



UNIVERSITÉ SIDI MOHAMED BEN ABDLLAH
FACULTÉ DES SCIENCES DHAR EL MAHRAZ DE FÈS



DÉPARTEMENT D'INFORMATIQUE



DATA MINING

Text Dataset Pre-Processing

Présenter par :
ABIBOU SOUKAYNA

Supervisé par :
RIFFI JAMAL

Année universitaire : 2020/2021

Filière MIDVI

I- Introduction :

Les données texte doivent être nettoyées et encodées en valeurs numériques avant de les transmettre à des modèles d'apprentissage automatique, ce processus de nettoyage et d'encodage est appelé *prétraitement de texte*.

Dans ce noyau, nous allons voir quelques étapes et techniques de base de nettoyage de texte pour encoder des données de texte. Nous allons voir :

- 1- **Comprendre les données** – Découvrez en quoi consistent les données. Ce qui doit être pris en compte pour le nettoyage des données (ponctuations, mots vides, etc.).
- 2- **Nettoyage de base** – Nous verrons quels paramètres doivent être pris en compte pour le nettoyage des données (comme les ponctuations, les mots vides, etc.) et son code.

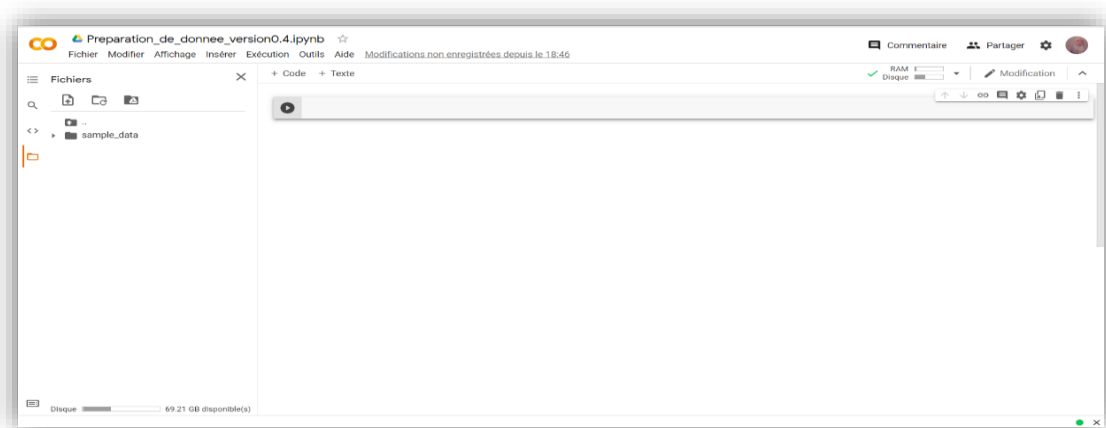
NB : Nous travaillerons sur l'environnement Google colab.

II- Prétraitement de texte :

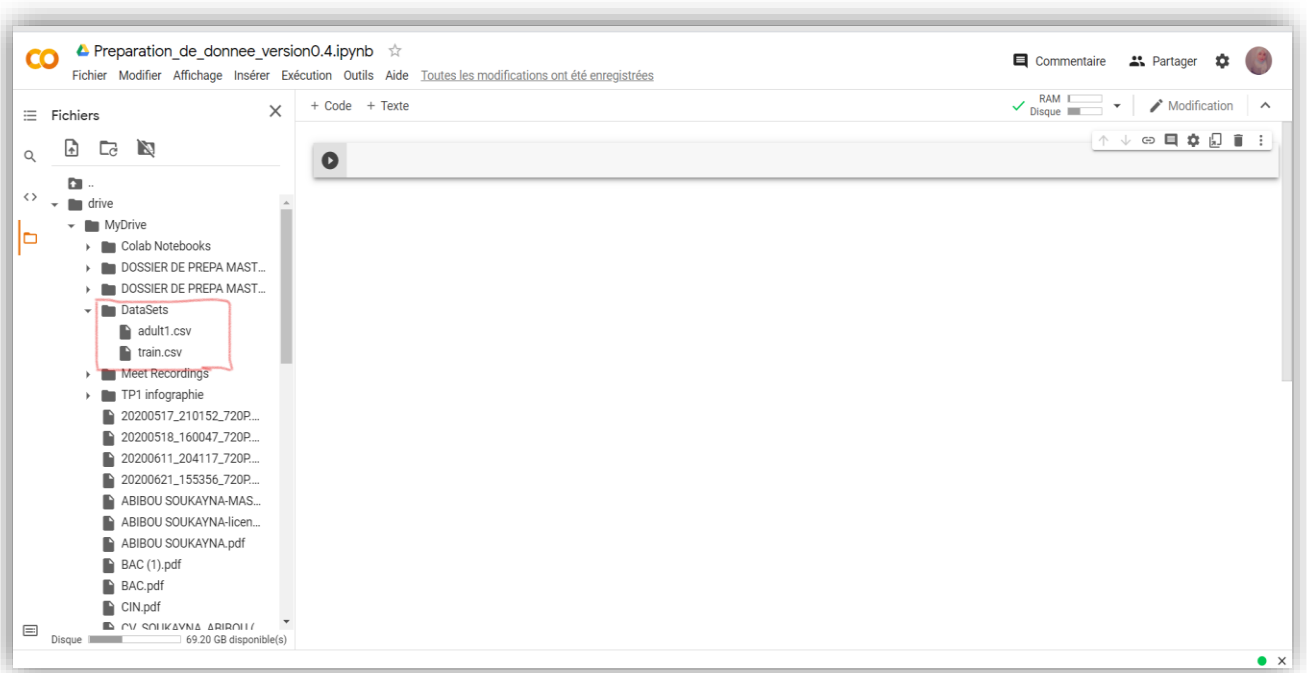
1. Télécharger les données :

