



Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté Informatique

Département IA & SD

Filière : Informatique

Spécialité : SII

Analyse des sentiments de Tweets : Une approche par réseau de neurones & apprentissage Profond

Réalisé par :

TAOUCI Kenza

OUCHAOU Chafaa

Professeur :

H. MOULAI

Table des matières

Table de figure :	- 4 -
Chapitre 1	6
1. Introduction :	6
- Contexte et objectifs.....	6
- Qu'est-ce qu'une analyse de sentiments sur twitter?	6
-Pourquoi l'analyse des sentiments est-elle importante ?	6
-Pourquoi utiliser Twitter pour l'analyse des sentiments ?	7
- Introduction au NLP	7
2- L'analyse de données	8
-Pourquoi l'analyse des données est-elle importante ?	8
-Affichage de quelques exemples de tweets	8
-Analyse de la distribution des tweets	8
-Analyse exploratoire des données (moyennes, quartiles, etc.)	9
Chapitre 2	11
1- La préparation des données	11
-Nettoyage des tweets.....	11
1. Conversion en Minuscule.....	11
2. Remplacement des emojis par leurs significations textuelles	11
3. Suppression des emails/mentions (avant la ponctuation)	12
4. Suppression des URLs	12
5. Suppression des balises HTML	13
6. Suppression de la ponctuation et des non-mots	13
7. Remplacement des abréviations pour une meilleure compréhension	13
8. Elimination des mots vides	14
9. Suppression de lettres consécutives.....	15
10. Suppression des chiffres	16
11. Tokenisation du texte des tweets et Radicalisation des mots.....	16
12. Lemmatisation	18
2-La visualisation des données en utilisant Word Cloud	19
3-Construction du vocabulaire.....	20
1. La création du fichier vocab.txt.....	20

2. Justification du choix K.....	21
3. Mapping des mots vers les index.....	26
4-Extraction des caractéristiques.....	27
Extraction de Caractéristiques pour la Classification de Tweets	28
1. Division des Données	28
2. Méthodes d'Extraction de Caractéristiques.....	28
3. Résultats et Analyse	30
Chapitre 3	31
1-Classification	31
- Utilisation des Classifiers Machine Learning.....	31
1. L'évaluation des modèles de classification	31
2. Justification du choix de la librairie sklearn :	32
3. Bernoulli NB.....	32
4. MVS (LinearSVC).....	37
5. Régression logistique	41
6. KNN	44
7. Décision Tree	47
8. Naïve Bayes	50
9. Réseau de neurones Classique.....	53
10. Réseau neuronal récurrent LSTM	57
Conclusion	60
Bibliography	61

Table de figure :

Figure 1: Analyse de sentiments sur twitter	6
Figure 2:Utilisation de twitter	7
Figure 3:Exemples de tweets	8
Figure 4:Diagramme circulaire des classes de tweets	9
Figure 5:La distribution des données.....	9
Figure 6:La distribution de la longueur de texte pour les tweets positifs	10
Figure 7:La distribution de la longueur de texte pour les tweets négatifs	10
Figure 8:Exemple avant Conversion en Minuscule.....	11
Figure 9::Exemple après Conversion en Minuscule.....	11
Figure 10:Exemple de tweets avec les emojis symbolises.....	12
Figure 11:Exemple de tweets avec les emojis écrits en texte	12
Figure 12:Exemple de tweets après la suppression des mentions et des URLS	12
Figure 13:Liste de ponctuations de String	13
Figure 14:Exemple des tweets après la suppression de la ponctuation et des non mots.....	13
Figure 15:L'exemple avec les abréviations	14
Figure 16:L'exemple après le remplacement des abréviations	14
Figure 17:Liste des stopwords dans la langue anglaise	15
Figure 18: Les exemples de tweets après la suppression des mots vides	15
Figure 19:Principe de la tokenisation	16
Figure 20:Exemple de tokenisation	17
Figure 21:Exemple de tweets avant tokenisation et radicalisation.....	17
Figure 22: Des exemples de tweets après radicalisation.....	18
Figure 23: Le temps CPU de la radicalisation.....	18
Figure 24:Difference entre le stemming et la lemmatisation.....	19
Figure 25:Nuage de mots des tweets positifs.....	19
Figure 26:Nuage de mots des tweets négatifs.....	20
Figure 27:Frequence des mots par ligne.....	21
Figure 28:Frequence des mots par ligne dans le corpus de tweets	22
Figure 29:Histogramme de distribution des fréquences des mots.....	23
Figure 30:Distribution détaillée des fréquences des mots	24
Figure 31:Nombre de mots dans le vocabulaire K=200	25
Figure 32:Nombre de mots dans le vocabulaire K=500	25
Figure 33:Nombre de mots dans le vocabulaire K=600	25
Figure 34:Nombre de mots dans le vocabulaire K=700	25
Figure 35:Nombre de mots dans le vocabulaire K=1000	25
Figure 36: Le mapping d'index	27
Figure 37:Separation des caractéristiques et labels	28
Figure 38:division des données pour l'entrainement et le test.....	28
Figure 39:extraction de caractéristiques par comptage.....	29
Figure 40:extraction de caractéristiques par TF-IDF.....	29
Figure 41: Recherche sur grille et validation croisée pour BernoulliNB model	35

Figure 42: Résultats de la recherche sur grille avec BernoulliNB	35
Figure 43: Evaluation de BernoulliNB TF-IDF	36
Figure 44: Matrice de confusion BernoulliNB TF-IDF	36
Figure 45: Evaluation de BernoulliNB avec CountVectorizer	37
Figure 46: Evaluation du modèle LinearSVC avec TF-IDF	39
Figure 47: Matrice de confusion de SVC avec TF-IDF	39
Figure 48: Evaluation du modèle LinearSVC avec CountVectorizer	40
Figure 49: Matrice de confusion de SVC avec CountVectorizer	40
Figure 50: Recherche sur grille et validation croisée pour le modèle Régression logistique	42
Figure 51: Meilleurs paramètres	43
Figure 52: Evaluation du modèle de régression logistique avec TF-IDF	43
Figure 53: Matrice de confusion du Modèle de régression linéaire avec TF-IDF	43
Figure 54: Evaluation du modèle de régression logistique avec CountVectorizer	43
Figure 55: Matrice de confusion de la régression logistique avec CountVectorizer	44
Figure 56: Evaluation du modèle KNN avec TF-IDF	45
Figure 57: Matrice de confusion de KNN TF-IDF	45
Figure 58: Evaluation du modèle KNN avec CountVectorizer	46
Figure 59: Matrice de Confusion de KNN avec CountVectorizer	47
Figure 60: Evaluation du modèle decision Tree avec TF-IDF	48
Figure 61: Matrice de confusion du modèle Decision Tree TF-IDF	48
Figure 62: Courbe ROC decision Tree	49
Figure 63: Evaluation du modèle decision Tree avec CountVectorizer	49
Figure 64: Evaluation de Naive Bayes avec TF-IDF	52
Figure 65: Matrice de confusion Naive Bayes TF-IDF	52
Figure 66: Evaluation de Naive Bayes avec CountVectorizer	52
Figure 67: Grille des hyperparamètres du réseau de neurones	54
Figure 68: Recherche des meilleurs paramètres pour NN	54
Figure 69: Meilleures paramètres et accuracy du NN	54
Figure 70: Sommaire du modèle RN	56
Figure 71: Entraînement du modèle NN	56
Figure 72: Résultats sur l'ensemble de validation du modèle NN	57
Figure 73: Entraînement du modèle LSTM	58
Figure 74: Test du modèle LSTM	58
Figure 75: Matrice de confusion de LSTM	59
Figure 76: Courbe ROC LSTM	59

Chapitre 1

1. Introduction :

- Contexte et objectifs

Dans le monde numérique actuel, les réseaux sociaux constituent une source intarissable d'informations et d'opinions. Parmi ces plateformes, Twitter se distingue par sa rapidité et son volume de données, générant des millions de tweets chaque jour. Cette immense quantité de données textuelles offre une opportunité unique d'analyser les sentiments et les opinions du public sur divers sujets.

Ce projet s'inscrit dans ce contexte et vise à explorer l'analyse des sentiments sur Twitter. L'objectif principal est de développer un modèle capable de classifier automatiquement les tweets en fonction de leur sentiment (positif, négatif ou neutre). Pour ce faire, nous implémenterons et comparerons différentes approches d'apprentissage automatique, incluant la régression logistique, le KNN, les SVM, les réseaux de neurones et le deep Learning.

- Qu'est-ce qu'une analyse de sentiments sur twitter?

L'analyse des sentiments sur Twitter, également appelée "sentiment analysis" ou "opinion mining", consiste à extraire et analyser les opinions et les sentiments exprimés dans les tweets. Cette analyse permet de comprendre la perception du public sur une marque, un produit, un événement ou un sujet d'actualité.



Figure 1: Analyse de sentiments sur twitter

-Pourquoi l'analyse des sentiments est-elle importante ?

L'analyse des sentiments présente de nombreux avantages pour divers domaines :

- Marketing et communication : Les entreprises peuvent utiliser l'analyse des sentiments pour mesurer l'impact de leurs campagnes marketing, identifier les points forts et faibles de leurs produits et services, et adapter leur communication en conséquence.

- Etudes de marché : L'analyse des sentiments permet de sonder l'opinion publique sur un produit ou un service avant son lancement, d'identifier les besoins et les attentes des clients, et de suivre l'évolution de la perception du marché.
- Veille informationnelle : L'analyse des sentiments peut être utilisée pour surveiller l'opinion publique sur un sujet d'actualité, identifier les risques et les opportunités, et réagir rapidement aux événements.

-Pourquoi utiliser Twitter pour l'analyse des sentiments ?

Twitter est une plateforme idéale pour l'analyse des sentiments pour plusieurs raisons :

- Volume de données : Twitter génère un volume important de données, ce qui permet d'obtenir des échantillons représentatifs de l'opinion publique.
- Rapidité : Les tweets sont publiés en temps réel, ce qui permet d'analyser les sentiments en temps réel et de suivre l'évolution des opinions.
- Diversité des sujets : Les tweets couvrent une grande variété de sujets, ce qui permet d'analyser les sentiments sur un large éventail de thématiques.



Figure 2:Utilisation de twitter

- Introduction au NLP

L'analyse des sentiments sur Twitter s'appuie sur le traitement du langage naturel (NLP), un domaine de l'intelligence artificielle qui vise à comprendre et à traiter le langage humain. Le NLP permet de transformer les tweets en données structurées que les modèles d'apprentissage automatique peuvent ensuite analyser. Cette introduction au NLP servira de fondement pour les étapes suivantes de préparation et d'analyse des données dans ce projet.

2- L'analyse de données

-Pourquoi l'analyse des données est-elle importante ?

L'analyse des données est une étape cruciale dans tout projet d'apprentissage automatique. Elle permet de comprendre les caractéristiques des données, d'identifier les patterns et les anomalies, et de sélectionner les variables les plus pertinentes pour le modèle. Dans le cadre de l'analyse des sentiments sur Twitter, l'analyse des données permet de :

- Comprendre la répartition des sentiments : Déterminer la proportion de tweets positifs, négatifs et neutres dans le dataset.
- Identifier les caractéristiques des tweets : Analyser la longueur des tweets, la fréquence des mots, la présence de hashtags et d'emojis, etc.
- Evaluer la qualité des données : Identifier les tweets aberrants ou incomplets.
- Sélectionner les features : Choisir les variables les plus discriminantes pour la classification des tweets.

-Affichage de quelques exemples de tweets

Afin d'illustrer la diversité des tweets et des sentiments exprimés, il est pertinent de présenter quelques exemples concrets.

	label	id	date	query	username	text
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

Figure 3:Exemples de tweets

-Analyse de la distribution des tweets

L'analyse de la distribution des tweets permet de visualiser la répartition des sentiments dans le dataset.

Diagramme circulaire des différentes classes de tweets

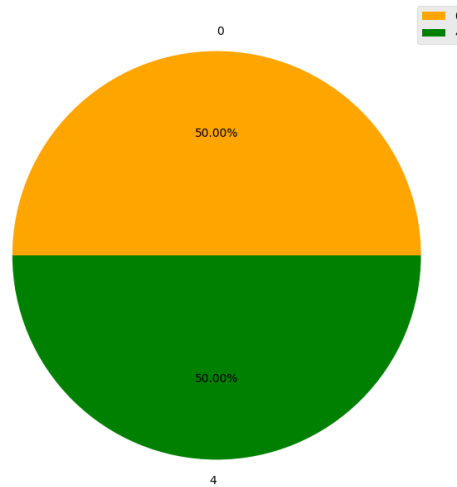


Figure 4:Diagramme circulaire des classes de tweets

Ce diagramme circulaire montre la proportion de tweets positifs et négatifs.

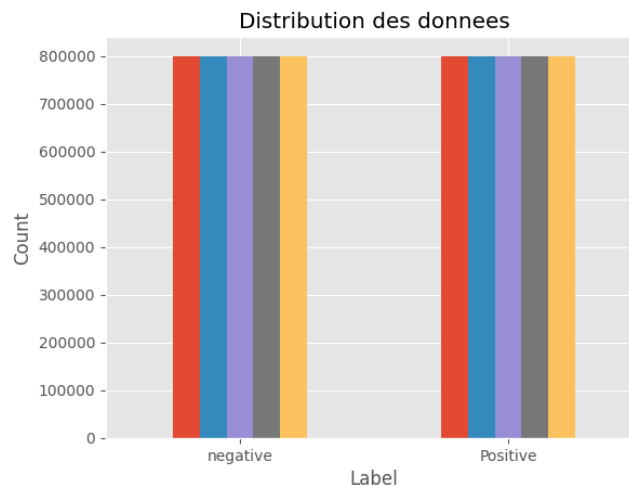


Figure 5:La distribution des données

Ce graphique montre la distribution des tweets pour chaque classe, on a donc 8000000 tweets négatifs et 800000 autres positifs.

-Analyse exploratoire des données (moyennes, quartiles, etc.)

L'analyse exploratoire des données (EDA) vise à découvrir des patterns et des insights cachés dans les données.

Sur la base de cette analyse, nous pouvons déterminer la longueur des tweets pour deux classes particulières de tweets :

Distribution de la longueur de texte pour positive sentiment tweets.

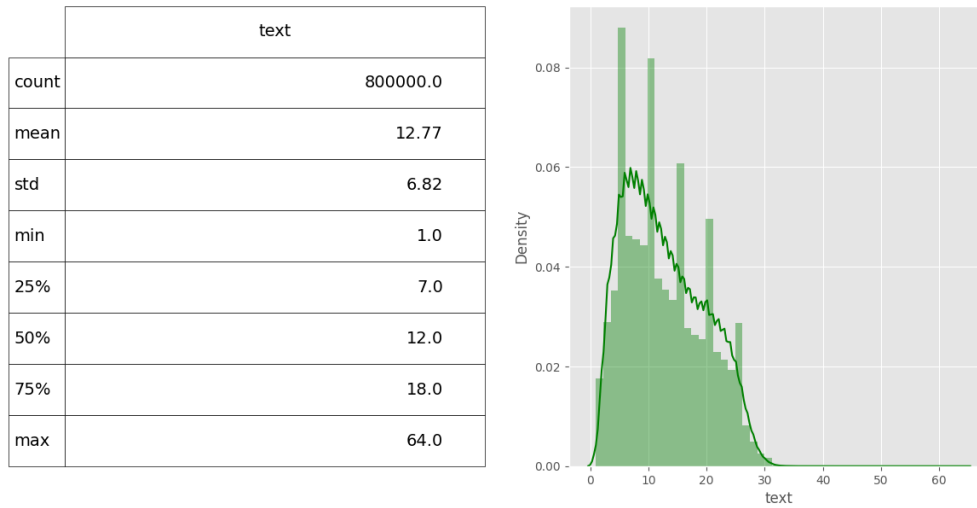


Figure 6: La distribution de la longueur de texte pour les tweets positifs

Distribution de la longueur de texte pour negative sentiment tweets.

