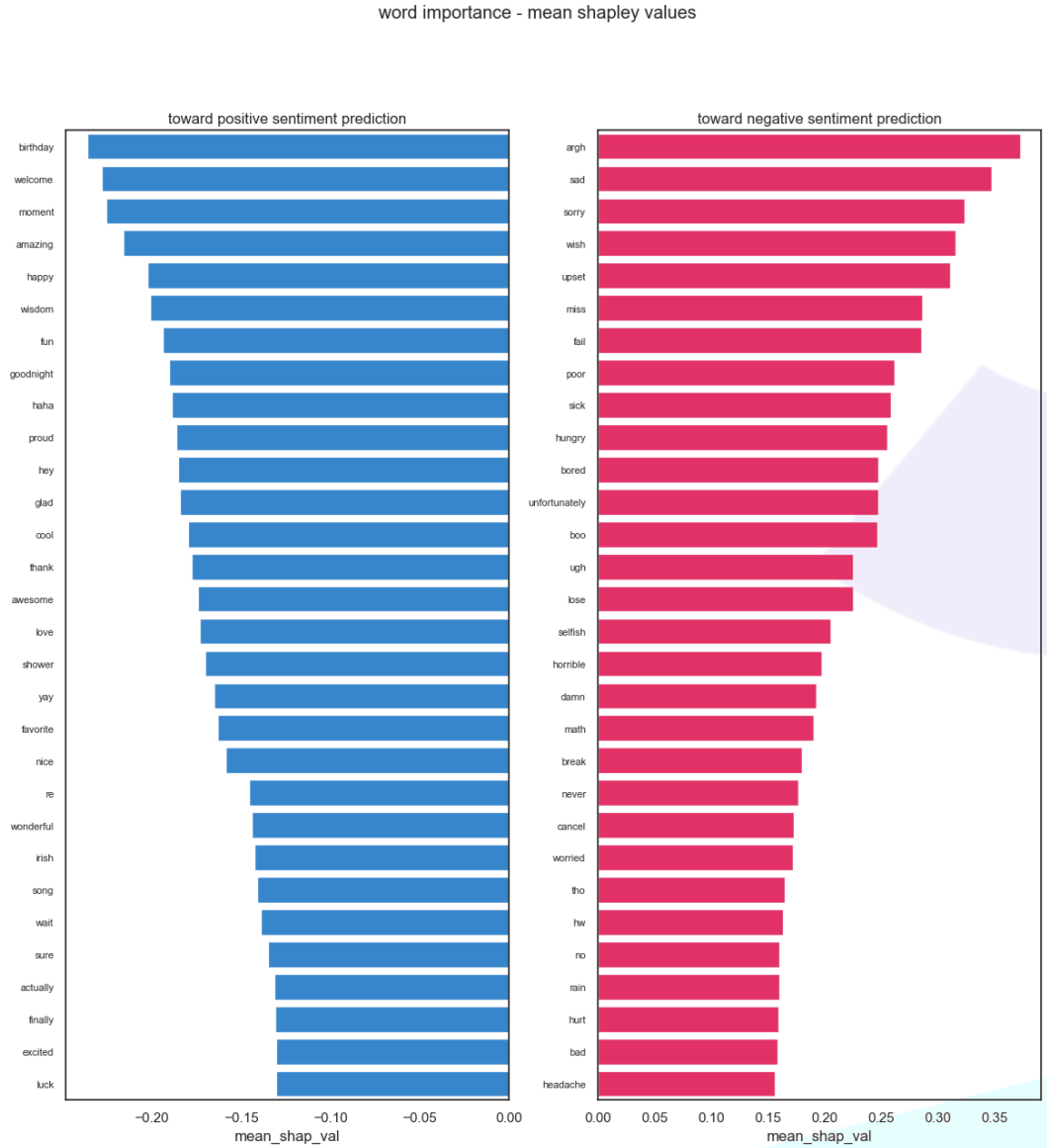
Modèle avancé - recherche hyperparamètres

