Rapport Projet BIG DATA

Réalisé par :
AIT BRIK Sara - ARI Chaymaa
ASSAG Khadija-ISLAH Zineb

Sujet : Analyse des sentiments

Encadré par :
Prof. Nassima SOUSSI

2023/2024

Résumé

Dans le cadre de notre projet Big Data, nous avons développé un système avancé d'analyse des sentiments, utilisant des algorithmes de machine learning pour classifier les sentiments exprimés en trois catégories : positif, négatif et neutre.

Pour ce projet, nous avons implémenté plusieurs algorithmes de machine learning, dont la régression logistique, les machines à vecteurs de support (SVM), la forêt d'arbres décisionnels (Random Forest), et le Naive Bayes. Chaque algorithme a été sélectionné pour ses atouts spécifiques en matière de traitement des données textuelles et de classification des sentiments. Après avoir testé et comparé ces algorithmes, nous avons retenu celui offrant la précision la plus élevée pour l'intégrer dans notre interface utilisateur.

L'infrastructure de notre projet repose sur PySpark, ce qui nous a permis de gérer et de traiter de grandes quantités de données efficacement et rapidement.

Une des fonctionnalités phares de notre système est une interface utilisateur conviviale. Cette interface permet aux utilisateurs de saisir un commentaire, et de recevoir instantanément une analyse de sentiment indiquant si le sentiment est positif, négatif ou neutre.

Pour la visualisation des résultats, nous avons utilisé Power BI. Cet outil nous a permis de présenter les données de manière claire et attrayante, facilitant ainsi l'interprétation des tendances et des insights obtenus à partir de l'analyse des tweets.

Notre projet se distingue par l'intégration harmonieuse de diverses technologies et algorithmes de machine learning, l'offre d'une interface utilisateur interactive, et des visualisations percutantes. Ce travail démontre la puissance des techniques de machine learning appliquées aux données textuelles et l'importance de la visualisation dans la communication des résultats analytiques.

Table de matière

Tables de Figures

Introduction

L'analyse de sentiment sur Twitter est devenue une pratique incontournable dans le domaine du traitement automatique des langues naturelles. En évaluant les opinions, les émotions et les attitudes exprimées par les utilisateurs à travers leurs tweets, cette technique offre un aperçu précieux de la façon dont le public perçoit divers sujets, événements ou entités.

Ce qui rend cette méthode particulièrement attrayante, c'est sa capacité à extraire des informations à grande échelle à partir d'un immense flux de données en temps réel. Avec des millions de tweets publiés chaque jour sur des sujets variés, Twitter devient une source riche et dynamique d'opinions et de sentiments.

Les entreprises utilisent l'analyse de sentiment pour évaluer la réception de leurs produits ou services, identifier les tendances émergentes, et ajuster leurs stratégies marketing en conséquence. Les chercheurs explorent ce terrain pour étudier les tendances sociologiques, les fluctuations de l'humeur collective, ou même surveiller les crises sanitaires ou politiques.

Les décideurs politiques trouvent également dans cette méthode un outil précieux pour évaluer le soutien ou l'opposition à certaines politiques, mesurer l'impact de leurs discours et campagnes, voire anticiper des mouvements sociaux.

Chapitre 1 : Présentation du projet

I. Objectif :

Réalisation d'une application pour analyser les sentiments des utilisateurs de twitter et les classifiant en trois catégorie positive négative et neutre

II. Description :

Notre projet repose sur un jeu de données composé de commentaires de tweets. Nous le traitons et appliquons divers algorithmes de machine learning afin de créer une application, développée avec Flask, capable de prédire le sentiment véhiculé par n'importe quelle phrase. Ainsi, nous avons réalisé une analyse approfondie du jeu de données et visualisé divers graphiques à l'aide de Power BI.

III. Méthodologies :

Pour créer ce système d'analyse des sentiments, nous nous sommes plongés dans la recherche des algorithmes les plus appropriés pour ce sujet.
Nous avons trouvé plusieurs articles utilisant chacun un modèle différent, et nous avons choisi de travailler avec les modèles suivants :

Régression logistique : est un modèle statistique utilisé pour la classification binaire. Elle repose sur la fonction logistique (ou sigmoïde) qui transforme une combinaison linéaire des caractéristiques en une probabilité.

SVM : (Support Vector Machine ou Machine à vecteurs de support) est un algorithme d'apprentissage automatique supervisé utilisé pour les problèmes de classification.
Il consiste à ramener un problème de classification ou de discrimination à un hyperplan (feature space) dans lequel les données sont séparées en plusieurs classes dont la frontière est la plus éloignée possible des points de données (ou marge maximale).

Random forest : un algorithme populaire d'apprentissage automatique, fusionne les résultats de plusieurs arbres de décision pour produire un seul résultat. Sa popularité tient à sa convivialité et à sa polyvalence, qui en font un outil adapté aux tâches de classification et de régression.

Naive bayes : c'est un ensemble d'algorithmes d'apprentissage supervisé basés sur l'application du théorème de Bayes avec l'hypothèse « naïve » de l'indépendance

conditionnelle entre chaque paire de caractéristiques compte tenu de la valeur de la variable de classe.

IV. Technologie :

1. Spark

Spark est un puissant système informatique distribué open source qui fournit une interface pour programmer des clusters entiers avec un parallélisme de données implicite et une tolérance aux pannes.

En Python, Spark est couramment utilisé via son API PySpark, qui nous permet d'interagir avec Spark à l'aide du code Python.

2. Flask

Flask est un micro framework open source écrit en python, offre aux développeurs un cadre de travail minimal pour qu'ils puissent créer des applications web de manière rapide, simple et efficace.

3. Mongo db (Base de donnees) :

MongoDB est une base de données orientée documents qui permet de stocker des données en utilisant un format flexible et JSON-like, appelé BSON.

4. Power BI

Microsoft Power BI est une solution d'analyse de données de Microsoft. Il permet de créer des visualisations de données personnalisées et interactives avec une interface suffisamment simple pour que les utilisateurs finaux créent leurs propres rapports et tableaux de bord.

Chapitre 2 : État de l'Art de l'Analyse des Sentiments

Introduction

Nous avons été inspirés par un article sur les "Avancées récentes et défis de l'analyse des sentiments basée sur la PNL" pour explorer un large éventail d'informations et tirer diverses connaissances.

L'analyse des sentiments, élément clé du traitement du langage naturel (NLP), identifie et catégorise les émotions dans les données textuelles, facilitant l'évaluation des opinions et attitudes. Elle utilise des techniques variées, allant de la classification basique des sentiments à l'identification de nuances émotionnelles complexes. Ce processus inclut la collecte de données, le prétraitement, l'extraction de fonctionnalités, et l'apprentissage de modèles. Les entreprises et chercheurs tirent profit de ces analyses pour la prise de décision basée sur les données, l'amélioration des produits, et la compréhension des opinions publiques, influençant ainsi les stratégies commerciales et les politiques sociales .

L'analyse des sentiments a récemment progressé grâce aux techniques d'apprentissage automatique (ML) et l'apprentissage profond (DL), améliorant la précision et l'évolutivité des modèles. Notre étude se distingue en fournissant une synthèse exhaustive des avancées récentes, en abordant les applications spécifiques, et les modèles MLet DL .

Applications de l'analyse des sentiments

L'analyse des sentiments a des applications variées dans de nombreux secteurs, offrant des informations précieuses sur les opinions et les sentiments du public. Voici quelques domaines critiques où elle est utilisée :

Soins de santé

L'analyse des sentiments est essentielle dans le secteur des soins de santé pour surveiller les commentaires des patients, les avis en ligne et l'opinion publique. Cela aide les prestataires de soins de santé à améliorer la satisfaction des patients, à prendre des décisions basées sur des données et à comprendre la perception du public à l'égard des politiques et des innovations en matière de santé (Singh et Singh, 2023). Les cas d'utilisation incluent :

- **Al-Mashhadany et al. (2022)** : Analyse des sentiments des commentaires Facebook pour classer les centres de beauté irakiens comme sains ou malsains.
- **Bansal et Kumar (2022)** : Comparaison des hôpitaux sur quatre dimensions clés par l'analyse des sentiments, résultant en un système d'évaluation complet.

- **Ghosh et coll. (2023)** : Système d'identification des sensations douloureuses à partir de l'analyse des expressions faciales utilisant des modèles d'apprentissage profond.
- **Saranya et coll. (2020)** : Prédiction en temps réel de l'impact des maladies à partir de tweets liés à la santé.

Marchés financiers

L'analyse des sentiments sur les marchés financiers permet aux investisseurs et institutions d'extraire des informations précieuses pour la prise de décision éclairée, l'évaluation des risques et la gestion des actifs. Applications récentes :

- **Guo et coll. (2022)** : Stratégie d'investissement utilisant l'apprentissage par renforcement profond et l'analyse des sentiments.

Service client

L'analyse des sentiments est cruciale dans le support client pour superviser le sentiment des clients et améliorer leur satisfaction. Applications courantes :

- **Diekson et al. (2023)** : Analyse des interactions des clients pour identifier les problèmes et évaluer la qualité du service.
- **Étage et parc (2022)** : Analyse des messages de forum d'assistance client pour obtenir des informations sur les émotions générées.

Divertissement

L'analyse des sentiments aide l'industrie du divertissement à évaluer les réactions et les préférences du public, permettant d'adapter le contenu et d'améliorer les stratégies marketing. Applications :

- **Modi et coll. (2022)** : Visualisation des données basée sur le Web pour l'analyse des sentiments des entreprises multinationales.

Éducation

Dans l'éducation, l'analyse des sentiments aide à évaluer l'efficacité des méthodes pédagogiques et à améliorer la satisfaction des étudiants. Applications :

- **Dake et Gyimah (2023)** : Exploration des commentaires qualitatifs des étudiants pour extraire des informations précieuses.

Commerce électronique

L'analyse des sentiments dans le commerce électronique permet d'évaluer la satisfaction des clients à partir des avis sur les produits et d'ajuster les stratégies marketing. Applications :

- **Huang et al. (2023)** : Analyse des avis sur les produits pour améliorer la création et le marketing des produits.

- **Yin et coll. (2022)** : Analyse des sentiments des données Twitter pour les entreprises de commerce électronique.

Méthodes de prétraitement courantes dans l'analyse des sentiments

- *Nettoyage des données*: Il s'agit d'un processus visant à identifier et à corriger les erreurs, les incohérences ou les inexactitudes dans les ensembles de données textuelles brutes. Ce processus est crucial pour améliorer la qualité et la fiabilité des données avant leur analyse.

- *Tokenisation*: Cette méthode implique la division du texte en mots individuels ou en jetons. C'est une étape fondamentale dans le traitement du langage naturel et l'analyse de texte, car elle permet de transformer le texte en une forme que les machines peuvent traiter et comprendre.

- *Sélection de fonctionnalités*: Lors de l'analyse des sentiments, il est souvent nécessaire d'améliorer les performances du modèle tout en réduisant la dimensionnalité des données. La sélection de fonctionnalités consiste à choisir les caractéristiques les plus pertinentes et informatives à partir de l'ensemble de données, ce qui permet de concentrer l'attention du modèle sur les aspects les plus importants du texte .

- *Extraction de caractéristiques*: Cette méthode implique la transformation des données textuelles en une nouvelle représentation qui capture les informations essentielles pour l'analyse tout en réduisant la dimensionnalité. L'extraction de caractéristiques peut inclure des techniques telles que la représentation vectorielle des mots (word embeddings) ou l'utilisation de méthodes comme TF-IDF (Term Frequency-Inverse Document Frequency) pour quantifier l'importance des mots dans un document .