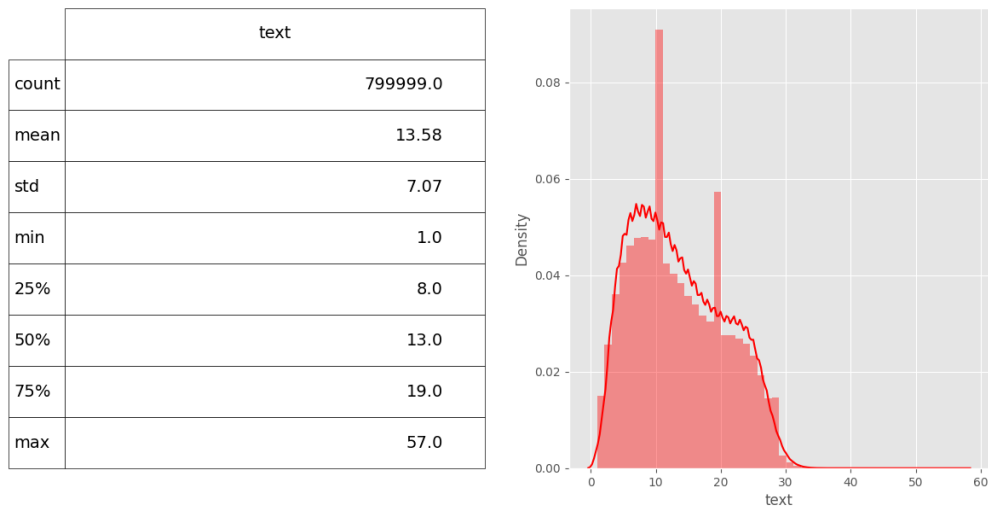


Figure 7: La distribution de la longueur de texte pour les tweets négatifs

Dans un ensemble de données aussi vaste, les tweets appartenant à deux classes ont des longueurs presque identiques. Cependant, la longueur moyenne des tweets pour la classe négative est d'environ 0,8 mots de plus.

Chapitre 2

1- La préparation des données

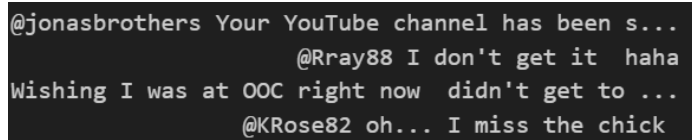
-Nettoyage des tweets

Le nettoyage des tweets est une étape essentielle pour améliorer la qualité des données et la performance du modèle d'apprentissage automatique. Cette étape vise à supprimer le bruit et les informations non pertinentes, et à normaliser le format du texte.

Etapes du nettoyage

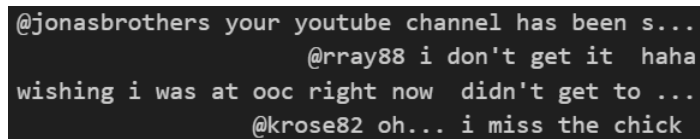
1. Conversion en Minuscule

Convertir tous les caractères en minuscules pour simplifier la comparaison des mots.



```
@jonasbrothers Your YouTube channel has been s...  
@Rray88 I don't get it haha  
Wishing I was at OOC right now didn't get to ...  
@KRose82 oh... I miss the chick
```

Figure 8:Exemple avant Conversion en Minuscule



```
@jonasbrothers your youtube channel has been s...  
@rray88 i don't get it haha  
wishing i was at ooc right now didn't get to ...  
@krose82 oh... i miss the chick
```

Figure 9::Exemple après Conversion en Minuscule

2. Remplacement des emojis par leurs significations textuelles

Les emojis sont omniprésents dans les communications numériques, et les tweets ne font pas exception. Cependant, leur nature symbolique peut créer des obstacles à l'analyse des sentiments. Pour lever ces obstacles, il est crucial de remplacer les emojis par leurs significations textuelles explicites.

Méthode

- Création d'un dictionnaire d'emojis :

Compiler un dictionnaire exhaustif d'emojis, en associant à chacun sa signification précise et ses variantes possibles.

Exploiter des sources fiables, comme des dictionnaires en ligne, des bases de données spécialisées ou des études sur l'utilisation des emojis.

- Remplacement des emojis :

Définir un ensemble d'emojis fréquents et leurs descriptions textuelles correspondantes.

```
@jonasbrothers ;) your youtube channel has bee...  
    @rray88 :) i don't get it haha  
wishing http://twitpic.com/6rmby i was at ooc...  
    @krose82 oh... i miss the chick  
so mutha effin bored
```

Figure 10: Exemple de tweets avec les emojis symbolisés

```
@jonasbrothers wink your youtube channel has b...  
    @rray88 smile i don't get it haha  
wishing http://twitpic.com/6rmby i was at ooc ...  
    @krose82 oh... i miss the chick  
so mutha effin bored
```

Figure 11: Exemple de tweets avec les emojis écrits en texte

Dans les étapes qui suivent on utilisera la bibliothèque **re** pour les expressions régulières :

Les expressions régulières (regex) sont un outil puissant pour rechercher et extraire des motifs de texte. Elles permettent de définir des règles complexes pour identifier et manipuler des parties spécifiques d'une chaîne de caractères.

Dans le contexte de la **suppression des emails/mentions, balises HTML, non mots, répétition des caractères et nombres**, les expressions régulières peuvent être utilisées pour les capturer et les supprimer afin d'extraire le texte brut.

3. Suppression des emails/mentions (avant la ponctuation)

Supprimer les adresses emails et les mentions d'utilisateurs (@username) pour simplifier le texte puisque ces derniers nous apportent rien à notre classification.

4. Suppression des URLs

Supprimer les liens hypertextes pour se concentrer sur le contenu textuel du tweet (on prend la figure 9 comme exemple de tweets avec des URLs et mentions)

```
wink your youtube channel has been suspended...  
    smile i don't get it haha  
wishing i was at ooc right now didn't get to...  
    oh... i miss the chick  
so mutha effin bored
```

Figure 12: Exemple de tweets après la suppression des mentions et des URLs

5. Suppression des balises HTML

Supprimer les balises HTML pour simplifier le format du texte

Impact de la ponctuation sur les balises HTML

Doit se faire avant la ponctuation puisque la ponctuation peut être incluse dans les balises HTML, ce qui peut perturber la suppression des balises si elle est effectuée en premier. Par exemple, la balise
 (saut de ligne), si la ponctuation est supprimée en premier, le signe < sera supprimé et donc on reconnaît plus les balises HTML)

6. Suppression de la ponctuation et des non-mots

Cette partie est cruciale dans la suppression du bruit du texte

Méthode **punctuation** de la bibliothèque string

La bibliothèque string en Python propose une méthode punctuation qui retourne une chaîne de caractères contenant tous les caractères de ponctuation communément utilisés en anglais. Cette méthode peut être utilisée pour supprimer la ponctuation d'une chaîne de caractères.

L'affichage de la liste de ponctuations nous donne ce qui suit :

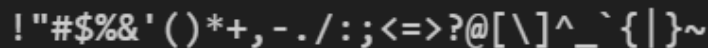


Figure 13: Liste de ponctuations de String

On prend la figure 10 comme exemple de tweets avant la suppression de la ponctuation et des non mots.

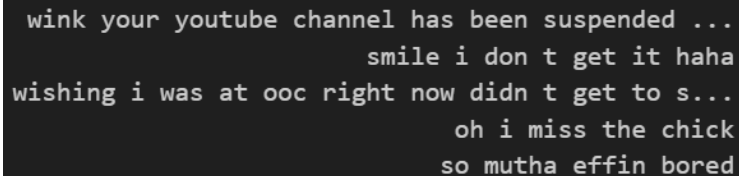


Figure 14: Exemple des tweets après la suppression de la ponctuation et des non mots

7. Remplacement des abréviations pour une meilleure compréhension

Le langage des réseaux sociaux regorge d'abréviations et d'emojis, qui peuvent obscurcir le sens des messages pour l'analyse des sentiments. Pour pallier ce problème, il est crucial de remplacer ces raccourcis par leurs formes complètes et explicites.

Méthode

- Création d'un dictionnaire d'abréviations :

Compiler un dictionnaire exhaustif d'emojis, en associant à chacun sa signification précise.

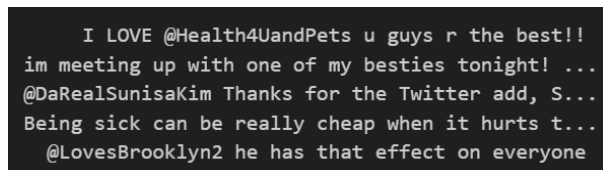
S'appuyer sur des sources fiables, comme des dictionnaires en ligne ou des bases de données spécialisées.

- Remplacement des abréviations :

Définir un ensemble d'abréviations courantes et leurs équivalents textuels complets.

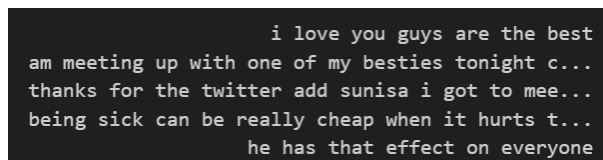
Utiliser des techniques de recherche et de remplacement de chaînes de caractères pour opérer le changement.

Veiller à traiter les abréviations avant la suppression des mots vides, car certaines en font partie.



```
I LOVE @Health4UandPets u guys r the best!!
im meeting up with one of my besties tonight! ...
@DaRealSunisaKim Thanks for the Twitter add, S...
Being sick can be really cheap when it hurts t...
@LovesBrooklyn2 he has that effect on everyone
```

Figure 15:L'exemple avec les abréviations



```
i love you guys are the best
am meeting up with one of my besties tonight c...
thanks for the twitter add sunisa i got to mee...
being sick can be really cheap when it hurts t...
he has that effect on everyone
```

Figure 16:L'exemple après le remplacement des abréviations

On remarque que 'u' et 'r' a été remplacé par 'You' et 'are' respectivement.

Avantages de cette approche

- Amélioration significative de la précision de l'analyse des sentiments.
- Meilleure compréhension du contexte et du ton des messages.
- Facilitation du traitement du texte et de l'extraction des sentiments.

8. Elimination des mots vides

Remarque

La ponctuation peut être adjacente aux stopwords, ce qui peut perturber la suppression des stopwords si elle est effectuée en premier. Par exemple, le stopword "my" peut être suivi d'une virgule (","). Si les stopwords sont supprimés en premier, ce dernier ne va pas être inclus en raison de la virgule, ce qui fait que l'étape de suppression de ponctuation doit se faire avant.

Méthode

- Utiliser la méthode stopwords de la bibliothèque NLTK pour supprimer les mots vides courants de la langue anglaise.

Objectif

- La suppression des mots vides permet de réduire la taille du texte et de se concentrer sur les mots significatifs pour l'analyse des sentiments.

```
> stopwords("english")
[1] "i" "me" "my" "myself" "we"
[6] "our" "ours" "ourselves" "you" "your"
[11] "yours" "yourself" "yourselves" "he" "him"
[16] "his" "himself" "she" "her" "hers"
[21] "herself" "it" "its" "itself" "they"
[26] "them" "their" "theirs" "themselves" "what"
[31] "which" "who" "whom" "this" "that"
[36] "these" "those" "am" "is" "are"
[41] "was" "were" "be" "been" "being"
[46] "have" "has" "had" "having" "do"
[51] "does" "did" "doing" "would" "should"
[56] "could" "ought" "i'm" "you're" "he's"
[61] "she's" "it's" "we're" "they're" "i've"
[66] "you've" "we've" "they've" "i'd" "you'd"
[71] "he'd" "she'd" "we'd" "they'd" "i'll"
[76] "you'll" "he'll" "she'll" "we'll" "they'll"
[81] "isn't" "aren't" "wasn't" "weren't" "hasn't"
[86] "haven't" "hadn't" "doesn't" "don't" "didn't"
[91] "won't" "wouldn't" "shan't" "shouldn't" "can't"
[96] "cannot" "couldn't" "mustn't" "let's" "that's"
[101] "who's" "what's" "here's" "there's" "when's"
[106] "where's" "why's" "how's" "a" "an"
```

Figure 17: Liste des stopwords dans la langue anglaise

La figure 16 est notre exemple avant l'élimination des stopwords :

```
love guys best
meeting one besties tonight cant wait girl talk
thanks twitter add sunisa got meet hin show dc...
sick really cheap hurts much eat real food plu...
effect everyone
```

Figure 18: Les exemples de tweets après la suppression des mots vides

Dans cet exemple la liste des mots vides supprimés est: 'i, you, are, im, up, with, off, my, for, the, it, he, has, that, on '.

9. Suppression de lettres consécutives

Méthode

- Détecter les séquences de trois lettres ou plus consécutives.
- Remplacer chaque séquence par deux lettres seulement.

Objectif

- Améliorer la précision de l'analyse des sentiments en réduisant l'impact des répétitions de lettres.

Exemple

- Avant: "loooooove", "goooooood", "geeezy"
- Après: "love", "good", "geezy"

Justification

- La suppression des répétitions permet de simplifier le texte et de se concentrer sur les mots significatifs.

10. Suppression des chiffres

Rôle des chiffres dans le langage

- Les chiffres sont utilisés pour quantifier des informations, exprimer des dates, des heures, des montants, etc.
- Ils peuvent également être utilisés à des fins stylistiques, comme dans les slogans ou les titres.

Impact des chiffres sur l'analyse du texte

- La présence de chiffres peut perturber l'analyse du sentiment, car ils ne sont pas porteurs d'émotions.
- Les chiffres peuvent également masquer des informations importantes, comme des expressions de sentiments.

Justification de la suppression des chiffres

- La suppression des chiffres permet de se concentrer sur les mots significatifs pour l'analyse du sentiment.
- Cela permet également d'améliorer la précision des tâches de NLP, comme dans notre cas la classification de texte.

Méthodes de suppression des chiffres

- Il existe plusieurs méthodes pour supprimer les chiffres du texte, comme l'utilisation d'expressions régulières (c'est ce qu'on utilise dans notre projet) ou de dictionnaires spécifiques.

11. Tokenisation du texte des tweets et Radicalisation des mots

11.1. Tokenisation

Principe

- La tokenisation consiste à découper le texte en unités plus petites appelées "tokens". Ces tokens peuvent être des mots, des ponctuations, des chiffres, etc.

