

PROJET DE FIN D'ETUDE

5éme Année en Ingénierie Informatique et Réseaux

Classification et Analyse des Sentiments sur les Marchés Publics au Maroc

Réalisé par :

Sara BEN CHIKH

Tuteur (s) :

Encadrant Professionnel :Mr. Souhayl MACHACH

Encadrant Pédagogique :Mr. Abdelali ZAKRANI

Au sein de Trésorerie Générale du Royaume :



الخزينة العامة للمملكة
• ٥٣٠ ٢١٨٩٠ ٣٤٦٩١٧ ٦٤٨٦٨٦
TRESORERIE GENERALE DU ROYAUME

Membres de jury : **Mr. Abdelali ZAKRANI**
Mr. Abderrahmane Daif
Mr. Said NOUH

Année universitaire : **2023/2024**

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

*Au nom de Dieu le Tout Miséricordieux,
le Très Miséricordieux*

Dédicaces

À Dieu tout puissant mon créateur

À mes très chers, aimables et honorables **parents**

Aucune dédicace ne saurait être assez éloquente pour exprimer mon respect,
mon amour éternel et ma considération pour vos sacrifices,
votre encouragement et votre confiance.

Je vous dois ce que je suis aujourd’hui et ce que je serai demain, et je ferai
toujours de mon mieux pour rester votre fierté.

Je vous aime.

À mes encadrants, académique et professionnel

Dont le soutien et les conseils ont été inestimables tout au long de ce parcours.

À mon encadrant académique, pour sa disponibilité, son expertise et ses conseils avisés. Votre
patience et votre rigueur ont grandement contribué à la réussite de ce projet. Je vous suis
profondément reconnaissant pour votre encadrement et votre bienveillance.

À mon encadrant professionnel, pour ses suggestions constructives et son accompagnement.
Votre enthousiasme et votre encouragement ont été une source de motivation constante.

À mes frères et ma sœur

pour leur aide précieuse et leur compréhension tout au long de cette aventure. Leur présence a
été une source de motivation inestimable.

À mes amis

pour leur soutien moral et leurs conseils judicieux. Leur amitié et leur encouragement ont été
essentiels pour surmonter les moments difficiles.

À tous les Emsistes

Merci pour les bons moments que nous avons passés ensemble.

À ceux qui croient en moi,

À ceux qui m’aiment,

À ceux que j’aime,

Je vous dédie ce travail.

Sara BEN CHIKH

Remerciement

Avant tout développement sur cette expérience, il apparaît nécessaire de commencer mon mémoire par des sincères remerciements. Pour cela, Je tiens à remercier **Dieu** le tout puissant d'avoir dessiné ce parcours pour moi, et toute personne qui a contribué à la réussite de ce travail.

Je saisie cette occasion pour exprimer ma plus profonde reconnaissance à **ma famille** et mes **amis** qui m'ont soutenu et motivé à réaliser ce mémoire de la meilleure façon possible.

J'adresse mes sincères remerciements et je témoigne toute ma reconnaissance à mon encadrant académique **Monsieur Abdelali ZAKRANI**, pour son soutien, sa disponibilité, ses précieuses remarques, ses conseils et recommandations qui n'ont cessé de me prodiguer durant toute la période du stage. Je vous présente Monsieur, ma profonde gratitude pour vos efforts.

J'exprime également ma grande gratitude à mon tuteur professionnel **Monsieur Souhayl MACHACH**, avec qui j'ai eu la chance de travailler, pour son accueil, sa confiance, son soutien, et sa disponibilité tout au long de la période de stage. Merci d'avoir accepté de diriger ce travail et de m'encadrer malgré votre charge de travail très serrée.

Enfin, mes sincères remerciements et respects aux **membres du Jury, Mr Daif, Mr NOUH et Mr ZAKRANI**, qui ont fait l'honneur d'évaluer ce travail, et d'assister à ma soutenance. Je tiens aussi à remercier le **corps professoral de l'EMSI** d'avoir assuré une formation de haute qualité.

Je remercie toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

Résumé

Le présent rapport est la synthèse du travail accompli dans le cadre de mon projet de fin d'études au sein de l'établissement public La Trésorerie Générale du Royaume, pour l'obtention du diplôme d'ingénieur d'état en informatique et réseaux à l'Ecole Marocaine des Sciences de l'Ingénieur EMSI, Casablanca.

Notre travail a pour objet de concevoir une application basé sur les techniques de la machine learning permettant d'analyser des emails et de donner une recommandation en s'inspirant des techniques de classification automatique de données textuelles. Dans un premier temps nous proposons d'identifier les emails qui expriment un avis ou représentant un bruit. Par la suite, pour les vrais avis, nous proposons une évaluation. Enfin, après calculer la moyenne des évaluations des avis pour savoir les sentiments des clients. Nous avons effectué une étude expérimentale sur différents algorithmes de classification avant de faire notre choix. Ainsi, les résultats expérimentaux obtenus pour notre solution sont satisfaisant.

Mots-clés : analyser des emails, sentiments des clients, algorithmes de classification, traitement du langage naturel.

Abstract

This report is a synthesis of the work accomplished as part of my final year project at the public institution, The General Treasury of the Kingdom, for the attainment of the state engineer diploma in computer science and networks from the Moroccan School of Engineering Sciences (EMSI), Casablanca.

Our project aims to design an application that analyzes emails and provides recommendations by leveraging automatic text classification techniques. Initially, we propose to identify emails that express an opinion versus those that represent noise. Subsequently, for genuine opinions, we propose an evaluation. Finally, after calculating the average evaluations of the opinions, we determine the customers' sentiments. We conducted an experimental study on different classification algorithms before making our choice. The experimental results obtained for our solution are satisfactory.

Keywords: analyzing emails, customer sentiment, classification algorithms, natural language processing

ملخص

هذا التقرير هو تلخيص للعمل المنجز كجزء من مشروع تخرجي في المؤسسة العامة، الخزينة العامة للمملكة، ، الدار (EMSI) للحصول على دبلوم مهندس دولة في علوم الكمبيوتر والشبكات من المدرسة المغربية لعلوم الهندسة البيضاء.

يهدف مشروعنا إلى تصميم تطبيق يحل الرسائل الإلكترونية ويوفر توصيات باستخدام تقنيات التصنيف النصي التلقائي. في البداية، نقترح تحديد الرسائل الإلكترونية التي تعبّر عن رأي مقابل ذلك الذي تمثل موضوعاً. بعد ذلك، بالنسبة للآراء الحقيقية، نقترح تقسيماً. وأخيراً، بعد حساب متوسط التقييمات للأراء، نحدد مشاعر العملاء. أجرينا دراسة تجريبية على خوارزميات تصنيف مختلفة قبل اتخاذ قرارنا. النتائج التجريبية التي تم الحصول عليها لحلنا كانت مرضية

الكلمات المفتاحية: تحليل الرسائل الإلكترونية، مشاعر العملاء، خوارزميات التصنيف، معالجة اللغة الطبيعية

Liste des abréviations

TGR : La trésorerie Générale du Royaume

AO : Appel d'Offres Ouvert

AR : Appel d'Offres Restreint

CR : Consultation Restreinte

MN : Marché Négocié

GG : Marché de Gré à Gré

AOP : Appel d'Offres avec Pré-qualification

NLP : Natural Language Processing

TALN : Traitement Automatique du Langage Naturel

TF-IDF : Term Frequency - Inverse Document Frequency (Fréquence Terme - Fréquence Inverse de Document)

SVM : Support Vector Machine (Machine à Vecteurs de Support)

F1-Score : Mesure de la performance d'un modèle de classification (moyenne harmonique entre précision et rappel)

NEG: Négatif (pour les sentiments négatifs)

NEU: Positif (pour les sentiments positifs)

POS: Positif (pour les sentiments positifs)

Liste des figures

| | |
|--|----|
| Figure 1: Siège de la TGR ----- | 17 |
| Figure 2: Organigramme de la TGR ----- | 22 |
| Figure 3: Organigramme de direction des ressources et du système d'information ----- | 22 |
| Figure 4: Site web des marchés publics ----- | 25 |
| Figure 5: Contact service marchés publics ----- | 26 |
| Figure 6: Processus d'une réclamation ----- | 26 |
| Figure 7 : Les niveaux d'analyse des sentiments ----- | 34 |
| Figure 8: régression logistique ----- | 41 |
| Figure 9 : SVM ----- | 42 |
| Figure 10 : Forêts Aléatoires----- | 43 |
| Figure 11: logo de GoogleColaboratory ----- | 46 |
| Figure 12: logo de Vs Code ----- | 47 |
| Figure 13: logo de Python ----- | 47 |
| Figure 14: logo de Excel ----- | 48 |
| Figure 15: logo de Facebook----- | 48 |
| Figure 16 : logo de Gmail ----- | 49 |
| Figure 17 : packages ----- | 51 |
| Figure 18 : bibliothèques utilisées ----- | 52 |
| Figure 19 : ressources nécessaires de NTLK----- | 53 |
| Figure 20 : sécurité de bibliothèques ----- | 54 |
| Figure 21 : importation de données----- | 54 |
| Figure 22 : dataset partie 1 ----- | 55 |
| Figure 23 : dataset partie 2 ----- | 55 |
| Figure 24 : dataset après importation----- | 56 |
| Figure 25 : suppression des colonnes non significatif ----- | 56 |
| Figure 26 : compression en majuscule, suppression des chiffres et de ponctuation ----- | 56 |
| Figure 27 : code pour ne garder que la langue française----- | 57 |
| Figure 28 : bouton de téléchargement de fichier ----- | 57 |
| Figure 29 : la réussite de téléchargement ----- | 57 |
| Figure 30 : résultat de suppression des autres langues ----- | 58 |
| Figure 31 : suppression des caractères spéciaux ----- | 58 |
| Figure 32 : suppression de quelques occurrences ----- | 58 |
| Figure 33 : suppression de stopwords ----- | 59 |
| Figure 34 : suppression des URLs ----- | 59 |
| Figure 35 : suppression de balises HTML ----- | 59 |
| Figure 36 : suppression de mots spécifiques ----- | 60 |
| Figure 37 : fréquence des mots ----- | 60 |
| Figure 38 : suppression des mots moins fréquents ----- | 61 |
| Figure 39 : suppression de lignes vides ----- | 61 |
| Figure 40 : correction orthographique ----- | 62 |
| Figure 41: suppression des lignes vides ----- | 62 |
| Figure 42: longueur des emails ----- | 63 |
| Figure 43 : graphe de longueur des emails----- | 63 |
| Figure 44 ; graphe des mots les plus fréquents ----- | 64 |
| Figure 45 : graphe des mots moins fréquents----- | 65 |
| Figure 46 : suppression des lignes contenant des valeurs manquantes ----- | 66 |
| Figure 47: lemmatisation----- | 66 |
| Figure 48 : nuage de mots----- | 67 |
| Figure 49 : code pour l'étiquetage des mots ----- | 68 |
| Figure 50 : résultat d'étiquetage ----- | 68 |
| Figure 51 : calcul de polarité ----- | 69 |
| Figure 52 : attribution des labels de sentiments ----- | 69 |

| | |
|---|-----------|
| Figure 53 : encodage des étiquettes de sentiments ----- | 69 |
| Figure 54 : séparation de données ----- | 70 |
| Figure 55 : vectorisation en Count Vectors ----- | 70 |
| Figure 56 : résultat de Count Vectors ----- | 71 |
| Figure 57 : vectorisation en tfidf vectorizer----- | 71 |
| Figure 58: résultat de Tfidfvectorizer ----- | 72 |
| Figure 59 : TF-IDF-vectorizer-ngram ----- | 72 |
| Figure 60 : résultat de TF-IDF-vectorizer-ngram ----- | 73 |
| Figure 61 : TF-IDF-WordLevel ----- | 73 |
| Figure 62: résultat TF-IDF-WordLevel----- | 74 |
| Figure 63 : application des modèles d'apprentissage----- | 77 |
| Figure 64 : Les deux modèles offrent une précision parfaite ----- | 77 |
| Figure 65 : Le modèle de régression logistique montre une excellente précision globale (98%) ----- | 78 |
| Figure 66: matrices des modèles----- | 78 |
| Figure 67: comparaison de la performance de modèles----- | 79 |
| Figure 68: optimisation de Random Forest ----- | 80 |
| Figure 69: résultatat de l'optimisation ----- | 80 |
| Figure 70: graphe de comparaison des accuracy des modèles----- | 81 |
| Figure 71 : graphe de répartition de sentiments dans les emails----- | 82 |
| Figure 72 : comptage des occurrences de sentiments----- | 83 |
| Figure 73: diagramme de répartition des sentiments----- | 83 |
| Figure 74: sauvegarde de dataframe----- | 83 |
| Figure 75: chargement de dataframe----- | 84 |
| Figure 76 : page d'accueil ----- | 84 |
| Figure 77 : chargement de fichier pkl----- | 85 |
| Figure 78 : répartition de sentiments----- | 86 |
| Figure 79 : code source de l'interface utilisateur----- | 88 |