Algorithmes ML

Dans l'analyse des sentiments, les algorithmes d'apprentissage automatique utilisent des modèles informatiques pour classer les données textuelles en sentiments positifs, négatifs ou neutres. Ces algorithmes reposent sur des techniques telles que le traitement du langage naturel et l'apprentissage supervisé pour fournir des informations automatisées sur le ton émotionnel du contenu textuel. Voici quelques techniques de ML couramment utilisées dans ce domaine :

Machine à Vecteurs de Support (SVM)

Le SVM (Support Vector Machine) est une méthode puissante d'apprentissage supervisé pour la régression et la classification. Il comprend deux principaux types : la régression des vecteurs de support (SVR) pour les tâches de régression et la classification des vecteurs de support (SVC) pour la classification. En ce qui concerne la classification, le SVC détermine l'hyperplan optimal pour les scénarios binaires et multi-classes, créant une limite linéaire ou non linéaire dans l'espace d'entrée. Les noyaux sont souvent utilisés avec les SVM pour définir la fonction de séparation. La formulation de SVM est donnée par :

$$\text{Minimize} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{Subject to } y_i(w \cdot x_i - b) \geq 1 - \xi_i \text{ for } i = 1, ..., n$$

$$\xi_i \geq 0 \text{ for } i = 1, ..., n$$

Dans ces équations :

- W représente les poids des caractéristiques,
- B est le terme de biais,
- C est le paramètre de régularisation,
- X(i) sont les vecteurs d'entraînement,
- y(i)sont les étiquettes de classe,
- ϵ(i) sont les variables d'écart,
- n est le nombre d'exemples d'entraînement.
-

Les SVM sont largement utilisés pour leur efficacité dans la classification de données textuelles dans l'analyse des sentiments.

Random forest

La forêt aléatoire est une méthode de machine learning supervisée utilisée pour la classification et la régression. Cette approche implique la création d'un ensemble complet d'arbres de décision et l'agrégation de leurs prédictions pour produire un résultat final. L'algorithme de forêt aléatoire intègre des concepts tels que les sous-espaces aléatoires et le "bagging", et a été formellement introduit par Breiman en 2001.

Dans cette méthode, plusieurs arbres de décision sont entraînés, chacun utilisant un échantillon légèrement différent des données. La forêt aléatoire est donc une collection d'algorithmes utilisant des arbres de décision comme prédicteurs individuels, tout en intégrant le bagging, la randomisation des sorties et le sous-espace aléatoire, y compris le Boosting. Reconnue comme une technique de classification très efficace, la forêt aléatoire peut catégoriser avec précision de grands ensembles de données.

Cette méthode d'apprentissage ensembliste génère plusieurs arbres de décision lors de la phase de formation et détermine le mode de classe en agrégeant les sorties des arbres individuels.

Naïve Bayes

Le classificateur Naïve Bayes est un exemple notable parmi les classificateurs de réseaux bayésiens, largement utilisés dans la classification supervisée . Ce modèle probabiliste, ancré dans le théorème de Bayes, repose sur l'hypothèse simplificatrice d'indépendance forte. Initialement développée pour la recherche de textes, cette méthode reste fondamentale dans la catégorisation de textes. Son objectif principal est d'attribuer des documents à des catégories en se basant sur la fréquence des mots. L'approche bayésienne de classification offre des algorithmes d'apprentissage pratiques et permet de combiner des connaissances antérieures avec des données observées. Dans Naïve Bayes, le concept fondamental consiste à calculer les probabilités de catégorie pour un document texte donné en évaluant les probabilités conjointes des mots et des catégories, en supposant l'indépendance des mots. Le théorème de Bayes sur la probabilité conditionnelle est à la base de ce concept, affirmant que, pour un point de données donné et une classe y :

$$P(y|x_1, x_2, ..., x_i) = \frac{P(y)P(x_1,x_2,...,x_i|y)}{P(x_1,x_2,...,x_i)}$$

De plus, en supposant que pour un point de données donné, x1 a xila probabilité d'occurrence de chaque attribut dans une classe spécifique est considérée comme indépendante. Ainsi, la probabilité de x1,x2,...,xix_1, x_2, ..., x_ix1,x2,...,xi peut être approximée comme suit :

$$P(x_1, x_2, ..., x_i|y) \approx P(x_1|y) \times P(x_2|y) \times ... \times P(x_i|y)$$

Cette formulation simplifie le calcul des probabilités conditionnelles en supposant que les attributs sont indépendants les uns des autres dans une classe spécifique.

Algorithmes DL

Le réseau neuronal récurrent (RNN)

Le réseau neuronal récurrent (RNN) est une architecture de réseau de neurones caractérisée par des connexions récurrentes entre les neurones, formant ainsi une structure en boucle. Cette conception permet aux RNN de traiter efficacement des données séquentielles en introduisant une notion de "mémoire" dans le réseau. Contrairement aux réseaux neuronaux standards, les RNN peuvent prendre en compte les éléments précédents d'une séquence lors du traitement de l'élément suivant. Cette capacité les rend particulièrement adaptés aux tâches impliquant des données séquentielles, comme le traitement de texte, la traduction automatique et la prédiction de séries temporelles.

Dans un RNN, chaque unité de temps dans une séquence d'entrée est associée à un pas de temps dans le réseau. A chaque pas de temps, le RNN reçoit une entrée xtx_txt et produit une sortie yty_tyt, tout en maintenant un état caché hth_tht qui capture les informations précédemment vues dans la séquence. Cet état caché hth_tht est calculé en fonction de l'entrée actuelle xtx_txt et de l'état caché précédent ht−1h_{t-1}ht−1. Cette rétroaction permet au RNN de conserver des informations sur les étapes précédentes de la séquence, ce qui lui donne une sorte de "mémoire" pour traiter efficacement les données séquentielles.

En résumé, le réseau neuronal récurrent (RNN) est une architecture de réseau de neurones capable de traiter des données séquentielles en utilisant une rétroaction récurrente pour maintenir et utiliser des informations sur les étapes précédentes de la séquence.

Conclusion

Ce chapitre explore l'état de l'art de l'analyse des sentiments, mettant en avant son rôle crucial dans divers secteurs. Des applications dans les soins de santé, les marchés financiers, le service client, le divertissement, l'éducation et le commerce électronique sont discutées. Les méthodes de prétraitement, telles que le nettoyage des données et la tokenisation, ainsi que les algorithmes ML et DL. Ces avancées promettent une amélioration continue de la précision et de la portée de l'analyse des sentiments, ouvrant de nouvelles opportunités pour son utilisation efficace dans des applications réelles.

Chapitre 3 : Réalisation du projet

I. Description Dataset :

Notre dataset comprend plusieurs variables clés qui permettent de mener une analyse de sentiment complète et contextualisée. Chaque enregistrement dans le dataset contient un texte de tweet complet, une partie sélectionnée du tweet représentant le sentiment, l'âge de l'utilisateur, le sentiment global du tweet, l'heure et la date du tweet, le pays d'origine de l'utilisateur, la population totale du pays et la superficie terrestre du pays.

Les variables incluses sont :

- text : Ce champ contient le texte complet du tweet. Il constitue la base de l'analyse de sentiment, permettant de déterminer le sentiment exprimé par l'utilisateur à travers ses mots.

- selected_text : Cette variable contient une partie spécifique du tweet que l'utilisateur a sélectionnée comme étant particulièrement représentative du sentiment. Cela peut aider à affiner l'analyse en se concentrant sur les mots ou phrases les plus significatifs.

- age of user : Cette colonne représente l'âge de l'utilisateur qui a posté le tweet. L'âge peut influencer les perspectives et les expressions de sentiment, et cette variable permet d'explorer ces variations.

- sentiment : Ce champ indique le sentiment global du tweet, qu'il soit positif, négatif ou neutre. C'est une variable cible essentielle pour l'analyse, permettant de catégoriser chaque tweet selon le sentiment exprimé.

- time of tweet : Cette variable enregistre l'heure et la date du tweet. Elle permet d'analyser les tendances temporelles, telles que les variations de sentiment au cours de la journée, de la semaine ou de l'année.

- Country : Ce champ indique le pays d'origine de l'utilisateur. Il est utile pour explorer les différences géographiques dans les sentiments et pour comprendre comment les contextes nationaux peuvent influencer les expressions de sentiment.

- population: Cette colonne représente la population totale du pays de l'utilisateur. Elle peut être utilisée pour contextualiser les données et pour effectuer des analyses démographiques.

- Land area : Ce champ indique la superficie totale du pays de l'utilisateur (en kilomètres carrés). Comme la population, cette variable peut fournir un contexte géographique utile pour l'analyse.

Ces variables offrent une richesse d'informations permettant de mener une analyse de sentiment détaillée. En combinant le texte des tweets avec des données démographiques, temporelles et géographiques, vous pouvez explorer comment les sentiments varient en fonction de divers facteurs et identifier des tendances ou des corrélations significatives.

II. Prétraitement :

1. Bibliothèques :

```
from pyspark.sql.functions import when
from pyspark.sql import SparkSession
import seaborn as sns
from pyspark.sql.functions import udf
import re
from string import punctuation
import string
import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from pyspark.sql.functions import regexp_replace, lower, col, concat_ws, split, array_except , lit ,collect_list
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
import numpy as np
```

Figure 1:Bibliothèques

- **PySpark Libraires :**
  - **pyspark.sql.functions :**
    - **when:** Utilisé pour les expressions conditionnelles dans les requêtes Spark SQL.
    - **regexp_replace:** Remplace les sous-chaînes qui correspondent à une expression régulière par une chaîne spécifiée.
    - **Lower :** Convertit le texte en minuscules.
    - **col :** Fait référence à une colonne dans un DataFrame.
    - **concat_ws:** Concatène plusieurs colonnes en une seule colonne avec un séparateur spécifié.
    - **Split :** Fractionne une colonne de chaînes en un tableau de chaînes en fonction d'un délimiteur spécifié.
    - **array_except:** Renvoie un tableau d'éléments du premier tableau qui ne sont pas présents dans le second tableau.
    - **Lit :** Crée une colonne de valeur littérale.
    - **collect_list:** Agrége les éléments dans une liste.

- **pyspark.sql.SparkSession:** Point d'entrée pour programmer Spark avec l'API Dataset et DataFrame.

- **pyspark.ml.feature:**
  - **HashingTF:** Met en correspondance une séquence de termes avec leurs fréquences en utilisant l'astuce du hachage.
  - **IDF:** Inverse Document Frequency (fréquence inverse des documents), utilisé pour mettre à l'échelle les fréquences des termes dans TF-IDF.
  - **Tokenizer:** Divise le texte en mots.
  - **StringIndexer:** Encode une colonne d'étiquettes en une colonne d'indices d'étiquettes.
- **pyspark.ml:**
  - **Pipeline:** Combine plusieurs étapes (transformateurs et estimateurs) en un seul pipeline pour faciliter la modélisation.
  - **classification.LogisticRegression:** Algorithme de régression logistique pour les tâches de classification.
  - **evaluation.MulticlassClassificationEvaluator:** Évalue les performances des modèles de classification à l'aide de mesures telles que la précision, le score F1, etc.

- **Python Libraries**
  - **seaborn:**
    Une bibliothèque de visualisation de données basée sur Matplotlib. Elle fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs.

  - **udf (User-Defined Functions):**
    Permet aux utilisateurs de définir des fonctions personnalisées en Python et de les utiliser dans les requêtes Spark SQL.

