Air Paradis

DÉTECTEZ LES BAD BUZZ GRÂCE AU DEEP LEARNING

Anticipation des sentiments associés à un tweet

Henri LINKE

SOMMAIRE

Introduction

Mission :
Produire une IA capable de classifier les tweet afin d'anticiper les bad buzz sur les réseaux sociaux pour le compte de Air Paradis qui veut redorer son image.

Données :
Il n'y a pas de données clients, on utilise donc les données Open source

Ce jeu comprends 1 600 000 tweet avec entre autres le sentiment associé positif (0) ou négatif (4) de répartition égale.

I

Présentation du projet, des données et problématiques

Réalisation du projet

Ce projet est divisé en 3 grosses parties :

1. **Un notebook, qui comprend à la fois la récupération des données, le développement et l'entraînement des différents modèles sur des données réduites - 40 000 commentaires - afin de limiter la lourdeur pour les calculs**
2. **Une deuxième partie, plus aboutie, est basée sur le notebook. . La mise en forme est plus aboutit, modulable et reproductible simplement à partir fichier main qui permettent de lancer ou le prétraitement des données ou l'entraînement de modèle avec des paramètres choisis. Enfin elle intègre la partie MLOps.**
3. **Et enfin, la dernière partie a été crée afin de déployer notre modèle le plus aboutit sur l'interface "Streamlit"**

Problématiques majeures rencontrées et solutions

Les 2 problèmes majeurs de ce projet sont liées aux limites des moyens à disposition, à savoir :

- **Limite de puissance de calcul**
- **Manque de plateforme pour stocker les données**

Les solutions mises en œuvres sont finalement les suivantes :

- **Limite de la quantité de données pour entraîner les modèles, (entre 40 000 et 100 000 pour 1 600 000 disponibles). Limitations des couches et nombres de neurones pour le modèle Bert et utilisation de *Google Colab* pour entraîner ce même modèle.**
- **Calcul et Stockage des données directement au sein de github avec git LFS**

II

Prétraitement des données textuelles

Uniformisation du texte

- **Majuscule → minuscule**
- **@[utilisateur] → "at_user"**
- **Abréviation → forme décontractée**
- **Smiley → mot proche de la signification**
- **http... → "url"**
- **#[word] → "[word]"**
- **Suppression caractères spéciaux**
- **Remplacement des suites de caractères répétés (ex : nooooo → noo)**
- **(Correction des fautes d'orthographe)**
- **Lemmatisation**
- **Suppression de "stop words"**
- **Gestion commentaires devenus vides → TOUS LES COMMENTAIRES ONT ÉTÉ GARDÉS**