- Bilan des recherches :



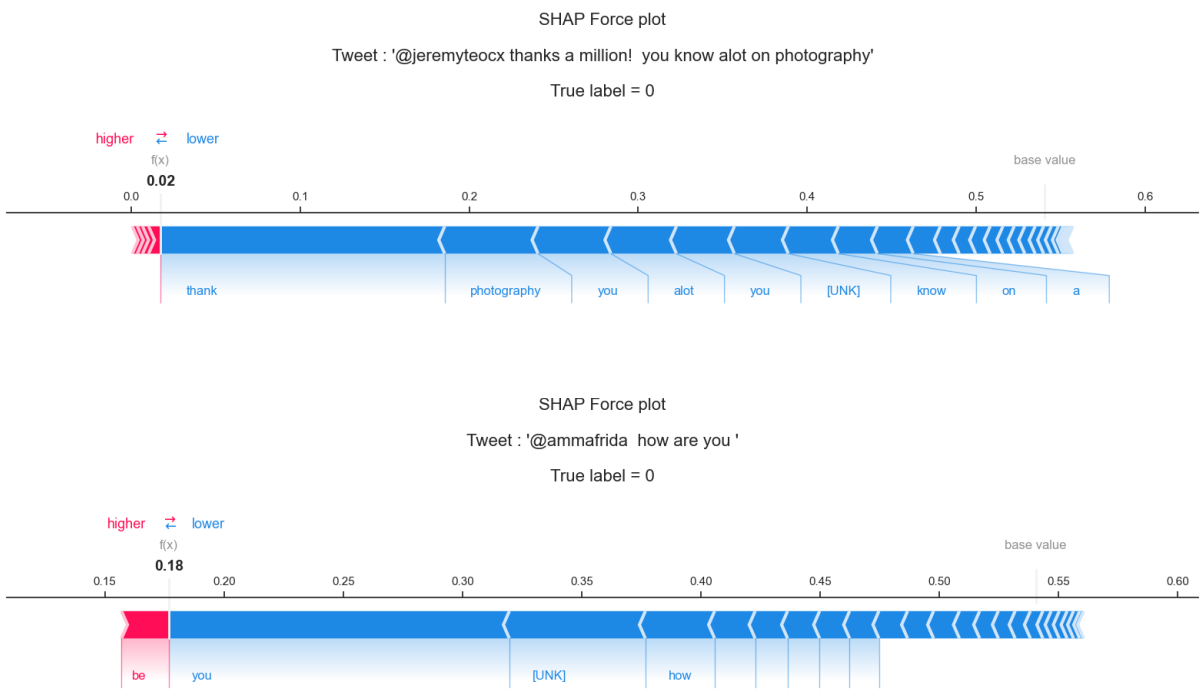
Modèle avancé – expliquer les résultats

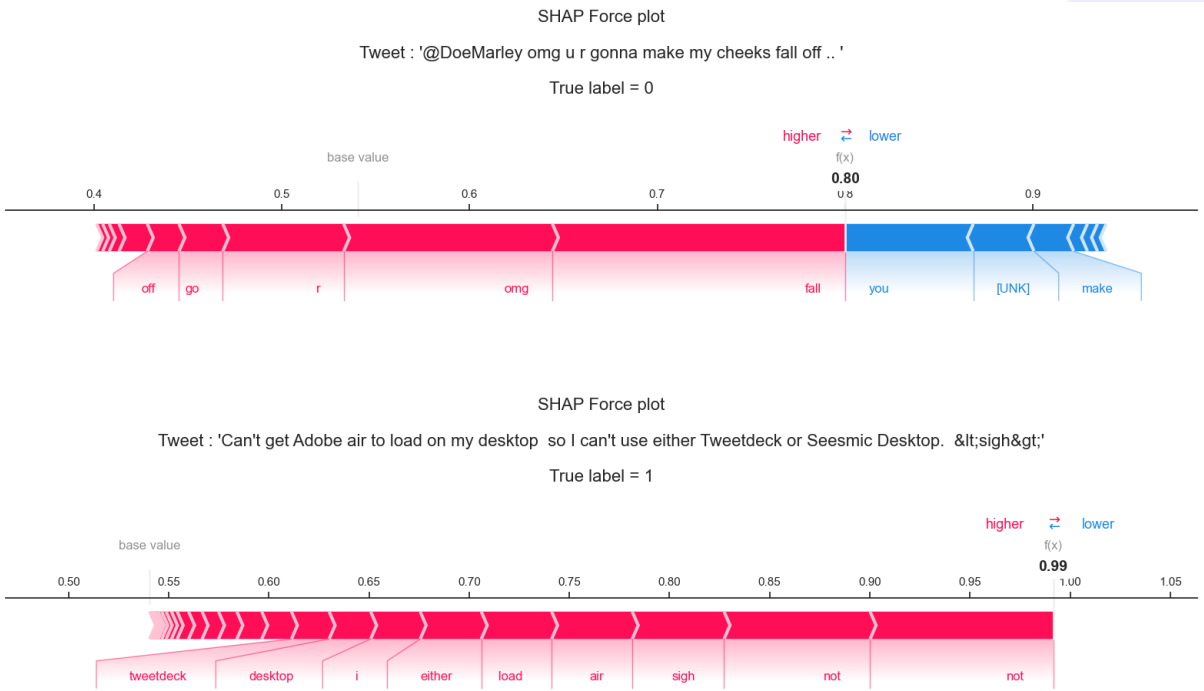
- Utilisation de Shap
- Au niveau **global** :

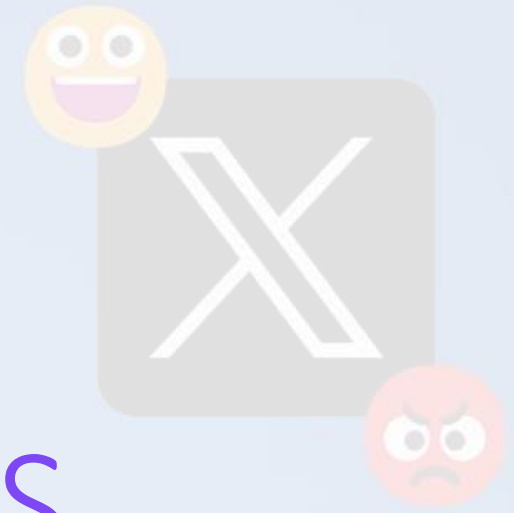


Modèle avancé – expliquer les résultats

- Utilisation de Shap
- Au niveau **local** :