  - **re(Regular Expressions):**
    Un module pour travailler avec des expressions régulières, utile pour le traitement de texte et la recherche de motifs.

  - **string:**
    Un module contenant diverses opérations de chaîne et constantes comme la ponctuation.

  - **nltk (Natural Language Toolkit):** A library for natural language processing (NLP) nltk.corpus.stopwords : fournit une liste de mots vides courants dans diverses langues, qui sont souvent supprimés lors du traitement de texte.

  - **wordcloud:**

Un outil permettant de créer des nuages de mots à partir de données textuelles, où la taille de chaque mot indique sa fréquence ou son importance.

❖ **matplotlib.pyplot:**
Une bibliothèque de traçage utilisée pour créer des visualisations statiques, interactives et animées en Python.

2. Configuration Spark :

- **Création d'une session Spark nommée "NB" :**

La ligne de code `spark = SparkSession.builder.appName("NB").getOrCreate()` initialise une SparkSession dans PySpark, qui sert de point d'entrée pour interagir avec Apache Spark. La méthode `builder` configure la SparkSession, et `appName("NB")` attribue le nom "NB" à l'application, la rendant facilement identifiable dans l'interface web de Spark. La méthode `getOrCreate()` garantit que soit une SparkSession existante est récupérée, soit une nouvelle est créée si aucune n'existe, évitant ainsi la création de multiples sessions pouvant entraîner des conflits de ressources. Cette SparkSession vous permet d'effectuer une variété de tâches, telles que le chargement et le traitement de grands ensembles de données, l'exécution de requêtes SQL et la construction de modèles d'apprentissage automatique, le tout dans un cadre évolutif et optimisé.

```
spark =SparkSession.builder.appName("NB").getOrCreate()
```

Figure 2:Création Session Spark

- **Affichage des Paramètres de Configuration de Spark avec PySpark :**

Cette partie permet d'accéder et d'afficher la configuration actuelle de Spark. En accédant à `SparkContext` via la session Spark, nous pouvons récupérer la configuration complète de Spark avec `sc.getConf()`. Ensuite, en parcourant chaque paire clé-valeur de cette configuration avec une boucle `for`, nous affichons tous les paramètres de configuration. Cela permet de vérifier et de valider les paramètres en vigueur, ce qui est essentiel pour le débogage, l'optimisation des performances et la documentation de l'application. Ces informations sont cruciales pour s'assurer que Spark est configuré correctement et fonctionne de manière optimale.

```
# Accéder à l'objet SparkContext via la session
sc = spark.sparkContext

# Obtenir la configuration Spark
spark_conf = sc.getConf()

# Afficher la configuration Spark
for conf in spark_conf.getAll():
    print(conf)
```

```
('spark.app.id', 'local-1717495007150')
('spark.app.submitTime', '1717495003426')
('spark.driver.extraJavaOptions', '-Djava.net.preferIPv6Addresses=false -XX:+IgnoreU
dd-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.refl
s=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-op
l.concurrent=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.atomic=ALL-UNNAM
va.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.base/sun.nio.cs=ALL-UNNAMED --add-op
sun.util.calendar=ALL-UNNAMED --add-opens=java.security.jgss/sun.security.krb5=ALL-U
('spark.driver.port', '39973')
('spark.executor.id', 'driver')
('spark.driver.host', 'e89920f3f973')
('spark.sql.warehouse.dir', 'file:/content/spark-warehouse')
('spark.app.startTime', '1717495003768')
('spark.rdd.compress', 'True')
('spark.executor.extraJavaOptions', '-Djava.net.preferIPv6Addresses=false -XX:+Ignor
--add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.r
pens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add
il.concurrent=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.atomic=ALL-UNNA
ava.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.base/sun.nio.cs=ALL-UNNAMED --add-o
e/sun.util.calendar=ALL-UNNAMED --add-opens=java.security.jgss/sun.security.krb5=ALL
('spark.serializer.objectStreamReset', '100')
('spark.app.name', 'NB')
('spark.master', 'local[*]')
('spark.submit.pyFiles', '')
('spark.submit.deployMode', 'client')
('spark.ui.showConsoleProgress', 'true')
```

Figure 3:Configuration Spark

3. Chargement et visualisation des données :

```
##  DATA LOADING
df_train = spark.read.csv('train.csv', inferSchema=True, header=True ,encoding="ISO-8859-1")
df_test = spark.read.csv('test.csv', inferSchema=True, header=True ,encoding="ISO-8859-1")
```

Figure 5:Chargement de données

```
df_train.show()
```

train dataset

textID	text	selected_text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km²)	Density (P/Km²)
cb774db0d1	I`d have respond...	I`d have responde...	neutral	morning	0-20	Afghanistan	38928346	652860.0	60
549e992a42	Sooo SAD I will ...	Sooo SAD	negative	noon	21-30	Albania	2877797	27400.0	105
088c60f138	my boss is bullyi...	bullying me	negative	night	31-45	Algeria	43851044	2381740.0	18
9642c003ef	what interview! ...	leave me alone	negative	morning	46-60	Andorra	77265	470.0	164
358bd9e861	Sons of ****, wh...	Sons of ****,|	negative	noon	60-70	Angola	32866272	1246700.0	26
28b57f3990	http://www.dotheb...	http://www.dotheb...	neutral	night	70-100	Antigua and Barbuda	97929	440.0	223
6e0c6d75b1	2am feedings for ...	fun	positive	morning	0-20	Argentina	45195774	2736690.0	17
50e14c0bb8	Soooo high	Soooo high	neutral	noon	21-30	Armenia	2963243	28470.0	104
e050245fbd	Both of you	Both of you	neutral	night	31-45	Australia	25499884	7682300.0	3
fc2cbefa9d	Journey!? Wow.....	Wow... u just bec...	positive	morning	46-60	Austria	9006398	82400.0	109
2339a9b08b	as much as i lov...	as much as i love...	neutral	noon	60-70	Azerbaijan	10139177	82658.0	123
16fab9f95b	I really really l...	like	positive	night	70-100	Bahamas	393244	10010.0	39

Figure 4:Affichage données d'entrainement

```
print("test dataset")
df_test.show()
```

```
test dataset
+----------+--------------------+---------+-------------+-----------+-------------------+----------------+---------------+---------------+
|    textID|                text|sentiment|Time of Tweet|Age of User|            Country|Population -2020|Land Area (Km²)|Density (P/Km²)|
+----------+--------------------+---------+-------------+-----------+-------------------+----------------+---------------+---------------+
|f87dea47db|Last session of t...|  neutral|      morning|       0-20|        Afghanistan|        38928346|       652860.0|             60|
|96d74cb729| Shanghai is also...| positive|         noon|      21-30|            Albania|         2877797|        27400.0|            105|
|eee518ae67|Recession hit Ver...| negative|        night|      31-45|            Algeria|        43851044|      2381740.0|             18|
|01082688c6|         happy bday!| positive|      morning|      46-60|            Andorra|           77265|         470.0|            164|
|33987a8ee5| http://twitpic.c...| positive|         noon|      60-70|             Angola|        32866272|      1246700.0|             26|
|726e501993| that`s great!! w...| positive|        night|     70-100|Antigua and Barbuda|           97929|         440.0|            223|
|261932614e|I THINK EVERYONE ...| negative|      morning|       0-20|          Argentina|        45195774|      2736690.0|             17|
|afa11da83f| soooooo wish i c...| negative|         noon|      21-30|            Armenia|         2963243|        28470.0|            104|
|e64208b4ef| and within a sho...|  neutral|        night|      31-45|          Australia|        25499884|      7682300.0|              3|
|37bcad24ca| What did you get...|  neutral|      morning|      46-60|            Austria|         9006398|        82400.0|            109|
|24c92644a4|My bike was put o...| negative|         noon|      60-70|         Azerbaijan|        10139177|        82658.0|            123|
|43b390b336| I checked.  We d...|  neutral|        night|     70-100|            Bahamas|          393244|        10010.0|             39|
|69d6b5d93e| .. and you`re on...|  neutral|      morning|       0-20|            Bahrain|         1701575|          760.0|           2239|
|5c1e0b61a1|I`m in VA for the...| negative|         noon|      21-30|         Bangladesh|       164689383|       130170.0|           1265|
|504e45d9d9|Its coming out th...| negative|        night|      31-45|           Barbados|          287375|          430.0|            668|
|ae93ad52a0|So hot today =_= ...| negative|      morning|      46-60|            Belarus|         9449323|       202910.0|             47|
|9fce30159a|            Miss you| negative|         noon|      60-70|            Belgium|        11589623|        30280.0|            383|
|00d5195223|        Cramps . . .| negative|        night|     70-100|             Belize|          397628|        22810.0|             17|
|33f19050cf| you guys didn`t ...| positive|      morning|       0-20|              Benin|        12123200|       112760.0|            108|
|f7718b3c23|I`m going into a ...|  neutral|         noon|      21-30|             Bhutan|          771608|        38100.0|             20|
+----------+--------------------+---------+-------------+-----------+-------------------+----------------+---------------+---------------+
```

Figure 6:Affichage données de test

Nettoyage et Préparation des données

En utilisant PySpark, nous avons récupéré avec succès les deux jeux de données "train.csv" et "test.csv". Nous allons maintenant appliquer un traitement approfondi pour préparer nos jeux de données pour le stockage dans MongoDB et la création de modèles pour l'analyse des sentiments. Ce processus de préparation des données inclura le nettoyage des données, la gestion des valeurs manquantes, la transformation des variables textuelles, et la normalisation des données. Une fois les données préparées, elles seront stockées dans MongoDB pour un accès rapide et efficace. Ensuite, nous utiliserons ces données pour entraîner et évaluer des modèles d'apprentissage automatique afin de réaliser une analyse de sentiments précise et fiable. Grâce à cette approche, nous pourrons tirer des insights précieux et comprendre les tendances et sentiments exprimés dans les tweets de manière plus approfondie.

