Compétition Kaggle: Natural Language Processing with Disaster Tweets

Présenté par: SEKPONA Kokou Sitsopé,
Etudiant en Ingénierie Machine Learning chez openclassrooms/Central Supélec

Introduction

En tant qu'étudiant Ingénieur Machine Learning, il nous est demandé de participer à une compétition Kaggle de notre choix.
Nous avons donc choisi de participer à ma compétition 'Natural Language Processing with Disaster Tweets' qui est une compétition en traitement naturel de Langage.
Ce projet social rend service énormément à la communauté car permettra aux secouristes d'etre alertés en cas de catastrophe à un endroit et donc vite intervenir, ce qui a suscité notre grand intérêt porté à cette compétition.
Nous allons donc présenter les détails concernant cette compétions à laquelle nous avons participés, les démarches et les résultats obtenus.

Problème

Les problèmes que la compétition tends à résoudre.

01

Existence de quantité énorme de tweets: Environ 6000 tweets par secondes : Impossibilité de classification par un humain

02

Identifier automatiquement un tweet qui parle de catastrophe pour alerter les secouristes

03

Les ordinateurs utilisent le langage binaire, sont donc numériques et ne travaillent qu'avec des nombres.

Contraintes du projet

01

Choix de la compétition: réelle, en cours, Mésurable,

02

S"appuyer sur les kernels partagés par d’autres participants

03

Avoir des étapes pertinents à partager avec la communauté.

Approche de Solution

01

Conversion du text en format numérique

02

Utilisation des modèles de machine leanring pour la classification

03

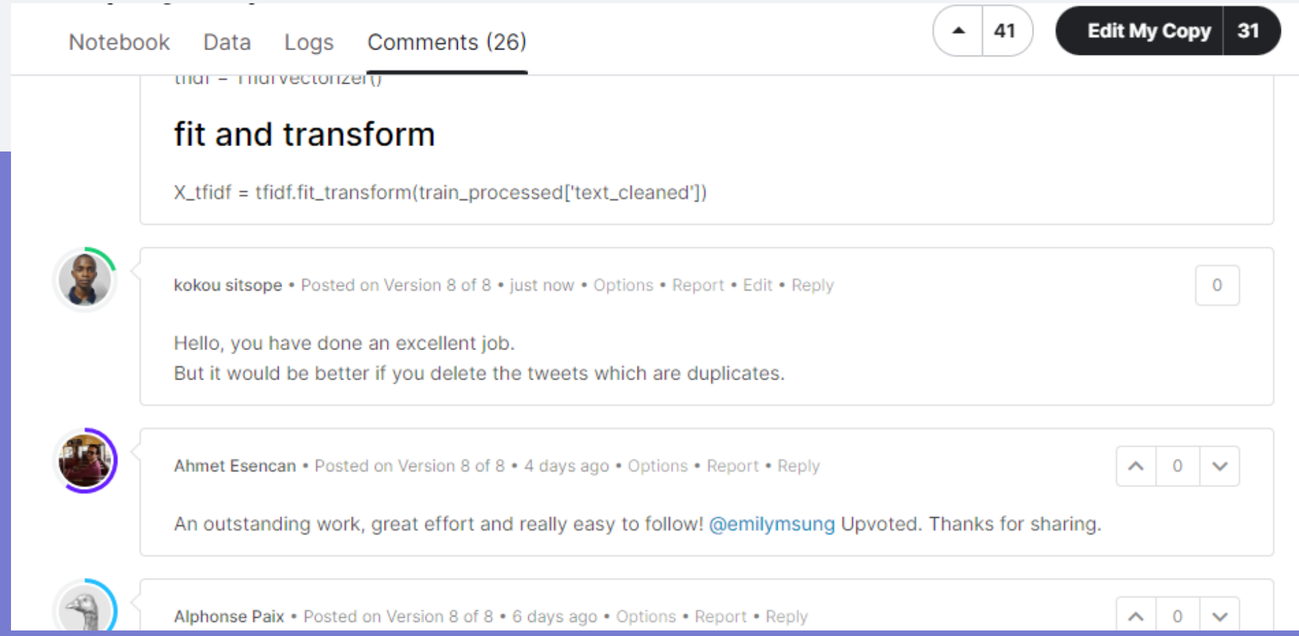
Décodage de la sortie en texte (Castastrope ou non)