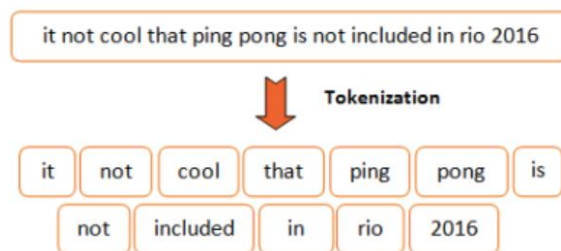


Figure 19: Principe de la tokenisation

Méthode

- Utiliser la méthode **RegexpTokenizer** de la bibliothèque NLTK pour appliquer la tokenisation.

Importance

- La tokenisation est une étape essentielle du traitement du langage naturel (NLP) car elle permet de transformer le texte brut en une séquence de tokens plus facile à traiter par les algorithmes.

Intérêt

- Simplifier le texte et de le rendre plus compréhensible pour les machines.
- Identifier les différents éléments du texte (mots, ponctuation, etc.).
- Faciliter l'application de techniques de NLP comme l'analyse lexicale, la lemmatisation et l'analyse syntaxique.

```
[wink, youtube, channel, suspended, weird]
                        [smile, get, haha]
[wishing, ooc, right, get, see]
                        [oh, miss, chick]
                        [mutha, effin, bored]
```

Figure 20:Exemple de tokenisation

11.2. Radicalisation des mots

Principe

- La radicalisation des mots consiste à supprimer les affixes (préfixes et suffixes) d'un mot pour obtenir sa racine, appelée "radical".

Méthode

- Utiliser la méthode **PorterStemmer()** de la bibliothèque NLTK pour appliquer la radicalisation.

Importance

- La radicalisation des mots est utile pour :
- Regrouper les mots de même famille (par exemple, "courir", "coureur", "courant" ont la même racine "cour").
- Améliorer la précision des tâches de NLP, comme la recherche d'information.
- Réduire la taille du vocabulaire utilisé, ce qui peut améliorer l'efficacité des algorithmes.

Intérêt

- Améliorer la précision de l'analyse lexicale.
- Réduire la complexité du traitement du langage naturel.
- Faciliter la comparaison de mots entre eux.

```
looking cat hard work go get sleep noww listen...
let see san diego versus maine yeah wanna go b...
                        guuys youtube account suspended
                        awwe wish
wink youtube channel suspended weird
```

Figure 21:Exemple de tweets avant tokenisation et radicalisation

```
[look, cat, hard, work, go, get, sleep, noww, ...  
[let, see, san, diego, versu, main, yeah, wann...  
      [guuy, youtub, account, suspend]  
                        [aww, wish]  
      [wink, youtub, channel, suspend, weird]
```

Figure 22: Des exemples de tweets après radicalisation

```
Le temps du CPU pour la radicalisation: 326.6173310279846
```

Figure 23: Le temps CPU de la radicalisation

12. Lemmatisation

Principe

- La lemmatisation est une technique de NLP qui consiste à ramener les mots à leur forme canonique, appelée "lemme". Le lemme est la forme non fléchie du mot, c'est-à-dire la forme qu'il prend dans le dictionnaire.

Méthode

- Utiliser la méthode **WordNetLemmatizer()** de la bibliothèque NLTK pour appliquer la radicalisation.

Importance

- Regrouper les mots de même famille (par exemple, « run, runs, ran, running, runner » ont le même lemme "run").
- Améliorer la précision des tâches de NLP, comme la recherche d'information et la traduction automatique.
- Réduire la taille du vocabulaire utilisé, ce qui peut améliorer l'efficacité des algorithmes.

Intérêt

- La lemmatisation permet de :
- Améliorer la précision de l'analyse lexicale.
- Réduire la complexité du traitement du langage naturel.
- Faciliter la comparaison de mots entre eux.

Différences entre la radicalisation et la lemmatisation

La radicalisation est une technique plus simple que la lemmatisation.

La lemmatisation est plus précise que la radicalisation car elle prend en compte les règles grammaticales de la langue.

vocabulaire. Ainsi, seuls les mots qui apparaissent au moins K fois dans le corpus seront conservés.

- Tri des mots par fréquence : Les mots filtrés sont ensuite triés en fonction de leur fréquence d'apparition, en ordre décroissant et sont utilisés pour constituer notre vocabulaire.

2. Justification du choix K

Détermination de la valeur de K pour un dataset volumineux :

Le choix de la valeur de K pour l'extraction de caractéristiques basée sur la fréquence des mots nécessite une approche réfléchie. Voici quelques étapes à suivre pour sélectionner une valeur de K appropriée :

1. Utilisation d'une ligne de fréquence des mots

Dans cette section, nous avons utilisé une ligne de fréquence des mots pour visualiser la distribution des fréquences des mots dans le corpus de tweets. Cette approche nous permet de mieux comprendre comment la fréquence des mots décroît en fonction de leur rang dans le corpus.

Nous avons représenté graphiquement la fréquence des mots en fonction de leur rang dans le corpus en utilisant une échelle logarithmique pour les deux axes. Cette représentation nous permet d'observer les variations de fréquence sur une large plage de valeurs.

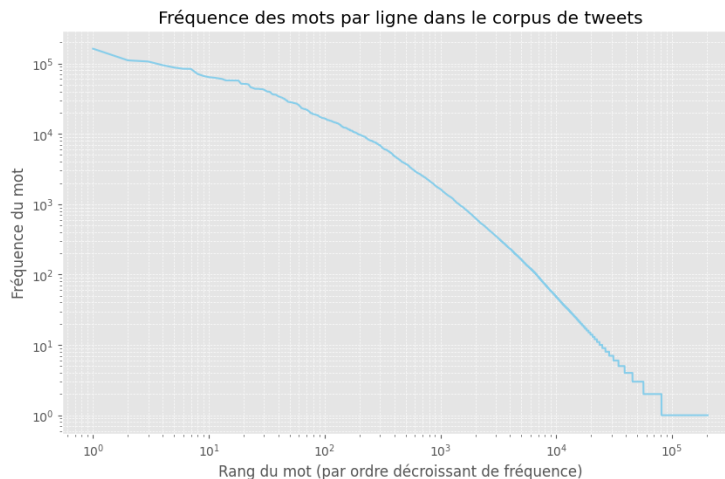


Figure 27:Fréquence des mots par ligne

Interprétation des résultats :

En examinant la ligne de fréquence des mots, nous pouvons faire plusieurs observations importantes :

- La ligne de fréquence des mots décroît rapidement, ce qui indique qu'il existe quelques mots très fréquents dans le corpus.

- La décroissance devient plus lente à mesure que nous progressons le long de l'axe des x, suggérant une longue queue de mots moins fréquents.
- L'utilisation d'une échelle logarithmique sur les axes nous permet de mieux visualiser les variations de fréquence sur un large spectre de valeurs.

En analysant la visualisation de la fréquence des mots par une ligne dans le corpus de tweets, nous avons identifié deux points d'inflexion marqués par des lignes rouges détectés par **l'algorithme de détection de ruptures**. Ces points d'inflexion représentent des changements significatifs dans la distribution des fréquences des mots.

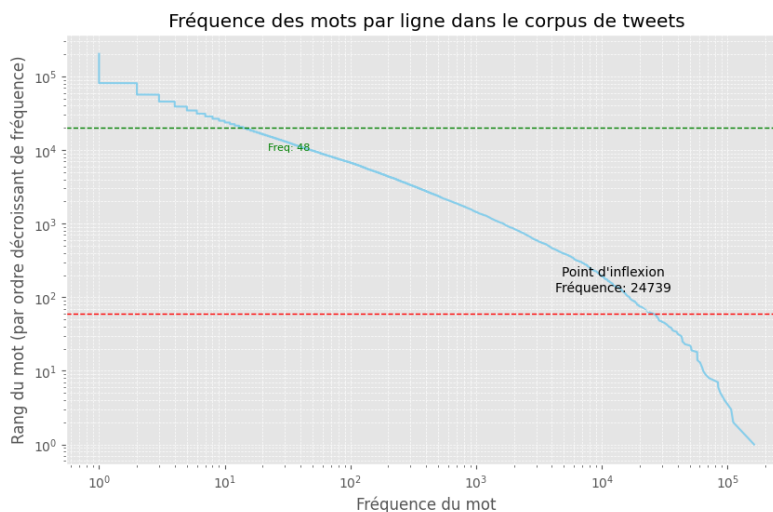


Figure 28: Fréquence des mots par ligne dans le corpus de tweets

Signification des points d'inflexion

Le point d'inflexion indique le rang à partir duquel la décroissance de la fréquence des mots devient plus lente, suggérant un seuil à partir duquel les mots deviennent moins fréquents dans le corpus.

Relation avec la valeur de K

Dans le contexte de la construction du vocabulaire, les points d'inflexion peuvent être utilisés comme guides pour estimer la valeur de K. Les rangs correspondant aux points d'inflexion peuvent être considérés comme des estimations de la fréquence minimale à laquelle les mots commencent à devenir moins fréquents et donc potentiellement moins significatifs pour la représentation du corpus.

Cette première ligne d'inflexion sépare les mots les plus fréquents des autres mots dans le corpus (si on veut faire une analyse approfondie on rajoute alors des mots moyennement fréquents et on exclut seulement les mots bruit (rares)).

Utilisation pour déterminer K

En utilisant les points d'inflexion comme guides, nous pouvons sélectionner une valeur appropriée pour K.

Remarque

On remarque qu'après 10^4 du rang des mots la ligne de fréquence change et devient plus lente, ou on peut déterminer un autre point (la ligne en vert).

Conclusion

On peut initialement, à l'aide de ce graphe conclure un intervalle de k : [50,24739]

2. Analyse de la distribution de fréquence des mots

- Visualisation de la distribution des fréquences :
 - ✓ Objectif : Comprendre la distribution des fréquences des mots dans le corpus de tweets.
 - ✓ Approche : Utiliser un histogramme ou des graphiques en camembert pour représenter visuellement la répartition des fréquences des mots.

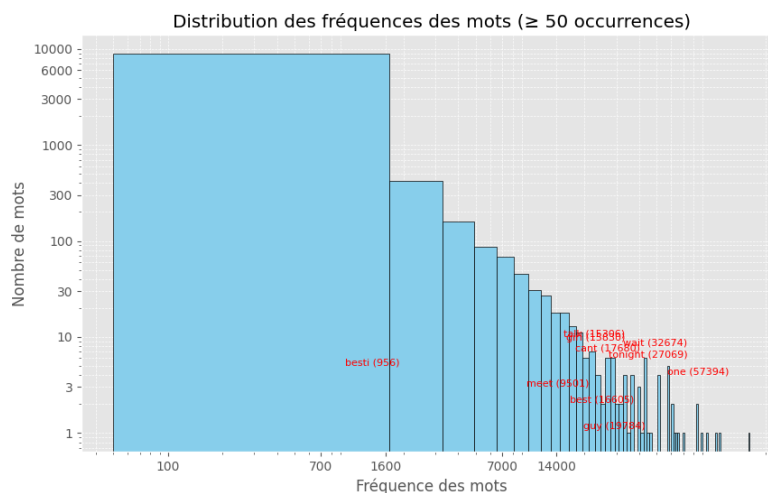


Figure 29: Histogramme de distribution des fréquences des mots

L'histogramme ci-dessus représente la distribution des fréquences des mots dans le corpus de tweets, en filtrant les mots ayant une fréquence d'apparition supérieure ou égale à 50 occurrences.

Interprétation de l'histogramme

- ✚ L'axe des x représente la fréquence des mots (en échelle logarithmique), tandis que l'axe des y représente le nombre de mots ayant cette fréquence.
- ✚ La distribution est fortement asymétrique, avec une forte concentration de mots ayant une fréquence faible et une diminution rapide du nombre de mots à mesure que la fréquence augmente.

- ✚ La majorité des mots ont une fréquence relativement faible, avec un nombre considérable de mots ayant une fréquence inférieure à 100 occurrences.
- ✚ Les quelques mots avec une fréquence plus élevée (probablement des mots très communs tels que "and", etc.) sont clairement visibles à l'extrémité gauche de l'histogramme.

Implications pour l'analyse

- ✚ Cette analyse peut orienter la sélection d'un seuil approprié (K) pour la construction du vocabulaire, en fournissant un aperçu du nombre d'occurrences minimum requis pour qu'un mot soit considéré comme significatif dans le corpus.

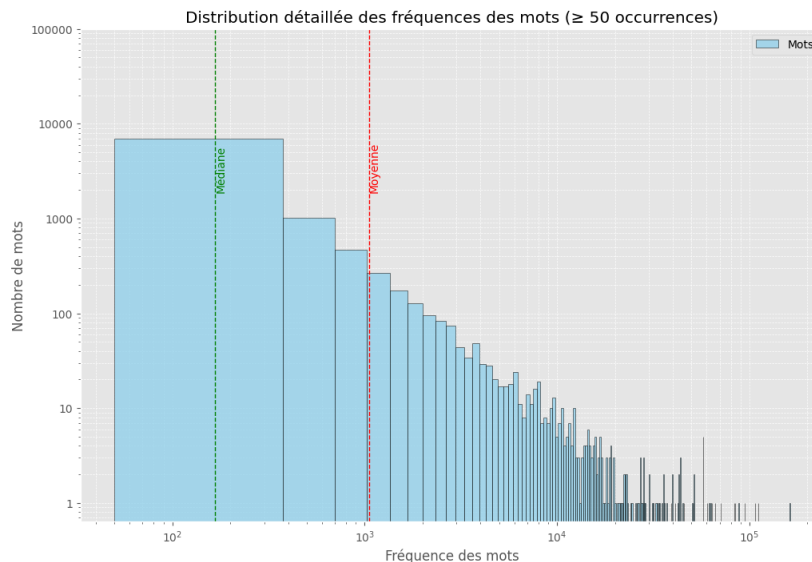


Figure 30: Distribution détaillée des fréquences des mots

Les lignes de moyenne et de médiane dans l'histogramme représentent les statistiques principales de la distribution des fréquences des mots filtrés. Voici comment les interpréter et les relier à la valeur de K :

Moyenne : La ligne rouge représentant la moyenne indique la valeur moyenne des fréquences des mots, calculée en additionnant toutes les fréquences et en les divisant par le nombre total de mots. La moyenne est sensible aux valeurs extrêmes et peut être affectée par les valeurs aberrantes dans la distribution.

Médiane : La ligne verte représentant la médiane indique la valeur médiane des fréquences des mots, c'est-à-dire la valeur au milieu de la distribution. La médiane est une mesure de la centralité d'une distribution qui représente la valeur au centre de l'ensemble de données une fois qu'il est trié par ordre croissant ou décroissant.

Relation avec K : on peut utiliser la moyenne ou la médiane comme point de référence pour sélectionner une valeur appropriée pour K. Par exemple, on peut choisir un seuil de K égal à la moyenne ou à la médiane si on veut inclure une proportion équivalente de mots dans votre vocabulaire.

- ✚ Si vous choisissez K pour être égal ou proche de la moyenne, cela signifie que vous sélectionnez un seuil où la fréquence moyenne des mots est utilisée comme critère de sélection. (Moyenne = approximativement 1000)
- ✚ Choisir K pour être égal ou proche de la médiane signifie que vous sélectionnez un seuil où environ la moitié des mots auront une fréquence égale ou supérieure à cette médiane. (Médiane = approximativement 200)

Conclusion

On peut donc réduire l'intervalle de K à [200, 1000].