Liste des tableaux

| | |
|--|-----------|
| Tableau 1 Fiche signalétique | 17 |
| Tableau 2 : comparaison de techniques de vectorisation..... | 38 |

Table des matières

| | |
|---|-----------|
| Dédicaces..... | 3 |
| Remerciement..... | 4 |
| Résumé | 5 |
| Abstract | 6 |
| Liste des abréviations | 8 |
| Liste des tableaux | 11 |
| Table des matières | 12 |
| Introduction générale..... | 15 |
| Chapitre I : Contexte Général du Projet | 16 |
| 1. Présentation de l'organisme d'accueil | 17 |
| 1.1 Identification de la TGR | 17 |
| 1.2 Fiche technique | 17 |
| 1.3 Histoire..... | 18 |
| 1.4 Missions de la trésorerie générale du royaume..... | 20 |
| 1.4.1 Le recouvrement des créances publiques | 20 |
| 1.4.2 Le contrôle et le paiement des dépenses publiques | 20 |
| 1.4.3 La gestion des finances locales..... | 20 |
| 1.4.4 La gestion des dépôts au Trésor | 21 |
| 1.4.5 La production de l'information financière et comptable..... | 21 |
| 1.5 Valeurs | 21 |
| 1.6 Organisation de la TGR | 21 |
| 1.7 Services proposés et modes de passation..... | 23 |
| 2 Contexte du projet | 24 |
| 2.1 Contexte général du projet | 24 |
| 2.2. Problématique : | 26 |
| 2.3 Besoins fonctionnels | 27 |
| 2.4 Besoins non-fonctionnels | 28 |
| 2 Connaissances sur le métier | 29 |

| | |
|---|-----------|
| 3.1. Métiers courants TGR marocaine | 29 |
| 3.2 Les marchés publics | 30 |
| 3.2.1 Définition des marchés publics | 30 |
| 3.2.2 Le principe des marchés publics | 30 |
| 3.2.3 Les types des marchés publics | 30 |
| 3.3. Services et modes de passation | 31 |
| Conclusion : | 32 |
| Chapitre II : Etat de l'Art & Techniques de Machine Learning | 33 |
| 1 Analyse de sentiments | 34 |
| 1.1 Définition | 34 |
| 1.2 Domaines d'application de l'analyse des sentiments | 34 |
| 1. Traitement Automatique du Langage Naturel (NLP) | 36 |
| 2.1. Définition | 36 |
| 2.2 Objectifs et Enjeux du NLP | 36 |
| 2.3 Applications du NLP | 36 |
| 3 Vectorisation du Texte | 37 |
| 3.1 TF-IDF Word Level | 37 |
| 3.3 TF-IDF Vectorizer | 38 |
| 3.4 Count Vectors (CountVectorizer) | 38 |
| 3.5 Comparaison des techniques de vectorisation | 38 |
| 4 Apprentissage automatique | 39 |
| 4.1. Définition | 39 |
| 4.2. Principe du Machine Learning | 39 |
| 4.3 Types d'Apprentissage | 40 |
| 4.4 Modèles d'apprentissage automatique | 40 |
| 4.4.1. Régression Logistique | 40 |
| 4.4.2. Support Vector Machines (SVM) | 42 |
| 5 Evaluation des Modèles | 44 |
| Chapitre III : Outils et Environnement | 45 |
| Introduction : | 46 |
| 1 Outils de développement | 46 |
| 1.1 Google Colaboratory | 46 |

| | | |
|--|--|-----------|
| 1.2 | Visual Studio Code (VS Code) | 46 |
| 1.3 | Python | 47 |
| 2 | Outils de gestion des données | 47 |
| 2.1 | Excel..... | 47 |
| 2.2 | Esuit..... | 48 |
| 3 | Plateformes de collecte de données..... | 48 |
| 3.1 | Facebook | 48 |
| 3.2 | Gmail..... | 48 |
| Chapitre IV : Collecte et Prétraitement | | 50 |
| 1 | Collecte des Données | 51 |
| 1.1. | Source de données | 51 |
| 1.2. | Description de la dataset : | 51 |
| 2 | Prétraitement de données | 51 |
| 2.1 | Préparation des Données..... | 51 |
| 2.2. | Nettoyage de données | 56 |
| 2.3 | Visualisation..... | 62 |
| 3 | Etiquetage de mots | 67 |
| 4 | Analyse de Sentiments | 68 |
| 5 | Séparation de données | 69 |
| 6 | Vectorisation des Textes | 70 |
| | Conclusion..... | 74 |
| Chapitre V : Réalisation et Résultats | | 75 |
| | Introduction : | 76 |
| 1 | Entraînement des Modèles d'Apprentissage Automatique | 76 |
| 2 | Optimisation des Hyperparamètres avec GridSearchCV | 79 |
| 3 | Affichage des Résultats | 80 |
| 4 | Interface graphique pour l'utilisateur..... | 84 |
| | Conclusion..... | 89 |
| | Webographie | 90 |

Introduction générale

Ce rapport s'inscrit dans le cadre de mon projet de fin d'études, dont l'objectif principal est la conception et le développement d'une solution complète d'analyse des sentiments appliquée aux échanges électroniques dans le cadre des marchés publics au Maroc. Avec l'évolution rapide des technologies de l'information et l'augmentation des volumes de données textuelles non structurées, telles que les emails, il devient essentiel d'exploiter ces sources de manière efficace pour en extraire des informations pertinentes.

Le projet se base sur l'utilisation de techniques avancées de traitement du langage naturel (NLP) et d'apprentissage automatique (machine learning), des domaines de plus en plus indispensables pour analyser et interpréter les textes. L'un des aspects fondamentaux de ce travail consiste à identifier les avis pertinents et à évaluer automatiquement les sentiments exprimés dans les emails. Cette analyse permet non seulement de classifier les messages en fonction de leur contenu, mais aussi de dégager des tendances émotionnelles (positives, négatives ou neutres) au sein des échanges, offrant ainsi une valeur ajoutée dans la prise de décision.

L'application de ces techniques à la gestion des marchés publics permet d'automatiser l'analyse de grandes quantités de données, facilitant ainsi la gestion des informations, la réactivité et l'efficacité des réponses apportées. Plus spécifiquement, cette solution vise à optimiser les processus décisionnels au sein de la Trésorerie Générale du Royaume en fournissant des insights sur les retours et avis liés aux transactions financières et aux demandes du secteur public.

Ce rapport présente, dans un premier temps, la phase de collecte et de prétraitement des données, puis les techniques utilisées pour l'analyse des sentiments et la classification des emails, et enfin, les résultats obtenus et les performances des modèles mis en œuvre. L'accent sera mis sur les défis rencontrés, les solutions adoptées, ainsi que les perspectives d'amélioration pour un usage à grande échelle.

Chapitre I : Contexte Général du Projet

Introduction

Ce chapitre présente l'organisme d'accueil, le contexte du projet et les objectifs. Nous décrirons la TGR, ses missions principales et son positionnement sur le marché. Ensuite, nous aborderons le contexte spécifique du projet, y compris les enjeux et défis. Enfin, nous définirons les objectifs du stage, tant pour le développement de compétences personnelles que pour les contributions à l'entreprise.

1. Présentation de l'organisme d'accueil

1.1 Identification de la TGR

La Trésorerie Générale du Royaume constitue l'une des administrations les plus importantes du Ministère des Finances et de la Privatisation. C'est à travers elle que transite l'ensemble des flux financiers et comptables de l'Etat et des collectivités locales au Maroc. Elle est également au centre d'un maillage institutionnel constitué d'administrations publiques, d'établissements publics, de collectivités locales et d'autres grandes institutions financières tous concernés par la gestion des deniers publics.



Figure 1: Siège de la TGR

Elle joue un rôle majeur dans l'économie. Elle coiffe les services du recouvrement des créances publiques et veille à l'exécution et le contrôle des dépenses publiques de l'État et à la bonne gestion des finances des Collectivités Territoriales. Elle procède également à assurer une activité bancaire et à produire l'information comptable et financière de l'État

1.2 Fiche technique

Tableau 1 Fiche signalétique

| | |
|-------------------------|-----------------------|
| Forme juridique | Administration public |
| Date de creation | 1916 |

| | |
|---------------------|--|
| Missions | <ul style="list-style-type: none"> • Recouvrement des créances publiques • Contrôle et paiement des dépenses publiques • La gestion des finances locales • Gestion des dépôts au Trésor • Production de l'information financière et comptable |
| Siège social | Rabat |
| Effectif | 74% de l'effectif global de la TGR, soit 3492 cadres et agents dont 95% affectés aux Trésoreries Préfectorales, Provinciales et aux Perceptions (soit 3306 cadres et agents) ; Agences Comptables à l'Etranger : 2% de l'effectif global de la TGR, soit 84 cadres et agents. |
| Site Web | www.tgr.gov.ma |
| Adresse | Ilot 31 (près de l'Av. Al Araar) Hay Riad, Rabat |

1.3 Histoire

La Trésorerie Générale du Royaume constitue l'une des administrations les plus importantes du Ministère de l'Economie et des Finances et à travers laquelle transite l'ensemble des flux financiers et comptables de l'Etat et des collectivités locales.

Elle est également au centre d'un maillage institutionnel constitué d'administrations publiques, d'établissements publics, de collectivités locales et d'autres grandes institutions financières tous concernés par la gestion des deniers publics. La TGR a initié, depuis 3 ans, un grand projet de modernisation dont la vision stratégique est sous-tendue par deux objectifs fondamentaux à savoir :

- La contribution à l'amélioration substantielle de la gestion des finances publiques.
- L'amélioration du service rendu aux clients et partenaires.

Parcours dans la gestion des finances publiques du Maroc

Constitué sous le règne du sultan Moulay Slimane (1792-1822), le corps des oumania fut organisé et structuré sous le règne du sultan Moulay El Hassan et comprenait une administration centrale et une Administration locale. Les oumania assuraient le recouvrement des impôts, le paiement des dépenses publiques et octroyaient des avances à l'Etat.

Un maillage local étendu

Au niveau local, il existait toute une variété d'oumania : les ournana des douanes, les oumania el mostafad et les oumania el khers.

Les oumania de douanes, installées dans les ports, étaient chargées de percevoir les droits d'exportation et d'importation. Comme ils détenaient la majeure partie des fonds recouvrés par le Trésor, le makhzen en avait fait ses banquiers, tirait sur eux pour ses paiements et leur demandaient, en cas de besoin, des avances de fonds.

Considérés comme chefs des services financiers dans les villes, Les oumania el mostafad centralisaient les droits de porte (hafer) et de marchés (nekas) et administraient les

biens domaniaux du makhzen, en assurant l'entretien et la location.

Les oumana el Khers (oumana el kabail) exerçaient dans les zones rurales. Ils évaluaient l'achour, estimaient les récoltes et encaissaient les impôts, qu'ils versaient à l'amin el mostafad de la ville la plus proche.

Une administration centrale organisée par mission

Au niveau central, on distinguait l'amin des rentrées, l'amin des dépenses, l'amin des comptes et l'amin el oumana.

L'amin des rentrées était chargé de centraliser le produit des recettes de l'Etat versées par les différents oumana qu'il versait ensuite au bit el mal en présence de deux adouls. Disposant d'une bénika au méchouar, il inscrivait sur son registre toutes les sommes ainsi versées.

Les oumana des dépenses (oumana el sayar) étaient chargés d'assurer le paiement des dépenses du makhzen (traitement des vizirs, solde des troupes et autres dépenses de l'Etat), sur les fonds du bit el mal.