Maintenant, l'ensemble de données que nous allons utiliser concerne "Android dmv in size", et si nous téléchargeons cet ensemble de données, cela peut prendre un certain temps, alors que devons-nous faire? Nous pouvons télécharger cet ensemble de données dans notre Google Drive.

La première étape consiste à télécharger notre ensemble de données dans cet environnement Google colab.

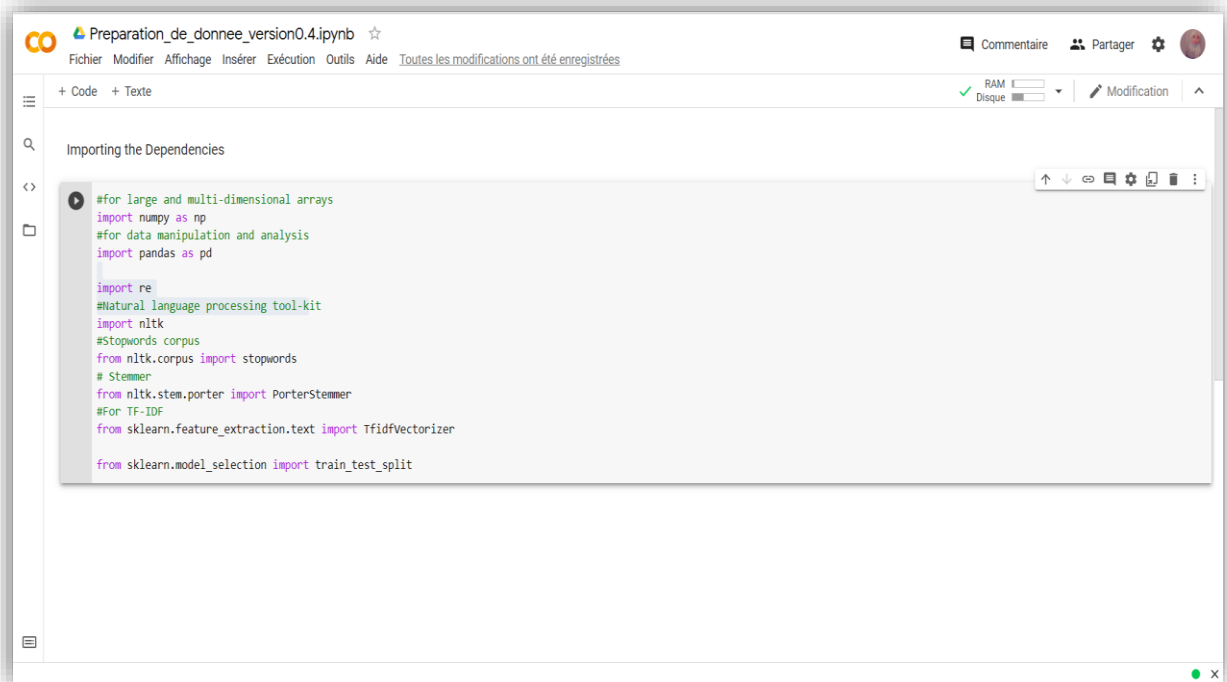


Alors vous pouvez accéder à ces options de lecteur de drive.



Lorsque vous donnez ce lecteur, votre Google drive sera lié à votre compte collaboratif Google, l'important est ce compte dans votre Google drive et dans Google Collaborate devrait être le même compte.