- Supprimer les valeurs nulles trouvée dans ‘train.csv’ et ‘test.csv’ en utilisant la fonction dropna() ,ensuite en affichant le nombre final des colonnes pour chaque Dataset en utilisant la fonction count():

```
# Supprimer les nulls
df_train= df_train.dropna()
# Afficher le nombre de lignes dans l'ensemble d'entraînement
df_train_count = df_train.count()
print("Nombre de lignes dans l'ensemble d'entraînement:", df_train_count)

# Supprimer les nulls
df_test = df_test.dropna()
# Afficher le nombre de lignes dans l'ensemble test
df_test_count = df_test.count()
print("Nombre de lignes dans l'ensemble d'entraînement:", df_test_count)
```

```
Nombre de lignes dans l'ensemble d'entraînement: 27478
Nombre de lignes dans l'ensemble d'entraînement: 3534
```

- Sélectionner uniquement les deux colonnes 'text' et 'sentiment' dans le jeu d'entraînement et de test, puis fusionner les deux en utilisant la fonction union() et afficher notre nouveau Data Frame df.

```
df_train=df_train['text', 'sentiment']
df_test=df_test['text', 'sentiment']
df = df_train.union(df_test)
df.show()
```

```
+--------------------+---------+
|                text|sentiment|
+--------------------+---------+
| I`d have respond...|  neutral|
| Sooo SAD I will ...| negative|
|my boss is bullyi...| negative|
| what interview! ...| negative|
| Sons of ****, wh...| negative|
|http://www.dotheb...|  neutral|
|2am feedings for ...| positive|
|          Soooo high|  neutral|
|         Both of you|  neutral|
| Journey!? Wow.....| positive|
| as much as i lov...|  neutral|
|I really really l...| positive|
|My Sharpie is run...| negative|
|i want to go to m...| negative|
|test test from th...|  neutral|
|Uh oh, I am sunbu...| negative|
| S`ok, trying to ...| negative|
|i`ve been sick fo...| negative|
|is back home now ...| negative|
|Hes just not that...|  neutral|
+--------------------+---------+
```

- Visualiser le nombre de sentiments distincts dans notre DataFrame

```
#visualiser les sentiment
df.groupBy('sentiment').count().show()
```

```
+---------+-----+
|sentiment|count|
+---------+-----+
| positive| 9685|
|  neutral|12547|
| negative| 8780|
+---------+-----+
```

- Nous encodons les trois types de sentiments en convertissant la colonne ‘sentiment’ en valeurs entières.

```
## Encoder les sentiments
# Définir le dictionnaire de remplacement
replacement_dict = {"positive": 1, "neutral": 0, "negative": 2}

# Remplacer les valeurs dans la colonne "sentiment" en fonction du dictionnaire de remplacement
df = df.withColumn("sentiment",
                   when(df["sentiment"] == "positive", replacement_dict["positive"])
                   .when(df["sentiment"] == "neutral", replacement_dict["neutral"])
                   .when(df["sentiment"] == "negative", replacement_dict["negative"])
                   .otherwise(df["sentiment"]))
#visualiser les sentiment
# Convertir la colonne "sentiment" en type entier
df = df.withColumn("sentiment", col("sentiment").cast("int"))

df.groupBy('sentiment').count().show()
```

```
+---------+-----+
|sentiment|count|
+---------+-----+
|        1| 9685|
|        2| 8780|
|        0|12547|
+---------+-----+
```

- Visualiser le décompte des trois types de sentiments en utilisant seaborn et la fonction countplot() après avoir converti le DataFrame Spark en DataFrame Pandas.

```
# Conversion du DataFrame Spark en DataFrame Pandas
df_pandas=df.toPandas()
sns.countplot(x='sentiment',data=df_pandas)
```

```
<Axes: xlabel='sentiment', ylabel='count'>
```

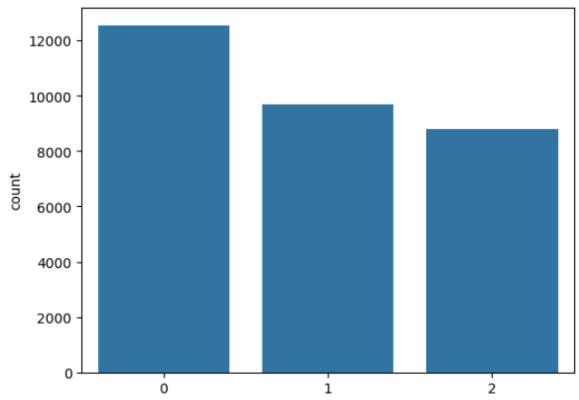

Figure 7:Visualisation du nombre de sentiments

- Échantillonnage pour équilibrer les classes de sentiments :

```
# Séparer les classes majoritaires et minoritaires
df_majority = df.filter(col("sentiment") == 0)
df_minority = df.filter(col("sentiment") == 2)
df_medium = df.filter(col("sentiment") == 1)

# Déterminer la taille de l'échantillon de la classe minoritaire
minority_count = df_minority.count()

# Échantillonnage aléatoire des données de la classe majoritaire pour correspondre à la taille de la classe minoritaire
df_majority_downsampled = df_majority.sample(False, minority_count / df_majority.count(), seed=1234)
df_medium_downsampled = df_medium.sample(False, minority_count / df_medium.count(), seed=1234)

# Combinez les classes majoritaires échantillonnées et les classes minoritaires
df = df_majority_downsampled.union(df_minority).union(df_medium_downsampled)
```

- Visualiser les données après l'échantillonnage : les 3 classes sont bien équilibrées

```
#visualiser les sentiment
df.groupBy('sentiment').count().show()
df_pandas=df.toPandas()
sns.countplot(x='sentiment',data=df_pandas)
```

```
+---------+-----+
|sentiment|count|
+---------+-----+
|        0| 8797|
|        2| 8780|
|        1| 8809|
+---------+-----+
```

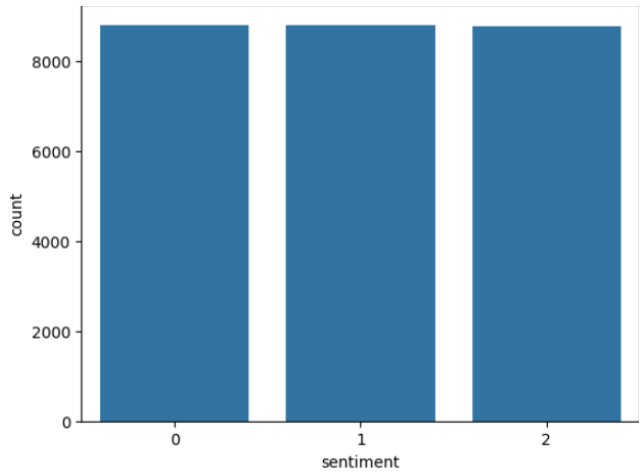

Figure 8:Visualisation du nombre des sentiments après Echantillonnage

- Appliquer des transformations sur la colonne text à savoir : Convertir le texte en minuscules, supprimer les caractères non alphanumériques,supprimer les ponctuations….

```
# Appliquer les transformations sur la colonne de texte
## Convertir le texte en minuscules
df = df.withColumn('processed_text', lower(df['text']))
# Supprimer les crochets et leur contenu
df = df.withColumn('processed_text', regexp_replace(df['processed_text'], '\\[.*?\\]', ''))

# Supprimer les caractères non alphanumériques
df = df.withColumn('processed_text', regexp_replace(df['processed_text'], '\\W', ' '))
### Supprimer les URLs
df = df.withColumn('processed_text', regexp_replace(df['processed_text'], 'https?://\\S+|www\\.\\S+', ''))
## Supprimer les balises HTML
df = df.withColumn('processed_text', regexp_replace(df['processed_text'], '<.*?>+', ''))
## Supprimer les ponctuations
df = df.withColumn('processed_text', regexp_replace(df['processed_text'], '[%s]' % re.escape(string.punctuation), ''))
## Supprimer les sauts de ligne
df = df.withColumn('processed_text', regexp_replace(df['processed_text'], '\\n', ''))
#
# Supprimer les chiffres et les mots contenant des chiffres
df = df.withColumn('processed_text', regexp_replace(df['processed_text'], '\\w*\\d\\w*', ''))
```

- Supprimer les stopwords sur la nouvelle colonne ‘processed_text’ en utilsant ntlk

```
# Télécharger la liste des stopwords si nécessaire
nltk.download('stopwords')
# Charger la liste des stopwords en français par exemple
stop_words = set(stopwords.words('english'))
## delete stop words
# Diviser le texte en mots
df = df.withColumn('words', split(df['processed_text'], ' '))
# Charger la liste des stopwords en tant qu'ensemble pour une recherche efficace
stop_words_set = set(stop_words)
# Convertir l'ensemble de stopwords en liste
stop_words_list = list(stop_words_set)
# Créer une colonne contenant la liste des mots
stop_words_col = lit(stop_words_list)
# Filtrer les mots qui ne sont pas des stopwords
df = df.withColumn('processed_words', array_except(col('words'), stop_words_col))Z
# Joindre les mots filtrés pour former le texte final
df = df.withColumn('processed_text', concat_ws(' ', col('processed_words')))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

- Afficher la colonne ‘processed_text’

```
# Afficher uniquement la colonne 'processed_text'
df.select('processed_text').show(truncate=False)
```

```
+--------------------------------------------------------------------------------------------------
|processed_text
+--------------------------------------------------------------------------------------------------
| go cruches next weeks
|coffee time  back later kisses
| heeeey dear finally found
| last night
|back uminaa
|grandpa telling used cut human bodies med school
|nuggets game everyoneeeee except
| ya never watch l
| theres thing
|rap battling second sat  moviesss haha swim lake sex matthew ghosts girlfriends past
|adventures jamie bethhh
| somehow cant reply message lol yes know thank youuu
| live germany costs lot wish could follow anyway
| sad wish something big kids may god bless always
| drive teeny tiny honda civic http myloc fry
|looking forward joy  hoping another disaterous race ferrari button gets well deserved win
|want followers
|uh  could come visit austin make gigantic moving decision closer st louis portland
| get although religious thing think let one slip
| thankies btw part scotland live never got visit uk lived europe
+--------------------------------------------------------------------------------------------------
only showing top 20 rows
```

- Agréger les textes en fonction du sentiment et en affichant le Word Cloud.

```
# Agréger les textes en fonction du sentiment
aggregated_df = df_train.groupby('sentiment').agg(concat_ws(' ', collect_list('text')).alias('text'))

# Convertir l'aggregated_df en Pandas DataFrame pour utiliser WordCloud
aggregated_pd_df = aggregated_df.toPandas()

# Créer un nuage de mots pour chaque sentiment
for index, row in aggregated_pd_df.iterrows():
    sentiment = row['sentiment']
    text = row['text']
    wordcloud = WordCloud(width=800, height=400, background_color='black').generate(text)