L'amin des comptes (amin al hisabat) avait pour mission de contrôler la comptabilité transmise régulièrement par les oumana en fonction sur l'ensemble du territoire, ainsi que les états relatifs aux arrêtés définitifs de leurs écritures, après cessation de leurs fonctions.

En effet, chaque amin était tenu de lui envoyer, en double exemplaire, un état hebdomadaire de ses recettes. De même qu'il devait lui expédier, dans les sept jours qui suivaient la fin de chaque mois, le compte détaillé du mois écoulé.

Des comptes apurés régulièrement

En outre, avant de quitter leurs fonctions, les oumana se présentaient au makhzen avec un compte général de leur gestion, afin qu'il leur en soit donné déchargé.

Un exemplaire des états ainsi fournis était soumis au Sultan qui le communiquait ensuite au contrôle de la bénika spéciale, faisant office à la fois de comptable supérieur du makhzen, chargé de centralisé l'ensemble de ses recettes et dépenses, et de juge des comptes ayant pour mission d'apurer les comptes des différents oumana.

L'amin el oumana, qui disposait également d'une bénika au mechouar, était placé à la tête du corps des oumana et dirigeait l'ensemble des services financiers. Il avait une parfaite connaissance de la situation financière de l'Etat, tant en ce qui concerne les recettes et les dépenses du makhzen que pour ce qui est de ses biens mobiliers et immobiliers dont il tenait constamment la situation à jour.

Phase transitoire

A compter de 1907, le rôle de trésorier général de l'Empire fut confié à la banque d'Etat du Maroc, par l'acte d'Algésiras en vertu duquel elle devait remplir à la fois de fonctions de " trésorier général de l'Empire " et d'"agent financier du Gouvernement ".

A partir de 1916, la Banque d'Etat du Maroc Perdit ses fonctions de Trésorier général de l'Empire. C'est désormais le trésorier général du Maroc qui fut chargé, dans la zone d'influence française, de centraliser les opérations de recettes et de dépenses de l'Etat, d'assurer le paiement des dépenses publiques et le mouvement de fonds et de gérer les réserves du trésor.

Le Trésorier général à la fois comptable marocain et comptable français

Le dahir du 9 juin 1917 sur la comptabilité publique confia le pouvoir financier aux comptables du Trésor, rendus responsable sur leurs deniers personnels de toute opération financière exécutée par leurs soins.

Bien que relevant directement du ministre français pour ce qui concernait les opérations métropolitaines, le trésorier général était placé sous l'autorité du directeur général des finances du protectorat pour les opérations concernant le budget marocain.

Il assumait donc à titre " Principal " les fonctions de comptable français et, à titre " accessoire ", celles de comptable marocain.

Cette situation a prévalu jusqu'à l'indépendance du Maroc, en 1956.

Le Trésorier général désormais comptable marocain

A la suite de la convention régissant les relations entre le Trésor marocain et le Trésor français signée le 31 décembre 1959, une paierie générale fut instituée auprès de l'ambassade de France à Rabat pour "exécuter ", sur le territoire marocain, les opérations financières du trésor français.

La Trésorerie Générale a donc connu une période transitoire (1959 à 1961) au cours de laquelle le premier trésorier général du Maroc indépendant fut de nationalité française avant qu'un cadre marocain ne lui succédaît le 1er octobre 1961.

Depuis la Trésorerie Générale du Royaume a franchi plusieurs étapes qui lui ont permis, après la phase de marocanisation de ses cadres, d'accompagner l'évolution de son environnement.

1.4 Missions de la trésorerie générale du royaume

1.4.1 Le recouvrement des créances publiques

La TGR assure, par le biais de son vaste réseau de comptables publics, la perception des recettes fiscales et non fiscales, à travers notamment :

- la gestion du contentieux administratif et judiciaire relatif au recouvrement et l'assistance des percepteurs en la matière;
- La prise en charge des ordres de recettes au titre du budget général de l'Etat, des budgets annexes et des comptes spéciaux du Trésor;
- La centralisation des prises en charges et des recouvrements au titre des amendes et condamnations péquéniaires;
- la gestion des comptes de prêts et d'avances accordées par le trésor et de «fonds de roulement» consentis par des organismes de financement des projets publics;
- l'élaboration des statistiques concernant la situation du recouvrement de créances publiques;

1.4.2 Le contrôle et le paiement des dépenses publiques

La TGR assure le contrôle et le règlement des dépenses publiques. Ainsi, le réseau de la TGR est chargé de contrôler la régularité des engagements de la quasi-totalité des dépenses de l'Etat. Elle assure à travers son réseau de comptables, le règlement desdites dépenses. En effet, au vu des propositions d'engagement et des ordres de paiement transmis par les ordonnateurs accrédités, les services de la TGR procèdent au règlement des créances de l'Etat.

La Trésorerie Générale assure également par le biais de la Paierie Principale des Rémunérations (PPR), le contrôle et le traitement de la paie de près 650.000 fonctionnaires.

1.4.3 La gestion des finances locales

A travers son réseau de trésoriers et receveurs communaux, la TGR assure la gestion des budgets de 1659 collectivités locales, de 86 groupements et de 41 arrondissements,

En effet, la TGR procède au recouvrement de leurs créances, au règlement des leurs dépenses et à la paie de leur personnel.

La TGR met à contribution également son expertise en offrant le conseil et l'assistance nécessaires aux collectivités locales .Ce conseil qui est de nature juridique et financière, concerne, entre autres, la modernisation des procédures comptables, l'analyse financière et l'élaboration des tableaux de bord.

1.4.4 La gestion des dépôts au Trésor

La TGR assure la mission de gestion des dépôts au Trésor. Elle participe à travers cette activité au financement de la trésorerie de l'Etat. A ce titre, elle gère les comptes des entreprises et établissements publics qui sont soumis à l'obligation de dépôt de leurs fonds au trésor . Cette activité est étendue également à la gestion des dépôts d'autres personnes morales ou privées.

1.4.5 La production de l'information financière et comptable

La TGR assure la centralisation des opérations comptables de l'Etat et des collectivités locales et de ce fait elle constitue une référence en matière de production et de valorisation de l'information comptable de l'Etat et des collectivités locales.

La production de l'information comptable permet ainsi de :

- décrire précisément les opérations budgétaires et financières.
- restituer rapidement une information fiable et indispensable à la prise de décision.
- préparer les documents relatifs à la reddition des comptes.

1.5 Valeurs

Ces valeurs partagées sont le socle commun de la Trésorerie Générale du Royaume auxquelles elle fait référer ses ambitions, fédère ses actions quotidiennes et fonde ses jugements moraux pour guider sa conduite sur un plan individuel et collectif.

✓ **Engagement** : Ethique – Intégrité - Solidarité et Proximité – Egalité - Motivation
Un contrat éthique qui se traduit par l'adhésion et l'implication de l'ensemble des femmes et des hommes de la Trésorerie Générale du Royaume dans la vision et les objectifs de l'institution qui place les clients et partenaires au centre de ses préoccupations.

✓ **Ouverture** : Transparence- Communication- Information- Ecoute
La volonté en externe et en interne de développer la communication, de partager l'information, de renforcer l'écoute et de respecter au quotidien le principe de transparence pour contribuer à l'efficacité de la gestion publique.

✓ **Performance** : Rigueur- Responsabilité- Professionnalisme- Service qualité
Encourager la responsabilité individuelle et collective, consolider le professionnalisme des équipes dans le cadre de la rigueur quotidienne et d'une recherche continue d'un service de qualité pour développer les métiers et les projets de la Trésorerie Générale du Royaume et créer de la valeur.

✓ **Innovation** : Créativité- Initiative- Modernité- Acteur de changement
Encourager la prise d'initiatives en initiant le principe de modernité et en valorisant la créativité à tous les niveaux pour être acteur du changement.

1.6 Organisation de la TGR

Cette sous-section comprendra la présentation de la TGR, ses missions principales, son organisation, et le lieu de déroulement de notre stage.

La TGR s'organise en six directions :

- Direction de la Recherche, de la Réglementation et de la Coopération Internationale
 - Direction des Finances Publiques
 - Direction des Dépenses du Personnel
 - Direction des Comptes Publics
 - Direction des Ressources et du Système d'Information
 - Direction du Contrôle, de l'Audit et de l'Inspection

Cette structure hiérarchique détaillée permet de clarifier les rôles et responsabilités au sein de la Trésorerie Générale du Royaume, facilitant ainsi la coordination et l'efficacité des opérations.

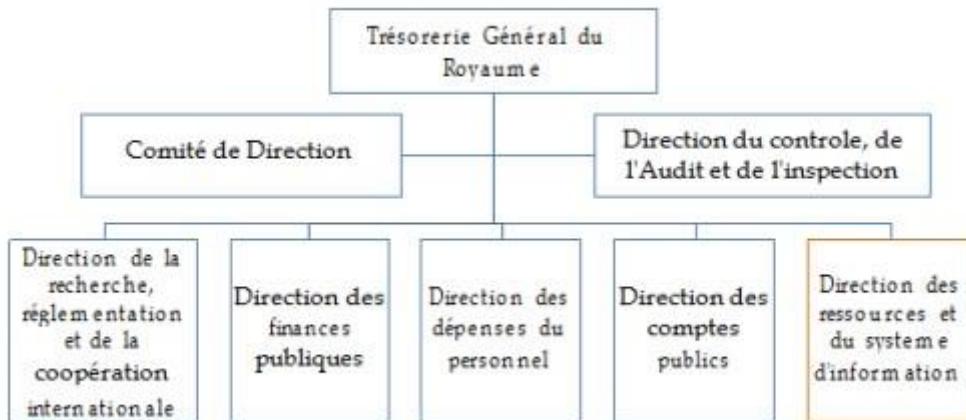


Figure 2: Organigramme de la TGR

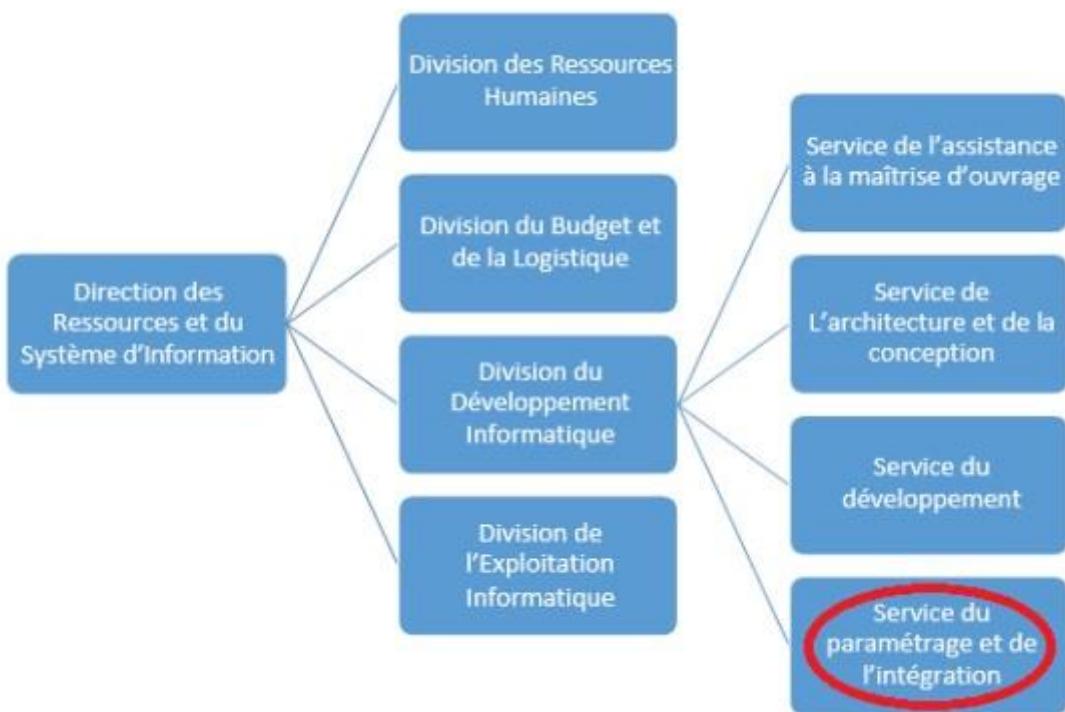


Figure 3: Organigramme de direction des ressources et du système d'information

Structure générale de la Trésorerie Générale du Royaume :

- 1. Comité de Direction :** C'est l'organe décisionnel principal qui définit les politiques, stratégies et orientations générales de la Trésorerie Générale du Royaume.
- 2. Direction de Contrôle, d'Audit et d'Inspection :** Elle est chargée d'évaluer les opérations

internes, de garantir la conformité et de gérer les risques. Cette direction assure la transparence et l'intégrité des opérations.