PARTIE 5 – TESTS AUTRES SOLUTIONS et BILAN

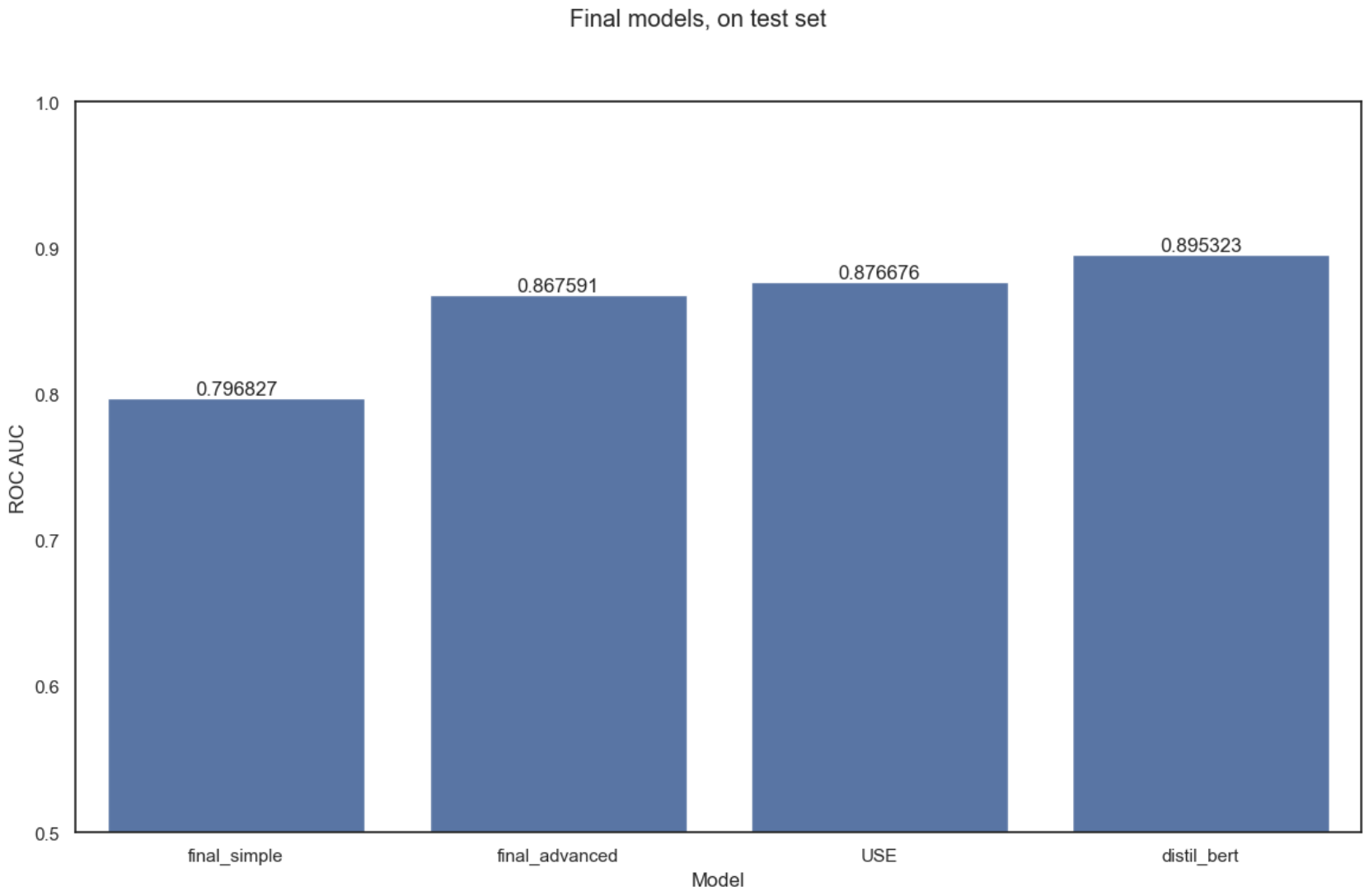
BERT (Bidirectional Encoder Representations from Transformers)

- Modèle ayant considérablement fait avancer le NLP en termes de performance
- Architecture basée sur la partie **encoder** du transfomer, et son **mécanisme d'attention**
- BERT n'a ainsi pas appris une tâche spécifique (comme de la traduction, etc.), mais une « **compréhension » très fine** du langage
- On peut alors appliquer, via transfer learning, **n'importe quel type de tâche**, comme de la classification de texte
- Ici : utilisation de **DistilBERT** (moins performant mais plus petit, plus rapide)

USE (Universal Sentence Encoder)

- Ici, ***sentence*** *embedding*
- Également développé par Google
- **Moins consommatrice** en ressource mémoire