2. Importation de bibliothèques:



```
#for large and multi-dimensional arrays
import numpy as np
#for data manipulation and analysis
import pandas as pd

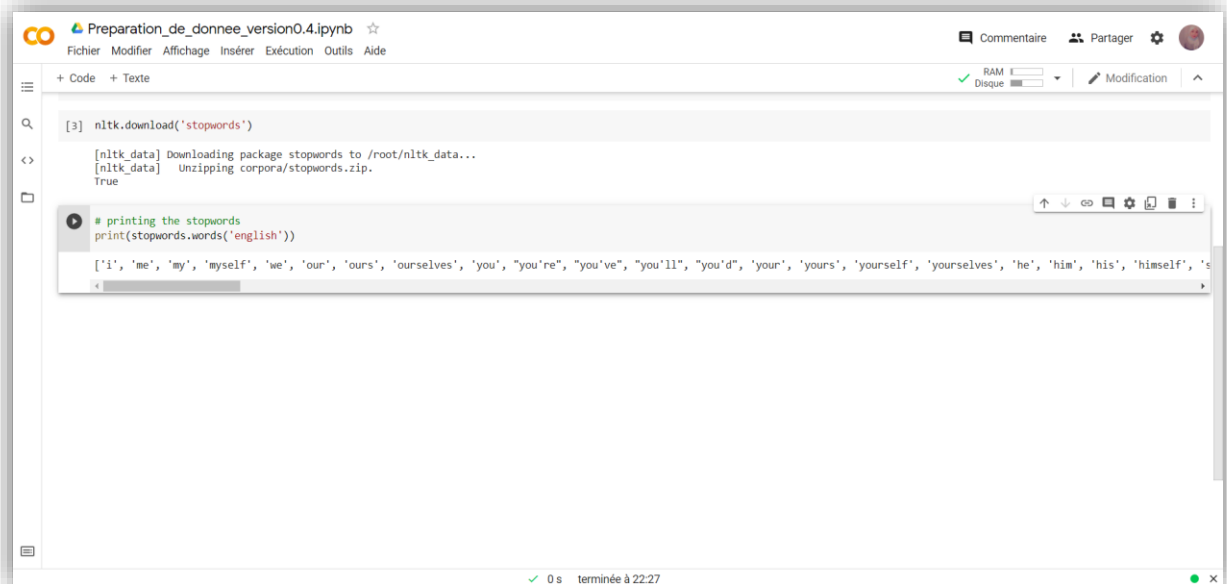
import re
#Natural language processing tool-kit
import nltk
#Stopwords corpus
from nltk.corpus import stopwords
# Stemmer
from nltk.stem.porter import PorterStemmer
#For TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split
```

« **Stopwords** », sont des mots qui peuvent être répétés plusieurs fois dans un paragraphe ou dans un document, mais ces mots ne véhiculent pas beaucoup de sens, nous devons donc les supprimer de notre **dataset**.

D’abord, nous devons télécharger ces « **Stopwords** » :

Et maintenant nous pouvons voir toute la liste des « **Stopwords** ».



The screenshot shows a Jupyter Notebook interface with the title 'Preparation_de_donnee_version0.4.ipynb'. The code cell contains two lines of Python code: `nltk.download('stopwords')` and `print(stopwords.words('english'))`. The output of the first line shows the download and unzipping process. The output of the second line displays a list of English stopwords.

```
[3] nltk.download('stopwords')

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
True

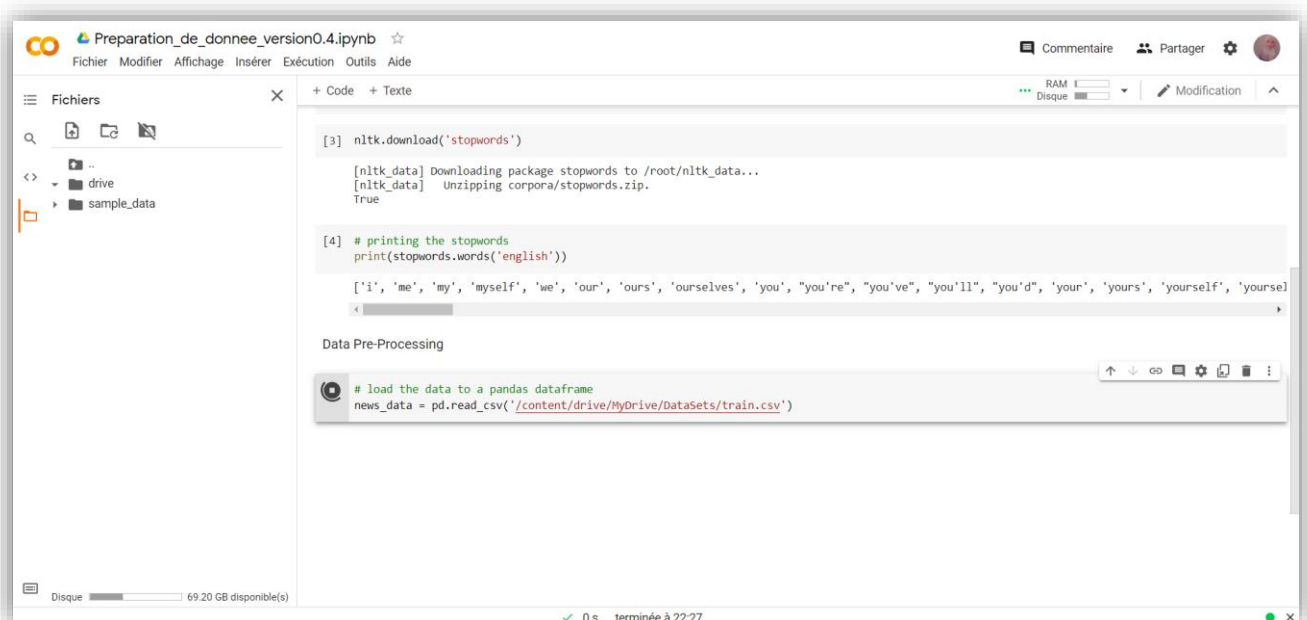
# printing the stopwords
print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 's'
```