III

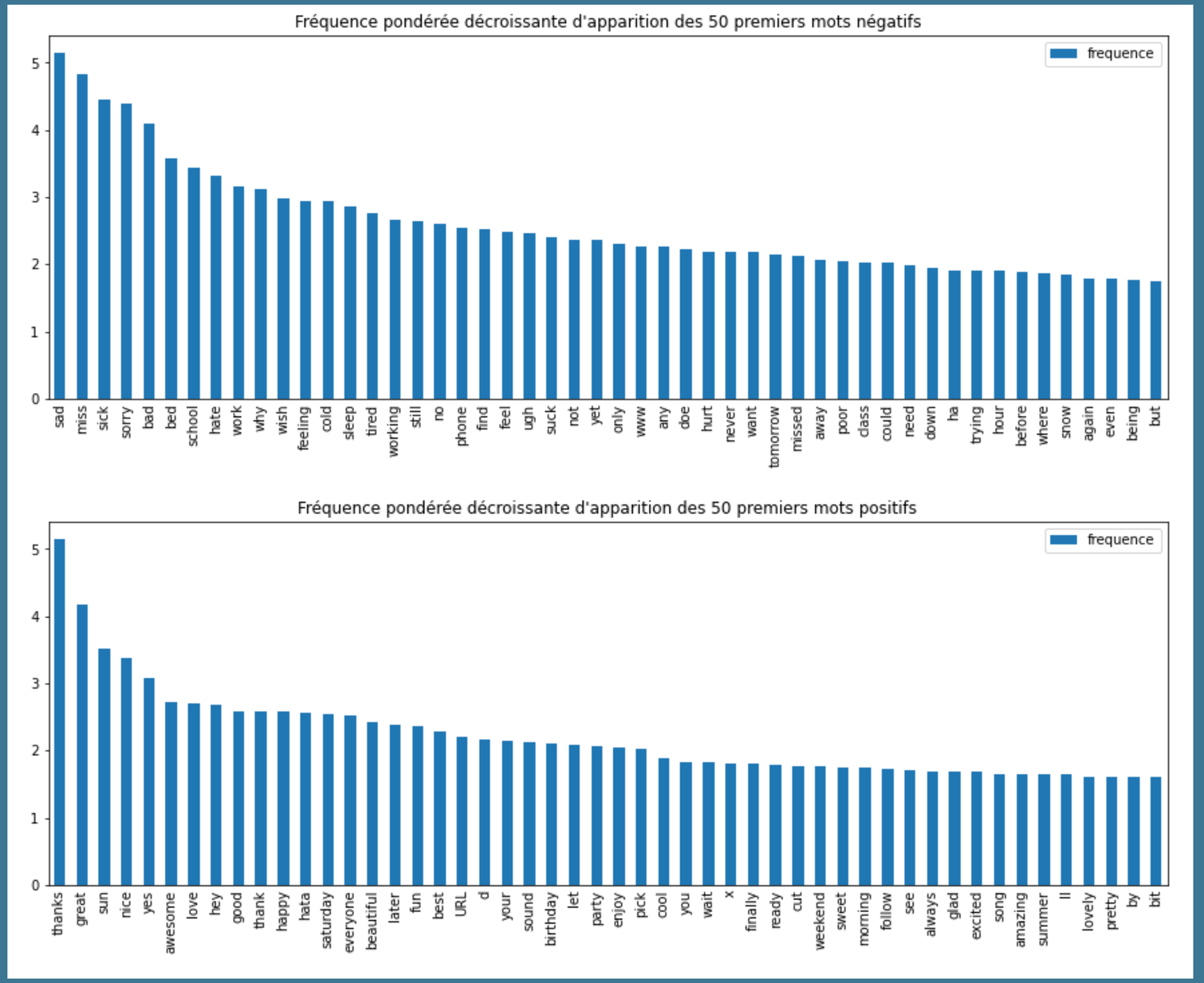
Analyse

Premier modèle

Ce modèle est un modèle tf-idf personnalisé.
Pour cela on pondère l'histogramme des mots pour chaque classe respective par la fréquence des mêmes mots dans l'autre classe.

De cette manière les mots prennent de l'importance lorsqu'ils sont présents significativement dans une classe plutôt que l'autre.

Accuracy obtenue : 71.425 %



Modèles classiques de ML avec scikit-learn

Pour faire marcher ces modèles, il faut leur donner en entrée un vecteur. Il y a donc une étape de word embedding qui précède l'entraînement du modèle.

3 méthodes de word embedding testées :

- **Word2Vec : représente les mots sous forme de vecteurs denses**
- **Doc2Vec : extension de Word2Vec qui permet de représenter non seulement les mots, mais aussi des documents entiers en l'occurence les tweet ici**
- **FastText : similaire à Word2Vec, mais avec une particularité importante. Cette méthode prend également en compte les sous-mots (ou n-grammes) des mots**

Pour Word2Vec et FastText il faut donc ensuite créer le "vecteur tweet" en combinant les "vecteurs mots".
2 solutions testées : la moyenne et le tf-idf

Modèles testés

On effectue une réduction de dimension à l'aide d'un PCA pour diminuer la colinéarité et augmenter l'efficacité des modèles.