Interaction avec la communauté

Nous avons récupérer un notebook de base sur Kaggle que nous avons améliorations comme:

- **Suppression des doublons**
- **Utilisation de différents model comme les Transformers et model pré entrainés.**

Le lien du notebook utilisé: https://www.kaggle.com/code/kokousitsope/step-by-step-nlp-with-disaster-tweets/edit



Plan de Travail

Nettoyage et analyse exploratoire des données

Nettoyage et visualisation de la dataset

Features Extractions

2 methodes: TfidfVectorizer et Doc2vec

Modélisation

Test de differntes models de classification et mésure de la précision

I. Nettoyage et analyse exploratoire des données

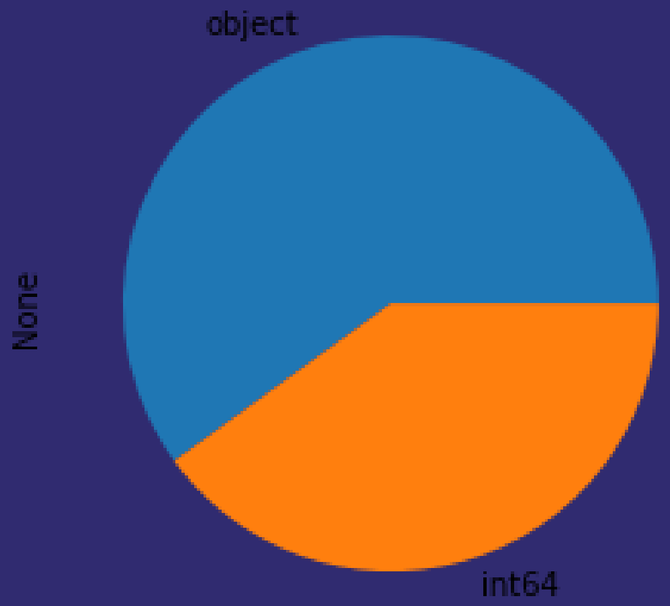
Notre dataset contient de train contient 5 colonnes et 7613 lignes.