3. Nettoyage du data :

Maintenant nous pouvons passer à l'étape de prétraitement des données.

Tout d'abord, nous chargeons notre **dataset** dans un **dataframe** pandas.



The screenshot shows the same Jupyter Notebook interface. The code cell now includes a third line of code: `news_data = pd.read_csv('/content/drive/MyDrive/DataSets/train.csv')`. The output of this line is a message indicating the file path. The file explorer on the left shows the 'sample_data' folder.

```
[3] nltk.download('stopwords')

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
True

[4] # printing the stopwords
print(stopwords.words('english'))

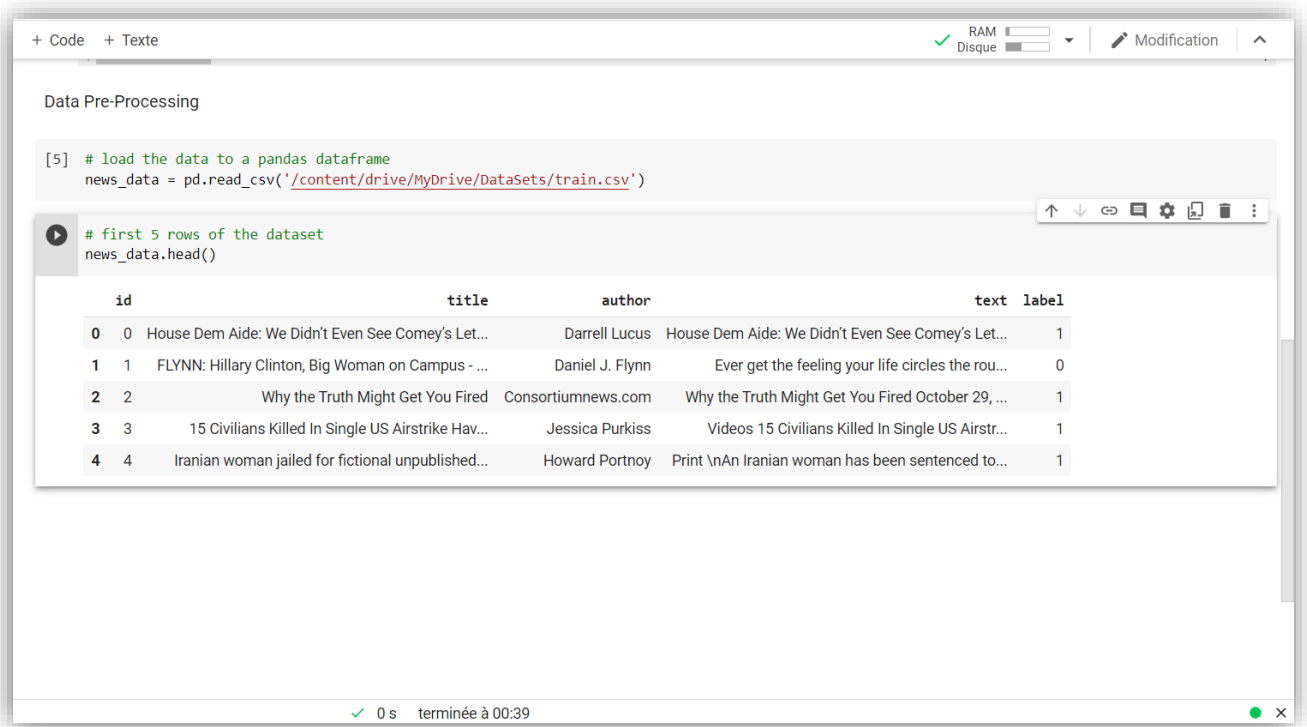
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yoursel]

Data Pre-Processing

# load the data to a pandas dataframe
news_data = pd.read_csv('/content/drive/MyDrive/DataSets/train.csv')
```