    # Afficher le nuage de mots
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(f'Word Cloud for Sentiment: {sentiment}')
    plt.axis('off')
    plt.show()
```

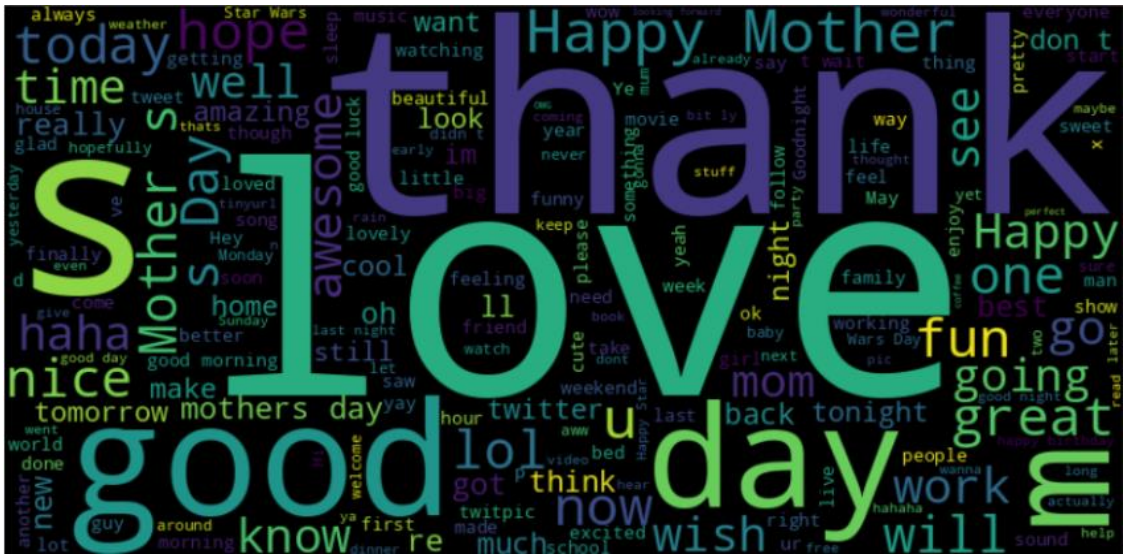

Figure 9: Word Cloud Positif

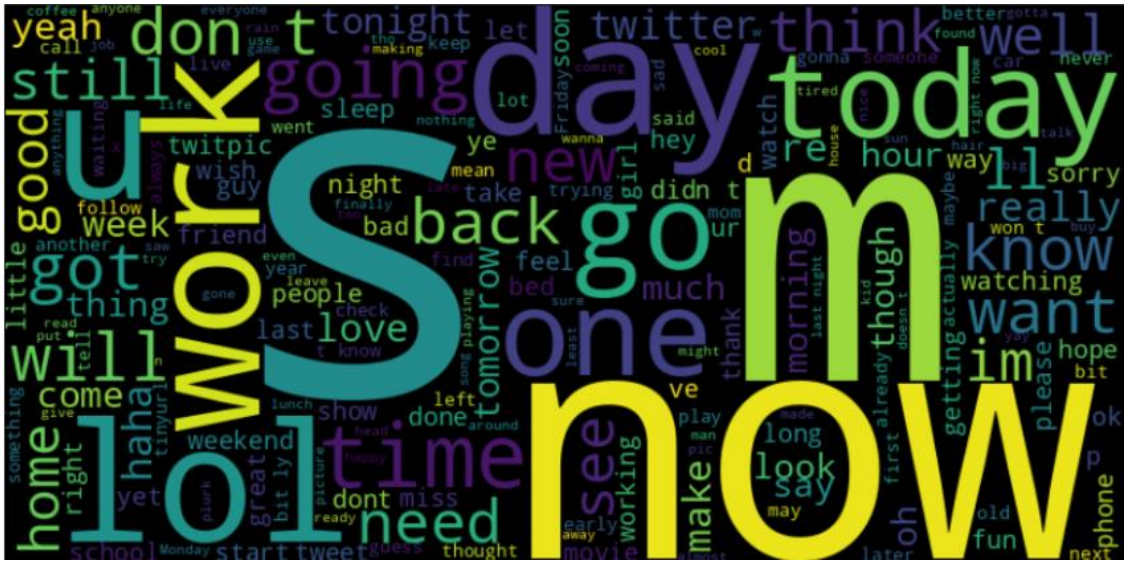

Figure 10: Word Cloud Neutre

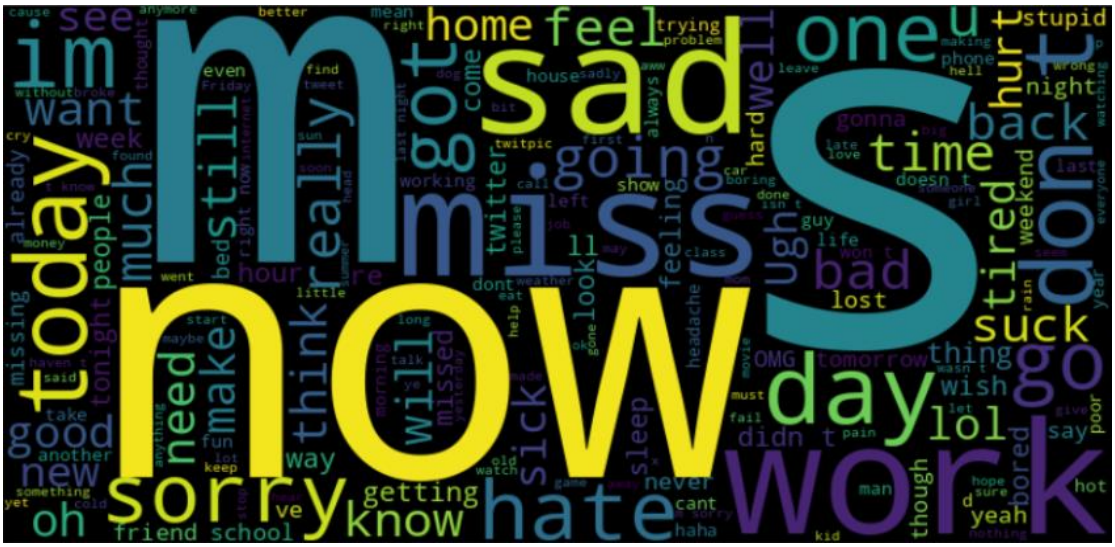

Figure 11:Word Cloud Négatif

Chapitre 4 : Visualisation Power BI

Après avoir établi la connexion entre MongoDB et Power BI en utilisant le pilote ODBC pour visualiser notre jeu de données stocké dans MongoDB, cette intégration permet de transférer les données de MongoDB vers Power BI, où nous pouvons les analyser et créer des visualisations interactives et informatives. Grâce à cette configuration, nous pouvons exploiter la puissance de Power BI pour explorer les données, identifier des tendances à partir des informations stockées dans MongoDB. Notre jeu de données est riche et contient des informations majeures sur les tweets, le nombre de tweets, les sentiments, et les pays d'origine, ce qui en fait une ressource précieuse pour des analyses approfondies.

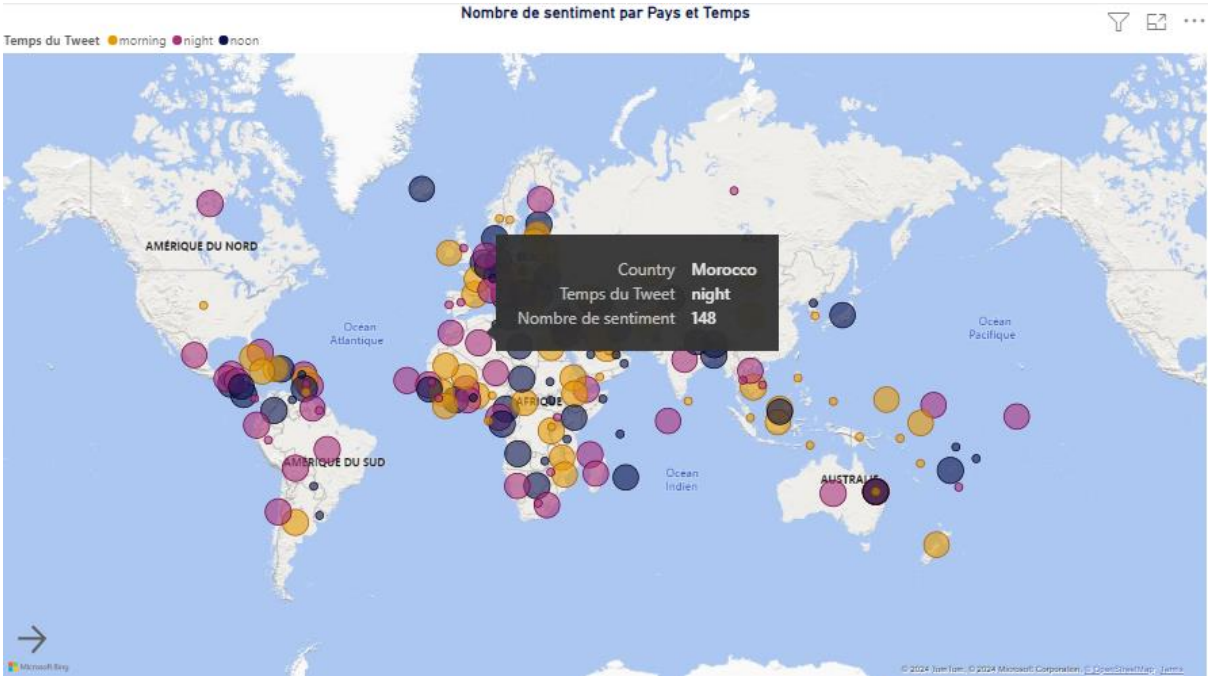

Figure 12:Nombre de sentiment par pays et temps

Cette carte interactive offre une perspective globale et détaillée de l'analyse de sentiment des tweets à travers le monde et au fil du temps. En utilisant les données extraites de Twitter, cette visualisation permet de cartographier les variations du sentiment exprimé dans différents pays à différentes périodes.

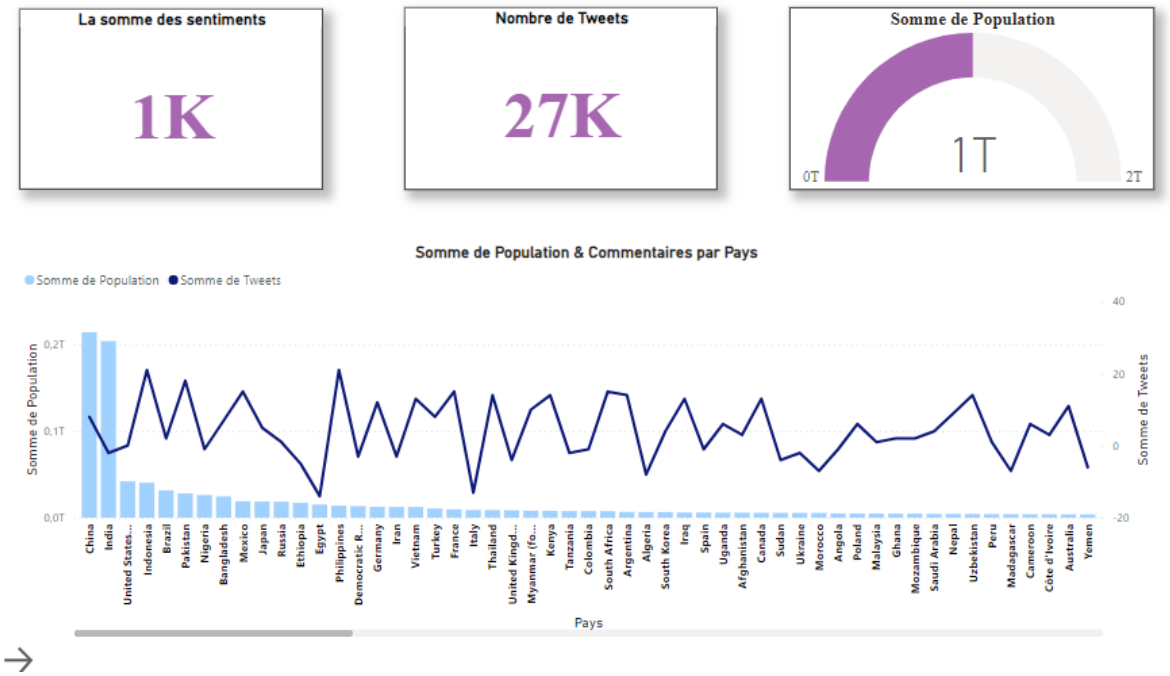

Figure 13:Statistiques et tableau de bord en Courbe et Histogramme

- Les trois cartes présentent des statistiques majeures concernant notre jeu de données :
  - La somme des sentiment :
  - Le nombre total de tweets : plus de 27 000 commentaires dans notre jeu de données.
  - La somme de la population dans notre jeu de données dépasse un trillion de personnes.
- Le tableau de bord, avec ses courbes et histogrammes groupés, offre une vue claire sur la somme de la population dans chaque pays ainsi que sur le nombre de tweets correspondants à chaque pays.

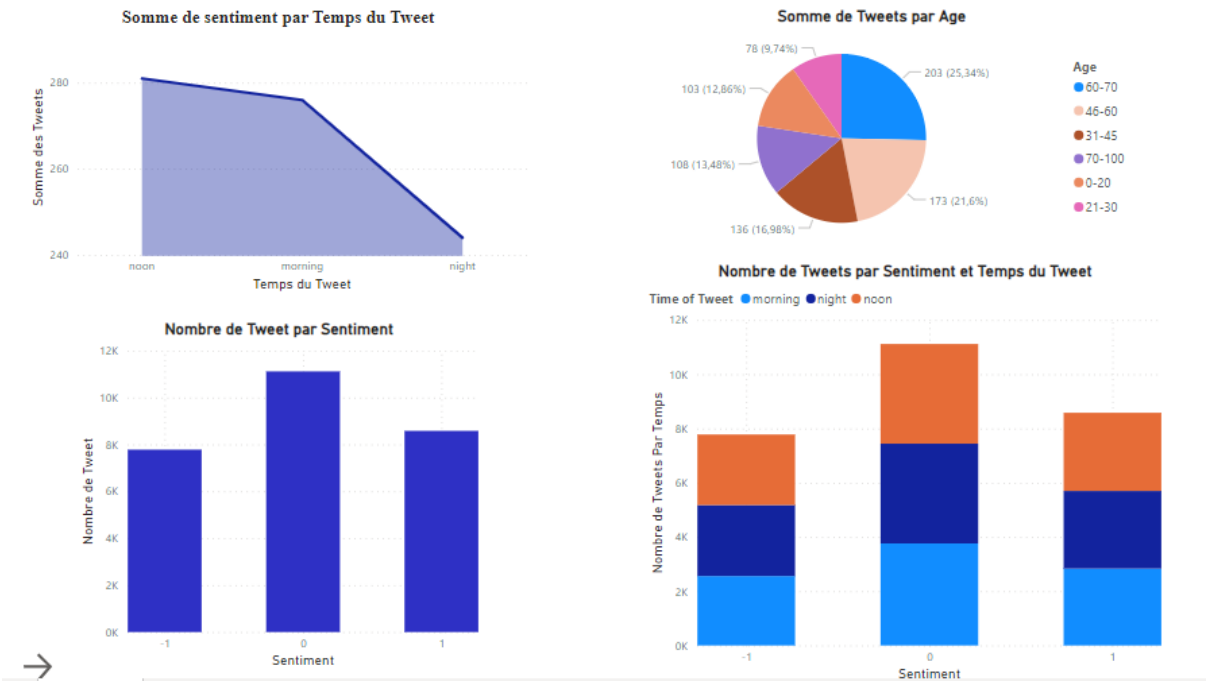

Figure 14:Tableaux de bord en aire ,en secteurs et Histogramme groupé

Ces graphiques visualisent diverses statistiques concernant les tweets classés par sentiment, temps de tweet, et âge des utilisateurs.

- **Somme de Tweets par Temps**

Ce graphique de zone montre la somme de sentiment des tweets au cours de la journée, divisée en trois périodes : noon (midi), morning (matin) et night (nuit). On observe que la somme des sentiments diminue progressivement de midi à la nuit.

- **Somme de Tweets par Age**

Ce diagramme circulaire représente la répartition des tweets par groupe d'âge des utilisateurs :
La tranche d'âge la plus active en tweeter c’est celle entre 60 et 70 ans alors que celle entre 21 et 30 ans et la moins active
Nombre de Tweet par Sentiment
Ce graphique à barres affiche le nombre de tweets pour chaque type de sentiment :

- Sentiment négatif (-1)
- Sentiment neutre (0)
- Sentiment positif (1)

On constate que le nombre de tweets neutres est légèrement plus élevé que celui des tweets positifs et négatifs.

- **Nombre de Tweets par Sentiment et Temps du Tweet**

Ce graphique empilé montre le nombre de tweets pour chaque sentiment (négatif, neutre, positif), en fonction du temps de tweet (morning, night, noon). Chaque segment de la barre représente le nombre de tweets pour une période spécifique. On remarque que :

- Les tweets neutres (sentiment 0) sont les plus fréquents, en particulier pendant la matinée (morning).
- Les tweets négatifs (-1) et positifs (1) sont plus équilibrés entre les différentes périodes de la journée, mais les tweets positifs sont légèrement plus fréquents le matin.

Chapitre 5 : Entrainement des modèles et comparaison

Les Modèles

- **Fonction permettant de Tracer la matrice de confusion**