Démarche de modélisation - bilan



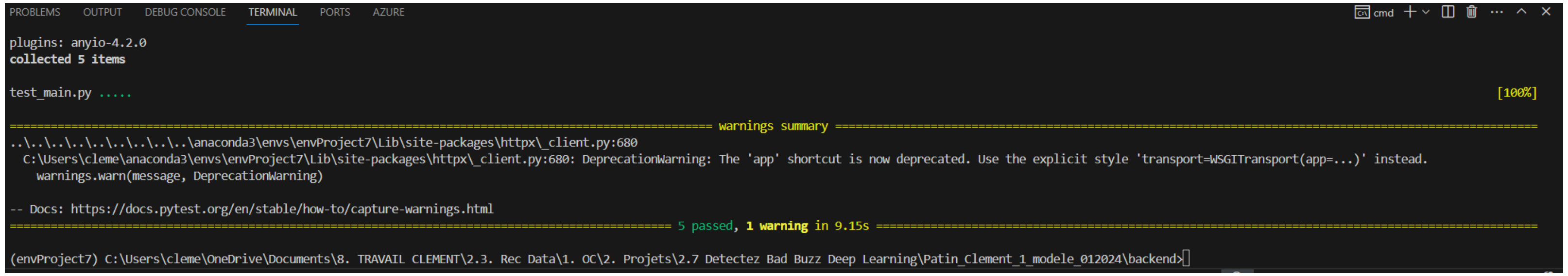
PARTIE 6 – DÉPLOIEMENT API et UI

API

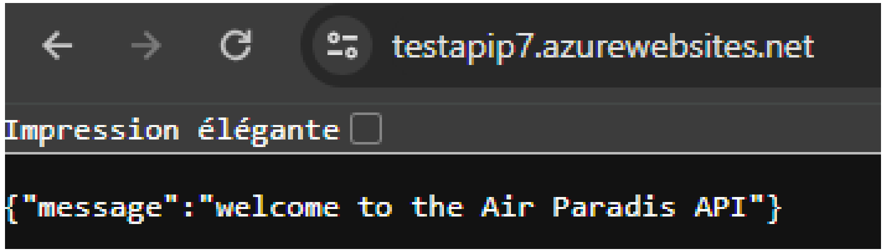
- entraîner modèle sur plus de données
- enregistrer TensorFlow Lite
- `main.py` FastAPI
- `test_main.py` pour les tests unitaires pytest
- `Dockerfile` docker
- Construire image docker docker
- *resource group*
- *appservice plan*
- *webapp*
- Déploiement continu + GitHub Actions

API

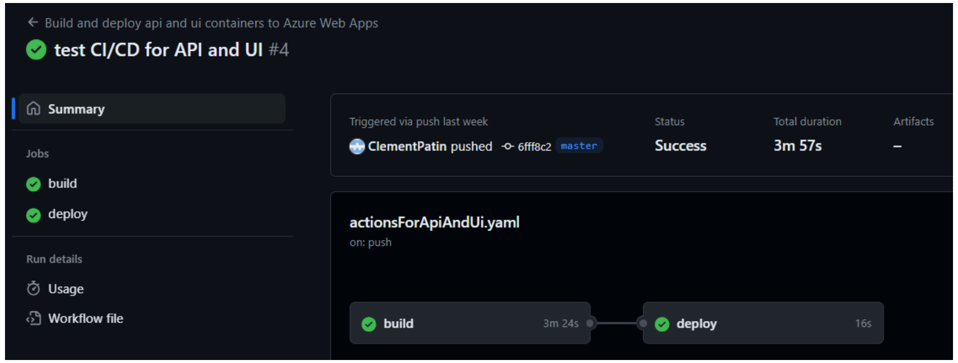
Tests unitaires :