Cela créera un nouveau dataframe qui contient toutes ces données.

Nous pouvons afficher les cinq premières lignes des données :



The screenshot shows a Jupyter Notebook interface with a tab labeled 'Code'. The notebook title is 'Data Pre-Processing'. The code cell contains two lines of Python code:

```
[5] # load the data to a pandas dataframe
news_data = pd.read_csv('/content/drive/MyDrive/DataSets/train.csv')
```

Below the code, the output shows the first 5 rows of the dataset using `news_data.head()`. The output is a table with 5 rows and 5 columns: `id`, `title`, `author`, `text`, and `label`.

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucas	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Aistr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

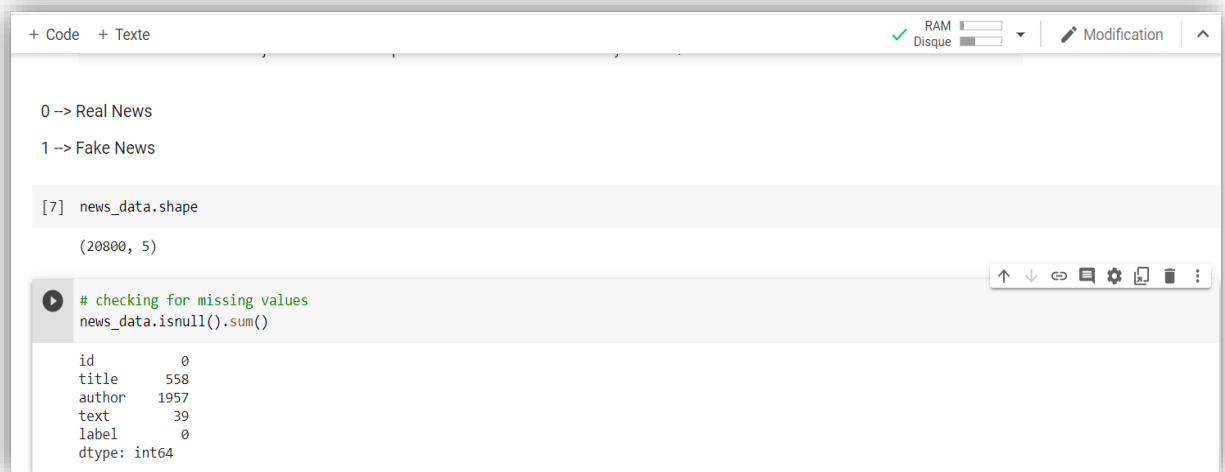
The status bar at the bottom indicates '0 s terminée à 00:39'.

C'est toujours une bonne pratique de mentionner ce que vous faites dans le code pour le rendre compréhensible !!!!

Donc nous avons : id, title, author, text et label.

- ✓ Si « label » est 0, cela signifie que les nouvelles sont vraies et si « label » est 1, les nouvelles sont fausses.
- ✓ L'idée derrière cet dataset est de former un modèle d'apprentissage automatique pour lui permettre de comprendre quelles nouvelles sont fausses et quelles nouvelles peuvent être réelles.

Nous pouvons voir le nombre total de points de données dont nous disposons, aussi si cet dataset contient des valeurs manquantes dans chaque colonne ;



The screenshot shows a Jupyter Notebook window with a toolbar at the top indicating RAM and disk usage, and a 'Modification' button. The code cell contains the following:

```
0 -> Real News
1 -> Fake News

[7]: news_data.shape

(20800, 5)

# checking for missing values
news_data.isnull().sum()
```

The output of the last cell is a summary of missing values for each column:

Column	Count
id	0
title	558
author	1957
text	39
label	0

The output also includes the dtype: int64.