La colonne id ne nous serra pas utile. Nous allons la supprimer.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   id        7613 non-null   int64 
 1   keyword   7552 non-null   object
 2   location  5080 non-null   object
 3   text      7613 non-null   object
 4   target    7613 non-null   int64 
dtypes: int64(2), object(3)
memory usage: 297.5+ KB
```

Notre dataset d'entrainement a plusieurs valeurs manquantes au niveay du keyword et de la location de text.
Aucune valeur manquante dans la colonne Text et target.

Types de donnés dans les colonnes et répartition



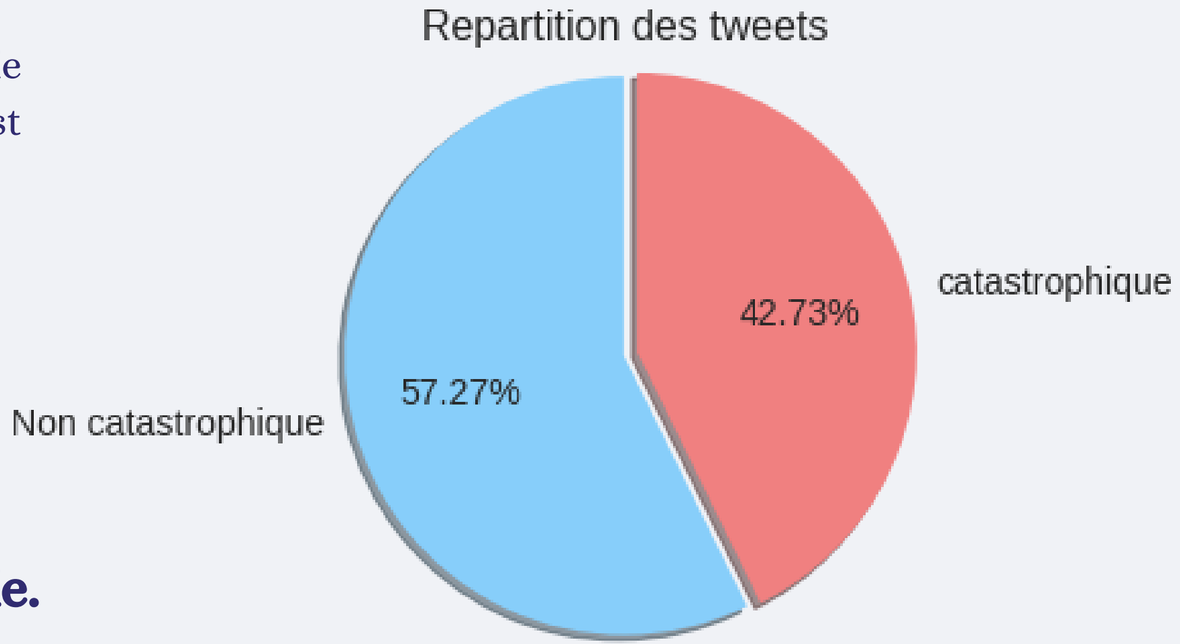
Environ 60% de nos colonnes sont de type object: keywords, text, location
Pres de 40% sont de type int: id et Target

Nous avons au total 70 doublons qui sont des tweet qui ont la meme location, keywords et text.

Nous allons les supprimer.

Répartition de la dataset