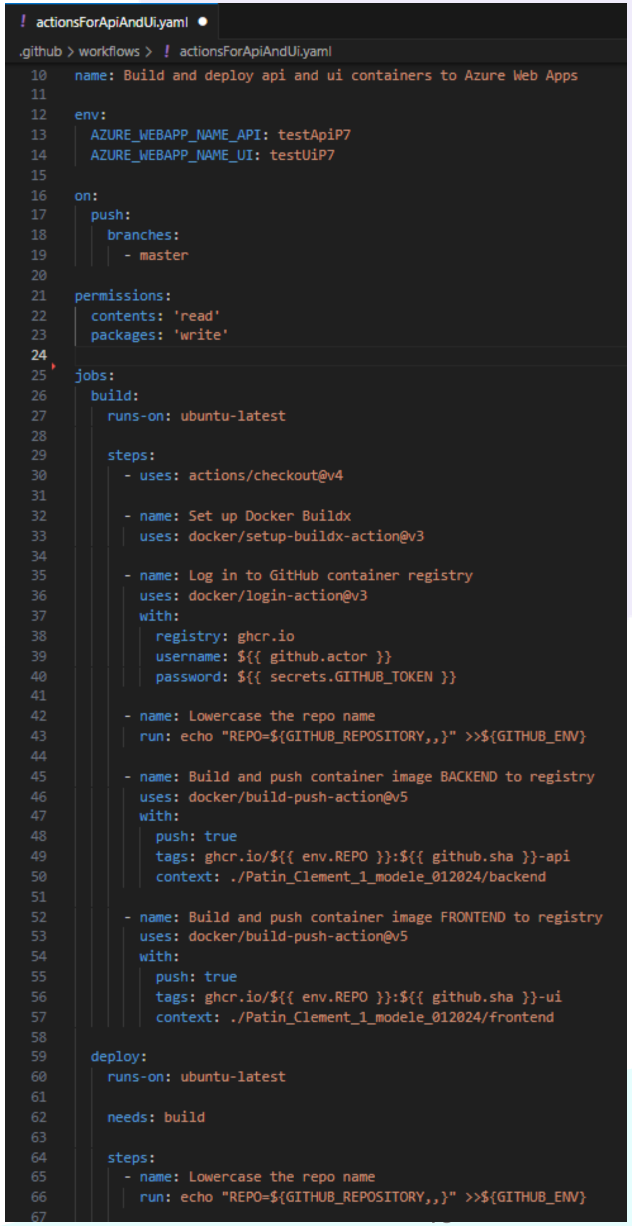
API déployée :



CI/CD:



Fichier YAML GitHub Actions :



API – test serving mlflow

- Via la ligne de commande :
  - se placer dans le dossier contenant */mlruns*
  - `mlflow models serve -m "models:/final_advanced_export/1" -p 1234 --no-conda`
- puis tester l'API grâce à la librairie `requests` :

```python
import requests

inference_request = {
    "inputs" : ["I hate humans", "I love humans", "I say I love humans but it's not true"]
}

endpoint = "http://127.0.0.1:1234/invocations"

response = requests.post(endpoint, json=inference_request)

print(response.text)
```

[1] ✓ 0.7s Python

```
{"predictions": [[0.9499937295913696], [0.11322450637817383], [0.9079648852348328]]}
```

PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER AZURE mlflow

```
WARNING:tensorflow:From C:\Users\cleme\anaconda3\envs\envProject7\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From C:\Users\cleme\anaconda3\envs\envProject7\Lib\site-packages\keras\src\saving\legacy\saved_model\load.py:107: The name tf.gfile.Exists is deprecated. Please use tf.io.gfile.exists instead.

2024-03-27 14:53:07.441054: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:tensorflow:From C:\Users\cleme\anaconda3\envs\envProject7\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\cleme\anaconda3\envs\envProject7\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

INFO:waitress:Serving on http://127.0.0.1:1234
1/1 [==============================] - 1s 587ms/step
```


UI

- `streamlit_app.py` 

- `Dockerfile` 

- Construire image docker 

- *resource group* 
- *appservice plan* 
- *webapp* 
- Déploiement continu  +  `GitHub Actions`