- ✓ Au total, nous avons vingt mille huit cents lignes et cinq colonnes.
- ✓ Ayant vingt mille huit cents lignes, cela signifie que nous avons de nombreuses utilisations différentes, nous appelons donc cela également « **data point** », c'est un ensemble de données assez volumineux mais ce n'est pas le plus grand.
- ✓ Si votre dataset est volumineux, vous pouvez faire de meilleures prédictions avec votre modèle, **plus il y a de données, les performances du modèle sont meilleures.**
- ✓ Nous pouvons voir que nous avons beaucoup de valeur manquante.
- ✓ Si une valeur manque, elle sera représentée par nan donc tout nan signifie n'est pas un nombre.
- ✓ Donc, dans ce cas particulier, nous pouvons remplacer toutes les valeurs manquantes par une chaîne nulle.

```

+ Code + Texte
# checking for missing values
news_data.isnull().sum()

id      0
title   558
author  1957
text     39
label    0
dtype: int64

# replacing the missing values with null string
news_data = news_data.fillna('')

# verification
news_data.isnull().sum()

id      0
title    0
author   0
text     0
label    0
dtype: int64

```

Nous allons analyser notre dataset uniquement avec deux fonctionnalités « title » et « text ».

- Nous allons combiner ces valeurs dans une autre colonne.

```

+ Code + Texte
[12] # merging the author name and news title
news_data['content'] = news_data['author']+' '+news_data['title']

# first 5 rows of the dataset
news_data.head()

```

	id	title	author	text	label	content
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1	Darrell Lucus House Dem Aide: We Didn't Even S...
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0	Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1	Consortiumnews.com Why the Truth Might Get You...
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Aistr...	1	Jessica Purkiss 15 Civilians Killed In Single ...
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1	Howard Portnoy Iranian woman jailed for fictio...

- Maintenant, nous allons utiliser cette colonne « content » pour un traitement ultérieur.

- Nous ferons ce prétraitement de texte sur cette colonne particulière et pas pour les autres.
- Séparons les deux colonnes importantes, donc ici nous avons besoin de cette colonne « **conten**t » et de cette colonne « **label** », nous n'avons besoin d'aucune autre colonne.

The screenshot shows a Jupyter Notebook with the following code and output:

```
[14] # separating feature and target
X = news_data.drop(columns='label', axis =1)
Y = news_data['label']
```

Below the code, the output of `print(X)` is displayed as a table with 5 columns: `id`, `...`, `...`, `...`, and `content`. The first few rows are:

	id	content
0	0	Darrell Lucus House Dem Aide: We Didn't Even S...
1	1	Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2	2	Consortiumnews.com Why the Truth Might Get You...
3	3	Jessica Purkiss 15 Civilians Killed In Single ...
4	4	Howard Portnoy Iranian woman jailed for fictio...

The output indicates there are 20800 rows and 5 columns. Below this, the output of `print(Y)` is shown as a single column of numerical values:

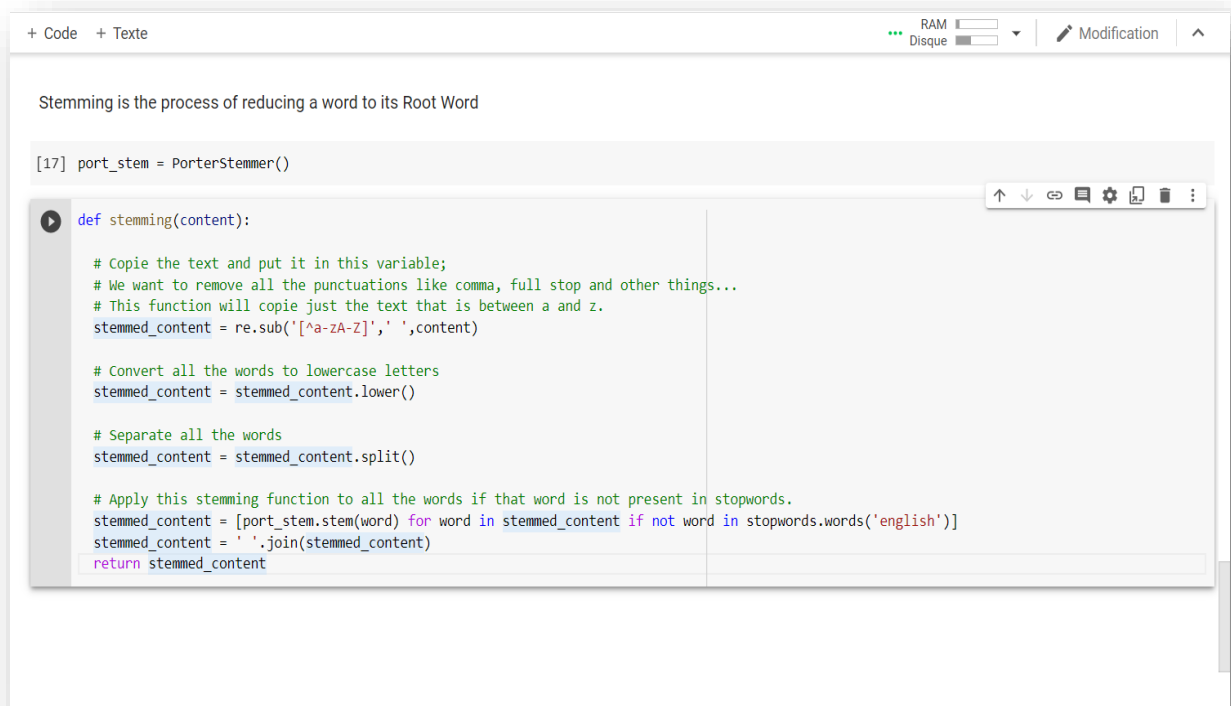
0	1
1	0
2	1
3	1
4	1
...	...
20795	0
20796	0
20797	0