3. Détermination de l'impact de K sur la taille du vocabulaire

- Calcul de la taille du vocabulaire pour différentes valeurs de K :
 - ✓ Sélectionner un intervalle de valeurs de K : [200, 1000]
 - ✓ Calculer la taille du vocabulaire correspondant pour chaque valeur de K.

K=200

Nombre de mots dans le vocabulaire : 4378

Figure 31: Nombre de mots dans le vocabulaire K=200

K=500

Nombre de mots dans le vocabulaire : 2361

Figure 32: Nombre de mots dans le vocabulaire K=500

K=600

Nombre de mots dans le vocabulaire : 2079

Figure 33: Nombre de mots dans le vocabulaire K=600

K=700

Nombre de mots dans le vocabulaire : 1880

Figure 34: Nombre de mots dans le vocabulaire K=700

K=1000

Nombre de mots dans le vocabulaire : 1440

Figure 35: Nombre de mots dans le vocabulaire K=1000

4. Évaluation de l'impact de K sur les performances du modèle

- **Entraînement et évaluation de modèles avec différentes valeurs de K :** Après l'implémentation de nos modèles d'analyse de sentiments des changements peuvent être effectués, un entraînement des modèles avec des vocabulaires générés à partir de différentes valeurs de K sera nécessaire.

5. Considérations supplémentaires

- **Taille du dataset** : Un dataset volumineux permet de choisir **une valeur de K plus élevée** sans sacrifier une couverture suffisante des mots importants.
- **Diversité des sujets** : Un dataset avec une grande diversité de sujets peut nécessiter **une valeur de K pas trop élevée** pour capturer la variété des termes pertinents.
- **Objectifs du modèle** : pour une analyse fine des sentiments, une valeur de K qui n'est pas **élevée** peut-être préférable pour conserver des mots plus nuancés.

Conclusion

Pour faire un compromis, on prend **K = 600** (comme compromis dans notre intervalle, la moyenne entre 200 et 1000).

On obtient donc un vocabulaire avec 2079 mots.

En résumé, le choix de la valeur de K est un processus itératif qui implique l'analyse de la distribution des fréquences des mots, l'évaluation de l'impact de K sur la taille du vocabulaire et les performances du modèle, et la prise en compte de la taille du dataset, de la diversité des sujets et des objectifs du modèle.

3. Mapping des mots vers les index

L'étape de mapping d'index est une étape importante dans le traitement du langage naturel, en particulier lors de l'utilisation de modèles de classification basés sur des vecteurs de mots tels que TF-IDF. Cette étape consiste à associer chaque mot d'un document à son index correspondant dans un dictionnaire de vocabulaire. Le mapping permet de :

- **Traduire les mots en nombres**: Chaque mot du vocabulaire est associé à un index unique, permettant une représentation numérique compacte des données textuelles.
- **Faciliter le traitement par les modèles**: Les modèles de classification et d'apprentissage automatique peuvent aisément manipuler des vecteurs numériques plutôt que des séquences de mots brutes.
- **Permettre l'apprentissage de relations entre les mots**: En analysant les index des mots dans les documents, les modèles peuvent apprendre des co-occurrences et des relations sémantiques entre les mots.

Deux approches principales de mapping sont couramment utilisées :

1. **Mapping d'index**: Chaque mot est associé à son index dans un dictionnaire de vocabulaire.
2. **Mapping vectoriel**: Chaque mot est converti en un vecteur numérique dense, représentant ses caractéristiques sémantiques.

Le choix de l'approche dépend de la tâche et du modèle utilisés. Le mapping d'index est simple et efficace, tandis que le mapping vectoriel peut capturer des informations sémantiques plus riches.

Dans le cadre de ce travail, nous avons utilisé **le mapping d'index** pour convertir les mots des tweets en représentations numériques exploitables par les modèles de classification.

Le processus de mapping d'index comprend les étapes suivantes :

1. Lecture du fichier vocabulaire.
2. Création du dictionnaire de vocabulaire :
 - Créer un dictionnaire de vocabulaire en associant chaque mot unique à un index numérique unique.
 - Gérer les mots rares ou inconnus (par exemple, en les remplaçant par un jeton spécial ou en les ignorant).
3. Mapping des mots des tweets : Pour chaque tweet prétraité :
 - Itérer sur chaque mot du tweet.
 - Rechercher le mot dans le dictionnaire de vocabulaire.
 - Si le mot est trouvé, récupérer son index correspondant.
 - Si le mot n'est pas trouvé, ignorer le mot (il ne sera pas inclus dans la représentation du tweet).
 - Stocker les index des mots trouvés dans une liste.
4. Affichage :

```
Lecture du fichier de vocabulaire...
Creation du dictionnaire de vocabulaire...
Mapping des mots des tweets vers les index du vocabulaire...
Tweets traités : 100000/1599999
Tweets traités : 200000/1599999
Tweets traités : 300000/1599999
Tweets traités : 400000/1599999
Tweets traités : 500000/1599999
Tweets traités : 600000/1599999
Tweets traités : 700000/1599999
Tweets traités : 800000/1599999
Tweets traités : 900000/1599999
Tweets traités : 1000000/1599999
Tweets traités : 1100000/1599999
Tweets traités : 1200000/1599999
Tweets traités : 1300000/1599999
Tweets traités : 1400000/1599999
Tweets traités : 1500000/1599999
Terminé.
```

Figure 36: Le mapping d'index

4-Extraction des caractéristiques

On fait d'abord la séparation des caractéristiques et labels.

```
#separation des caracteristiques et labels
X = data.text
y = data.label
```

Figure 37: Separation des caractéristiques et labels

Extraction de Caractéristiques pour la Classification de Tweets

1. Division des Données

Pour entraîner et évaluer efficacement les modèles de classification de tweets, il est crucial de diviser les données en deux ensembles distincts : les données d'entraînement et les données de test.

Données d'entraînement: Cet ensemble représente la majorité des données et est utilisé pour entraîner les modèles. Les modèles apprennent à identifier les modèles et les relations au sein de ces données, leur permettant de faire des prédictions précises sur de nouvelles données.

Données de test: Cet ensemble représente une plus petite partie des données et n'est jamais utilisé pendant l'entraînement. Il est uniquement utilisé pour évaluer les performances des modèles entraînés. En évaluant les modèles sur des données qu'ils n'ont jamais vues auparavant, on obtient une mesure impartiale de leur capacité à généraliser et à effectuer des prédictions précises sur de nouveaux tweets.

Dans notre projet on a pris 80% des données pour l'entraînement le reste pour le test.

```
print(X_train.shape)
print(y_train.shape)
print(X_val.shape)
print(y_val.shape)

(1279999,)
(1279999,)
(320000,)
(320000,)
```

Figure 38: division des données pour l'entraînement et le test

2. Méthodes d'Extraction de Caractéristiques

L'extraction de caractéristiques est un processus crucial dans le traitement du langage naturel (NLP), car elle permet de convertir des données textuelles brutes en représentations numériques que les modèles d'apprentissage automatique peuvent comprendre et traiter. Dans le contexte de la classification de tweets, l'objectif est de représenter chaque tweet sous la forme d'un vecteur de caractéristiques qui capture les informations essentielles sur son contenu.

Dans ce projet, deux méthodes d'extraction de caractéristiques ont été explorées :

a) Comptage (CountVectorizer) :

- Cette méthode crée un vecteur de caractéristiques pour chaque tweet, où chaque élément du vecteur correspond à la fréquence d'un mot du dictionnaire dans le tweet.
- Le dictionnaire est constitué de tous les mots uniques présents dans l'ensemble des données.
- La méthode CountVectorizer ne prend pas en compte l'importance relative des mots dans le tweet, mais uniquement leur présence ou leur absence.

```
Vectoriser fitted.  
Nombre des mots caracteristiques: 2079  
Data Transformed.
```

Figure 39:extraction de caractéristiques par comptage

b) TF-IDF (Term Frequency-Inverse Document Frequency):

- Cette méthode va plus loin que le comptage en tenant compte à la fois de la fréquence d'un mot dans un tweet (TF) et de son importance relative dans l'ensemble des documents (IDF).
- Le TF est calculé comme la fréquence d'un mot dans le tweet.
- L'IDF est calculé comme le logarithme du nombre total de documents divisé par le nombre de documents contenant le mot.
- La pondération TF-IDF donne plus d'importance aux mots qui sont fréquents dans un tweet mais rares dans l'ensemble des données, car ils sont considérés comme plus discriminants.

```
Vectoriser fitted.  
Nombre des mots caracteristiques: 2079  
Data Transformed.
```

Figure 40:extraction de caractéristiques par TF-IDF

Choix des Méthodes:

L'utilisation des deux méthodes, CountVectorizer et TF-IDF, permet de comparer leurs performances et de choisir la plus adaptée à la tâche de classification de tweets et à chaque modèle.

Justification du Choix:

Le choix de la méthode d'extraction de caractéristiques dépend de plusieurs facteurs, tels que la nature des données, la complexité du modèle et les objectifs de la classification.

- Dans le cas de la classification de tweets, où les tweets sont généralement courts et peuvent contenir des répétitions de mots, la méthode CountVectorizer peut être suffisante pour capturer les informations essentielles.
- Cependant, si l'objectif est de mettre en évidence les mots les plus discriminants et les plus informatifs, la méthode TF-IDF peut être plus efficace.

L'expérimentation avec les deux méthodes permet d'évaluer empiriquement laquelle offre les meilleures performances pour la tâche de classification spécifique.

3. Résultats et Analyse

L'évaluation des performances des modèles entraînés avec les vecteurs de caractéristiques générés par chaque méthode (CountVectorizer et TF-IDF) permettra de déterminer laquelle est la plus adaptée à la classification de tweets dans le contexte de ce projet.

Comparaison des Résultats

- Les métriques d'évaluation telles que la **précision**, le **rappel** et la **F1-score** seront utilisées pour comparer les performances des modèles entraînés avec les deux types de vecteurs de caractéristiques.
- Une analyse approfondie des résultats permettra de déterminer si une méthode se distingue de l'autre en termes de performance de classification.

Facteurs Influent

- La nature des tweets et la distribution des mots dans le corpus peuvent influencer l'efficacité de chaque méthode.
- La complexité du modèle et les hyperparamètres utilisés peuvent également jouer un rôle dans la performance globale.

Conclusion

En explorant deux méthodes d'extraction de caractéristiques, CountVectorizer et TF-IDF, ce projet vise à identifier la méthode la plus adaptée aux modèles spécifiques dans la classification de tweets dans le contexte spécifique des données utilisées. L'analyse des résultats permettra de tirer des conclusions sur l'efficacité relative de chaque méthode et de fournir des recommandations pour des projets ultérieurs de classification de tweets.

Chapitre 3

1-Classification

- Utilisation des Classifieurs Machine Learning
 - 1. L'évaluation des modèles de classification

Pour ce qui suit, Les métriques d'évaluation telles que la **précision**, le **rappel**, la **F1-score**, **accuracy**, **Macro Avg**, **Weighted Avg** et la **matrice de confusion** seront utilisées pour comparer les performances des modèles entraînés.

Précision, Rappel et F1-score

a) Précision: La précision représente la proportion des tweets prédits comme positifs qui sont réellement positifs. En d'autres termes, elle mesure la capacité du modèle à éviter les faux positifs (tweets négatifs classés à tort comme positifs).

b) Rappel: Le rappel représente la proportion des tweets réellement positifs qui sont correctement identifiés comme tels par le modèle. En d'autres termes, il mesure la capacité du modèle à capturer tous les vrais positifs.

c) F1-score: Le F1-score est une mesure composite qui combine la précision et le rappel en une seule valeur. Il est calculé comme la moyenne harmonique de la précision et du rappel. Le F1-score est souvent utilisé comme mesure globale de la performance d'un modèle de classification, car il prend en compte à la fois la capacité du modèle à éviter les erreurs et sa capacité à identifier correctement tous les cas positifs.

Accuracy, Macro Avg et Weighted Avg

- a) **Accuracy:** L'accuracy représente la proportion totale de tweets correctement classés par le modèle (0.75 dans ce cas).
- b) **Macro avg:** Le macro average calcule la moyenne des métriques (précision, rappel, F1-score) en prenant en compte le nombre de tweets dans chaque classe.
- c) **Weighted avg:** Le Weighted average calcule la moyenne des métriques en pondérant les valeurs par le nombre de tweets dans chaque classe.

Matrice de confusion

La matrice de confusion présente des informations détaillées sur les performances du modèle pour chaque classe (positif et négatif). Dans la matrice fournie :

- La ligne 0 correspond à la classe "Négatif" (positifs prédits = 0).
- La ligne 1 correspond à la classe "Positif" (positifs prédits = 1).
- La colonne "Support" indique le nombre total de tweets dans chaque classe réelle.

- Les valeurs dans les cellules de la matrice représentent le nombre de tweets classés par le modèle dans chaque catégorie.

Maximiser les métriques d'évaluation pour la classification des sentiments

L'objectif de la classification des sentiments est d'obtenir un modèle qui maximise la précision, le rappel et le F1-score.

1. Comprendre les compromis entre les métriques:

- **Précision** : Minimiser les faux positifs (tweets négatifs classés positifs). Crucial lorsque les erreurs de classification positive ont des conséquences élevées (domaine médical ou financier).
- **Rappel** : Minimiser les faux négatifs (tweets positifs classés négatifs). Important lorsque la capture de tous les vrais positifs est essentielle (détection des spams, surveillance des réseaux sociaux).
- **F1-score** : Compromis entre précision et rappel, les pondérant également. Souvent utilisé comme mesure globale de la performance du modèle, mais ne tient pas compte des coûts des faux positifs et faux négatifs.

2. Définir les priorités du projet:

- **Faux positifs coûteux** : Maximiser la précision.
- **Identification de tous les vrais positifs** : Maximiser le rappel.
- **Compromis équilibré entre précision et rappel** : Maximiser le F1-score.

2. Justification du choix de la librairie sklearn :

La bibliothèque scikit-learn (sklearn) est un choix populaire pour l'implémentation d'algorithmes d'apprentissage automatique en Python. Elle offre une implémentation efficace et bien documentée des modèles machine Learning que'on va utiliser (BernoulliNB, Régression logistique, MVS (Linear Support Vector Classification (LinearSVC), KNN, Decision Tree, régression linéaire, naïve bayes et Random Forest) ainsi que des fonctionnalités utiles pour le prétraitement des données, l'optimisation des hyperparamètres et l'évaluation des performances.

3. Bernoulli NB

3.1. *Principe de l'algorithme de BernoulliNB*

Le modèle Naïve Bayes de Bernoulli (BernoulliNB) est un algorithme de classification probabiliste basé sur le théorème de Bayes et l'hypothèse d'indépendance des caractéristiques. Il suppose que la probabilité d'une classe donnée un ensemble de caractéristiques est le produit des probabilités individuelles de chaque caractéristique donnée cette classe.

Formule de classification :

$$P(C | x) = \prod_i P(x_i | C) * P(C)$$

Où:

- $P(C | x)$ est la probabilité de la classe C étant donné les caractéristiques x .
- $P(x_i | C)$ est la probabilité de la caractéristique x_i étant donné la classe C .
- $P(C)$ est la probabilité a priori de la classe C .

Hypothèse d'indépendance des caractéristiques:

L'hypothèse d'indépendance des caractéristiques suppose que la présence ou l'absence d'une caractéristique n'a aucun effet sur la présence ou l'absence d'une autre caractéristique, étant donné la classe. Cette hypothèse simplifie considérablement les calculs et permet une implémentation efficace du modèle.