3. Direction de la Recherche, de la Réglementation et de la Coopération Internationale :

Cette direction s'occupe de la recherche et de l'analyse des réglementations financières, ainsi que de la coopération avec les organismes internationaux.

4. Direction des Finances Publiques : Elle est responsable de la gestion des finances publiques, de l'élaboration du budget de l'État et du suivi des dépenses publiques.

5. Direction des Dépenses du Personnel : Cette direction gère les dépenses liées au personnel de l'État, incluant les salaires, les avantages sociaux et les pensions.

6. Direction des Comptes Publics : Elle supervise la comptabilité publique, assurant l'exactitude des comptes de l'État et la transparence financière.

7. Direction des Ressources et du Système d'Information : Cette direction gère les ressources humaines, le budget, la logistique et les systèmes d'information de la Trésorerie Générale du Royaume.

Divisions au sein de la direction des ressources et du système d'Information

- **Division des Ressources Humaines :** Elle s'occupe de la gestion du personnel, incluant le recrutement, la formation, et le développement des compétences des employés.
- **Division du Budget et de la Logistique :** Elle gère le budget interne et la logistique, assurant que les ressources sont utilisées de manière efficace et efficiente.
- **Division du Développement Informatique :** Cette division développe les systèmes informatiques et les logiciels nécessaires au bon fonctionnement de la Trésorerie.
- **Division de l'Exploitation Informatique :** Elle assure le fonctionnement quotidien des systèmes informatiques, incluant la maintenance et le support technique.

Services au sein de la Division du Développement Informatique

Service de l'Assistance à la Maîtrise d'Ouvrage : Ce service aide les différentes unités de la Trésorerie à définir leurs besoins en systèmes informatiques et à gérer les projets informatiques.

Service de l'Architecture et de la Conception: Il conçoit l'architecture des systèmes informatiques, assurant qu'ils répondent aux besoins de l'organisation et sont évolutifs.

Service du Développement: Ce service développe les applications et logiciels nécessaires, les adapte et les met à jour en fonction des besoins de la Trésorerie.

Service du Paramétrage et de l'Intégration: Il configue et intègre les différents systèmes et applications, assurant qu'ils fonctionnent ensemble de manière cohérente.

1.7 Services proposés et modes de passation

Au Maroc, les marchés publics sont régis par la loi n° 52-05 relative aux marchés publics. Cette loi définit différents services et modes de passation des marchés publics. Voici une liste des principaux services proposés et des modes de passation utilisés au Maroc :

Services proposés :

- **Fournitures :** Acquisition de biens matériels tels que équipements, fournitures de bureau, matériel informatique, etc.
- **Services :** Prestation de services intellectuels, techniques, professionnels ou administratifs tels

- que consultant, formation, maintenance, etc.
- **Travaux** : Réalisation de travaux de construction, d'aménagement, de réhabilitation, etc.

Modes de passation :

Appel d'Offres Ouvert (AO) : Procédure où tous les fournisseurs intéressés peuvent soumissionner en réponse à un appel d'offres publié par l'entité adjudicatrice.

Appel d'Offres Restreint (AR) : Procédure similaire à l'AO, mais seuls les fournisseurs pré-qualifiés sont invités à soumissionner.

Consultation Restreinte (CR) : Procédure où un nombre restreint de fournisseurs est invité à soumissionner par l'entité adjudicatrice.

Marché Négocié (MN) : Procédure dans laquelle l'entité adjudicatrice négocie directement avec un ou plusieurs fournisseurs sans passer par une mise en concurrence formelle.

Marché de Gré à Gré (GG) : Procédure où l'entité adjudicatrice conclut directement un contrat avec un fournisseur sans mise en concurrence préalable. Cette procédure est soumise à des restrictions strictes.

Concours : Procédure utilisée pour la passation de marchés de services intellectuels où les soumissionnaires sont évalués sur la base de critères de qualité.

Appel d'Offres avec Pré-qualification (AOP) : Procédure où les soumissionnaires doivent préalablement être qualifiés sur la base de critères techniques, financiers ou professionnels avant de soumissionner.

Ces services et modes de passation offrent une certaine flexibilité aux entités adjudicatrices pour répondre à leurs besoins spécifiques tout en garantissant la transparence, l'égalité de traitement et la concurrence dans le processus d'attribution des marchés publics au Maroc.

2 Contexte du projet

2.1 Contexte général du projet

Un marché public est un contrat, conclu à titre onéreux, entre d'une part un maître et, d'autre part, une personne physique ou morale appelée entrepreneur, fournisseur ou prestataire de services pour l'exécution de travaux, livraison de fournitures ou la réalisation de prestations de services avec contrepartie.

La Trésorerie Générale du Royaume (TGR), essentielle à la gestion des finances publiques et des marchés publics au Maroc, se trouve confrontée à la nécessité de comprendre et d'analyser les perceptions des citoyens, des entreprises et des parties prenantes concernant les marchés publics. Ces perceptions influent directement sur la confiance publique, l'efficacité des politiques publiques et la transparence des processus administratifs.

Afin de collecter les feedbacks et les réclamations des clients, TGR a mis en place une application en ligne, www.marchespublics.gov.ma, spécifiquement de marchés publics et dans laquelle il y a la fenêtre Contact qui contient l'adresse de dépôt de réclamations client, marchespublics@tgr.gov.ma. Ensuite, selon l'objet de la réclamation, il existe deux techniciens qui reçoivent et répondent aux emails selon le sujet de chaque email qui peut être un problème technique ou un problème métier.

La TGR reçoit chaque jour de dizaines voire des centaines d'emails. Ce qui rend la réponse à cet énorme nombre une tâche fastidieuse et épique. En plus, l'analyse de ces emails

afin d'évaluer la qualité de service est complexe.

Dans ce contexte, le projet vise à concevoir, développer et implémenter un programme informatique performant pour analyser les sentiments exprimés en ligne à propos des marchés publics. L'objectif est de fournir des insights pertinents et actionnables afin d'améliorer les pratiques de gestion de la TGR, de renforcer la transparence et de consolider la confiance publique. Par l'intégration de technologies de Data Science et de Business Intelligence, ce projet ambitionne de transformer la gestion des marchés publics, en offrant des outils avancés pour la surveillance, l'analyse et l'optimisation des processus, tout en répondant aux attentes des citoyens et des parties prenantes.

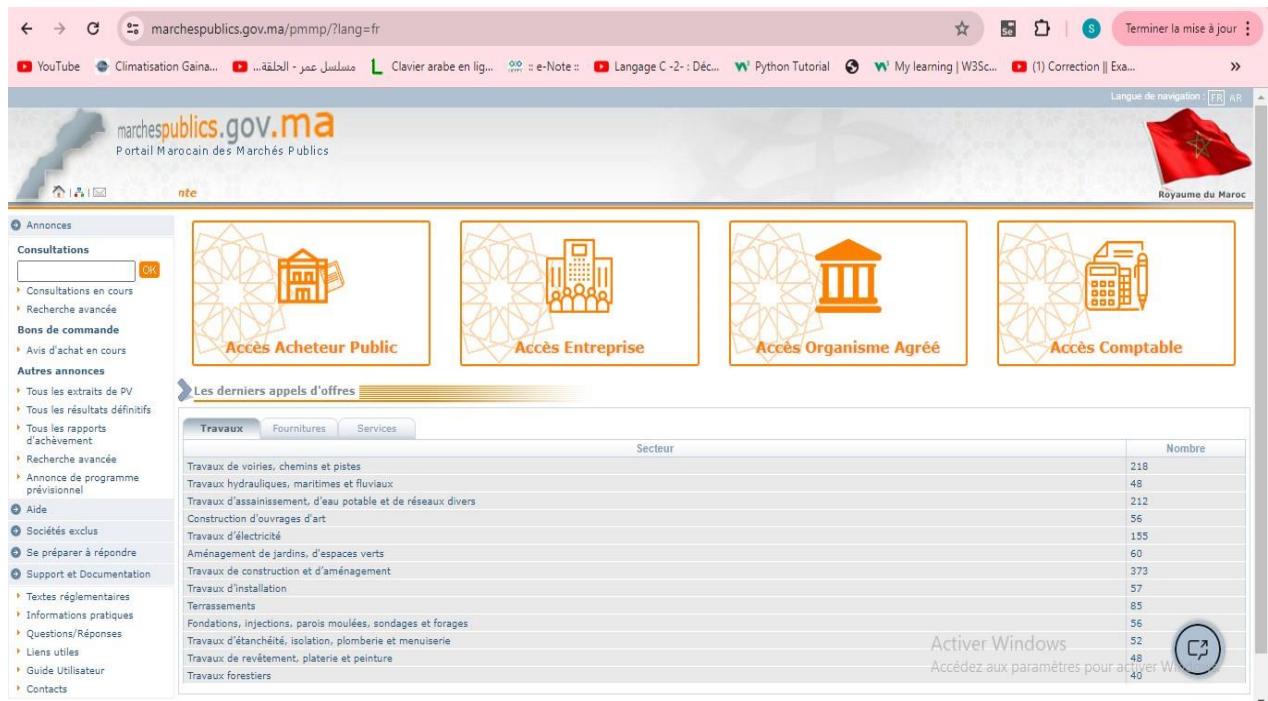


Figure 4: Site web des marchés publics

Figure 5: Contact service marchés publics

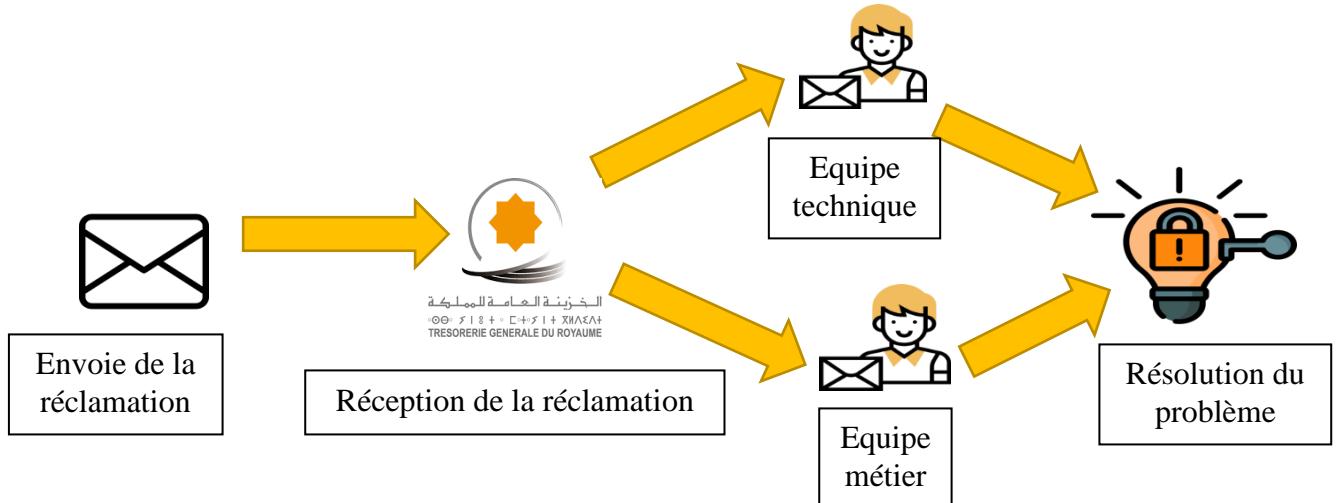


Figure 6: Processus d'une réclamation

2.2. Problématique :

La plate-forme de marchés publics de la Trésorerie Générale du Royaume (TGR) joue un rôle central dans la gestion et la régulation des marchés publics au sein du pays. Elle sert de point de convergence pour toutes les transactions liées aux marchés publics, permettant une transparence accrue et une gestion plus efficace des processus d'achat. En regroupant les appels d'offres, les contrats, et les autres documents pertinents en un seul endroit, la plate-forme facilite l'accès à l'information pour les entreprises et les organismes publics, assurant ainsi une concurrence équitable et une allocation optimale des ressources.