```
from pyspark.mllib.evaluation import MulticlassMetrics
def plot_confusion_matrix(confusion_matrix, classes,
                          normalize=True,
                          title=None,
                          cmap=plt.cm.Blues):
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    if normalize:
        confusion_matrix = confusion_matrix.astype('float') / confusion_matrix.sum(axis=1)[:, np.newaxis]
    plt.imshow(confusion_matrix, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f'
    thresh = confusion_matrix.max() / 2.
    for i in range(confusion_matrix.shape[0]):
        for j in range(confusion_matrix.shape[1]):
            plt.text(j, i, format(confusion_matrix[i, j], fmt),
                     ha="center", va="center",
                     color="white" if confusion_matrix[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

Figure 12: Matrice de Confusion

- **Pipeline pour la transformation des données textuelles en vecteurs TF-IDF et division des données**

```
# Créer un indexeur de chaînes pour la colonne 'sentiment'
string_indexer = StringIndexer(inputCol="sentiment", outputCol="label")

# Tokenizer pour diviser le texte en mots
tokenizer = Tokenizer(inputCol="processed_text", outputCol="word")

# Créer un pipeline pour indexer la chaîne, tokenizer, et convertir les mots en vecteurs TF-IDF
hashing_tf = HashingTF(inputCol="word", outputCol="raw_features", numFeatures=10000)
idf = IDF(inputCol="raw_features", outputCol="features")
pipeline = Pipeline(stages=[string_indexer, tokenizer, hashing_tf, idf])

# Adapter le pipeline sur les données
pipeline_model = pipeline.fit(df)

# Transformer les données
transformed_df = pipeline_model.transform(df)

# Diviser les données en ensembles de formation et de test
train_data, test_data = transformed_df.randomSplit([0.9, 0.1], seed=42)
```

Figure 13: Pipeline

Le pipeline crée un index des sentiments, tokenise le texte, et génère des vecteurs TF-IDF. Ensuite, il s'adapte aux données pour les transformer en un ensemble de caractéristiques. Enfin, les données sont divisées en ensembles d'entraînement et de test pour la modélisation.

a. Modèle 1 : Régression logistique

Pour la régression logistique nous avons initialisé notre modèle en précisant les colonnes de caractéristiques (`featuresCol`), la colonne cible (`labelCol`) et le nombre maximal d'itérations ensuite il est entraîné sur les données d'entraînement et utilisé pour faire les prédictions sur les données de test .
Vers la fin il est évalué à l'aide de MutlticlassClassificationEvaluator qui calcule l'exactitude (`accuracy`) du modèle sur les données d'entraînement et de test.
Pour les données nous avons trouvé comme accuracy 0.92 or pour le test nous avons trouvé 0.60

```
# Créer le modèle de régression logistique
lr = LogisticRegression(featuresCol='features', labelCol='label', maxIter=10)

# Entraîner le modèle sur les données d'entraînement
lr_model = lr.fit(train_data)

# Faire des prédictions sur les données de test
predictions_train = lr_model.transform(train_data)
predictions_test = lr_model.transform(test_data)

# Évaluer la performance du modèle
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy_train = evaluator.evaluate(predictions_train)
accuracy_test = evaluator.evaluate(predictions_test)
print("Régression linéaire :")
print("accuracy_train:", accuracy_train)
print("accuracy_test:", accuracy_test)

predictions_train.show()
```

```
Régression linéaire :
accuracy_train: 0.9182627688172043
accuracy_test: 0.595810705973623
```

Figure 14: Régression Logistique

A partir des prédictions du modèle nous avons généré la matrice de confusion afin de les comparer avec les étiquettes réelles.

```
predictionTest_label_rdd = predictions_test.select("prediction", "label").rdd.map(tuple)
predictionTrain_label_rdd = predictions_train.select("prediction", "label").rdd.map(tuple)

# Initialiser MulticlassMetrics avec la RDD
metricsTest = MulticlassMetrics(predictionTest_label_rdd)
metricsTrain = MulticlassMetrics(predictionTrain_label_rdd)

# Obtenir la matrice de confusion
confusion_matrixTest = metricsTest.confusionMatrix().toArray()
confusion_matrixTrain = metricsTrain.confusionMatrix().toArray()

# Définir les classes
classes = [str(i) for i in range(confusion_matrixTest.shape[0])]

# Afficher la matrice de confusion
plot_confusion_matrix(confusion_matrixTrain, classes=classes, title='Confusion Matrix train')
plt.show()
plot_confusion_matrix(confusion_matrixTest, classes=classes, title='Confusion Matrix test')
plt.show()

# Stockage des résultats
results={}
results['Regression Logistique'] = {"Accuracy": accuracy_test, "Confusion Matrix": confusion_matrixTest}
```

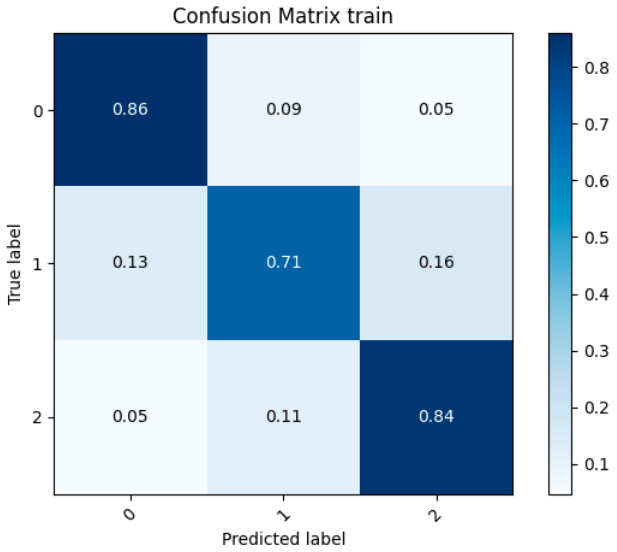

Figure 16:Matrice de confusion_train(RL)

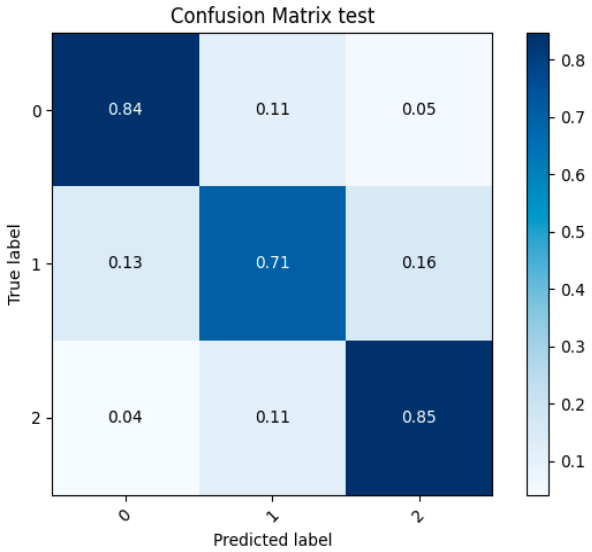

Figure 15:Matrice de confusion_Test(RL)

b. Modèle 2 : SVM

Pour SVM nous avon crée le modèle SVM en indiquant le nombre maximal d'itérations et le paramètre de régularisation , puis nous avons Instancier le modèle OneVsRest avec SVM comme classifieur de base et nous avons entraine notre modèle pour faire la prédiction sur l'ensemble d'entraînement et de test. Par la suite, nous avons évalué la précision de notre modèle a l'aide de la fonction MulticlassClassificationEvaluator().

```python
from pyspark.ml.classification import OneVsRest
from pyspark.ml.classification import LinearSVC

# Création du modèle SVM
svm = LinearSVC(maxIter=10, regParam=0.1)

# Instanciation du modèle OneVsRest avec SVM comme classifieur de base
ovr = OneVsRest(classifier=svm)

# Entraînement du modèle
model = ovr.fit(train_data)

predictions_train = model.transform(train_data)
# Prédiction sur l'ensemble de test
predictions = model.transform(test_data)

model.write().overwrite().save("model")

# Évaluation de la précision du modèle
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy_train = evaluator.evaluate(predictions_train)
accuracy = evaluator.evaluate(predictions)
print("SVM :")
print("acuuracy train :" ,accuracy_train)
print("Accuracy test:", accuracy)
predictions_train.show()
```