Nous observons si la dataset d'entrainement est bien répartie c'est à dire si le nombre d'observations positives est sensiblement égale au nombre d'observations négatives



- **57% de cas négatifs**
- **42% de cas positifs.**

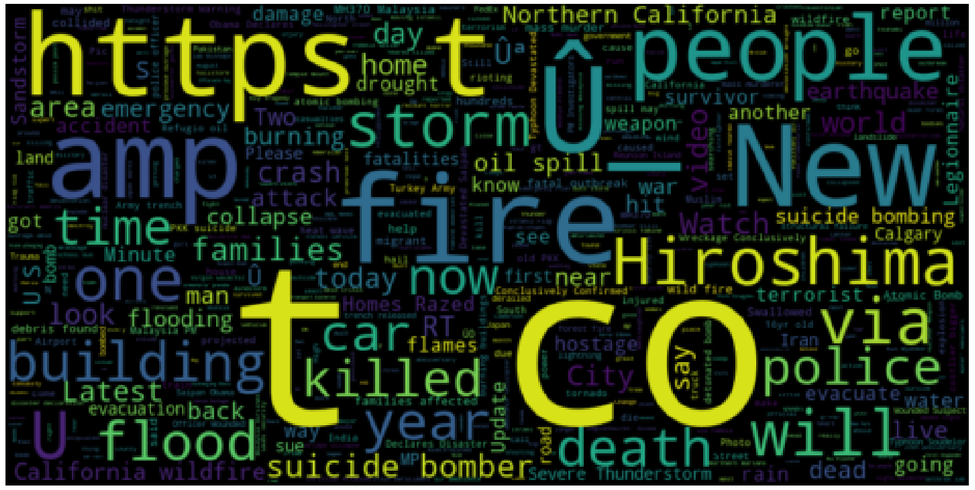
Notre dataset d'entrainement est bien répartie.

Affichage du texte brut par classes avec wordcloud

Nous afichons le texte pour observer les mots qui reviennent suivant par classe

Tweets non catastrophiques

Tweets Catastrophiques



- **Nous observons que les mots « incendie », « inondation » et « tempête » reviennent souvent dans la classe des tweets liés aux catastrophe**

Prétraitement du Test

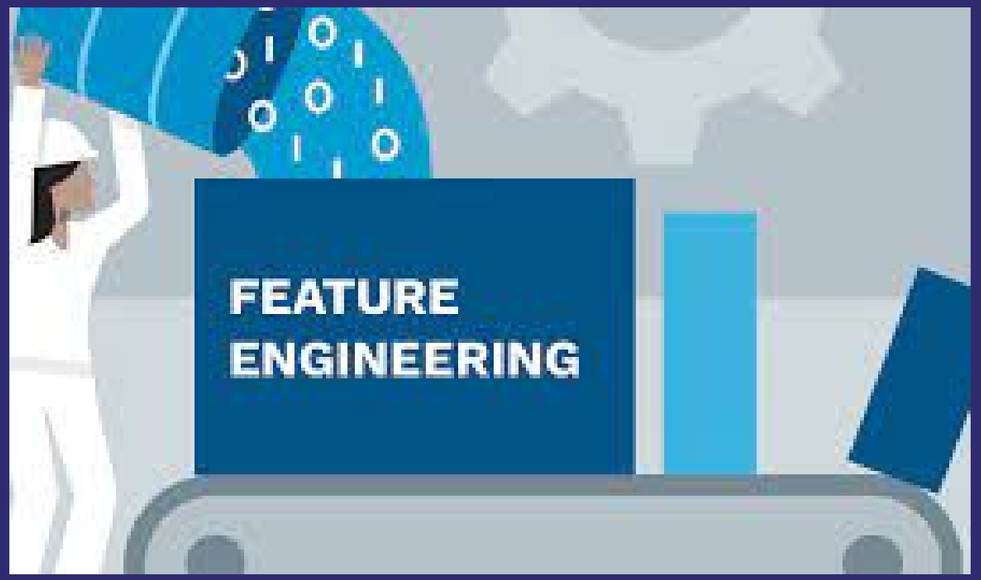
1.
Supression:

- URL
- Balises HTML
- références de personnages
- caractères non imprimables
- valeurs numériques

2.

- lemmatisons le texte
- Conversion en minuscules.
- Supression des caractères répétés dans les mots allongés,
- Supression des mots vides