De plus, la plate-forme de la TGR contribue à la modernisation et à la digitalisation des procédures administratives. En automatisant et en standardisant les processus de soumission et de gestion des marchés publics, elle réduit les délais et les coûts associés à la bureaucratie traditionnelle. Cette efficacité accrue non seulement améliore l'expérience utilisateur pour les entreprises soumissionnaires mais aussi renforce la capacité de l'administration publique à suivre et à évaluer les projets en temps réel. En intégrant des outils de suivi et d'analyse, la plate-forme permet également une meilleure supervision et une gestion plus proactive des contrats, garantissant ainsi la bonne utilisation des fonds publics et l'atteinte des objectifs de développement économique et social.

Cependant, l'absence d'un outil automatisé d'évaluation de la satisfaction des utilisateurs constitue un problème majeur. Cette lacune empêche de recueillir des retours d'expérience précis et réguliers de la part des entreprises et des organismes publics qui utilisent la plateforme. Sans ces retours, il devient difficile d'identifier les points à améliorer et de répondre efficacement aux besoins et aux attentes des utilisateurs.

Cette analyse approfondie des sentiments en ligne est indispensable pour éclairer les décisions stratégiques et opérationnelles de la TGR. Ainsi, la question fondamentale qui se pose est la suivante : Comment concevoir, développer et implémenter un programme informatique performant pour analyser les sentiments exprimés en ligne sur les marchés publics, afin d'informer et d'améliorer les pratiques de gestion de la Trésorerie Générale du Royaume, dans un souci de renforcement de la confiance publique, de l'efficacité des politiques publiques et de la transparence des processus administratifs ?

Objectifs :

- Identifier et comprendre les principaux sentiments exprimés sur les marchés publics et leur évolution dans le temps.
- Évaluer l'impact de ces sentiments sur la perception publique et la performance des marchés publics.
- Développer un programme informatique robuste pour l'analyse des sentiments exprimés sur les marchés publics, spécifiquement pour les besoins de la TGR.
- Fournir des insights pertinents et actionnables pour améliorer la gestion des marchés publics et renforcer la transparence et la confiance publique.

2.3 Besoins fonctionnels

Collecte des Données :

Le programme doit pouvoir recueillir des données à partir de diverses sources en ligne : Emails de messagerie en site web de la TGR, Commentaires des groupes FACEBOOK des marchés publics au Maroc.

Il doit prendre en charge le format de données texte (xlsx).

Nettoyage et Prétraitement des Données :

Les données collectées doivent être nettoyées pour éliminer les bruits.

Le programme doit être capable de normaliser les données en gérant les synonymes, les fautes d'orthographe, et les variations linguistiques.

Analyse des Sentiments :

Le programme doit analyser les textes pour déterminer les sentiments (positif, négatif, neutre).

Il doit être capable de gérer le traitement du langage naturel (NLP) pour extraire les opinions et les sentiments exprimés.

Visualisation des Données :

Les résultats de l'analyse doivent être présentés sous forme de graphiques et de tableaux de bord interactifs.

Les utilisateurs doivent pouvoir filtrer et explorer les données selon différents critères (période, source, type de sentiment, etc.).

Interface Utilisateur :

Le programme doit offrir une interface utilisateur qui permet d'afficher les pourcentages des sentiment en chargeant le fichier pkl du système.

Il doit permettre aux utilisateurs de personnaliser les paramètres d'analyse et de visualisation.

2.4 Besoins non-fonctionnels

○ Performance :

- Le programme doit être capable de traiter de grandes quantités de données en temps réel ou quasi temps réel.
- Il doit offrir des temps de réponse rapides pour les requêtes et les analyses.

○ Scalabilité :

- Le système doit être évolutif pour gérer une augmentation du volume de données sans dégradation des performances.
- Il doit pouvoir s'adapter aux nouvelles sources de données et aux évolutions des besoins de la TGR.

○ Sécurité :

- Les données collectées et analysées doivent être protégées contre les accès non autorisés.
- Le programme doit respecter les réglementations en matière de protection des données personnelles et de confidentialité.

○ Fiabilité :

- Le programme doit garantir un haut niveau de disponibilité et de résilience face aux pannes.
- Il doit inclure des mécanismes de sauvegarde et de récupération en cas de défaillance.

○ Maintenabilité :

- Le code du programme doit être bien documenté pour faciliter les mises à jour et les améliorations futures.
- Il doit être modulable pour permettre des ajouts ou des modifications sans affecter l'ensemble du système.

○ **Compatibilité :**

- Le programme doit être compatible avec les autres systèmes et infrastructures informatiques existants de la TGR.
- Il doit pouvoir s'intégrer facilement avec des outils de gestion et de reporting utilisés par la TGR.

○ **Utilisabilité :**

- L'interface utilisateur doit être intuitive et facile à utiliser, même pour les utilisateurs non techniques.
- Une formation minimale doit être nécessaire pour que les utilisateurs puissent exploiter pleinement les fonctionnalités du programme.

2 Connaissances sur le métier

3.1. Métiers courants TGR marocaine

Au Maroc, la Trésorerie Générale du Royaume (TGR) joue un rôle central dans la gestion des finances publiques. Voici quelques-uns des métiers courants que l'on retrouve généralement au sein de la TGR marocaine :

Trésorier général : Responsable de la supervision de toutes les activités financières de l'État et de la mise en œuvre des politiques financières définies par le gouvernement.

Contrôleur financier : Chargé de veiller à la conformité des opérations financières avec les lois et réglementations en vigueur, ainsi que de l'analyse des données financières et de la préparation de rapports.

Gestionnaire de trésorerie : Responsable de la gestion quotidienne des liquidités de l'État, s'assurant que les fonds nécessaires sont disponibles pour les dépenses courantes et optimisant les investissements à court terme.

Analyste financier : Chargé d'analyser les données financières et économiques pour soutenir les décisions de gestion financière et formuler des recommandations stratégiques.

Comptable public : Responsable de la comptabilisation des opérations financières de l'État, y compris l'enregistrement des transactions, l'établissement des états financiers et la conformité aux normes comptables.

Expert en gestion de la dette publique : Chargé de la gestion de la dette souveraine, notamment l'émission et le remboursement des obligations, ainsi que du développement de stratégies visant à minimiser les coûts d'emprunt.

Spécialiste en trésorerie et paiements électroniques : Responsable de la mise en œuvre et de la gestion des systèmes de paiement électronique pour l'administration publique, favorisant l'efficacité et la transparence des transactions financières.

Responsable des relations bancaires : Entretient des relations avec les banques et autres institutions financières pour faciliter les opérations de trésorerie et négocier les meilleures conditions pour l'État.

Auditeur financier : Effectue des audits des opérations financières de l'État pour garantir la fiabilité, la légalité et la transparence des processus financiers.

3.2 Les marchés publics

3.2.1 Définition des marchés publics

Un marché public est un contrat, conclu à titre onéreux, entre d'un part un maître d'ouvrage et, d'autre part, une personne physique ou morale appelée entrepreneur, fournisseur ou prestataire de services pour l'exécution de travaux, la livraison de fournitures ou la réalisation de prestations de services avec contrepartie.

Les prestations et fournitures acquises dans le cadre d'un marché public donnent lieu au versement d'un prix par la personne publique. Les prestations que cette dernière se procure à titre gratuit ne sont donc pas des marchés publics.

3.2.2 Le principe des marchés publics

Pour que vous puissiez comprendre le concept d'appel d'offre, il vous faut comprendre le principe de marché public. En effet, les marchés publics constituent l'outil principal de réalisation des achats publics. Ainsi, un marché public est un contrat qui est conclu, à titre onéreux, entre un maître d'ouvrage et un entrepreneur, fournisseur ou prestataire de services. L'objet de ce contrat peut être l'exécution de travaux, la livraison de fournitures ou la réalisation de prestations de services. Les marchés publics au Maroc peuvent être conclus selon différentes procédures, notamment par des appels d'offres, des marchés négociés ou des bons de commande.

3.2.3 Les types des marchés publics

On peut distinguer trois types de marchés publics : le marché de travaux, des fournitures et des services. Le premier type de marché concerne la construction, à la reconstruction, à la démolition, à la réparation ou à la rénovation, à l'aménagement et à l'entretien d'un bâtiment, d'un ouvrage ou d'une structure ainsi que les travaux de reboisement. Le deuxième marché a pour objet l'achat ou la location avec option d'achat de produits ou de matériel. Ces dits produits existent dans le commerce et ne sont pas fabriqués sur des spécifications techniques particulières. Et finalement, le dernier marché concerne la réalisation de prestations de services. Il peut s'agir des marchés d'études, de formation, prestations de laboratoires ou prestations architecturales.

Les marchés négociés ou les bons de commande

Les marchés négociés représentent un mode de passation des marchés publics. Tout simplement, le marché négocié s'inscrit dans le cadre des modes exceptionnels de passation des marchés. Il permet au maître d'ouvrage de négocier les conditions du marché avec un ou plusieurs candidats. Pour ce qui est des bons de commande, c'est un mode qui permet

l'acquisition de fournitures et la réalisation de travaux ou services. La condition est que le montant du bon de commande ne doit pas dépasser deux cent mille (200 000,00) dirhams toutes taxes comprises.

Un appel d'offres

Les marchés publics peuvent être conclus par appel d'offres. Ce dernier permet la mise en concurrence de plusieurs fournisseurs ou prestataires de services. Les appels d'offre peuvent prendre plusieurs formes. Il y a d'abord, l'appel d'offre ouvert. Ce type d'appel d'offre permet à tout concurrent d'obtenir le dossier de consultation et présenter sa candidature. Par la suite, il y a l'appel d'offre restreint. Il s'agit dans ce type d'appel d'offre lorsque le maître d'ouvrage décide des concurrents qui peuvent participer. Ainsi, seuls ces derniers peuvent remettre leurs offres. Et finalement, il y a l'appel d'offres avec présélection. Dans ce cas de figure, une commission d'amission choisit des concurrents sur la base de leurs capacités techniques et financières. Et par la suite, seuls ces concurrents présélectionnés ont la possibilité de présenter leurs offres.

3.3. Services et modes de passation

Au Maroc, les marchés publics sont régis par la loi n° 52-05 relative aux marchés publics. Cette loi définit différents services et modes de passation des marchés publics. Voici une liste des principaux services proposés et des modes de passation utilisés au Maroc :

Services proposés :

Fournitures : Acquisition de biens matériels tels que les équipements, fournitures de bureau, matériel informatique, etc.

Services : Prestation de services intellectuels, techniques, professionnels ou administratifs tels que consultant, formation, maintenance, etc.

Travaux : Réalisation de travaux de construction, d'aménagement, de réhabilitation, etc.

Modes de passation :

Appel d'Offres Ouvert (AO) : Procédure où tous les fournisseurs intéressés peuvent soumissionner en réponse à un appel d'offres publié par l'entité adjudicatrice.

Appel d'Offres Restreint (AR) : Procédure similaire à l'AO, mais seuls les fournisseurs préqualifiés sont invités à soumissionner.

Consultation Restreinte (CR) : Procédure où un nombre restreint de fournisseurs est invité à soumissionner par l'entité adjudicatrice.

Marché Négocié (MN) : Procédure dans laquelle l'entité adjudicatrice négocie directement avec un ou plusieurs fournisseurs sans passer par une mise en concurrence formelle.

Marché de Gré à Gré (GG) : Procédure où l'entité adjudicatrice conclut directement un contrat avec un fournisseur sans mise en concurrence préalable. Cette procédure est soumise à des restrictions strictes.

Concours : Procédure utilisée pour la passation de marchés de services intellectuels où les soumissionnaires sont évalués sur la base de critères de qualité.

Appel d'Offres avec Pré-qualification (AOP) : Procédure où les soumissionnaires doivent préalablement être qualifiés sur la base de critères techniques, financiers ou professionnels avant de soumissionner.