- Maintenant nous pouvons faire notre traitement sur ce X.
- Y n'a besoin d'aucun traitement car il se présente déjà sous forme de valeurs numériques.
- Le processus que nous allons suivre appeler **Stemming**.

❖ Stemming :

Stemming est le processus de réduction d'un mot à son mot racine.

- Mot racine signifie par exemple : les mots « **enjoyable** », « **enjoying** », « **enjoyed** » ces mots leur mot racine est « **enjoy** ».
- Nous utilisons ce traitement car les mots racines sont petits et le traitement peut être plus rapide et ce sera plus facile.



The screenshot shows a Jupyter Notebook interface. At the top, there are tabs for '+ Code' and '+ Texte'. On the right, there are indicators for RAM and Disque usage, and a 'Modification' button. The main area contains a text box with the sentence: 'Stemming is the process of reducing a word to its Root Word'. Below this, there is a code cell with the following Python code:

```
[17] port_stem = PorterStemmer()

def stemming(content):
    # Copie the text and put it in this variable;
    # We want to remove all the punctuations like comma, full stop and other things...
    # This function will copie just the text that is between a and z.
    stemmed_content = re.sub('[^a-zA-Z]', ' ', content)

    # Convert all the words to lowercase letters
    stemmed_content = stemmed_content.lower()

    # Separate all the words
    stemmed_content = stemmed_content.split()

    # Apply this stemming function to all the words if that word is not present in stopwords.
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
```

- Maintenant nous pouvons appliquer cette fonction sur notre colonne :

```

+ Code + Texte
✓ RAM
Disque
Modification

[20] news_data['content'] = news_data['content'].apply(stemming)

print(news_data['content'])

0      darrel lucu hous dem aid even see comey letter...
1      daniel j flynn flynn hillari clinton big woman...
2      consortiumnew com truth might get fire
3      jessica purkiss civilian kill singl us airstri...
4      howard portnoy iranian woman jail fiction unpu...
...
20795   jerom hudson rapper trump poster child white s...
20796   benjamin hoffman n f l playoff schedul matchup...
20797   michael j de la merc rachel abram maci said re...
20798   alex ansari nato russia hold parallel exercis ...
20799   david swanson keep f aliv
Name: content, Length: 20800, dtype: object