Régression linéaire

Forêt aléatoire

Gradient boosting tree

Combinaison des précédents modèles

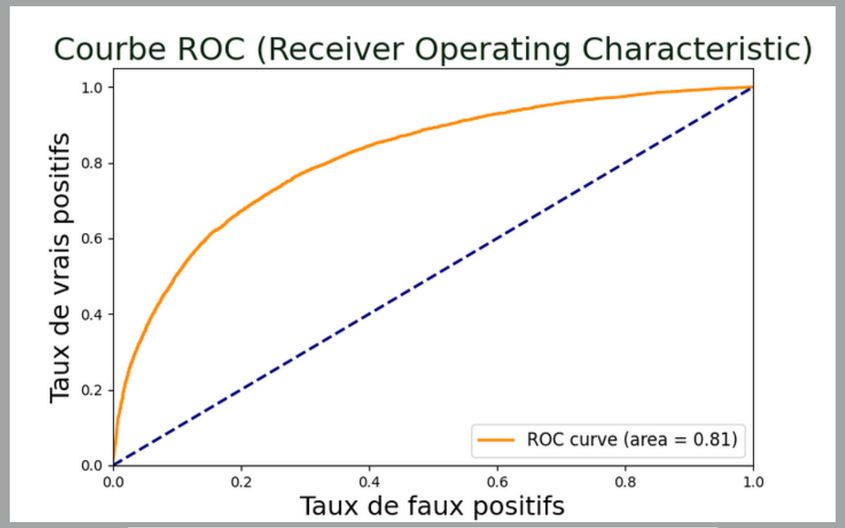
Modèle Bert :

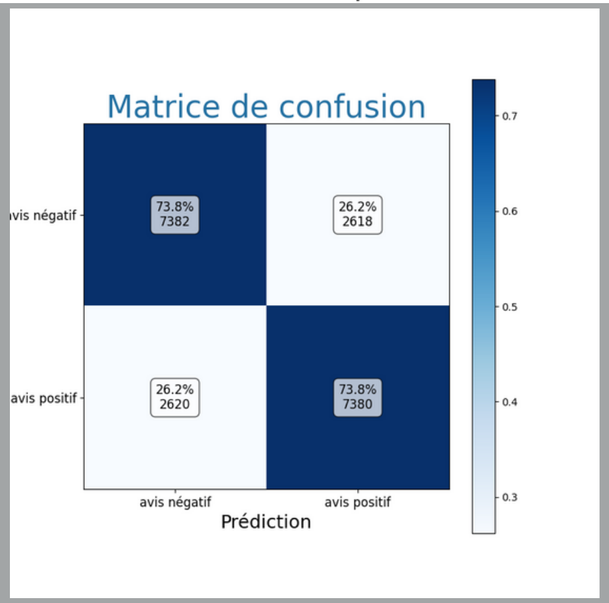
BERT (Bidirectional Encoder Representations from Transformers) est un modèle de langage de pointe en traitement du langage naturel qui utilise une méthode d'apprentissage préalable supervisée pour capturer des informations contextuelles riches sur les mots.

- **Récupération du modèle déjà entraîné sur tensorflow : Modèle anglais basique.**
- **Ajout de notre couche de sortie adapté à notre problème de classification.**
- **Choix d'une métrique d'optimisation : Accuracy et AUC.**
- **Choix de la méthode de convergence.**
- **Entraînement du modèle.**
- **Ré-entraînement du modèle avec d'autres données**
- **Enregistrement de la couche de sortie.**