Ces services et modes de passation offrent une certaine flexibilité aux entités adjudicatrices pour répondre à leurs besoins spécifiques tout en garantissant la transparence, l'égalité de traitement et la concurrence dans le processus d'attribution des marchés publics au Maroc.

Conclusion :

Le premier chapitre du rapport présente le contexte général du projet en expliquant le rôle de la Trésorerie Générale du Royaume (TGR), l'organisme d'accueil, dans la gestion des finances publiques au Maroc. Ce chapitre met en lumière les missions essentielles de la TGR, telles que le recouvrement des créances publiques et le contrôle des dépenses publiques, tout en introduisant le cadre du projet, qui vise à développer une application d'analyse des sentiments à partir d'emails, afin d'améliorer la réactivité et l'efficacité dans le traitement des échanges liés aux marchés publics

Chapitre II : Etat de l'Art & Techniques de Machine Learning

Introduction

Ce chapitre est généralement structuré en deux parties : une revue de l'état de l'art des techniques et méthodes existantes, et une discussion sur les techniques de machine learning appliquées dans le domaine pertinent.

1 Analyse de sentiments

1.1 Définition

L'analyse des sentiments, également connue sous le nom d'opinion mining, est une technique d'analyse de données textuelles qui vise à déterminer le sentiment global, l'attitude ou l'émotion exprimée dans un texte. Cette méthode est largement utilisée dans divers domaines, notamment le marketing, la veille stratégique, la gestion de la réputation en ligne, et même la politique pour comprendre les opinions du public sur différents sujets.

En résumé, l'analyse des sentiments est une technique puissante pour extraire des informations précieuses à partir de grandes quantités de données textuelles, permettant aux entreprises et aux organisations de comprendre les opinions, les attitudes et les émotions du public à l'égard de leurs produits, services ou sujets d'intérêt.

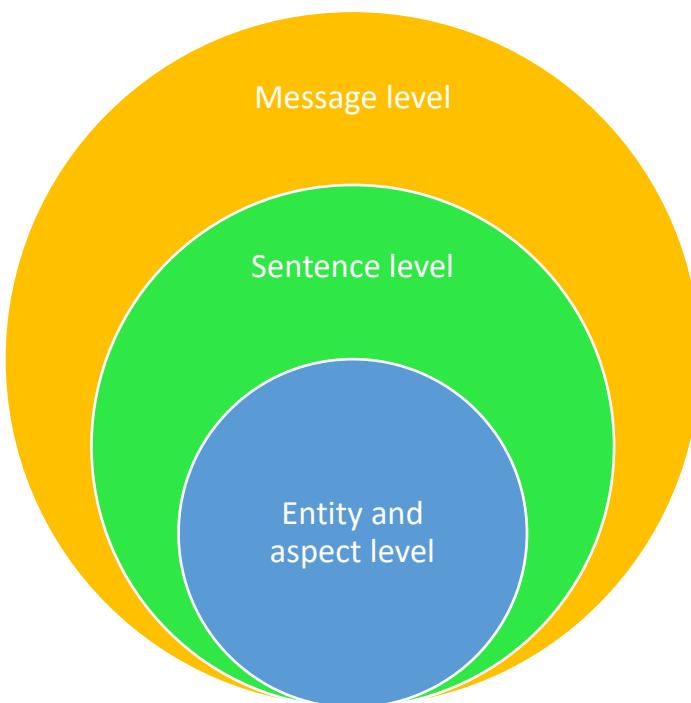


Figure 7 : Les niveaux d'analyse des sentiments

1.2 Domaines d'application de l'analyse des sentiments

L'analyse des sentiments, une branche du Traitement Automatique du Langage Naturel (TALN), joue un rôle essentiel dans de nombreux secteurs. Elle permet d'extraire des opinions, des émotions et des attitudes à partir de textes non structurés, tels que des critiques, des publications sur les réseaux sociaux ou des avis clients. Voici les principaux domaines d'application de cette technologie :

○ Marketing et Étude de la Satisfaction Client

L'un des usages les plus courants de l'analyse des sentiments se trouve dans l'évaluation de la satisfaction des clients. Les entreprises peuvent analyser les avis laissés par leurs clients sur des plateformes telles qu'Amazon, Yelp ou Google pour :

- Comprendre la perception des produits ou services.
- Identifier les points de friction dans l'expérience client.
- Améliorer leurs offres en fonction des retours négatifs. L'analyse de sentiments est également utilisée pour évaluer l'impact des campagnes marketing et ajuster les stratégies en temps réel.

○ Analyse des Réseaux Sociaux

Les réseaux sociaux représentent une source massive de données d'opinions. L'analyse des sentiments permet de :

- Suivre la réputation en ligne des marques, personnalités publiques ou événements.
- Déetecter les tendances émergentes dans l'opinion publique.
- Réagir aux crises médiatiques en identifiant rapidement les opinions négatives massives (bad buzz).

○ Politique et Analyse d'Opinion Publique

Dans le domaine politique, l'analyse des sentiments est utilisée pour :

- Mesurer l'opinion publique lors d'élections ou de débats politiques.
- Évaluer l'impact des discours politiques et des décisions gouvernementales. Les analystes peuvent ainsi suivre en temps réel l'évolution des sentiments de l'électorat à travers les réseaux sociaux, les forums et les sondages en ligne.

○ Finance et Prévision des Marchés

Les émotions jouent un rôle important dans les décisions financières. Les investisseurs et les analystes financiers utilisent l'analyse des sentiments pour :

- Prévoir les fluctuations du marché boursier en se basant sur les opinions exprimées dans les médias, blogs, ou forums financiers.
- Suivre les réactions aux annonces de résultats financiers ou aux changements stratégiques des entreprises.
- Évaluer la confiance des investisseurs à partir des commentaires sur les réseaux sociaux et les forums spécialisés.

○ Ressources Humaines

L'analyse des sentiments peut être utilisée pour :

- Évaluer le climat interne dans une organisation en analysant les retours des employés dans des enquêtes ou des feedbacks anonymes.
- Améliorer la gestion des talents en détectant les sentiments de frustration ou de satisfaction au travail, permettant ainsi de prévenir le turnover.

○ Domaines Juridique et Sécurité

L'analyse des sentiments peut être appliquée dans le cadre de la sécurité et de la surveillance pour :

- Déetecter des discours haineux ou incitant à la violence sur les réseaux sociaux.
- Identifier des menaces potentielles dans les messages publics ou privés. Dans le domaine juridique, elle peut également être utilisée pour analyser l'opinion publique sur des affaires judiciaires ou des lois en vigueur.

○ Médias et Divertissement

Les industries des médias et du divertissement utilisent l'analyse des sentiments pour :

- Mesurer l'accueil du public à l'égard de films, séries, émissions ou contenus musicaux.
- Adapter les campagnes promotionnelles en fonction des réactions du public.
- Suivre la notoriété des artistes ou des productions à travers les commentaires des utilisateurs sur des plateformes comme Twitter ou YouTube.

○ Santé

Dans le domaine de la santé, l'analyse des sentiments permet de :

- Suivre l'opinion des patients concernant les traitements, les hôpitaux ou les professionnels de santé.
- Détecter les signes de détresse émotionnelle chez les patients à travers l'analyse des publications sur les forums de santé ou des plateformes de discussion. Elle peut également être utilisée pour évaluer les réactions du public face à des crises sanitaires, comme les pandémies, et ajuster la communication en conséquence.

1. Traitement Automatique du Langage Naturel (NLP)

2.1. Définition

Le Traitement Automatique du Langage Naturel (TALN), ou Natural Language Processing (NLP) en anglais, est une discipline qui se situe à la croisée de l'intelligence artificielle, de la linguistique et de l'informatique. Elle se concentre sur l'interaction entre les humains et les machines à travers le langage naturel, qu'il soit écrit ou parlé. L'objectif principal du TALN est de permettre aux ordinateurs de comprendre, analyser et générer du texte de manière similaire à celle des humains.

2.2 Objectifs et Enjeux du NLP

L'une des principales difficultés dans le traitement du langage naturel réside dans la diversité des structures linguistiques et la richesse sémantique des langues humaines. Les systèmes de NLP cherchent à :

- Comprendre le sens d'un texte ou d'un discours.
- Extraire de l'information pertinente.
- Répondre de manière cohérente à des requêtes ou des questions.
- Analyser des sentiments ou des opinions contenus dans un texte.

2.3 Applications du NLP

Le NLP est utilisé dans une variété d'applications industrielles, dont certaines sont courantes dans la vie quotidienne :

- **Assistants vocaux** comme Siri, Alexa ou Google Assistant, qui interprètent les commandes vocales et interagissent avec les utilisateurs.
- **Analyse de sentiments** pour comprendre les opinions exprimées dans des commentaires, des tweets ou des avis clients.

- **Chatbots** et agents conversationnels qui automatisent les réponses aux utilisateurs dans les services clientèle.
- **Systèmes de traduction automatique** comme Google Translate, permettant de traduire des textes d'une langue à une autre.
- **Filtrage de spams** dans les emails, pour distinguer les courriers légitimes des messages indésirables.

3 Vectorisation du Texte

Les techniques de vectorisation visent à conserver le sens et la structure des textes tout en les simplifiant dans un format compréhensible pour les modèles d'apprentissage. Les deux principales méthodes utilisées dans ce projet sont le Count Vectorizer et le TF-IDF Vectorizer.

3.1 TF-IDF Word Level

Le TF-IDF (Term Frequency-Inverse Document Frequency) au niveau des mots est une technique qui permet de pondérer les mots d'un texte en fonction de leur importance dans un ensemble de documents. Le calcul est basé sur deux composants principaux :

Term Frequency (TF) : Nombre de fois qu'un mot apparaît dans un document, divisé par le nombre total de mots dans ce document.

Inverse Document Frequency (IDF) : Mesure qui diminue le poids des mots courants dans l'ensemble de documents et augmente celui des mots rares. Elle est calculée comme le logarithme du nombre total de documents divisé par le nombre de documents contenant le mot.

Cette méthode est utilisée pour pondérer les mots de manière à donner plus d'importance aux mots qui apparaissent fréquemment dans un document mais rarement dans l'ensemble des documents.

Utilisation dans un projet : Dans ton projet, le TF-IDF Word Level permet de transformer chaque document en un vecteur où chaque dimension représente un mot. Ce vecteur contient des valeurs représentant l'importance de chaque mot dans le document, ce qui est ensuite utilisé pour des modèles de machine learning comme la classification des sentiments.

3.2 TF-IDF Vectorizer with N-grams

Le TF-IDF avec des N-grams est une extension du TF-IDF standard, qui prend en compte non seulement des mots individuels (unigrammes), mais aussi des séquences de mots contigus appelés n-grams. Les N-grams sont des groupes de mots adjacents dans un texte.

- **Unigram** : Mot unique (ex : "chat")
- **Bigram** : Deux mots consécutifs (ex : "chat noir")
- **Trigram** : Trois mots consécutifs (ex : "chat noir dort")

L'utilisation de N-grams permet de capturer des informations sur des expressions courantes ou des séquences de mots qui peuvent être importantes pour l'analyse.

Utilisation dans un projet : Le TF-IDF avec N-grams est utile dans des cas où des expressions complètes ou des combinaisons de mots sont plus significatives que des mots individuels. Par exemple, dans l'analyse des sentiments, la phrase "pas content" est négative, mais un TF-IDF classique pourrait mal interpréter le mot "content". L'utilisation de bigrams peut résoudre ce problème en considérant "pas content" comme une seule entité.

3.3 TF-IDF Vectorizer

Le TF-IDF Vectorizer est l'outil qui permet de convertir une collection de documents en une matrice TF-IDF. Il est couramment utilisé dans les projets d'analyse de texte pour transformer les textes bruts en matrices numériques, que les algorithmes de machine learning peuvent comprendre.

Le TF-IDF Vectorizer dans scikit-learn, par exemple, permet de :

- Transformer des documents textuels en vecteurs pondérés TF-IDF.
- Spécifier des options pour inclure des unigrammes, des bigrams, ou d'autres N-grams.
- Exclure des termes fréquents non pertinents ou inclure uniquement les mots importants.

Utilisation dans un projet : En utilisant un **TF-IDF Vectorizer**, chaque document dans ton corpus sera transformé en un vecteur avec des scores TF-IDF, facilitant l'entraînement des modèles de machine learning.