- Conservation des hashtags car ils peuvent fournir des informations précieuses sur ce projet particulier.

Features Engineering



Nous créons 10 colonnes qui sont :

- Nombre de phrases
- Nombre de mots
- Nombre de caractères
- Nombre de hashtags
- Nombre de mentions
- Nombre de mots tout en majuscules
- Longueur moyenne des mots
- Nombre de noms propres (PROPN)
- Nombre de noms non propres (NOM)
- Pourcentage de caractères qui sont de la ponctuation

II. Extraction des Features

Cette etape va nous permettre d'encoder nos texte en vecteurs que le model peux utiliser pour la prédiction

1. TfidfVectorizer

	__	aa	aan	aand	aar	aashiqui	ab	aba	abandon	abandoned	...	zotar	zouma	zrnf	zss	zumiez	zurich	word_count
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	14.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	8.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	24.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	9.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	18.0

5 rows × 11986 columns

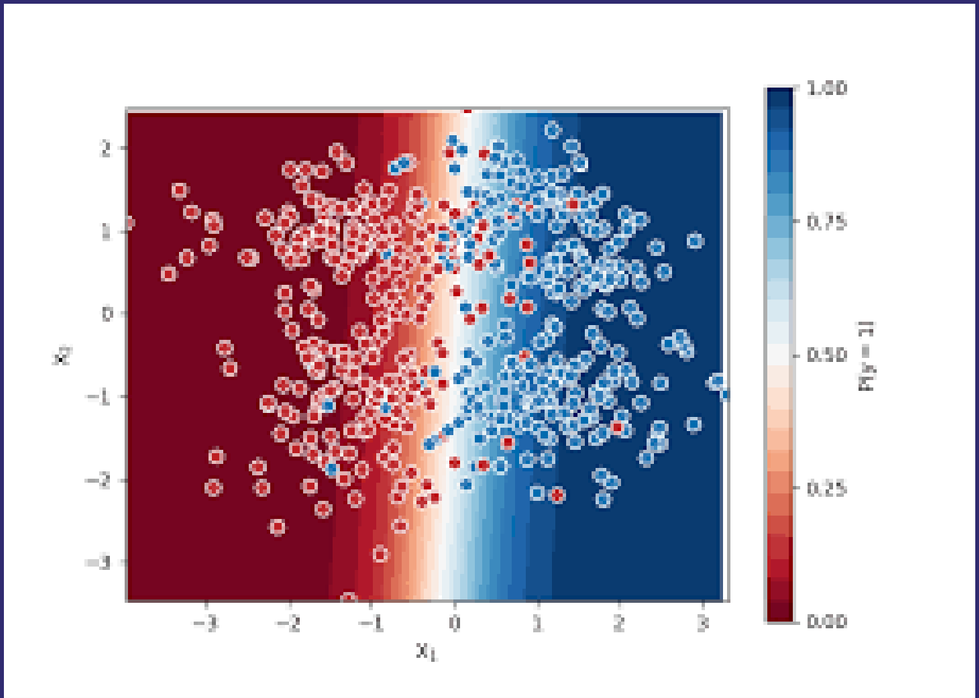
2. Doc2vec

	0	1	2	3	4	5	6	7	8	9	...	90	9
0	-0.002671	-0.013938	-0.002306	-0.009692	0.016702	0.012870	0.016330	0.010423	0.013950	0.003004	...	0.024148	0.01521
1	-0.001862	-0.023862	-0.008151	-0.000994	0.013673	-0.002422	0.030008	-0.005985	0.011767	-0.004005	...	0.018086	0.02007
2	-0.012431	-0.057285	-0.026106	0.020399	0.031340	0.011255	0.040662	0.018954	0.023757	0.014061	...	0.058772	0.01918
3	0.003006	-0.021050	-0.008683	-0.001035	0.020931	0.007115	0.028014	-0.002684	0.011862	-0.004147	...	0.018131	0.00752
4	-0.007600	-0.021517	-0.022451	-0.005334	0.022792	0.000829	0.041967	-0.001303	0.018972	-0.014071	...	0.037112	0.03223

5 rows × 100 columns

III. Modélisation

Nous utilisons 3 models pour la classification



01

Logistic Regressor

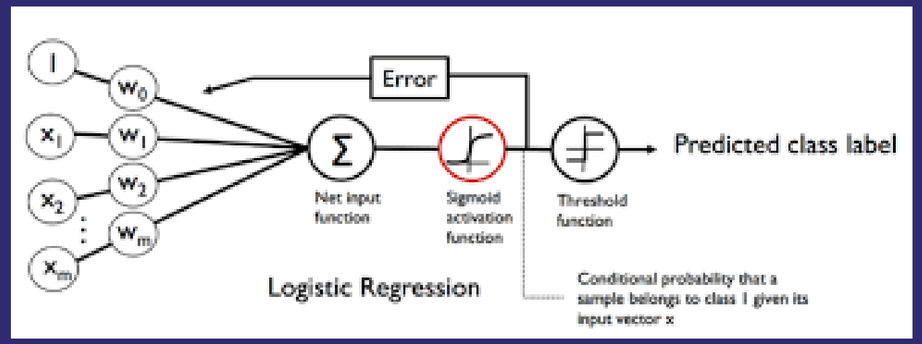