3.2. *Choix du modèle BernoulliNB pour l'analyse des sentiments*

BernoulliNB est bien adapté aux données binaires telles que les représentations en bag-of-words.

Le modèle BernoulliNB présente plusieurs avantages qui le rendent pertinent pour l'analyse des sentiments des tweets :

- **Simplicité et efficacité:** Le modèle BernoulliNB est simple à implémenter et à interpréter, ce qui le rend adapté aux tâches de classification de grande envergure comme l'analyse des sentiments.
- **Gestion des données textuelles:** Le modèle BernoulliNB est conçu pour traiter des données textuelles représentées par des vecteurs de mots binaires, ce qui correspond parfaitement à la représentation des tweets après l'extraction de caractéristiques par comptage ou TF-IDF.
- **Robustesse au bruit:** Le modèle BernoulliNB est relativement robuste au bruit et aux valeurs aberrantes dans les données, ce qui peut être important pour l'analyse des sentiments, car les tweets peuvent contenir des informations non pertinentes ou des erreurs de frappe.

3.3. *Recherche sur grille avec validation croisée pour l'optimisation de l'hyperparamètre alpha*

La recherche sur grille est une méthode d'optimisation d'hyperparamètres qui consiste à évaluer les performances d'un modèle pour différentes combinaisons de valeurs d'hyperparamètres. La validation croisée est une technique utilisée pour estimer les performances d'un modèle de manière fiable.

On va utiliser pour cela la fonction **GridSearchCV** de la bibliothèque scikit-learn (sklearn).

- Principe de GridSearchCV

GridSearchCV est une méthode d'optimisation d'hyperparamètres couramment utilisée dans l'apprentissage automatique. Son objectif est d'identifier les valeurs d'hyperparamètres qui maximisent les performances d'un modèle sur les données d'entraînement.

- Fonctionnement de GridSearchCV

Le fonctionnement de GridSearchCV se décompose en plusieurs étapes :

a) Définition du modèle et de la grille de recherche :

- On commence par définir un modèle d'apprentissage automatique (dans ce cas, BernoulliNB).
- On définit ensuite une grille de recherche qui spécifie les différentes combinaisons d'hyperparamètres à évaluer. La grille de recherche est un dictionnaire où les clés correspondent aux noms des hyperparamètres et les valeurs correspondent aux listes de valeurs à tester pour chaque hyperparamètre.

b) Validation croisée :

- **GridSearchCV** utilise la validation croisée pour estimer les performances du modèle sur différentes partitions des données d'entraînement. Cela permet d'éviter le surajustement, qui survient lorsque le modèle s'adapte trop aux données d'entraînement et perd sa capacité à généraliser aux nouvelles données.
- La validation croisée fonctionne en divisant les données d'entraînement en plusieurs sous-ensembles. Le modèle est ensuite entraîné sur un sous-ensemble (ensemble d'entraînement) et évalué sur un autre sous-ensemble (ensemble de validation). Ce processus est répété pour chaque sous-ensemble, ce qui permet d'obtenir une estimation plus robuste des performances du modèle.

c) Sélection des meilleurs hyperparamètres :

GridSearchCV évalue les performances du modèle pour chaque combinaison d'hyperparamètres dans la grille de recherche et sélectionne finalement la combinaison qui maximise une métrique d'évaluation spécifique (par exemple, la précision, le rappel ou le score F1).

- Avantages

L'utilisation de GridSearchCV présente plusieurs avantages dans le contexte de ce projet :

- **Identification des hyperparamètres optimaux:** GridSearchCV permet d'identifier les valeurs d'hyperparamètres qui optimisent les performances du modèle, ce qui peut améliorer considérablement la précision et la généralisation du modèle.
- **Éviter le surajustement:** La validation croisée intégrée à GridSearchCV aide à éviter le surajustement en évaluant le modèle sur différentes partitions des données d'entraînement.
- **Simplicité d'utilisation:** GridSearchCV est une fonction facile à utiliser et à comprendre, ce qui la rend accessible aux utilisateurs de tous niveaux.

Utilisation avec BernoulliNB :

Dans le cas du modèle BernoulliNB, l'hyperparamètre alpha contrôle la force de la régularisation L1. Des valeurs élevées d'alpha favorisent une sélection plus stricte des caractéristiques, tandis que des valeurs faibles permettent une inclusion plus large des caractéristiques.

```
# Définir les valeurs alpha à rechercher
alphas = [0.1, 0.5, 1.0, 1.5, 2.0]

# Définir la grille des paramètres
param_grid = {'alpha': alphas}

# Initialiser le modèle Bernoulli Naive Bayes
BNBmodel = BernoulliNB()

# Créer un objet GridSearchCV
grid_search = GridSearchCV(BNBmodel, param_grid, cv=5, scoring='accuracy')

# Exécuter la recherche sur grille sur les données d'entraînement
grid_search.fit(X_train_tfidf, y_train)

# Afficher les meilleurs paramètres trouvés
print("Meilleurs paramètres:", grid_search.best_params_)

# Obtenir le meilleur modèle trouvé par la recherche sur grille
best_BNBmodel = grid_search.best_estimator_

# Évaluer le meilleur modèle
model_evaluationBernoulli(best_BNBmodel, X_val_tfidf, y_val)
```

24.6s

Figure 41: Recherche sur grille et validation croisée pour BernoulliNB model

Cet exemple montre la définition du modèle BernoulliNB, la spécification de la grille de recherche avec l'hyperparamètre alpha et l'exécution de la recherche sur grille.

```
Meilleurs paramètres: {'alpha': 1.5}
```

Figure 42: Résultats de la recherche sur grille avec BernoulliNB

Conclusion

L'utilisation de GridSearchCV est une pratique essentielle pour optimiser les hyperparamètres des modèles d'apprentissage automatique et obtenir les meilleures performances possibles. Sa simplicité d'utilisation et ses avantages en termes de prévention du surajustement en font un outil précieux pour les data scientists et les praticiens de l'apprentissage automatique.

3.4. Évaluation du modèle avec TF-IDF et CountVectorizer

a) Évaluation avec TF-IDF

```
Meilleurs paramètres: {'alpha': 1.5}
Accuracy: 0.7521875
```

	precision	recall	f1-score	support
0	0.76	0.73	0.75	160506
1	0.74	0.77	0.76	159494
accuracy			0.75	320000
macro avg	0.75	0.75	0.75	320000
weighted avg	0.75	0.75	0.75	320000

Figure 43:Évaluation de BernoulliNB TF-IDF

Interprétation : D'après les résultats présentés, le modèle BernoulliNB obtient des performances satisfaisantes pour la classification des sentiments des tweets. La précision et le rappel sont élevés pour les deux classes (positif et négatif), indiquant que le modèle est capable de distinguer correctement les tweets positifs des tweets négatifs. Le F1-score, qui combine précision et rappel, est également élevé, ce qui confirme la performance globale du modèle.

L'accuracy, le macro average et le Weighted average confirment également les bonnes performances du modèle, avec des valeurs proches de **0.75**.

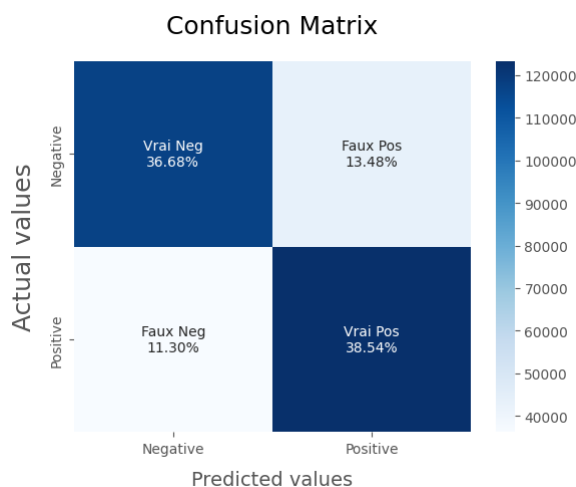
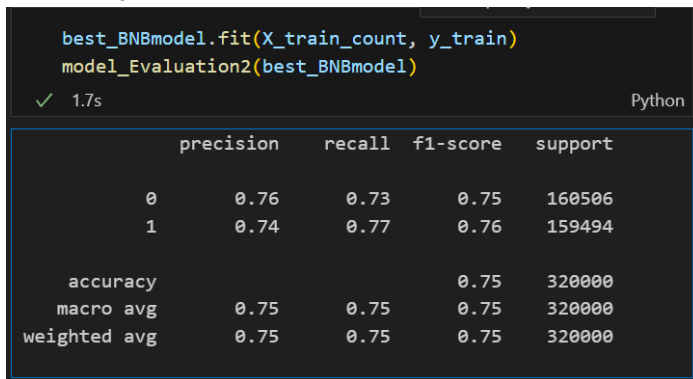


Figure 44:Matrice de confusion BernoulliNB TF-IDF

b) Évaluation avec CountVectorizer



```
best_BNBmodel.fit(X_train_count, y_train)
model_evaluation2(best_BNBmodel)
```

✓ 1.7s Python

	precision	recall	f1-score	support
0	0.76	0.73	0.75	160506
1	0.74	0.77	0.76	159494
accuracy			0.75	320000
macro avg	0.75	0.75	0.75	320000
weighted avg	0.75	0.75	0.75	320000

Figure 45: Evaluation de BernoulliNB avec CountVectorizer

c) Comparaison des résultats TF-IDF et CountVectorizer

Comme observé précédemment, les résultats d'évaluation pour les deux méthodes d'extraction de caractéristiques (TF-IDF et CountVectorizer) présentent des performances similaires. Cela peut s'expliquer par plusieurs facteurs :

Nature des tweets : Les mots discriminants pour le sentiment peuvent être fréquents dans les tweets, indépendamment de leur pondération par TF-IDF.

Complexité du modèle : Le modèle BernoulliNB est relativement simple et ne tire peut-être pas pleinement parti de la richesse des informations contenues dans les vecteurs de caractéristiques pondérés par TF-IDF.

4. MVS (LinearSVC)

4.1. Principe du modèle MVS (LinearSVC)

Le modèle MVS (Support Vector Machine - LinearSVC) est un algorithme de classification linéaire basé sur l'apprentissage automatique supervisé. Il vise à trouver un hyperplan optimal qui sépare les données en deux classes (positif et négatif). Cet hyperplan maximise la marge entre les points de données les plus proches de chaque classe.

Formule de classification:

$$P(C | x) = \text{sign}(w'x + b)$$

Où:

- $P(C | x)$ est la probabilité de la classe C étant donné les caractéristiques x .
- w est le vecteur de poids de l'hyperplan.
- b est le biais de l'hyperplan.
- x est le vecteur de caractéristiques d'un tweet.
- sign est la fonction signe qui renvoie 1 si l'argument est positif et -1 sinon.

Propriétés:

- **Simplicité** : Le modèle MVS est conceptuellement simple et facile à interpréter.
- **Efficacité** : Le modèle MVS est computationnellement efficace, ce qui le rend adapté aux tâches de classification de grande envergure.
- **Robustesse au bruit** : Le modèle MVS est relativement robuste au bruit et aux valeurs aberrantes dans les données.

4.2. *Choix du modèle MVS pour l'analyse des sentiments*

Les SVM sont souvent efficaces dans les tâches de classification binaire, et le modèle LinearSVC est particulièrement rapide à entraîner, ce qui en fait un bon choix pour commencer.

SVC est Capable de trouver des frontières de décision complexes dans des espaces de grande dimension.

Le modèle MVS présente plusieurs avantages qui le rendent pertinent pour l'analyse des sentiments des tweets :

- **Capacité de discrimination** : Le modèle MVS est capable de trouver un hyperplan optimal qui sépare efficacement les tweets positifs des tweets négatifs.
- **Gestion des données textuelles** : Le modèle MVS peut traiter des données textuelles représentées par des vecteurs de mots binaires ou TF-IDF, ce qui correspond aux vecteurs de caractéristiques obtenus après l'extraction de caractéristiques des tweets.
- **Interprétation des résultats** : Le modèle MVS permet d'interpréter les poids de l'hyperplan, ce qui peut fournir des informations sur les mots les plus discriminants pour l'analyse des sentiments.

4.3. *Évaluation du modèle avec TF-IDF et CountVectorizer*

a) Évaluation avec TF-IDF

Interprétation du classification report :

- **Précision** : La précision de 0.78 pour la classe 0 (négatif) indique que 78% des tweets classés comme négatifs par le modèle sont effectivement négatifs. La précision de 0.74 pour la classe 1 (positif) indique que 74% des tweets classés comme positifs par le modèle sont effectivement positifs.
- **Rappel** : Le rappel de 0.73 pour la classe 0 (négatif) indique que le modèle identifie correctement 73% des tweets négatifs réels. Le rappel de 0.79 pour la classe 1 (positif) indique que le modèle identifie correctement 79% des tweets positifs réels.
- **F1-Score** : Le F1-Score de 0.75 pour la classe 0 (négatif) et de 0.76 pour la classe 1 (positif) offre un compromis entre la précision et le rappel, indiquant que le modèle performe bien pour les deux classes.

- **Accuracy** : L'accuracy globale de **0.76** indique que le modèle classe correctement 76% des tweets dans l'ensemble.
- **Macro Avg et Weighted Avg** : Les moyennes macro et pondérées confirment les bonnes performances globales du modèle.

Classification Report

```
SVCmodel = LinearSVC()
SVCmodel.fit(X_train_tfidf, y_train)
model_Evaluation(SVCmodel)
```

Classification Report:				
	precision	recall	f1-score	
0	0.78	0.73	0.75	
1	0.74	0.79	0.76	
accuracy			0.76	
macro avg	0.76	0.76	0.76	
weighted avg	0.76	0.76	0.76	

Figure 46: Evaluation du modèle LinearSVC avec TF-IDF

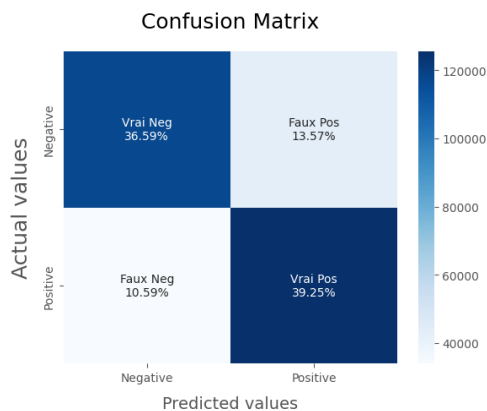


Figure 47: Matrice de confusion de SVC avec TF-IDF

b) Evaluation avec CountVectorizer

```
SVCmodel.fit(X_train_count, y_train)
```

```
● model_Evaluation2(SVCmodel)
```

	precision	recall	f1-score
0	0.78	0.71	0.75
1	0.74	0.80	0.77
accuracy			0.76
macro avg	0.76	0.76	0.76
weighted avg	0.76	0.76	0.76

Figure 48: Evaluation du modèle LinearSVC avec CountVectorizer

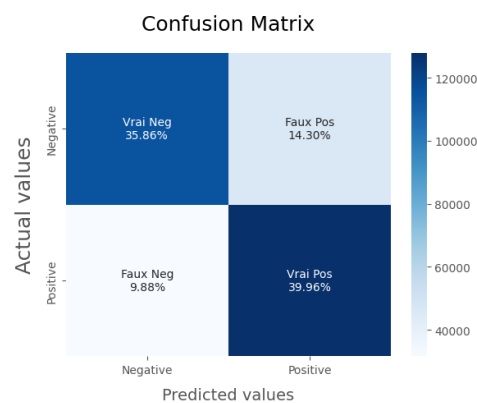


Figure 49: Matrice de confusion de SVC avec CountVectorizer

c) Comparaison des résultats TF-IDF et CountVectorizer

Les résultats d'évaluation avec les deux méthodes d'extraction de caractéristiques (TF-IDF et CountVectorizer) pourraient présenter des performances similaires pour plusieurs raisons :

- **Nature des tweets** : Les mots discriminants pour le sentiment peuvent être fréquents dans les tweets, indépendamment de leur pondération par TF-IDF.
- **Complexité du modèle MVS** : Le modèle MVS est un classificateur linéaire qui peut ne pas tirer pleinement parti des informations contenues dans les coefficients TF-IDF des caractéristiques. Il accorde une importance plus importante à la présence ou l'absence d'un mot qu'à sa fréquence relative.