```

- Maintenant nous avons les données de texte importantes, nous pouvons convertir ces données en valeur numérique.
- Séparez les données de leurs **features** et **target** correspondantes :

```

+ Code + Texte
name: content, length: 20800, dtype: object
✓ RAM
Disque
Modification

[22] X = news_data['content'].values
     Y = news_data['label'].values

[23] print(X)

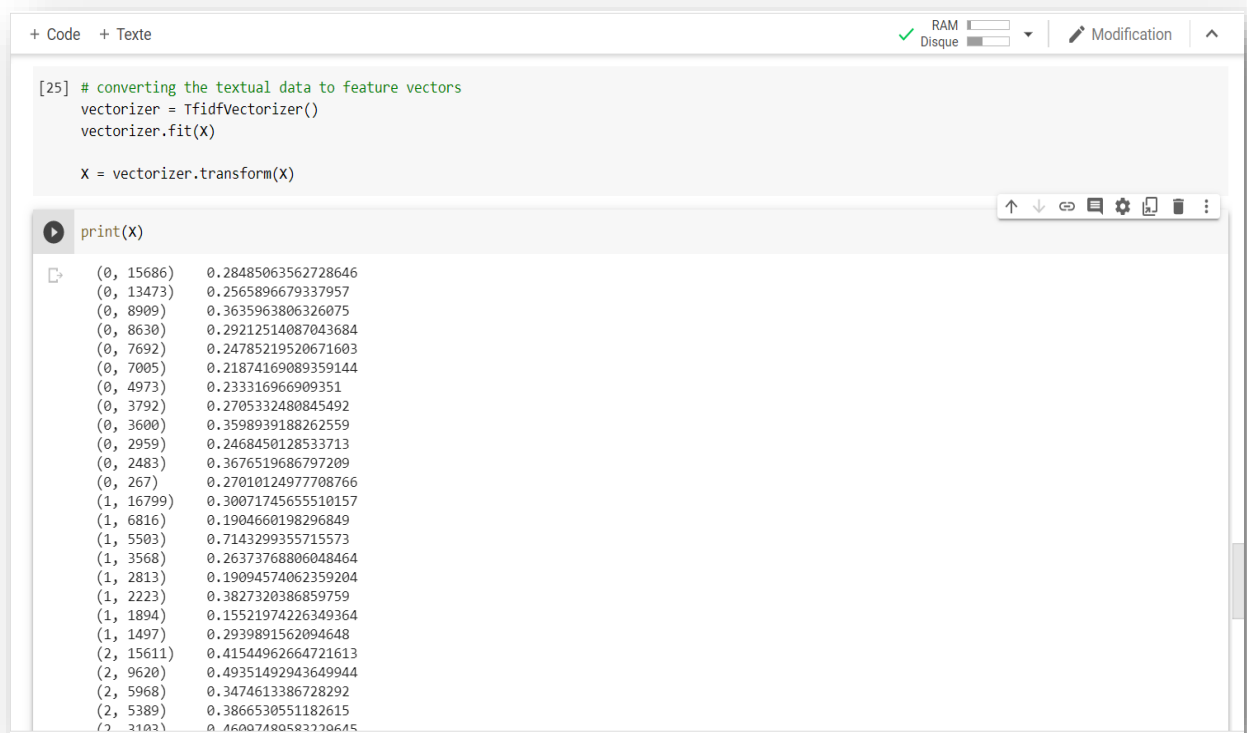
['darrel lucu hous dem aid even see comey letter jason chaffetz tweet'
'daniel j flynn flynn hillari clinton big woman campu breitbart'
'consortiumnew com truth might get fire' ...
'michael j de la merc rachel abram maci said receiv takeov approach hudson bay new york time'
'alex ansari nato russia hold parallel exercis balkan'
'david swanson keep f aliv']

print(Y)

[1 0 1 ... 0 1 1]

```

- Nous allons convertir tous ces mots :



The screenshot shows a Jupyter Notebook with two cells. The first cell contains code for converting textual data to feature vectors using TFIDF. The second cell displays the output of the `print(X)` command, showing a list of (word_id, value) pairs.

```
[25] # converting the textual data to feature vectors
vectorizer = TfidfVectorizer()
vectorizer.fit(X)

X = vectorizer.transform(X)
```

print(X)

(0, 15686)	0.28485063562728646
(0, 13473)	0.2565896679337957
(0, 8909)	0.3635963806326075
(0, 8630)	0.29212514087043684
(0, 7692)	0.24785219520671603
(0, 7005)	0.21874169089359144
(0, 4973)	0.233316966909351
(0, 3792)	0.2705332480845492
(0, 3600)	0.3598939188262559
(0, 2959)	0.2468450128533713
(0, 2483)	0.3676519686797209
(0, 267)	0.27010124977708766
(1, 16799)	0.30071745655510157
(1, 6816)	0.1904660198296849
(1, 5503)	0.7143299355715573
(1, 3568)	0.26373768806048464
(1, 2813)	0.19094574062359204
(1, 2223)	0.3827320386859759
(1, 1894)	0.15521974226349364
(1, 1497)	0.2939891562094648
(2, 15611)	0.41544962664721613
(2, 9620)	0.49351492943649944
(2, 5968)	0.3474613386728292
(2, 5389)	0.3866530551182615
(2, 2102)	0.4600718055822720645

- Maintenant tous les mots ont leurs valeurs numériques correspondantes.

III- Conclusion :

Tout au long de ce projet, nous avons vu différentes techniques pour encoder des données textuelles en vecteurs numériques.

Mais la technique appropriée pour notre modèle d'apprentissage automatique dépend de la structure des données et de l'objectif de notre modèle.

Webographie

➤ Code :

<https://colab.research.google.com/drive/1M41Atvk55hbnnsoZ4II7F-Rr3-39Pxv?usp=sharing>

➤ Autre méthode :

<https://drive.google.com/file/d/1yOWmtBHtEfRXOLWAvUjrceLehGAi pwu0/view?usp=sharing>