02

Transformers

03

Pre Entrained Model **nnlm-en-dim50**

Logistic Regressor



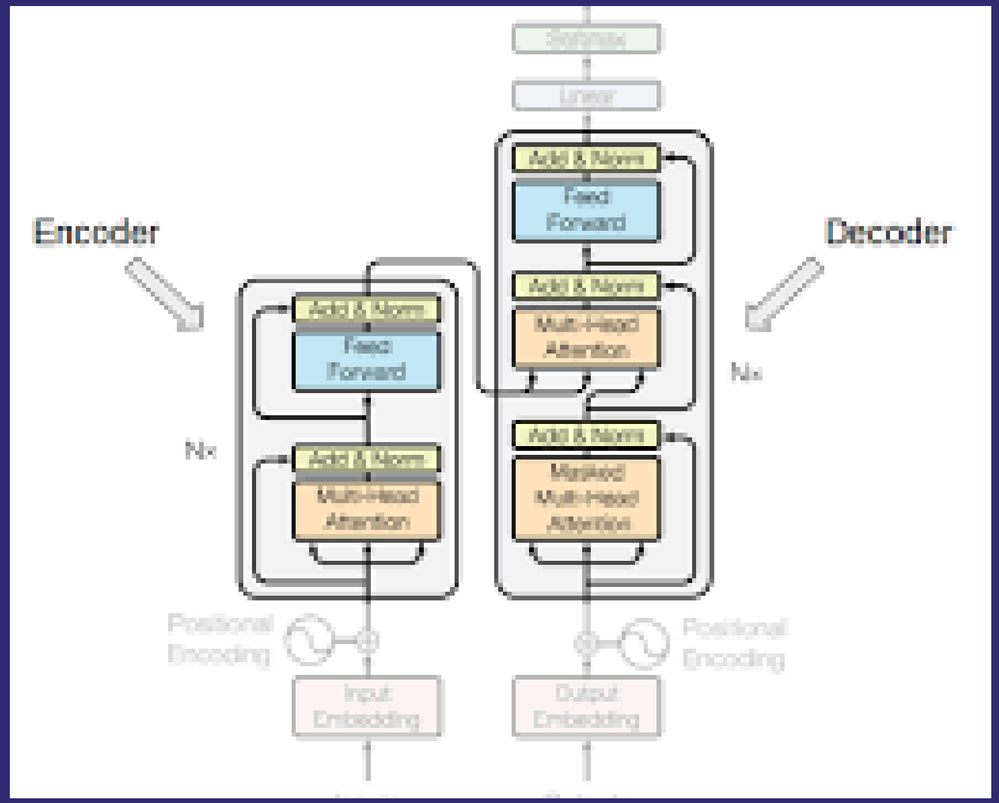
1. With TFIDFVectorizer

- Nous recherchons les meilleurs hyperparametres avec GridSearchCV
- accuracy : 0.7809139784946236

2. With Doc2vec

- Nous recherchons les meilleurs hyperparametres avec GridSearchCV
- accuracy : 0.6503311258278146

Transformers



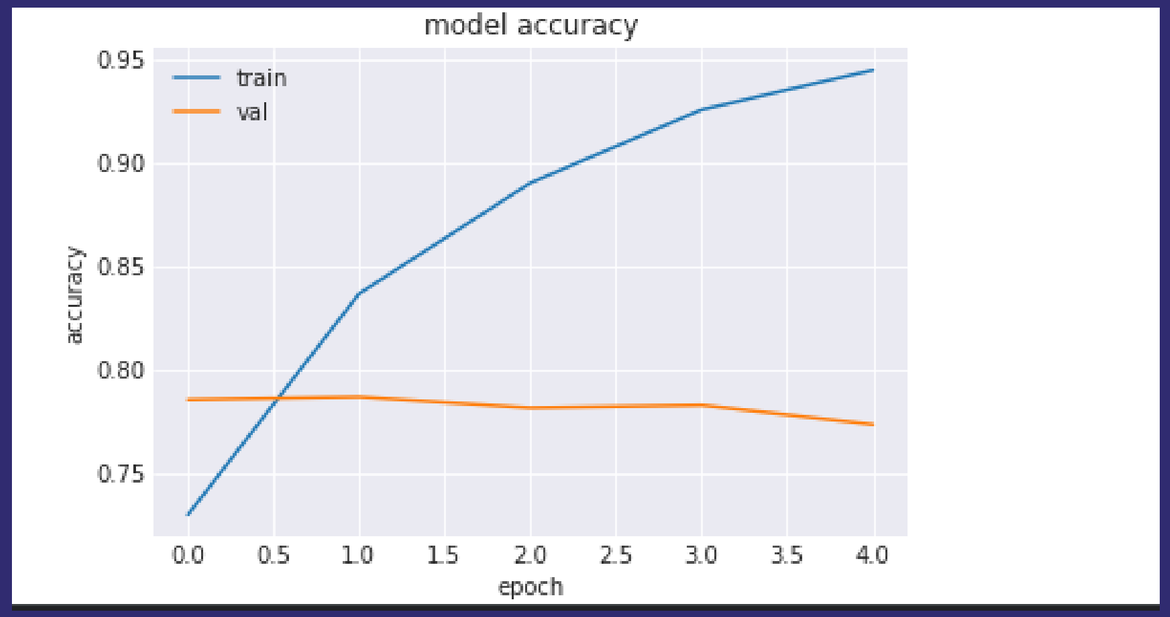
1. With TFIDFVectorizer

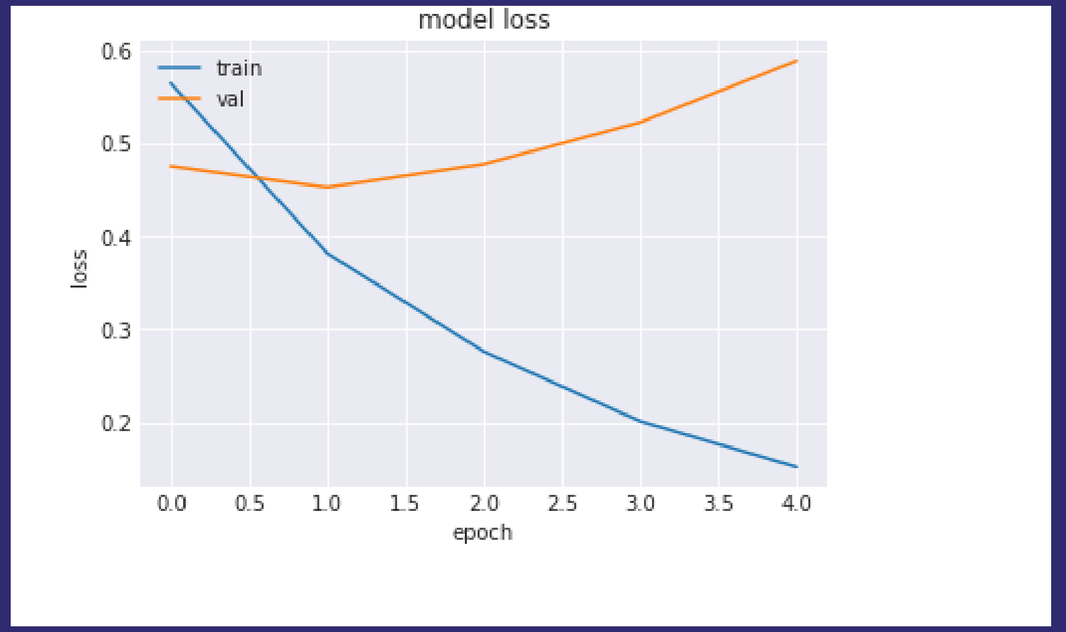
- accuracy : 0.569

2. With Doc2vec

accuracy : 0.558

Pre entrained model nnlm-en-dim50





accuracy: 0.7775

Soumissions

Nous avons soumis plusieurs versions de notre travail sur kaggle : Ce sont les resultats obtenus pour chaque model et extraction de features utilisés

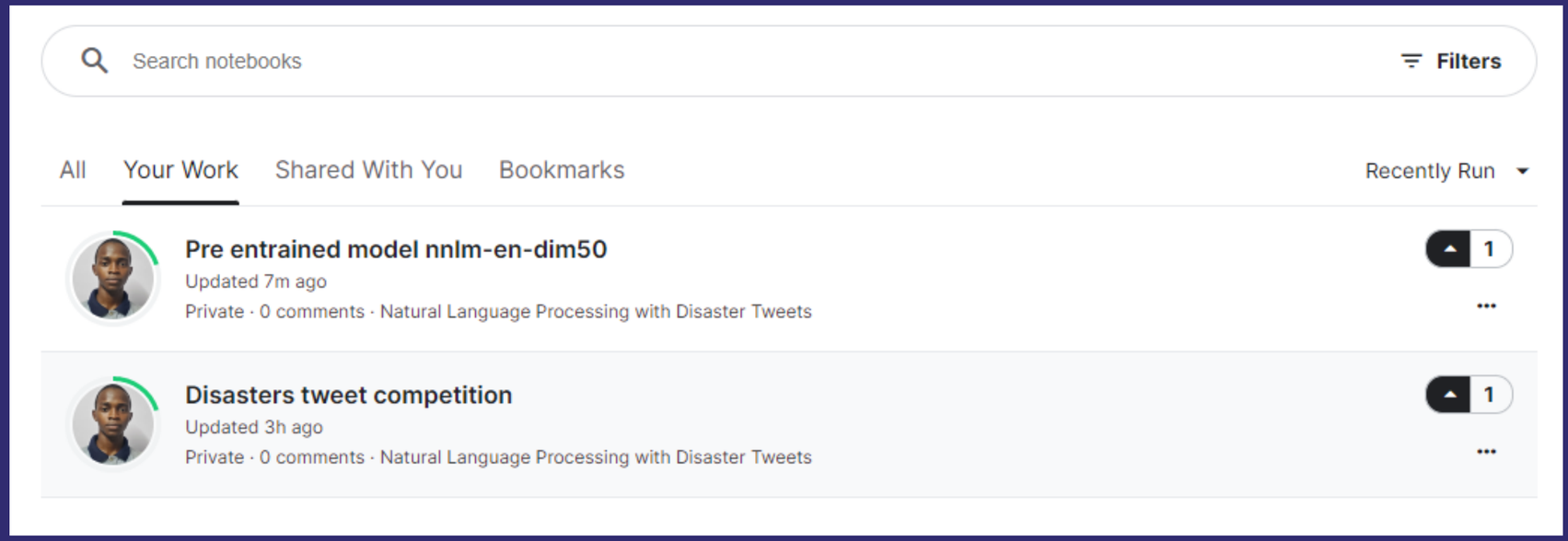
Overview Data Code Discussion Leaderboard Rules Team Submissions **Submit Predictions**

Complete · 9h ago · Tfidf with Transformers	0.42966