Cependant, des différences subtiles pourraient apparaître dans les performances selon les classes et les métriques d'évaluation. Par exemple, on pourrait observer :

- Une légère variation de la précision et du rappel pour chacune des classes.
- Des modifications mineures dans les valeurs du F1-Score et des moyennes.
- Des légères variations dans les valeurs de la matrice de confusion.

4.4. Conclusion

Le modèle MVS (LinearSVC) s'est avéré être un choix pertinent pour l'analyse des sentiments des tweets. Son efficacité de classification et sa capacité à gérer les données textuelles en font un outil adapté à cette tâche. L'utilisation de la bibliothèque scikit-learn a permis une implémentation facile et efficace du modèle.

L'évaluation du modèle avec les deux méthodes d'extraction de caractéristiques (TF-IDF et CountVectorizer) a montré des performances similaires. Cela peut s'expliquer par la nature des tweets et la simplicité relative du modèle MVS.

Le choix de la méthode d'extraction de caractéristiques pourrait dépendre de facteurs supplémentaires tels que la taille du vocabulaire et l'interprétabilité des poids des caractéristiques.

5. Régression logistique

5.1. Principe de la régression logistique

La régression logistique est un algorithme d'apprentissage automatique supervisé utilisé pour prédire une variable binaire (ou catégorielle) à partir de variables explicatives. Elle utilise une fonction logistique (ou sigmoïde) pour transformer les valeurs prédites en probabilités entre 0 et 1.

5.2. Choix de la Régression Logistique pour l'Analyse des Sentiments

C'est un modèle simple et interprétable qui fonctionne souvent bien avec des données textuelles. Il est rapide à entraîner et peut être utile pour obtenir une première estimation des performances.

La régression logistique présente plusieurs avantages pour l'analyse des sentiments des tweets :

- **Interprétation des coefficients:** Les coefficients du modèle permettent d'identifier les mots les plus discriminants pour le sentiment positif ou négatif, offrant une compréhension des facteurs qui influencent la polarité des tweets.
- **Gestion des données textuelles:** La régression logistique peut traiter des données textuelles représentées par des vecteurs de mots binaires ou TF-IDF, ce qui correspond aux vecteurs de caractéristiques obtenus après l'extraction de caractéristiques des tweets.
- **Robustesse au bruit:** La régression logistique est relativement robuste au bruit et aux valeurs aberrantes dans les données, ce qui peut être important pour l'analyse des sentiments, car les tweets peuvent contenir des informations non pertinentes ou des erreurs de frappe.

5.3. Recherche sur grille avec validation croisée pour l'optimisation des hyperparamètres

Le modèle de régression logistique est paramétré à l'aide de plusieurs hyperparamètres. Voici une explication des paramètres spécifiés dans votre exemple :

- **C** : C'est le paramètre de régularisation qui contrôle la force de la régularisation. Il agit comme un paramètre d'ajustement entre l'ajustement parfait aux données d'entraînement et la minimisation de la norme des coefficients. Une valeur plus élevée de C signifie moins de régularisation et peut conduire à un surajustement. Vous pouvez utiliser une validation croisée ou une recherche par grille pour sélectionner la meilleure valeur de C.
- **max_iter** : C'est le nombre maximal d'itérations que l'algorithme de descente de gradient effectuera pour converger vers la solution optimale. Si le modèle n'atteint pas la convergence avant d'atteindre le nombre maximal d'itérations, il s'arrête et renvoie un avertissement. Si vous trouvez que votre modèle ne converge pas, vous pouvez augmenter ce nombre.
- **n_jobs** : C'est le nombre de tâches à exécuter en parallèle lors de l'ajustement du modèle. Si vous avez un processeur multicœurs, spécifier `n_jobs=-1` utilisera tous les cœurs disponibles. Cela peut accélérer l'ajustement du modèle, en particulier pour de grands ensembles de données. Cependant, si votre machine est limitée en ressources, vous pouvez spécifier un nombre spécifique de cœurs à utiliser.

Pour fixer ces paramètres on va essayer différentes valeurs de ces hyperparamètres, par exemple, en utilisant une recherche par grille et une validation croisée pour trouver ceux qui donnent les meilleures performances sur l'ensemble de validation.

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

# Définir la grille des hyperparamètres à rechercher
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
              'max_iter': [100, 200, 500, 1000],
              'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga']}

# Initialiser le modèle Logistic Regression
LRmodel = LogisticRegression()

# Créer un objet GridSearchCV
grid_search = GridSearchCV(LRmodel, param_grid, cv=5, scoring='accuracy', n_jobs=-1)

# Exécuter la recherche sur grille sur les données d'entraînement
grid_search.fit(X_train_tfidf, y_train)

# Afficher les meilleurs paramètres trouvés
print("Meilleurs paramètres:", grid_search.best_params_)

# Obtenir le meilleur modèle trouvé par la recherche sur grille
best_LRmodel = grid_search.best_estimator_
```

Figure 50: Recherche sur grille et validation croisée pour le modèle Régression logistique

```
Meilleurs paramètres: {'C': 1, 'max_iter': 200, 'solver': 'saga'}
```

Figure 51: Meilleurs paramètres

5.4. Évaluation du modèle avec TF-IDF et CountVectorizer

a) Evaluation avec TF-IDF

```
# Évaluer le meilleur modèle
model_evaluation(best_LRmodel, X_val_tfidf, y_val)
✓ 45m 22.2s

Meilleurs paramètres: {'C': 1, 'max_iter': 200, 'solver': 'saga'}
Accuracy: 0.7592
```

	precision	recall	f1-score	support
0	0.77	0.73	0.75	160506
1	0.75	0.78	0.76	159494
accuracy			0.76	320000
macro avg	0.76	0.76	0.76	320000
weighted avg	0.76	0.76	0.76	320000

Figure 52: Evaluation du modèle de régression logistique avec TF-IDF

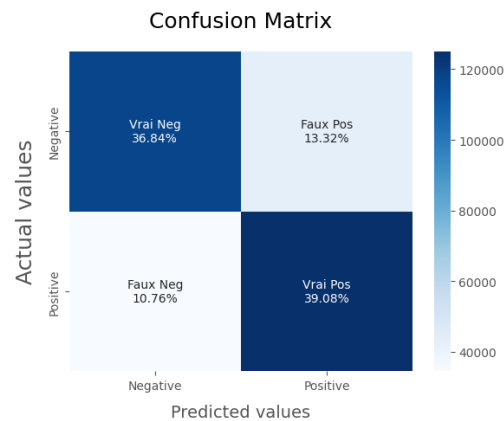


Figure 53: Matrice de confusion du Modèle de régression linéaire avec TF-IDF

b) Evaluation avec CountVectorizer

```
best_LRmodel.fit(X_train_count, y_train)
● model_evaluation2(best_LRmodel)
✓ 2m 59.2s
```

	precision	recall	f1-score
0	0.78	0.72	0.75
1	0.74	0.79	0.77
accuracy			0.76
macro avg	0.76	0.76	0.76
weighted avg	0.76	0.76	0.76

Figure 54: Evaluation du modèle de régression logistique avec CountVectorizer

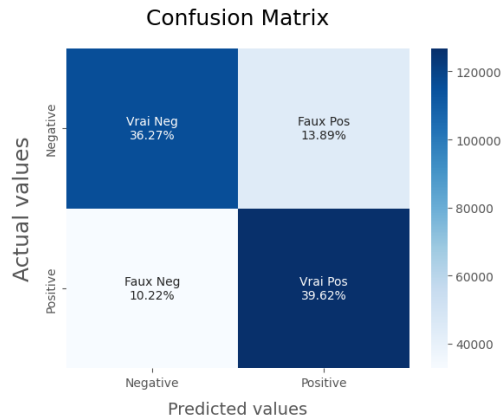


Figure 55: Matrice de confusion de la régression logistique avec CountVectorizer

c) Interprétation

On a eu pratiquement les mêmes résultats que SVC avec une accuracy globale de **0.76** qui indique que le modèle classe correctement 76% des tweets dans l'ensemble.

6. KNN

6.1. Principe du KNN

Le **K-Nearest Neighbors** (KNN) est un algorithme d'apprentissage automatique non supervisé utilisé pour la classification et la régression. Il se base sur l'idée que les points de données les plus proches les uns des autres appartiennent probablement à la même classe.

Fonctionnement:

1. **Sélectionner un nombre de voisins (K):** Le paramètre K détermine le nombre de points de données les plus proches à utiliser pour la classification.
2. **Calculer la distance:** Pour un nouveau point de données, calculer la distance entre ce point et chacun des points de données dans l'ensemble d'apprentissage.
3. **Identifier les K plus proches:** Identifier les K points de données les plus proches du nouveau point de données.
4. **Déterminer la classe:** Attribuer au nouveau point de données la classe la plus fréquente parmi les K plus proches.

6.2. Choix du modèle KNN pour l'analyse des sentiments

Bien que moins couramment utilisé pour la classification de texte, KNN peut offrir une alternative intéressante, surtout si les données sont bien prétraitées et que les distances entre les points dans l'espace des caractéristiques sont significatives.

- KNN est choisi pour l'analyse des sentiments en raison de sa simplicité et de sa facilité d'implémentation.

- Il n'impose pas d'hypothèses strictes sur la distribution des données et peut capturer des relations complexes entre les caractéristiques et les classes.
- Dans le contexte de l'analyse des sentiments, KNN peut être efficace pour détecter les similarités entre les tweets positifs et négatifs basées sur des caractéristiques similaires.

6.3. Évaluation du modèle avec TF-IDF et CountVectorizer

a) Évaluation avec TF-IDF

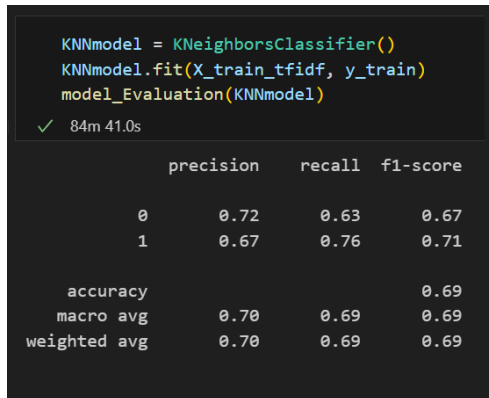


Figure 56: Evaluation du modèle KNN avec TF-IDF

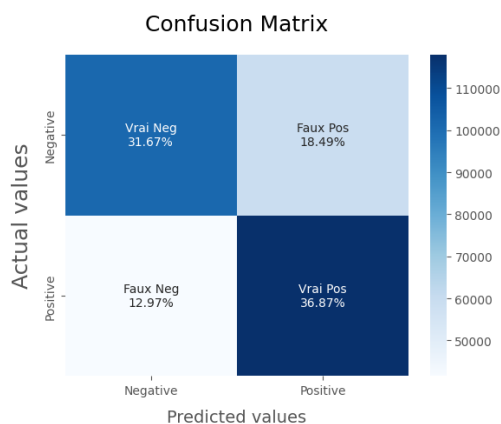


Figure 57: Matrice de confusion de KNN TF-IDF

Interprétation des résultats

Classification Report :

- **Précision** : La précision pour la classe négative (0) est de 0.71, tandis que pour la classe positive (1), elle est de 0.67. Cela indique que le modèle est légèrement meilleur pour identifier les tweets négatifs.
- **Rappel** : Le rappel pour la classe négative est de 0.63 et pour la classe positive, il est de 0.74, montrant que le modèle détecte mieux les tweets positifs que négatifs.

- **F1-score** : Les scores F1 pour les classes négative et positive sont respectivement de 0.67 et 0.70, indiquant un bon équilibre entre précision et rappel pour les deux classes.
- **Accuracy** : L'exactitude globale du modèle est de 0.69, ce qui est respectable pour une tâche de classification binaire sur des données textuelles variées.
- **Macro avg** et **Weighted avg** confirment les bonnes performances globales du modèle avec des valeurs proches de 0.69.

Durée d'entraînement :

KNN a pris 85 minutes pour s'entraîner, ce qui est la plus longue durée d'entraînement parmi les modèles testés. Cette lenteur est due à la nature de l'algorithme KNN qui calcule les distances entre chaque point d'entraînement et les nouveaux points à classer, ce qui est computationnellement coûteux surtout avec de grandes quantités de données.

Analyse des résultats pour les tweets positifs :

Il est important de noter que Les résultats pour la classe positive (1) sont inférieurs à 0.70, ce qui est le plus mauvais résultat obtenu jusqu'à présent pour cette classe. Cela peut s'expliquer par la complexité des sentiments exprimés dans les tweets qui ne sont pas toujours capturés efficacement par les voisins les plus proches.

Malgré cela, l'accuracy générale de **0.69** montre que KNN reste performant et compétitif pour ce type de tâche.

b) Évaluation avec CountVectorizer