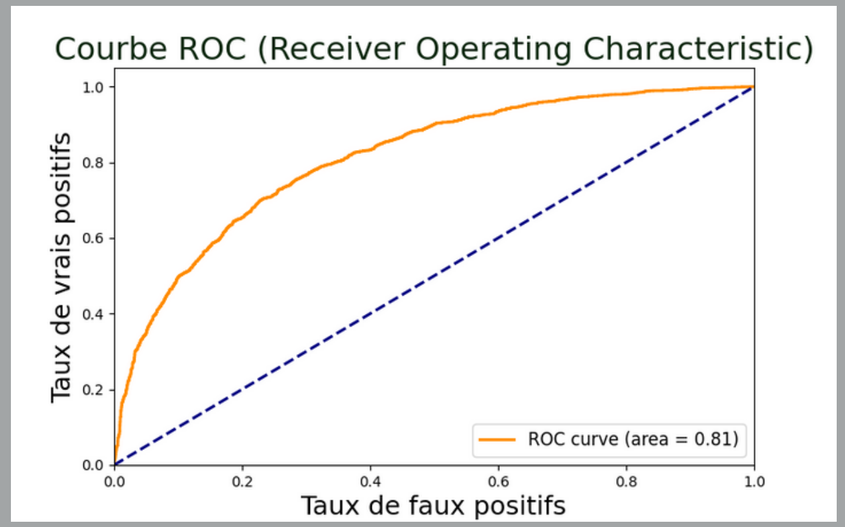
Meilleurs résultats

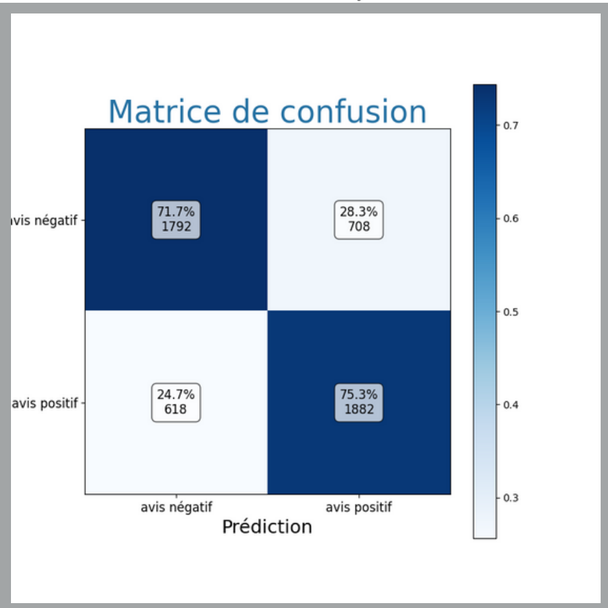
Forêt Aléatoire : 0.738



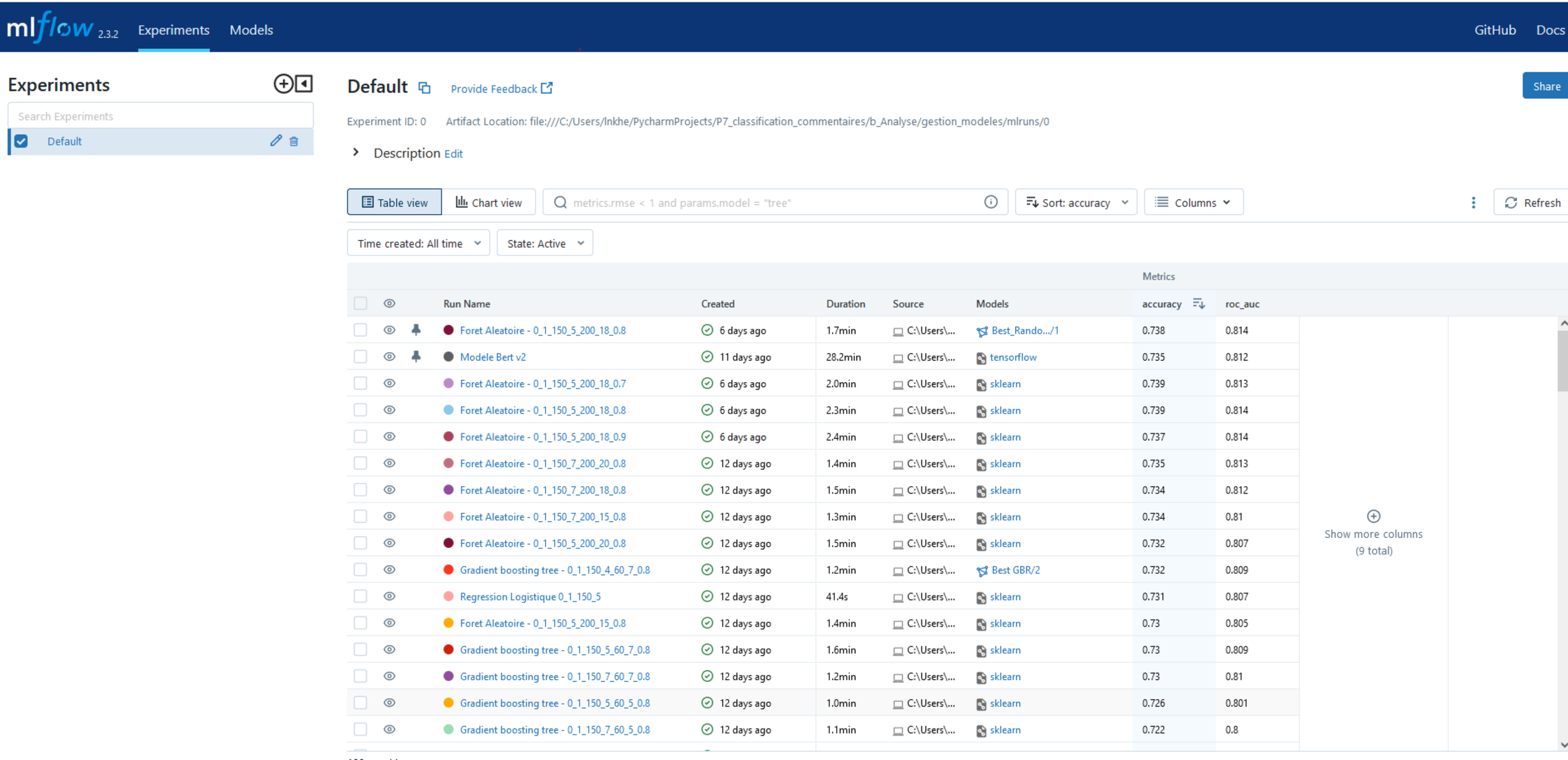


Modèle BERT : 0.735





MLOps


mlflow 2.3.2
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Default
Provide Feedback
Share
Experiment ID: 0
Artifact Location: file:///C:/Users/lnkhe/PycharmProjects/P7_classification_commentaires/b_Analyse/gestion_modeles/mlruns/0
Description Edit
Table view
Chart view
metrics.rmse < 1 and params.model = "tree"
Sort: accuracy
Columns
Refresh
Time created: All time
State: Active
Metrics
Run Name
Created
Duration
Source
Models
accuracy
roc_auc
Foret Aleatoire - 0_1_150_5_200_18_0.8 6 days ago 1.7min C:\Users\... Best_Rando.../1 0.738 0.814
Modele Bert v2 11 days ago 28.2min C:\Users\... tensorflow 0.735 0.812
Foret Aleatoire - 0_1_150_5_200_18_0.7 6 days ago 2.0min C:\Users\... sklearn 0.739 0.813
Foret Aleatoire - 0_1_150_5_200_18_0.8 6 days ago 2.3min C:\Users\... sklearn 0.739 0.814
Foret Aleatoire - 0_1_150_5_200_18_0.9 6 days ago 2.4min C:\Users\... sklearn 0.737 0.814
Foret Aleatoire - 0_1_150_7_200_20_0.8 12 days ago 1.4min C:\Users\... sklearn 0.735 0.813