3.4 Count Vectors (CountVectorizer)

Le Count Vectorizer est une méthode plus simple de vectorisation du texte que TF-IDF. Il transforme un texte en un vecteur où chaque dimension représente un mot, et les valeurs correspondent à la fréquence de chaque mot dans un document. Contrairement au TF-IDF, qui pondère les mots selon leur importance, le Count Vectorizer ne fait que compter les occurrences des mots.

Utilisation dans un projet :

Le Count Vectorizer est souvent utilisé lorsque la fréquence brute des mots dans un texte est suffisante pour une analyse. C'est une technique de base utilisée pour la classification de texte ou d'autres analyses linguistiques simples.

Il est plus facile à comprendre et à mettre en œuvre, mais il peut ne pas être aussi performant que TF-IDF dans les cas où des mots courants peuvent perturber les résultats (car il ne pondère pas en fonction de l'importance).

3.5 Comparaison des techniques de vectorisation

Les trois techniques de vectorisation ont été testées dans le cadre du projet afin d'évaluer leur impact sur la qualité des modèles d'apprentissage automatique. Le tableau suivant résume les avantages et inconvénients observés lors de leur utilisation :

Tableau 2 : comparaison de techniques de vectorisation

| Technique | Avantages | Inconvénients |
|---------------------|--|---|
| CountVectorizer | Simplicité, rapide à implémenter | Ne tient pas compte de l'importance relative des mots dans l'ensemble du corpus |
| TF-IDF | Pondération des mots en fonction de leur importance dans le corpus | Plus complexe, peut être coûteux en temps de calcul |
| TF-IDF avec N-grams | Capture des relations contextuelles entre les mots (bigrams, trigrams, etc.) | Nécessite plus de ressources et de mémoire pour traiter les N-grams |

4 Apprentissage automatique

4.1. Définition

Le Machine Learning (ou apprentissage automatique) est une branche de l'intelligence artificielle qui permet aux ordinateurs d'apprendre à partir de données sans être explicitement programmés pour une tâche spécifique. Contrairement aux approches traditionnelles de programmation, où les règles sont définies à l'avance, le Machine Learning repose sur des algorithmes qui apprennent et s'améliorent en fonction des exemples de données fournis.

4.2. Principe du Machine Learning

Le Machine Learning consiste à créer des modèles capables de faire des prédictions ou de prendre des décisions en se basant sur des données. Ces modèles sont entraînés à partir d'un ensemble de données appelé ensemble d'entraînement. L'idée est que, grâce à l'analyse des données passées, le modèle puisse être en mesure de généraliser et d'effectuer des prédictions précises sur des données qu'il n'a jamais vues auparavant.

Le processus de Machine Learning se décompose en plusieurs étapes clés :

- 1. Collecte de données** : La première étape consiste à rassembler les données nécessaires pour l'entraînement du modèle. La qualité et la quantité des données sont cruciales pour les performances du modèle.
- 2. Prétraitement des données** : Avant de pouvoir entraîner un modèle, les données doivent être nettoyées et préparées. Cela inclut la gestion des valeurs manquantes, la normalisation, la suppression des doublons et la transformation des données en un format adapté à l'algorithme.
- 3. Sélection du modèle** : En fonction de la tâche à accomplir (classification, régression, clustering, etc.), un ou plusieurs algorithmes de Machine Learning sont choisis pour entraîner le modèle.

4. Entraînement du modèle : Le modèle est entraîné à partir des données d'entraînement. L'algorithme ajuste les paramètres internes du modèle pour minimiser l'erreur entre les prédictions et les valeurs réelles.

5. Évaluation du modèle : Une fois entraîné, le modèle est testé sur un ensemble de données distinct, appelé **ensemble de test**, pour évaluer ses performances et sa capacité à généraliser sur des données nouvelles.

6. Optimisation : Si les performances du modèle ne sont pas satisfaisantes, des ajustements peuvent être faits, tels que la modification des hyperparamètres, l'ajout de nouvelles données ou la sélection d'un autre algorithme.

4.3 Types d'Apprentissage

Le Machine Learning se divise généralement en trois grandes catégories selon la manière dont l'algorithme apprend à partir des données :

- **Apprentissage supervisé** : Dans ce type d'apprentissage, l'algorithme apprend à partir de données étiquetées, c'est-à-dire des données pour lesquelles la réponse correcte est déjà connue. Il s'agit de la méthode la plus courante dans les applications pratiques, comme la classification (par exemple, l'analyse de sentiments) et la régression (par exemple, la prévision des ventes). L'algorithme apprend à prédire la sortie à partir des données d'entrée et des étiquettes associées.

- **Apprentissage non supervisé** : Ici, les données fournies ne sont pas étiquetées, et l'algorithme doit découvrir des modèles ou des relations dans les données par lui-même. Les tâches d'apprentissage non supervisé incluent le clustering (regroupement des données similaires), la réduction de dimension et la détection d'anomalies.

- **Apprentissage par renforcement** : Cette approche consiste à entraîner un modèle à prendre des décisions séquentielles. Un agent apprend en interagissant avec son environnement et en recevant des récompenses ou des punitions en fonction des actions entreprises. L'objectif est de maximiser les récompenses cumulées au fil du temps. L'apprentissage par renforcement est souvent utilisé dans des domaines comme la robotique et les jeux vidéo.

4.4 Modèles d'apprentissage automatique

4.4.1. Régression Logistique

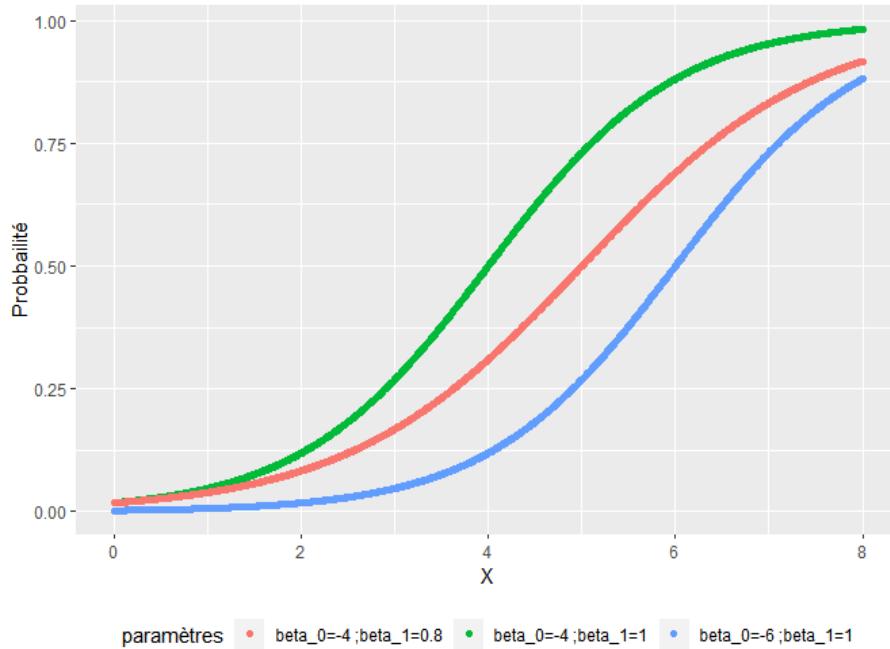


Figure 8: régression logistique

La régression logistique est un modèle statistique souvent utilisé pour les tâches de classification binaire. Contrairement à la régression linéaire, qui prédit une valeur continue, la régression logistique prédit la probabilité qu'une observation appartienne à une classe spécifique. Le modèle repose sur la fonction sigmoïde, qui convertit les valeurs en probabilités dans l'intervalle 0,10, 10,1.

Fonctionnement : La régression logistique cherche à établir une relation linéaire entre les variables explicatives (caractéristiques) et la variable cible (étiquette), puis applique la fonction sigmoïde pour calculer une probabilité. Si la probabilité dépasse un certain seuil (généralement 0,5), l'observation est classée dans une des deux classes (par exemple, positif/négatif).

Applications :

- Analyse des sentiments (positif/négatif/neutre).
- Filtrage de spam.
- Prédiction d'événements comme la défection des clients.

Avantage :

- Simple à comprendre et à implémenter.
- Très efficace pour les problèmes de classification binaire.
- Interprétable : les coefficients associés aux caractéristiques fournissent une indication sur l'importance de chaque variable.

Limite :

- Efficace seulement pour les classes linéairement séparables.

- Moins performant pour les problèmes complexes avec des relations non linéaires.

4.4.2. Support Vector Machines (SVM)

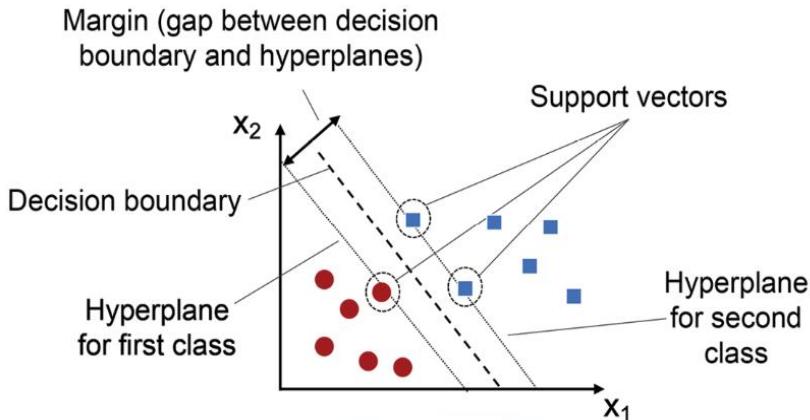


Figure 9 : SVM

Le Support Vector Machine (SVM) est un algorithme de classification supervisé qui cherche à trouver un hyperplan optimal séparant les classes dans un espace à plusieurs dimensions. Il est particulièrement adapté aux problèmes où les classes sont difficiles à séparer de manière linéaire.

Fonctionnement :

Le SVM fonctionne en identifiant l'hyperplan qui maximise la marge entre les points de données de chaque classe. Seuls les points de données les plus proches de la frontière de séparation (appelés vecteurs de support) sont utilisés pour construire cet hyperplan. En cas de non-linéarité, le SVM peut utiliser des kernels pour transformer les données et trouver une séparation dans un espace de dimension supérieure.

Applications :

- Classification d'images et de textes.
- Reconnaissance de formes.
- Bioinformatique (analyse des séquences d'ADN, prédition de la structure des protéines).

Avantage :

- Efficace pour les problèmes de classification complexe, même dans les espaces de grande dimension.
- Flexible grâce à l'utilisation des kernels pour résoudre des problèmes non linéaires.
- Robuste face à des données bruitées.

Limite :

- Plus complexe à comprendre et à paramétriser.
- Temps de calcul élevé pour les grands ensembles de données.
- Difficilement interprétable par rapport à des modèles plus simples comme la régression logistique.

4.4.3. Forêts Aléatoires (Random Forest)

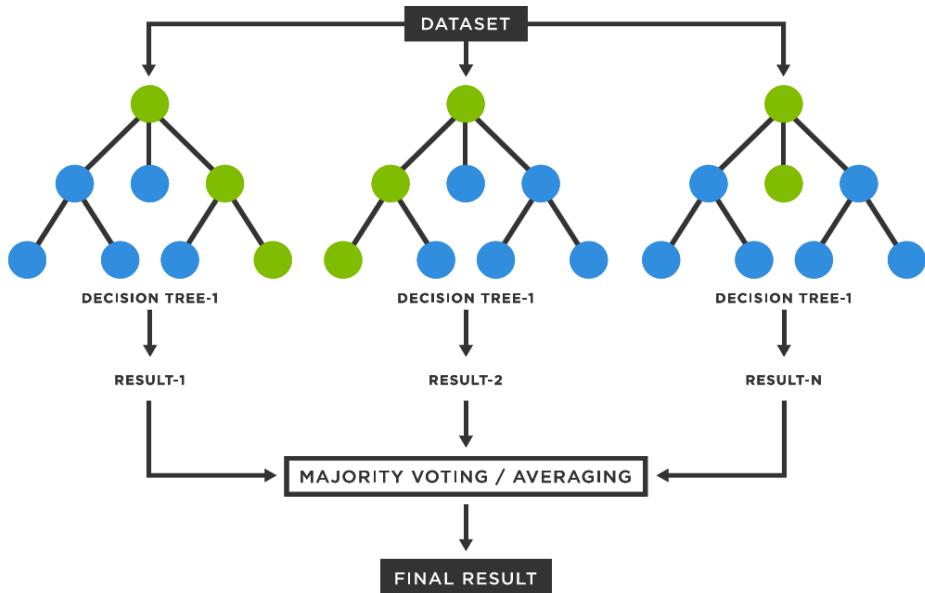


Figure 10 : Forêts Aléatoires

Le modèle de Forêts Aléatoires est un algorithme d'apprentissage supervisé basé sur un ensemble d'arbres de décision. Il combine plusieurs arbres de décision pour produire une prédition plus robuste et précise en réduisant la variance et le risque de surapprentissage (overfitting).