Figure 20:SVM

```
SVM :
acuuracy train : 0.8025033602150538
Accuracy test: 0.7975174553917765
```

Pour générer la matrice de confusion nous avons effectué une conversion des prédictions du modèle en Pandas DataFrame pour une manipulation plus facile, puis nous avons affiché nos matrices pour les comparer avec les étiquettes réelles.

```python
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import pandas as pd
# Prédiction sur l'ensemble d'entraînement
predictions_train = model.transform(train_data)
# Prédiction sur l'ensemble de test
predictions_test = model.transform(test_data)

# Convertir les prédictions en Pandas DataFrame pour une manipulation plus facile
predictions_train_df = predictions_train.select("label", "prediction").toPandas()
predictions_test_df = predictions_test.select("label", "prediction").toPandas()

# Matrice de confusion pour l'ensemble d'entraînement
cm_train = confusion_matrix(predictions_train_df['label'], predictions_train_df['prediction'])
# Matrice de confusion pour l'ensemble de test
cm_test = confusion_matrix(predictions_test_df['label'], predictions_test_df['prediction'])

# Affichage des matrices de confusion
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
sns.heatmap(cm_train, annot=True, cmap='Blues', fmt='d', ax=axes[0])
axes[0].set_title('Training Confusion Matrix')
axes[0].set_xlabel('Predicted Label')
axes[0].set_ylabel('True Label')
sns.heatmap(cm_test, annot=True, cmap='Blues', fmt='d', ax=axes[1])
axes[1].set_title('Testing Confusion Matrix')
axes[1].set_xlabel('Predicted Label')
axes[1].set_ylabel('True Label')
results['SVM'] = {"Accuracy": accuracy, "Confusion Matrix": confusion_matrixTest}
```

Voici les deux matrices de confusion pour les données de test et celles d'entraînement en utilisant SVM

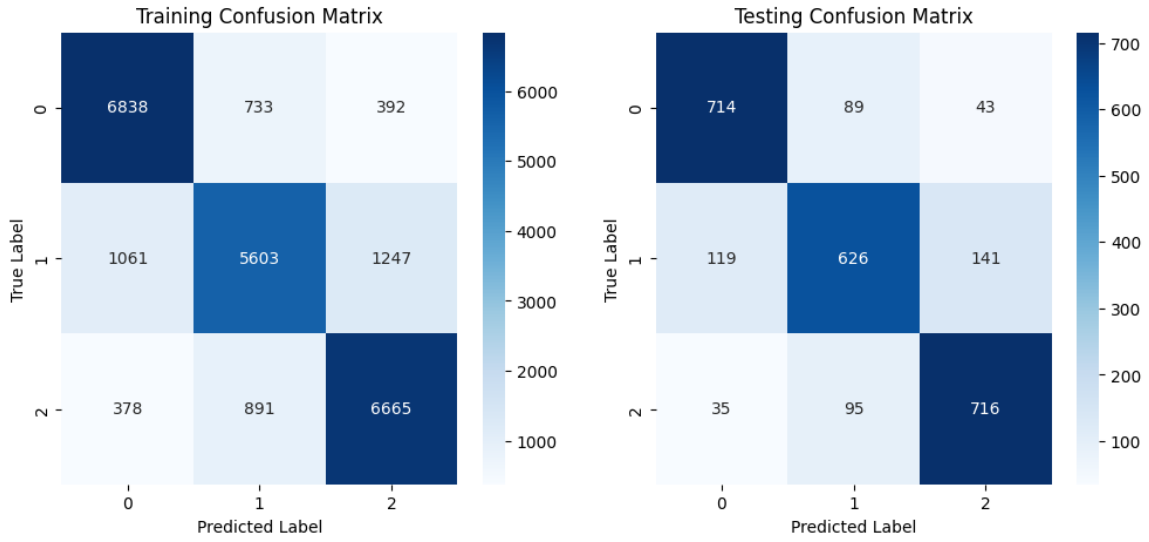

Figure 21: Matrices de confusion pour SVM

c. Modèle 3 : Naive Bayes:

On a choisi le modèle Naive Byes, un algorithme de classification basé sur le théorème de Bayes avec une hypothèse d'indépendance naïve, en utilisant Spark MLlib pour entraîner et évaluer ce modèle de classification sur les données d'entraînement et de test, puis nous avons les résultats d'exactitude et les prédictions.

```
from pyspark.ml.classification import NaiveBayes

# Créer le modèle de Naive Bayes
lr = NaiveBayes(smoothing=1)
# Entraîner le modèle sur les données d'entraînement
lr_model = lr.fit(train_data)

# Faire des prédictions sur les données de test
predictions_train = lr_model.transform(train_data)
predictions_test = lr_model.transform(test_data)

# Évaluer la performance du modèle
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy_train = evaluator.evaluate(predictions_train)
accuracy_test = evaluator.evaluate(predictions_test)
print("Naive Bayes :")
print("accuracy_train:", accuracy_train)
print("accuracy_test:", accuracy_test)

predictions_train.show()
```

Figure 22: Naive Byes

Résultat :

```
Naive Bayes :
accuracy_train: 0.7810819892473119
accuracy_test: 0.5713731574864236
```

Matrices de confusion du Naive Bayes :

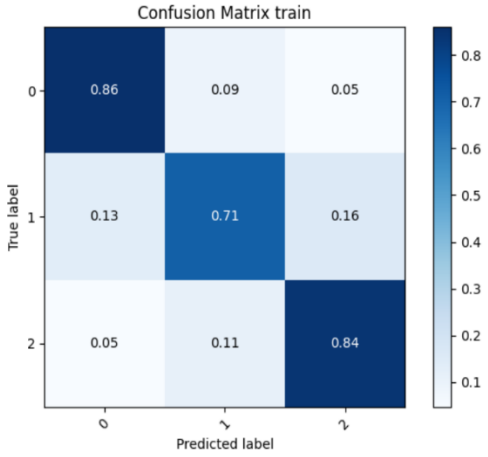

Figure 18:Matrice confusion_Test(NB)

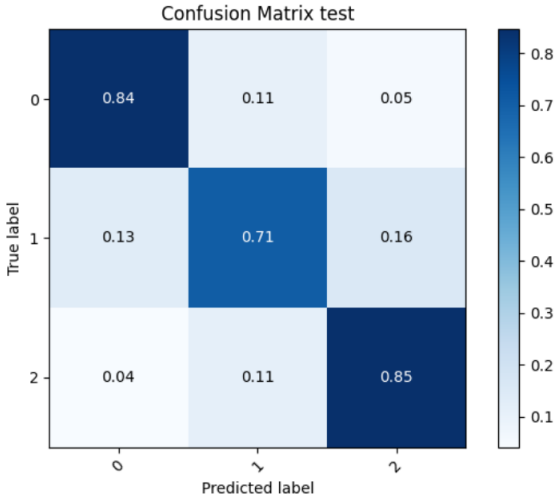

Figure 17: Matrice de confusion_Test(NB)

d. Modèle 4: Random Forest

Nous avons choisi d'utiliser le modèle 'Random Forest' comme un algorithme de classification permet d'ordonner les features plus ou moins discriminantes (les termes les plus fréquents).

Après avoir importé RandomForestClassifier de la bibliothèque pyspark.ml.classification, nous avons entrainé le modèle sur notre jeu données d'entraînement 'train_data' en utilisant la fonction fit(),et faire les prédictions sur les données d'entrainement ainsi que de test .

Utiliser MutlticlassClassificationEvaluator pour évaluer les prédictions trouvées en affichant les deux décisions.

```python
from pyspark.ml.classification import RandomForestClassifier
# Créer le modèle de régression logistique
rf= RandomForestClassifier(featuresCol='features', labelCol='label',numTrees = 100, maxDepth = 4, maxBins = 32)

# Entraîner le modèle sur les données d'entraînement
rf_model = rf.fit(train_data)

# Faire des prédictions sur les données de test
predictions_train = rf_model.transform(train_data)
predictions_test = rf_model.transform(test_data)

# Évaluer la performance du modèle
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy_train = evaluator.evaluate(predictions_train)
accuracy_test = evaluator.evaluate(predictions_test)
print("Random Forest :")
print("accuracy_train:", accuracy_train)
print("accuracy_test:", accuracy_test)

predictions_train.show()
```

```
Random Forest :
accuracy_train: 0.6521757392473119
accuracy_test: 0.6276183087664856
```

Figure 19:Random Forest

Voici les deux matrices de confusion pour les données de test et celles d'entraînement en utilisant Random Forest:

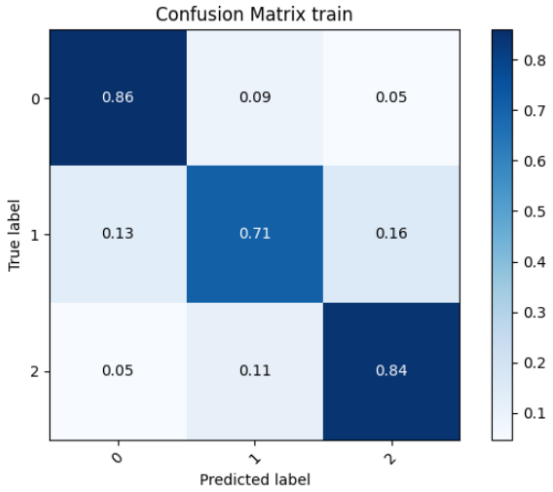

Figure26 :Matrice de confusion_Train(RF)

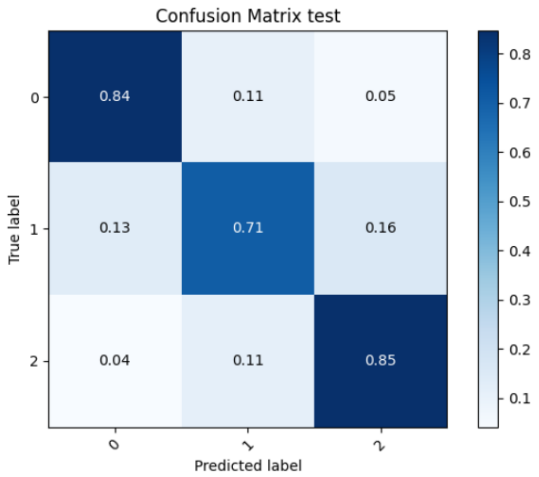

Figure 27:Matrice de confusion_Test(RF)

Comparaison et test

```
for name, result in results.items():
    print(f"Modèle: {name}")
    print(f"Accuracy: {result['Accuracy']}")
    print("Matrice de confusion:")
    print(result['Confusion Matrix'])
    print("\n")
```

- **Comparaison**

```
Modèle: Regression Logistique
Accuracy:
0.595810705973623
Matrice de confusion:
[[544. 201. 101.]
[180. 461. 245.]
[96. 219. 531.]]

Modèle: SVM
Accuracy: 0.7975174553917765
Matrice de confusion:
[[714.  89.  43.]
 [119. 626. 141.]
 [ 35.  95. 716.]]

Modèle: Naive Bayes
Accuracy: 0.5713731574864236
Matrice de confusion:
[[537. 194. 115.]
[242. 392. 252.]
[91. 211. 544.]]

Modèle: Random Forest
Accuracy: 0.6097750193948798
Matrice de confusion:
[[522. 250. 74.]
[146. 578. 162.]
[ 81. 293. 472.]]
```

Figure 20:Comparaison entre les modèles

On constate que SVM affiche la meilleure performance parmi tous les modèles testés. Il a la plus haute précision et montre une gestion efficace des confusions entre les classes.Pour random forest,il est en dessous du SVM, mais il reste supérieur aux modèles de régression logistique et Naive Bayes. Il réduit considérablement les confusions, surtout entre les classes 1 et 2.