UI

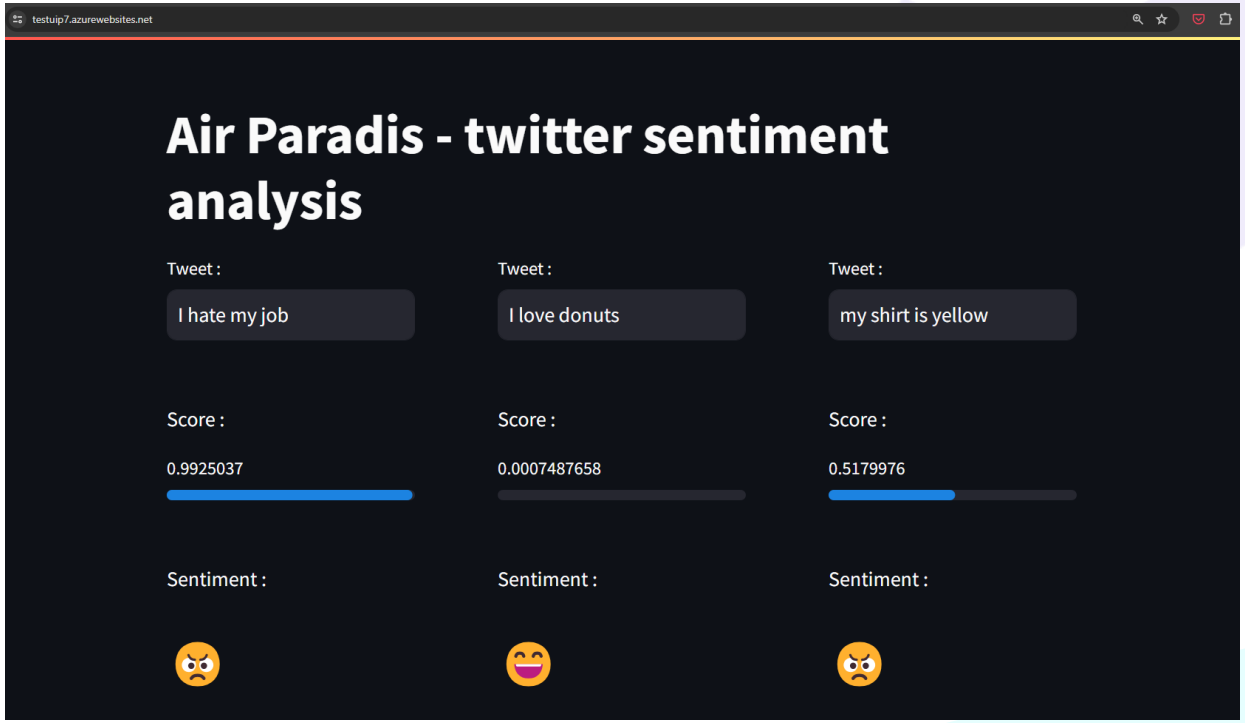
app streamlit :

streamlit_app.py

Patin_Clement_1_modele_012024 > frontend > streamlit_app.py > ...

```python
import streamlit as st
import requests

st.title("Air Paradis - twitter sentiment analysis")

col1, col2, col3 = st. columns(3, gap="large")
baseInputs = ["I hate my job", "I love donuts", "my shirt is yellow"]

for i,(col,baseInput) in enumerate(zip([col1, col2, col3], baseInputs)) :
    with col :

        tweet = st.text_input(label="Tweet :", value=baseInput, key=i)

        st.write("")
        st.write("")


        API_URL = "https://testapip7.azurewebsites.net"

        response = requests.post(API_URL+"/predict", json={"text":tweet}).json()
        # response = requests.post(API_URL+"/predict?text="+tweet)

        if "score" not in response.keys() :
            st.write(":exclamation: Error :exclamation: :")
            st.write("Detail : "+response["detail"])

        else :
            st.write("Score :")
            st.progress(float(response["score"]), text = response["score"])
            st.write("")
            st.write("")

            st.write("Sentiment :")
            if response["sentiment"] == "positive" :
                st.header(":smile:")
            else :
                st.header(" :angry:")
```

app streamlit :



CONCLUSION

merci