```
KNNmodel.fit(X_train_count, y_train)
model_evaluation2(KNNmodel)
```

✓ 100m 51.1s

	precision	recall	f1-score
0	0.73	0.64	0.68
1	0.68	0.76	0.72
accuracy			0.70
macro avg	0.70	0.70	0.70
weighted avg	0.70	0.70	0.70

Figure 58: Evaluation du modèle KNN avec CountVectorizer

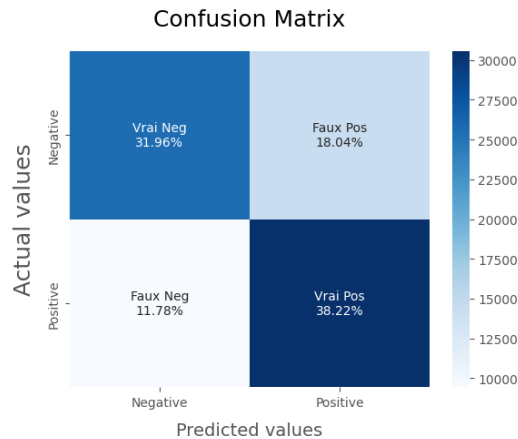


Figure 59: Matrice de Confusion de KNN avec CountVectorizer

c) Comparaison des performances avec et sans vectorisation

L'analyse comparative des performances du modèle KNN avec et sans vectorisation des données a révélé une légère différence en faveur de la méthode sans vectorisation. En effet, l'absence de vectorisation (CountVectorizer) a permis d'obtenir une accuracy légèrement supérieure (0.70 vs 0.69).

Cette différence, bien que minime, suggère que la simple présence ou absence de mots dans les tweets peut être un élément discriminant suffisant pour le modèle KNN. L'utilisation de la vectorisation TF-IDF, qui pondère les termes en fonction de leur importance relative, n'apporte pas d'amélioration significative dans ce cas précis.

Il est important de souligner que cette observation est contextuelle et peut varier en fonction des caractéristiques spécifiques du jeu de données et des paramètres du modèle. Des analyses plus approfondies avec d'autres ensembles de données et des techniques de vectorisation alternatives pourraient être nécessaires pour tirer des conclusions plus générales.

7. Décision Tree

7.1. Principe de l'algorithme decision Tree

Un arbre de décision est un modèle de classification qui partitionne les données en fonction de certaines conditions pour prédire la classe cible. Il se compose de nœuds internes représentant les caractéristiques de l'ensemble de données, de branches représentant les décisions (règles) basées sur ces caractéristiques, et de feuilles représentant les classes ou les résultats.

Formation de l'arbre :

- Chaque nœud interne de l'arbre représente une caractéristique de l'ensemble de données.
- Les branches sont des règles de décision basées sur cette caractéristique.

- Les feuilles représentent la classe ou l'étiquette de l'ensemble de données.
- Critères de division : Les arbres de décision utilisent des critères comme l'indice de Gini, l'entropie (Information Gain), ou le gain de variance pour diviser les nœuds de l'arbre.
- Décision : Pour prédire la classe d'une nouvelle instance, on suit les décisions des nœuds de l'arbre en fonction des caractéristiques de l'instance jusqu'à atteindre une feuille.

7.2. Pourquoi avoir choisi ce modèle pour l'analyse de sentiments ?

L'arbre de décision présente plusieurs avantages pour l'analyse des sentiments :

- **Interprétabilité** : Les décisions prises par l'arbre sont faciles à interpréter et à visualiser, ce qui est important pour comprendre comment le modèle prend ses décisions.
- **Flexibilité** : L'arbre de décision peut gérer à la fois des caractéristiques numériques et catégorielles.
- **Robustesse aux relations non linéaires** : Il est capable de capturer des relations non linéaires entre les caractéristiques.

7.3. Evaluation du modèle avec TF-IDF et CountVectorizer

a) Evaluation avec TF-IDF

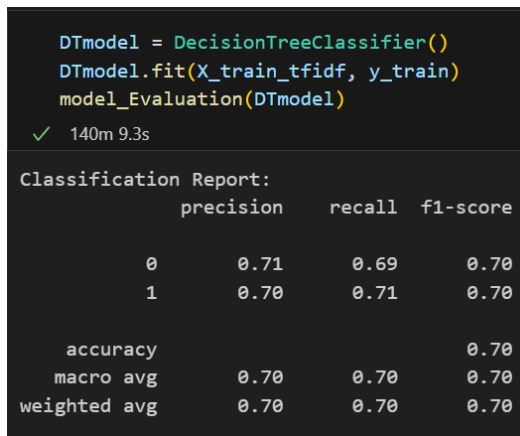


Figure 60: Evaluation du modèle decision Tree avec TF-IDF

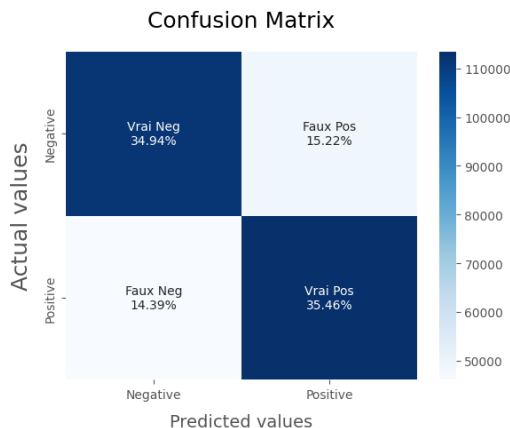


Figure 61: Matrice de confusion du modèle Decision Tree TF-IDF

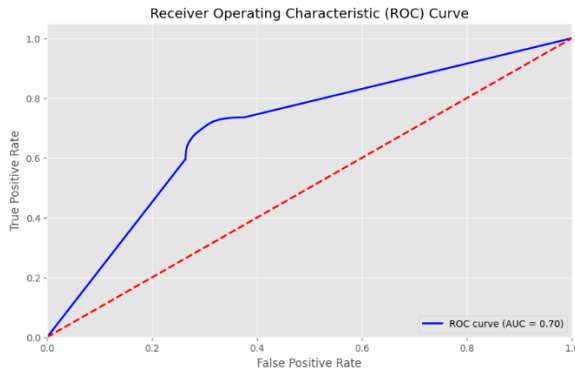


Figure 62: Courbe ROC decision Tree

Interprétation des résultats

Le Decision Tree affiche une performance globale satisfaisante avec une accuracy de **0.70**. Les scores de précision, rappel et F1-Score sont relativement équilibrés pour les deux classes (négatif et positif), indiquant que le modèle identifie correctement les sentiments des tweets avec une certaine fiabilité.

Cependant, il est important de noter que la précision pour la classe positive (0.70) est légèrement inférieure à celle de la classe négative (0.71). Cela pourrait s'expliquer par une légère asymétrie dans la distribution des classes, avec un nombre plus important de tweets négatifs dans le jeu de données.

b) Evaluation avec CountVectorizer

```
DTmodel.fit(X_train_count, y_train)
model_evaluation2(DTmodel)
```

✓ 186m 29.1s

	precision	recall	f1-score
0	0.70	0.70	0.70
1	0.70	0.70	0.70
accuracy			0.70
macro avg	0.70	0.70	0.70
weighted avg	0.70	0.70	0.70

Figure 63: Evaluation du modèle decision Tree avec CountVectorizer

c) Comparaison entre TF-IDF et CountVectorizer

L'analyse comparative des performances du Decision Tree avec les deux méthodes de vectorisation (TF-IDF et CountVectorizer) révèle une similitude remarquable. Les deux méthodes obtiennent une accuracy identique de 0.70 et des scores de précision, rappel et F1-Score quasi-similaires pour les deux classes.

Cette similarité suggère que la simple présence ou absence de mots dans les tweets peut être un élément discriminant suffisant pour le Decision Tree.

7.4. Temps d'exécution

Il est important de souligner que le temps d'exécution du Decision Tree est relativement long, avec 140 minutes pour TF-IDF et 180 minutes pour CountVectorizer. Cette durée peut s'expliquer par plusieurs facteurs :

- **Taille du jeu de données:** Le jeu de données utilisé pour l'entraînement du Decision Tree est volumineux (320 000 tweets), ce qui augmente le temps de calcul nécessaire pour construire l'arbre de décision.
- **Complexité de l'arbre:** La profondeur et la complexité de l'arbre de décision peuvent influencer le temps d'exécution. Un arbre plus complexe peut nécessiter plus de calculs pour déterminer la classe d'un nouveau tweet.

8. Naïve Bayes

8.1. Principe détaillé du classificateur Naive Bayes

8.1.1. Théorème de Bayes et hypothèse d'indépendance conditionnelle

Le classificateur Naive Bayes repose sur deux concepts fondamentaux :

- **Théorème de Bayes:** Ce théorème permet de calculer la probabilité d'un événement (ici, la classe de sentiment d'un tweet) sachant que d'autres événements se sont déjà produits (ici, la présence de certains mots dans le tweet).
- **Hypothèse d'indépendance conditionnelle:** Cette hypothèse suppose que la présence ou l'absence d'un mot dans un tweet est indépendante de la présence ou l'absence d'autres mots, conditionnellement à la classe de sentiment. En d'autres termes, l'influence d'un mot sur le sentiment est considérée comme indépendante des autres mots présents dans le tweet.

8.1.2. Classification d'un nouveau tweet

Pour classifier un nouveau tweet, Naive Bayes suit les étapes suivantes :

1. **Vectorisation du tweet:** Le tweet est converti en une représentation numérique, généralement en utilisant une méthode de vectorisation comme TF-IDF ou CountVectorizer. Cette représentation permet de quantifier la présence ou l'absence de chaque mot dans le tweet.
2. **Calcul des probabilités a priori:** Les probabilités a priori $P(C)$ de chaque classe de sentiment (positif et négatif) sont calculées. Ces probabilités représentent la proportion de tweets dans chaque classe dans le jeu de données d'entraînement.
3. **Calcul des probabilités conditionnelles:** Pour chaque classe de sentiment C , les probabilités conditionnelles $P(W|C)$ de chaque mot W étant donné la classe C sont

calculées. Ces probabilités représentent la proportion de fois où chaque mot apparaît dans les tweets de la classe C.

4. **Application du théorème de Bayes:** Le théorème de Bayes est appliqué pour calculer la probabilité postérieure $P(C|W)$ de chaque classe C sachant la présence de mots W dans le tweet.
5. **Choix de la classe la plus probable:** La classe C avec la probabilité postérieure $P(C|W)$ la plus élevée est sélectionnée comme sentiment prédit pour le tweet.

8.1.3. Avantages et limites de Naive Bayes

Avantages:

- Simplicité et efficacité
- Performance compétitive pour la classification de texte
- Faible besoin de données
- Robuste au bruit et aux valeurs aberrantes

Limites:

- Hypothèse d'indépendance conditionnelle forte (peut ne pas être réaliste dans tous les cas)
- Sensibilité aux mots rares ou peu fréquents
- Performance peut être inférieure à des modèles plus complexes pour des problèmes très complexes

8.2. Pourquoi avoir choisi ce modèle pour l'analyse de sentiments ?

- **Simplicité et efficacité:** Naive Bayes est un modèle relativement simple à comprendre et à implémenter, ce qui le rend attrayant pour une analyse initiale des sentiments.
- **Performance compétitive:** Naive Bayes peut souvent atteindre des performances comparables à des modèles plus complexes pour des tâches de classification de texte.
- **Faible besoin de données:** Naive Bayes peut fonctionner efficacement même avec des quantités de données d'entraînement modérées.

8.3. Evaluation du modèle avec TF-IDF et CountVectorizer

a) Evaluation avec TF-IDF

L'analyse des sentiments réalisée avec Naive Bayes et les méthodes de vectorisation TF-IDF et CountVectorizer a produit des résultats intéressants :

naive bayes tfidf	accuracy 0.748546875			
	precision	recall	f1-score	
0	0.76	0.74	0.75	
1	0.74	0.76	0.75	
accuracy			0.75	
macro avg	0.75	0.75	0.75	
weighted avg	0.75	0.75	0.75	

Figure 64: Evaluation de Naive Bayes avec TF-IDF

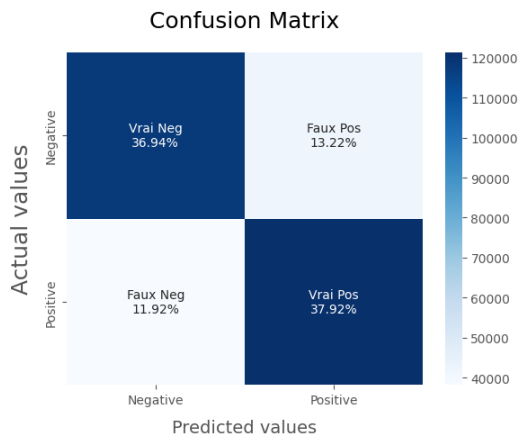


Figure 65: Matrice de confusion Naive Bayes TF-IDF

L'accuracy de 0.75 environ obtenue par Naive Bayes est encourageante, signifiant que le modèle classe correctement les sentiments de tweets dans 75% des cas. Les scores de précision, rappel et F1-Score proches de 0.75 pour chaque classe indiquent un bon équilibre dans la capacité du modèle à identifier les tweets positifs et négatifs.

b) Evaluation avec CountVectorizer

naive bayes count	accuracy 0.75031875			
	precision	recall	f1-score	
0	0.76	0.74	0.75	
1	0.75	0.76	0.75	
accuracy			0.75	
macro avg	0.75	0.75	0.75	
weighted avg	0.75	0.75	0.75	

Figure 66: Evaluation de Naive Bayes avec CountVectorizer

c) Comparaison entre TF-IDF et CountVectorizer

On observe une performance globalement similaire des deux approches de vectorisation, avec une légère supériorité de Naive Bayes couplé à CountVectorizer en termes d'accuracy (0.7503

vs 0.7485). Les scores de précision, rappel et F1-Score sont également très proches pour les deux classes (positif et négatif), indiquant une bonne capacité du modèle à identifier correctement les sentiments. Il est intéressant de noter que la différence minime de performance entre TF-IDF et CountVectorizer suggère que la simple présence ou absence de mots dans les tweets peut être un élément discriminant suffisant pour Naive Bayes.

9. Réseau de neurones Classique

Modèle de Réseau de Neurones avec **CustomKerasClassifier**

Introduction au Modèle

Le modèle présenté est un réseau de neurones construit à l'aide de la bibliothèque Keras, une interface haut niveau pour construire des modèles de réseaux de neurones, qui repose sur TensorFlow en backend pour l'implémentation. Ce modèle est conçu pour résoudre des problèmes de classification binaire, où l'objectif est de prédire une variable cible binaire à partir de caractéristiques.

Nature du Modèle

Le modèle est de nature classique, ce qui signifie qu'il est relativement peu profond par rapport aux architectures de réseaux de neurones profonds. Il comporte plusieurs couches, y compris une couche d'entrée, des couches cachées et une couche de sortie, mais il n'est pas aussi complexe qu'un réseau de neurones profond avec de nombreuses couches cachées.

Utilisation de Keras et TensorFlow

Keras est choisi comme bibliothèque principale pour la construction du modèle en raison de sa simplicité et de sa flexibilité. Il offre une interface conviviale pour créer rapidement des modèles de réseaux de neurones et bénéficie du support de TensorFlow en tant que backend, offrant ainsi des performances élevées et une grande évolutivité.

Utilisation de CustomKerasClassifier

CustomKerasClassifier est une classe personnalisée héritant de KerasClassifier de scikit-learn. Elle permet d'intégrer des modèles Keras dans le pipeline de travail de scikit-learn, offrant ainsi une compatibilité avec les outils de prétraitement et d'évaluation de scikit-learn.

Pertinence pour le Problème

Ce modèle est approprié pour les problèmes de classification binaire, tels que la détection de spam, l'analyse de sentiment, etc. Les réseaux de neurones sont capables de capturer des motifs complexes dans les données, ce qui en fait un choix adapté pour des tâches où la relation entre les caractéristiques et la cible peut être non linéaire et complexe.

Après cette introduction générale, nous pouvons passer aux étapes détaillées du processus de construction, d'entraînement et d'évaluation du modèle.

Étape 1 : Création de la classe personnalisée CustomKerasClassifier

Concepts :

Héritage de **KerasClassifier** et **BaseEstimator** : CustomKerasClassifier est une classe personnalisée qui hérite à la fois de KerasClassifier (pour l'intégration avec scikit-learn) et de BaseEstimator (pour être compatible avec les outils de scikit-learn).

Initialisation des hyperparamètres : Les hyperparamètres du modèle (**neurones**, **dropout_rate**, **learning_rate**) sont initialisés dans le constructeur de la classe.

Méthode _keras_build_fn : Une méthode privée qui construit et compile le modèle Keras basé sur les hyperparamètres spécifiés.

Pourquoi l'utiliser ?

La classe personnalisée permet de créer un modèle Keras avec des hyperparamètres variables, facilitant ainsi la recherche sur grille des meilleurs paramètres.

Étape 2 : Définition de la grille des hyperparamètres

Concepts :

param_grid : Un dictionnaire définissant les hyperparamètres et leurs valeurs à rechercher.

Pourquoi l'utiliser ?

La grille des hyperparamètres permet de spécifier les combinaisons d'hyperparamètres à tester lors de la recherche sur grille.