Foret Aleatoire - 0_1_150_7_200_18_0.8 12 days ago 1.5min C:\Users\... sklearn 0.734 0.812
Foret Aleatoire - 0_1_150_7_200_15_0.8 12 days ago 1.3min C:\Users\... sklearn 0.734 0.81
Foret Aleatoire - 0_1_150_5_200_20_0.8 12 days ago 1.5min C:\Users\... sklearn 0.732 0.807
Gradient boosting tree - 0_1_150_4_60_7_0.8 12 days ago 1.2min C:\Users\... Best GBR/2 0.732 0.809
Regression Logistique 0_1_150_5 12 days ago 41.4s C:\Users\... sklearn 0.731 0.807
Foret Aleatoire - 0_1_150_5_200_15_0.8 12 days ago 1.4min C:\Users\... sklearn 0.73 0.805
Gradient boosting tree - 0_1_150_5_60_7_0.8 12 days ago 1.6min C:\Users\... sklearn 0.73 0.809
Gradient boosting tree - 0_1_150_7_60_7_0.8 12 days ago 1.2min C:\Users\... sklearn 0.73 0.81
Gradient boosting tree - 0_1_150_5_60_5_0.8 12 days ago 1.0min C:\Users\... sklearn 0.726 0.801
Gradient boosting tree - 0_1_150_7_60_5_0.8 12 days ago 1.1min C:\Users\... sklearn 0.722 0.8
Show more columns
(9 total)

Conclusion

Pistes d'amélioration ou d'utilisation

- Sélection du modèle selon qu'on veuille optimiser vrai positif ou négatif
- Meilleur sélection des données d'entraînement pour de meilleurs résultats ou un résultat plus ciblé.