- ## Test

Ce script prend une nouvelle phrase, la transforme en utilisant le pipeline déjà défini , et utilise le modèle de classification pour prédire le sentiment de la phrase (positif, neutre ou négatif).

```
# Nouvelle phrase à prédire
new_phrase = "I feel sad"

# Créer un DataFrame Spark contenant la nouvelle phrase
new_df = spark.createDataFrame([(new_phrase,)], ["processed_text"])

# Appliquer le pipeline sur le nouveau DataFrame
new_processed_df = pipeline_model.transform(new_df)

# Utiliser le modèle entraîné pour faire des prédictions sur la nouvelle phrase
predictions = model.transform(new_processed_df)

# Interpréter les prédictions
predicted_label = predictions.select("prediction").collect()[0][0]

if predicted_label == 1.0:
    print("Positive user comment")
elif predicted_label == 0:
    print(" Neutral user comment")
else:
    print("Negative user comment")
```

```
Negative user comment
```

Figure 21:Test

Chapitre 6 : Application avec Flask

Dans ce chapitre, nous avons mis en place une interface avec Flask pour classifier les tweets en positifs, négatifs et neutres en utilisant un modèle de Naive Bayes. Le principe de ce code est de charger et préparer les données de tweets. Après avoir traité les valeurs manquantes et encodé les sentiments en valeurs numériques, les données sont divisées en ensembles d'entraînement et de test. Un `TfidfVectorizer` est utilisé pour transformer les textes en vecteurs de caractéristiques, et un classificateur Naive Bayes Multinomial est entraîné sur ces vecteurs. Le modèle et le vectoriseur sont ensuite sauvegardés.

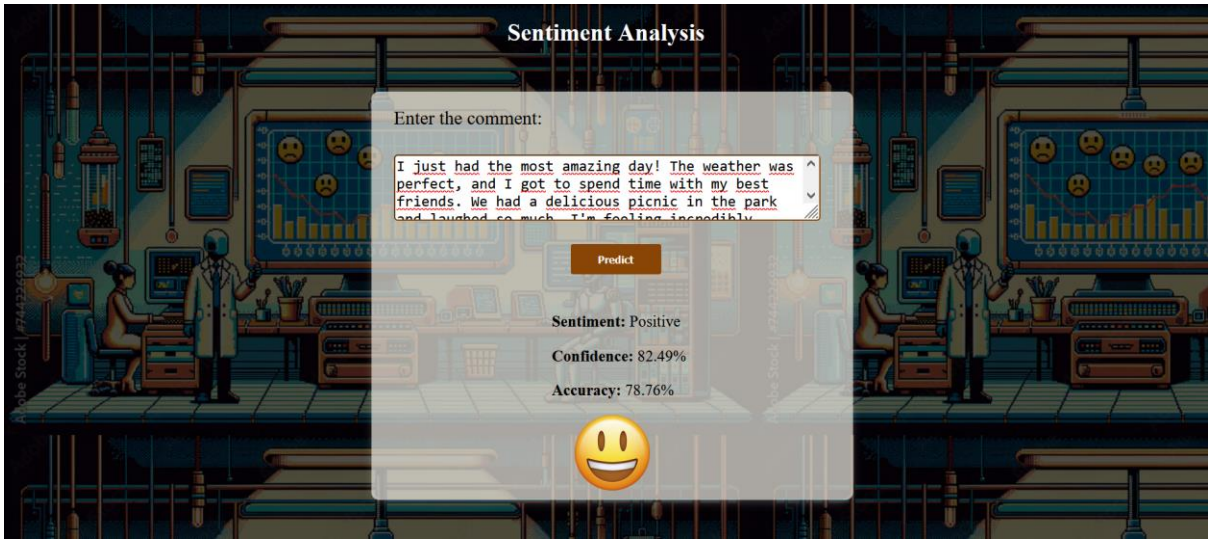

Figure 22:Cas Positif

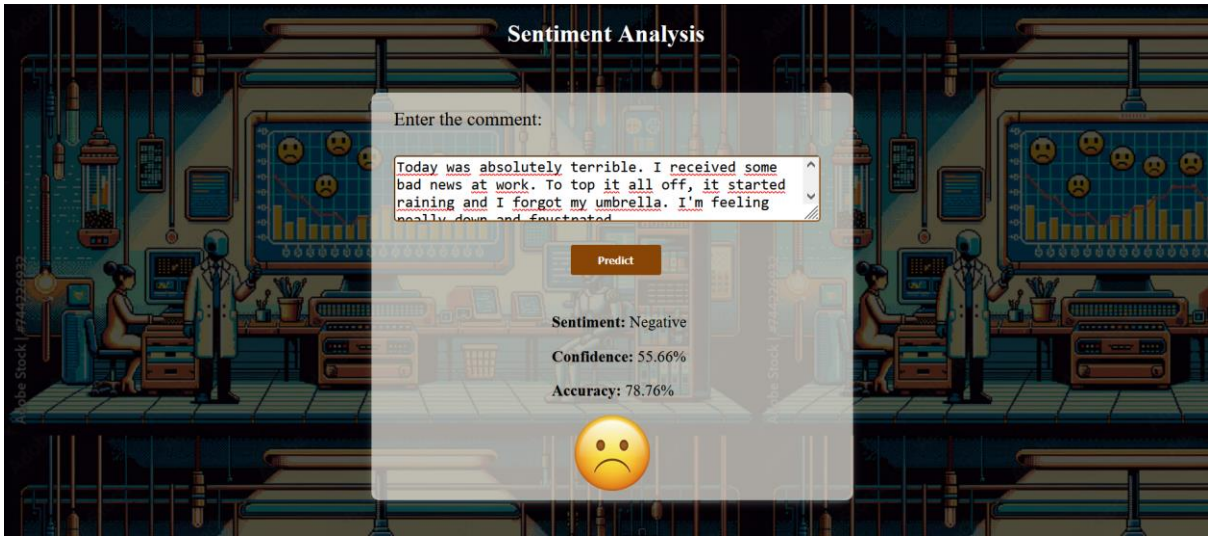

Figure 30 :Cas Négatif

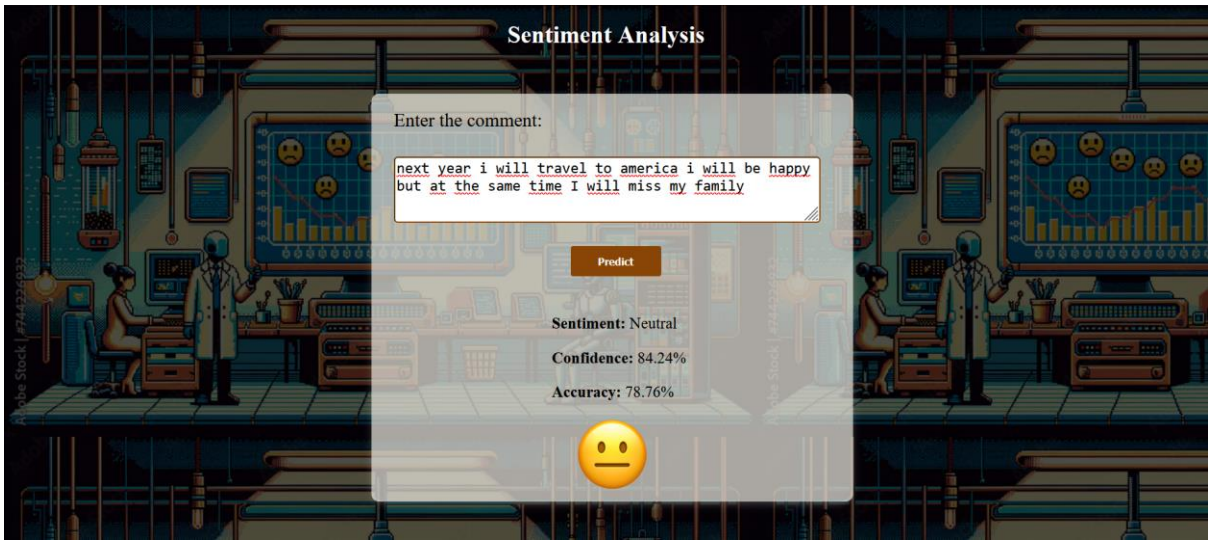

Figure 31:Cas Neutre

Dans la fonction `predict` de l'application Flask, le processus commence par la récupération du texte saisi par l'utilisateur via un formulaire HTML. Ce texte est ensuite vectorisé en utilisant le `TfidfVectorizer` précédemment sauvegardé, transformant ainsi le texte en vecteurs de caractéristiques numériques.

Le modèle Naive Bayes Multinomial, préalablement entraîné, est alors utilisé pour prédire le sentiment de ce texte. La prédiction, exprimée sous forme de valeur numérique, est traduite en un label de sentiment ("Neutral", "Positive", "Negative") et associée à un emoji correspondant.

Parallèlement, la confiance de la prédiction est calculée, et la précision globale du modèle est évaluée en le testant sur un ensemble de données de test.

Enfin, les résultats, incluant le sentiment prédit, l'emoji, la précision globale et la confiance de la prédiction, sont structurés en un dictionnaire et renvoytés au client sous forme de réponse JSON. Cette approche permet une interaction dynamique e en temps réel avec l'utilisateur, offrant une classification des tweets précise et visuellement intuitive.

Conclusion

Dans le cadre de ce projet, nous avons développé quatre modèles de classification pour l'analyse de sentiments sur Twitter, en utilisant les algorithmes suivants : Régression Logistique, SVM (Support Vector Machine), Naive Bayes, et Random Forest. Après une évaluation comparative basée sur des critères de performance tels que la précision, le modèle SVM s'est révélé être le plus performant.

Fort de cette constatation, nous avons ensuite déployé une application web conviviale en utilisant le framework Flask. Cette application fournit une interface utilisateur simple et intuitive où les utilisateurs peuvent saisir un commentaire. En temps réel, l'application analyse le sentiment du commentaire saisi et indique si le sentiment est positif, négatif ou neutre. Cette fonctionnalité permet aux utilisateurs d'obtenir instantanément des feedbacks sur les émotions véhiculées par leurs messages.

Parallèlement, nous avons intégré Power BI pour créer des visualisations interactives et dynamiques des données et des caractéristiques des sentiments extraits des tweets. Ces visualisations incluent des graphiques, des tableaux de bord et d'autres représentations visuelles, facilitant ainsi l'interprétation des tendances et des insights obtenus. Les données sont stockées dans une base de données MongoDB, ce qui permet une gestion efficace et évolutive des informations. Power BI permet de transformer ces données brutes en insights exploitables, offrant une compréhension claire et détaillée des sentiments exprimés sur Twitter.

En somme, ce projet combine des modèles de machine learning, une interface utilisateur fluide et des outils de visualisation avancés pour offrir une solution complète d'analyse de sentiments sur les réseaux sociaux.

Références

- https://www.sciencedirect.com/science/article/pii/S2949719124000074#b50