sample_submission.csv Complete · 9h ago · Doc2vec logistic Regressor with features engineering columns	0.65337
sample_submission.csv Complete · 2d ago · Doc2vec model with logistic Refression	0.64296
sample_submission.csv Complete · 2d ago · Logistic regression Tfidf Vectorizer commit	0.79098
sample_submission.csv Complete · 2d ago · Logistic Regression with Tfidf Vectorizer	0
sample_submission.csv	

Au total 10 soumissions dont un échec.
Le meilleur score obtenu est de 79.098

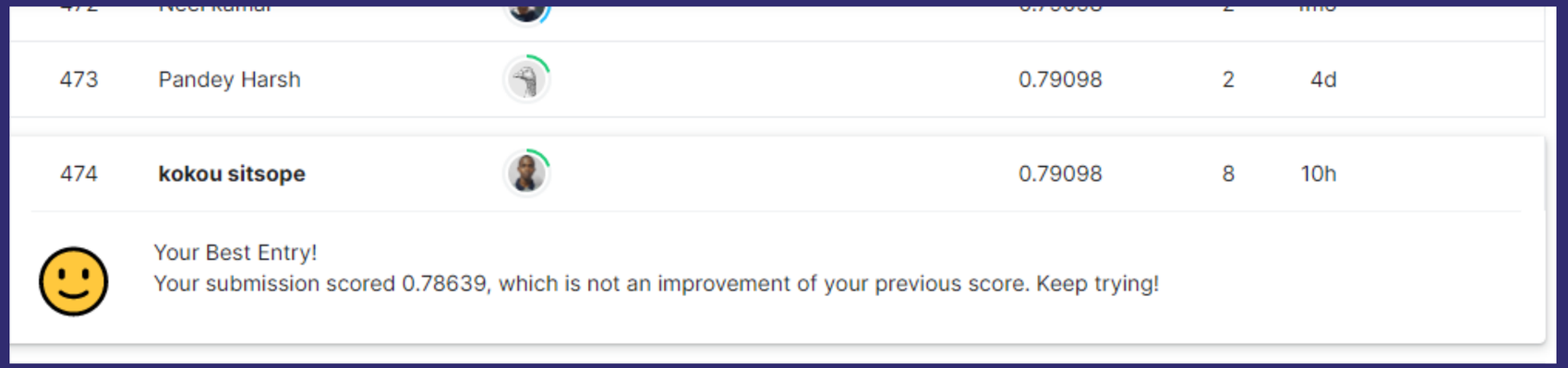
Partage du code avec la communauté

Apres notre travail, nous avons paragé avec la communauté le model pré entrainé découvert ainsi que notre code en jupyter notebook



Conclusion et Rang

Dans tous les models éssayés, le Logistic Regressor utilissé evc le TFIDFVectorizer a eu le meilleur resultat, ce qui nous a permis d'avoir un rang de 474ème sur un total de 870 participants.



Merci !