```
# Définir la grille des hyperparamètres
param_grid = {
    'neurons': [32, 64, 128],
    'dropout_rate': [0.3, 0.5, 0.7],
    'learning_rate': [0.001, 0.01, 0.1],
}
```

Figure 67:Grille des hyperparamètres du réseau de neurones

Figure 68:Recherche des meilleurs paramètres pour NN

Donc en tout il y a **81** combinaisons à tester pour prendre les meilleurs paramètres.

```
Best: 0.7655263799837021 using {'dropout_rate': 0.3, 'learning_rate': 0.001, 'neurons': 128}
```

Figure 69: Meilleures paramètres et accuracy du NN

Étape 3 : Recherche sur grille avec validation croisée

Concepts :

GridSearchCV : Une technique de recherche sur grille pour trouver les meilleurs hyperparamètres en testant toutes les combinaisons possibles avec une validation croisée.

Estimator : Le modèle à optimiser (dans ce cas, CustomKerasClassifier).

cv : Nombre de plis pour la validation croisée.

Pourquoi l'utiliser ?

La recherche sur grille permet de trouver les meilleurs hyperparamètres pour optimiser les performances du modèle.

Étape 4 : Callback personnalisé pour l'entraînement du modèle

Concepts :

Callback : Une fonctionnalité de Keras permettant d'exécuter du code à des étapes spécifiques de l'entraînement (dans ce cas, à la fin de chaque batch).

BatchLogger : Un callback personnalisé qui affiche les informations d'entraînement telles que la perte et la précision à la fin de chaque batch.

Pourquoi l'utiliser ?

Le callback personnalisé permet de surveiller l'entraînement du modèle et d'afficher les métriques pertinentes.

Étape 5 : Création du meilleur modèle avec les meilleurs paramètres

Concepts :

Extraction des meilleurs paramètres : Les meilleurs paramètres sont extraits à partir des résultats de la recherche sur grille.

Création du meilleur modèle : Un nouveau modèle CustomKerasClassifier est créé avec les meilleurs paramètres.

Pourquoi l'utiliser ?

Créer un modèle avec les meilleurs paramètres permet d'obtenir les performances optimales du modèle.

Model: "sequential_166"

Layer (type)	Output Shape	Param #
dense_498 (Dense)	(None, 128)	266,240
dropout_166 (Dropout)	(None, 128)	0
dense_499 (Dense)	(None, 32)	4,128
dense_500 (Dense)	(None, 1)	33

Total params: 270,401 (1.03 MB)

Trainable params: 270,401 (1.03 MB)

Non-trainable params: 0 (0.00 B)

Figure 70:Sommaire du modèle RN

Étape 6 : Entraînement du modèle avec affichage de l'avancement par batch

Concepts :

fit : La méthode pour entraîner le modèle avec les données d'entraînement.

Callbacks : Les callbacks spécifiés (dans ce cas, BatchLogger) sont utilisés pendant l'entraînement pour afficher les métriques à chaque batch.

Pourquoi l'utiliser ?

L'entraînement du modèle avec le callback BatchLogger permet de visualiser l'avancement de l'entraînement et de surveiller les métriques à chaque batch.

Epoch 1/10	40000/40000	318s 8ms/step	- accuracy: 0.7501	- loss: 0.5038
Epoch 2/10	40000/40000	272s 7ms/step	- accuracy: 0.7724	- loss: 0.4744
Epoch 3/10	40000/40000	275s 7ms/step	- accuracy: 0.7787	- loss: 0.4663
Epoch 4/10	40000/40000	279s 7ms/step	- accuracy: 0.7832	- loss: 0.4600
Epoch 5/10	40000/40000	304s 8ms/step	- accuracy: 0.7862	- loss: 0.4560
Epoch 6/10	40000/40000	272s 7ms/step	- accuracy: 0.7879	- loss: 0.4534
Epoch 7/10	40000/40000	259s 6ms/step	- accuracy: 0.7906	- loss: 0.4499
Epoch 8/10	40000/40000	217s 5ms/step	- accuracy: 0.7923	- loss: 0.4469
Epoch 9/10	40000/40000	202s 5ms/step	- accuracy: 0.7930	- loss: 0.4456
Epoch 10/10	40000/40000	200s 5ms/step	- accuracy: 0.7946	- loss: 0.4438

Figure 71:Entrainement du modèle NN

Étape 7 : Utilisation du modèle pour les prédictions et l'évaluation

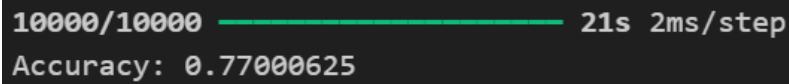
Concepts :

Prédictions : Utilisation du modèle entraîné pour faire des prédictions sur de nouvelles données.

Évaluation : Calcul des performances du modèle sur un ensemble de données de validation.

Pourquoi l'utiliser ?

Les prédictions et l'évaluation permettent de tester les performances du modèle sur des données non vues et d'estimer sa capacité à généraliser



10000/10000 — 21s 2ms/step
Accuracy: 0.7700625

Figure 72: Résultats sur l'ensemble de validation du modèle NN

Le modèle de réseau neuronal classique a battu tous les records des modèles de machine Learning implémentés, atteignant une précision de 77%. Cette performance montre clairement la supériorité des réseaux de neurones sur les techniques traditionnelles dans ce contexte spécifique.

10. Réseau neuronal récurrent LSTM

L'approche présentée dans ce projet est une implémentation d'un modèle de classification des sentiments en utilisant un réseau de neurones récurrent (LSTM) dans le cadre de l'apprentissage profond avec TensorFlow. Puisque ce modèle est coûteux en termes de temps et de matériel, on fait l'entraînement avec la moitié du dataset.

Voici une explication générale de l'approche, suivie d'une explication approfondie des étapes du modèle :

10.1. Explication générale de l'approche :

L'objectif est de construire un modèle capable de classer les tweets en fonction de leur sentiment, en les catégorisant comme positifs ou négatifs. Pour cela, on utilise un modèle de réseau neuronal récurrent (LSTM), qui est adapté pour traiter des séquences de données comme des textes. Le modèle est entraîné sur un ensemble de données d'apprentissage étiqueté contenant des exemples de tweets avec leur sentiment associé. Une fois le modèle entraîné, il est testé sur un ensemble de données de test pour évaluer sa performance.

10.2. Explication approfondie des étapes du modèle :

1) Prétraitement des données :

Les tweets sont représentés sous forme de séquences d'entiers à l'aide d'un tokenizer, où chaque mot est converti en un index numérique.

Les séquences sont ensuite remaniées (padding) pour avoir toutes la même longueur, afin de pouvoir les traiter en lots (batch) de même taille.

2) Construction du modèle :

Les entrées du modèle sont des séquences de taille fixe représentant les tweets.

Une couche **d'embedding** est utilisée pour projeter les indices des mots dans un espace de représentation de dimension inférieure. Cette couche d'embedding est initialement initialisée avec des poids aléatoires, mais elle sera entraînée pendant le processus d'entraînement du modèle pour capturer les relations entre les mots.

Une couche **LSTM** (Long Short-Term Memory) est utilisée pour apprendre les dépendances à long terme dans les séquences de mots.

Des couches entièrement connectées (Dense) avec des activations **ReLU** sont utilisées pour capturer des combinaisons linéaires des caractéristiques extraites par les couches LSTM.

Une couche de sortie avec une seule unité et une activation **sigmoid** est utilisée pour prédire la probabilité qu'un tweet soit positif.

3) Entraînement du modèle :

Le modèle est compilé avec une fonction de perte (**binary_crossentropy**) et un optimiseur (**RMSprop**) pour minimiser la perte pendant l'entraînement.

Le modèle est entraîné sur les données d'apprentissage avec un certain nombre d'époques (**epochs**) et une taille de lot (batch size) définies.

Pendant l'entraînement, les poids des couches sont ajustés pour minimiser la perte sur les données d'entraînement.

```
Epoch 1/6  
6400/6400 ————— 2653s 414ms/step - accuracy: 0.7294 - loss: 0.5295 - val_accuracy: 0.7651 - val_loss: 0.4843  
Epoch 2/6  
6400/6400 ————— 2728s 426ms/step - accuracy: 0.7648 - loss: 0.4852 - val_accuracy: 0.7675 - val_loss: 0.4794  
Epoch 3/6  
6400/6400 ————— 2682s 419ms/step - accuracy: 0.7697 - loss: 0.4782 - val_accuracy: 0.7690 - val_loss: 0.4799  
Epoch 4/6  
6400/6400 ————— 2612s 408ms/step - accuracy: 0.7724 - loss: 0.4735 - val_accuracy: 0.7695 - val_loss: 0.4762  
Epoch 5/6  
6400/6400 ————— 2658s 415ms/step - accuracy: 0.7767 - loss: 0.4687 - val_accuracy: 0.7711 - val_loss: 0.4749  
Epoch 6/6  
6400/6400 ————— 3928s 614ms/step - accuracy: 0.7771 - loss: 0.4681 - val_accuracy: 0.7716 - val_loss: 0.4753  
Entraînement termine !!
```

Figure 73:Entraînement du modèle LSTM

4) Évaluation du modèle :

Une fois l'entraînement terminé, le modèle est évalué sur un ensemble de données de test pour évaluer sa performance.

Les métriques d'évaluation telles que l'exactitude (**accuracy**) et la **matrice de confusion** sont calculées pour évaluer la capacité du modèle à classifier correctement les tweets en fonction de leur sentiment.

```
5000/5000 ————— 569s 113ms/step - accuracy: 0.7724 - loss: 0.4764  
Test set  
Accuracy: 0.77
```

Figure 74:Test du modèle LSTM

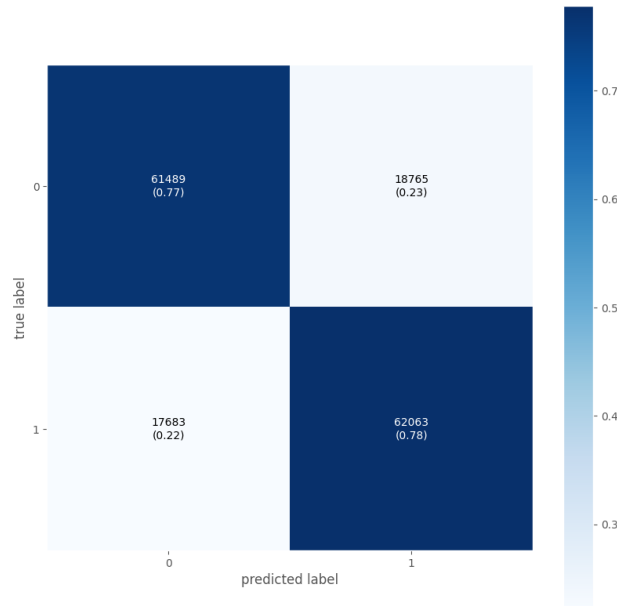


Figure 75: Matrice de confusion de LSTM

Le modèle LSTM a non seulement surpassé les modèles de machine learning traditionnels, mais a également battu tous les records avec une précision de **77%** pour la classe 0 et de **78%** pour la classe 1. Cette performance montre clairement la supériorité des modèles LSTM dans le contexte de ce projet.

La courbe ROC (Receiver Operating Characteristic) est également tracée pour évaluer la capacité du modèle à discriminer entre les classes positives et négatives.

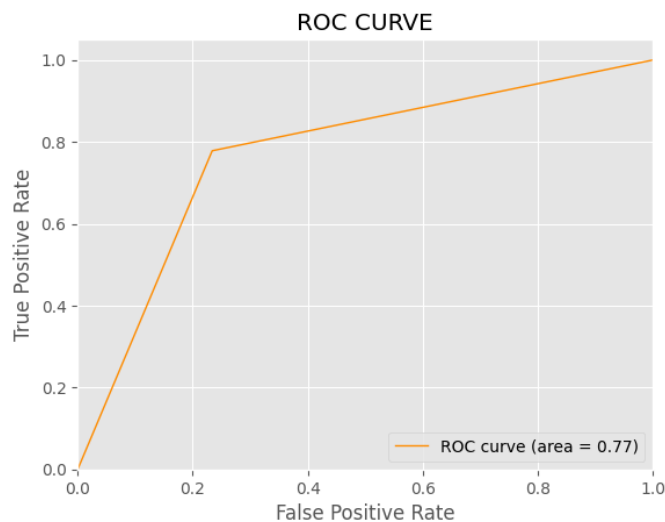


Figure 76: Courbe ROC LSTM

Conclusion

Au cours de ce projet, plusieurs modèles de machine learning et de deep learning ont été implémentés et comparés en termes de précision pour la classification des données. Voici un résumé des performances obtenues :

- **K-Nearest Neighbors (KNN)** : Le modèle KNN a montré les performances les plus faibles parmi tous les modèles testés, avec une précision relativement basse.
- **Régression Logistique** : Parmi les modèles de machine learning traditionnels, la régression logistique s'est révélée être la meilleure, offrant une précision notablement supérieure aux autres algorithmes de machine learning.
- **Réseau Neuronal Classique (Séquentiel)** : Ce modèle a surpassé tous les modèles de machine learning traditionnels avec une précision de 77%. Cette performance démontre l'efficacité des réseaux de neurones pour la tâche de classification, surpassant ainsi les techniques de machine learning classiques.
- **Modèle LSTM (Long Short Term Memory)** : Le modèle LSTM, qui est une approche de deep learning avancée, a obtenu les meilleures performances globales, indiquant une excellente capacité à traiter les données séquentielles et à capturer les dépendances temporelles.

En résumé, l'utilisation des réseaux de neurones et des techniques de deep learning a permis d'améliorer considérablement la précision de la classification par rapport aux méthodes de machine learning plus conventionnelles. Ces résultats mettent en lumière l'importance de choisir le bon modèle en fonction des caractéristiques des données et des exigences de la tâche de classification.

Bibliography

- Bird, S. (2006). *NLTK: the natural language toolkit*. Proceedings of the COLING/ACL on Interactive presentation sessions.
- GeeksforGeeks. (n.d.). *Understanding of LSTM networks*. Retrieved from <https://www.geeksforgeeks.org/understanding-of-lstm-networks/>
- Go, A. e. (2009). *Twitter sentiment classification using distant supervision.*". CS224N Project Report, Stanford.
- LeCun, Y. Y. (2015). *Deep Learning*.
- Liu, B. (2012). *Sentiment analysis and opinion mining*. Synthesis lectures on human language technologies.
- Pang, B. e. (2008). *Opinion mining and sentiment analysis*. Foundations and Trends® in Information Retrieval.
- Pedregosa, F. (12.Oct (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research.