Fonctionnement

Chaque arbre de décision est entraîné sur un sous-échantillon aléatoire des données. La décision finale du modèle est obtenue en agrégeant les prédictions de chaque arbre (par vote majoritaire pour la classification ou par moyenne pour la régression). Cette approche permet de corriger les erreurs potentielles des arbres individuels en réduisant les corrélations entre eux.

Applications :

- Détection de fraude.
- Classification d'images.
- Analyse prédictive dans les domaines de la finance, de la santé, et du marketing.

Avantage :

- Performant pour les tâches de classification et de régression.
- Réduit le risque de surapprentissage par l'utilisation de plusieurs arbres.

- Peut gérer des données avec un grand nombre de caractéristiques sans avoir besoin de beaucoup de prétraitements.

Limite :

- Moins interprétable que les modèles individuels comme les arbres de décision.
- Plus coûteux en termes de calculs et de mémoire, surtout pour les grands ensembles de données.
- Moins performant pour les données rares (sparses) ou à haute dimension sans ajustements appropriés.

5 Evaluation des Modèles

Validation Croisée : Technique pour évaluer la performance des modèles en utilisant plusieurs sous-ensembles des données.

Métriques d'Évaluation

Précision, Rappel, F1-Score : Mesures pour évaluer la qualité des prédictions des modèles.

Conclusion

Ce chapitre devrait offrir une vue d'ensemble des techniques et méthodes existantes dans le domaine du traitement du langage naturel et de l'analyse des sentiments, ainsi que des techniques de machine learning utilisées pour ces tâches.

Chapitre III : Outils et Environnement

Introduction :

Dans ce chapitre, nous présenterons les différents outils et environnements utilisés pour la réalisation de ce projet. Ces outils jouent un rôle crucial dans le développement, l'implémentation et l'automatisation des tâches liées à l'analyse des sentiments et au traitement des données. Nous aborderons à la fois les logiciels, les bibliothèques et les plateformes choisies, ainsi que leurs caractéristiques spécifiques qui ont facilité la mise en œuvre des différentes étapes du projet.

1 Outils de développement

1.1 Google Colaboratory

Google Colab a été choisi comme environnement principal pour l'écriture et l'exécution des scripts Python. Sa puissance de calcul, notamment l'accès à des GPU, et son intégration facile avec les bibliothèques Python ont permis de traiter rapidement de grandes quantités de données textuelles.



Figure 11: logo de GoogleColaboratory

- **Avantages :** Collaboration en temps réel, ressources de calcul gratuites, accès direct à Google Drive pour stocker les données.
- **Utilisation :** Prétraitement des données, entraînement des modèles de machine learning, génération de nuages de mots et visualisation des résultats.

1.2 Visual Studio Code (VS Code)

VS Code a été utilisé comme environnement de développement intégré (IDE) pour le débogage et l'écriture des scripts locaux. Il a permis de gérer le code source de manière structurée avec des extensions facilitant le développement.

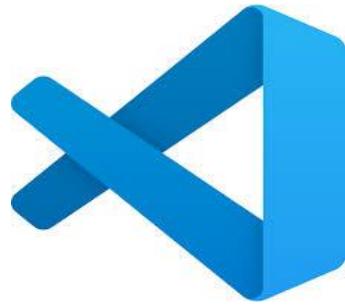


Figure 12: logo de Vs Code

- **Avantages :** Personnalisation, gestion de versions Git, extensions pour Python.
- **Utilisation :** Développement et modification des scripts Python pour le traitement des données.

1.3 Python

Python a été le langage principal utilisé pour le traitement des données et la construction des modèles d'analyse des sentiments.



Figure 13: logo de Python

- **Packages utilisés :**
 - NLTK et TextBlob pour le traitement du langage naturel.
 - Matplotlib et WordCloud pour la visualisation des résultats.
- **Utilisation :** Traitement des emails, nettoyage des données, construction des modèles d'analyse de sentiment et création de l'interface utilisateur.

2 Outils de gestion des données

2.1 Excel

Excel a été utilisé pour manipuler les données brutes collectées, comme les emails ou les commentaires, avant leur importation dans l'environnement Python pour l'analyse.



Figure 14: logo de Excel

- **Utilisation :** Prévisualisation des données, formatage des fichiers, gestion de bases de données simples avant traitement.

2.2 Esuit

Esuit a été utilisé pour gérer certaines tâches administratives et de reporting liées aux données et aux résultats du projet.

- **Utilisation :** Stockage et partage des fichiers, gestion des données et des rapports et extraction des commentaires.

3 Plateformes de collecte de données

3.1 Facebook

Facebook a servi de plateforme principale pour la collecte de données, notamment pour les commentaires et les avis des utilisateurs sur les marchés publics.



Figure 15: logo de Facebook

- **Utilisation :** Extraction des commentaires publics pour l'analyse des sentiments concernant les marchés publics.

3.2 Gmail

Gmail a été utilisé pour recueillir et analyser les emails envoyés à la Trésorerie Générale du Royaume, afin d'évaluer les retours et les réclamations des utilisateurs.



Figure 16 : logo de Gmail

- **Utilisation :** Extraction d'emails pour une analyse approfondie des sentiments.

Conclusion

L'ensemble de ces outils et plateformes a été essentiel pour réussir le projet. Google Colab et VS Code ont offert une flexibilité maximale pour le développement et l'exécution des scripts, tandis que Python a permis d'implémenter les techniques avancées de traitement du langage naturel et de machine learning. Les plateformes Facebook et Gmail ont constitué des sources précieuses de données textuelles pour l'analyse des sentiments, et les outils comme Excel et Esuit ont assuré une gestion efficace des données et des rapports.

Chapitre IV : Collecte et Prétraitement

Introduction

Dans cette section, nous aborderons les étapes de la collecte et du prétraitement des données utilisées pour l'analyse des sentiments. Ces deux étapes sont cruciales pour assurer la qualité et la pertinence des données avant de les soumettre aux modèles de machine learning.

1 Collecte des Données

La collecte de données est le processus de rassemblement d'informations nécessaires pour analyser, comprendre et résoudre des problèmes ou des questions spécifiques. Elle est cruciale pour effectuer des recherches, évaluer des situations et prendre des décisions éclairées.

1.1. Source de données

La collecte des données est une étape clé pour tout projet de machine learning, en particulier pour l'analyse de texte. Dans ce projet, les données sont principalement des emails en français issus des échanges entre les clients de la Trésorerie Générale du Royaume (TGR) et les techniciens, qu'ils soient techniques ou métiers. Ces emails offrent une diversité d'informations pertinentes pour l'analyse de sentiments.

1.2. Description de la dataset :

Les données collectées étaient principalement sous forme de texte brut, stockées dans un fichier Excel. Ce jeu de données comprend 19 colonnes, dont une colonne intitulée Corps qui contient le texte des emails. Il est composé de 5174 lignes, chacune représentant un email. Ces données sont collectées entre Janvier et Avril 2024.

2 Prétraitement de données

Le prétraitement des données est une étape cruciale dans tout projet d'analyse ou de traitement de données. Il consiste à préparer les données brutes pour qu'elles soient utilisables par des modèles ou des algorithmes de traitement, notamment dans des projets d'apprentissage automatique ou de traitement du langage naturel.

2.1 Préparation des Données

Cette partie permet de préparer le dataset avant le nettoyage et le prétraitement.

➤ Importation de packages nécessaire

Dans le cadre de mon projet, plusieurs bibliothèques ont été installées pour le prétraitement et l'analyse de données textuelles :

```
!pip install nltk
!pip install textblob
!pip install pyspellchecker
!pip install wordcloud
```

Figure 17 : packages

- **NLTK** : Utilisée pour des tâches comme la tokenisation, le lemmatisation et l'analyse syntaxique, elle est essentielle pour manipuler et préparer les textes.
- **TextBlob** : Simplifie l'analyse de sentiments, la traduction, et la correction grammaticale, facilitant ainsi le traitement rapide de texte.

- **pyspellchecker** : Fournit un correcteur orthographique automatique pour améliorer la qualité des textes avant analyse.

- **WordCloud** : Génère des nuages de mots pour visualiser la fréquence des termes dans un texte, utile pour l'analyse exploratoire.

➤ Importation de bibliothèques

Ce code importe plusieurs bibliothèques Python utilisées pour le traitement du langage naturel (NLP) et pour des tâches d'apprentissage automatique.

```
from warnings import filterwarnings
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import nltk
import re
from nltk.sentiment import SentimentIntensityAnalyzer
from bs4 import BeautifulSoup
from spellchecker import SpellChecker
from collections import Counter
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, GridSearchCV, cross_validate
from sklearn.preprocessing import LabelEncoder
from textblob import Word, TextBlob
from wordcloud import WordCloud, STOPWORDS
import matplotlib.patches as patches
from nltk.corpus import words
from google.colab import files
import io
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.svm import SVC
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from nltk import pos_tag
from nltk.tokenize import word_tokenize
```

Figure 18 : bibliothèques utilisées

- **nltk** : Permet de travailler sur le traitement du langage naturel, avec des outils comme l'analyse des sentiments, la tokenisation des mots et l'étiquetage des parties du discours.

- **SentimentIntensityAnalyzer** : Utilisé pour analyser les sentiments dans les textes.
- **pos_tag, word_tokenize** : Pour le traitement grammatical et la segmentation des phrases en mots.
- **BeautifulSoup** : Sert à analyser et extraire des données à partir de fichiers HTML et XML, souvent utilisé pour nettoyer du texte après le scraping web.

- **SpellChecker** : Permet de corriger les fautes d'orthographe dans les textes.
- **RandomForestClassifier, LogisticRegression, SVC** : Ce sont des algorithmes d'apprentissage automatique pour la classification de données.
- **CountVectorizer, TfidfVectorizer** : Techniques pour convertir des documents textuels en vecteurs de mots, utilisés pour transformer les textes en données exploitables pour l'algorithme.
- **WordCloud** : Outil de visualisation qui permet de générer des nuages de mots à partir d'un corpus textuel.
- **GridSearchCV, cross_val_score** : Méthodes pour optimiser les hyperparamètres et évaluer les modèles par validation croisée.
- **Matplotlib (plt)** : Bibliothèque utilisée pour générer des graphiques et visualisations.

➤ Les ressources nécessaires pour travailler avec NTLK

```
filterwarnings('ignore')
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 200)
pd.set_option('display.float_format', lambda x: '%.2f' % x)
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('words')
```

Figure 19 : ressources nécessaires de NTLK

- **filterwarnings('ignore')** : Cette commande désactive les avertissements dans le code, permettant d'éviter d'afficher des messages d'avertissement qui peuvent être gênants mais qui n'affectent pas l'exécution du programme.
- **pd.set_option('display.max_columns', None)** : Permet d'afficher toutes les colonnes des DataFrames Pandas, ce qui est utile lorsque tu travailles avec de grands ensembles de données et que tu souhaites tout visualiser sans restriction.
- **pd.set_option('display.width', 200)** : Ajuste la largeur d'affichage des DataFrames pour qu'ils s'affichent correctement sur des lignes plus larges dans la console, ici jusqu'à 200 caractères.
- **pd.set_option('display.float_format', lambda x: '%.2f' % x)** : Change le format d'affichage des nombres à virgule flottante pour qu'ils soient arrondis à deux décimales, rendant les données numériques plus lisibles.
- **nltk.download('punkt')** : Télécharge le package punkt de nltk, qui contient un outil de tokenisation pour diviser les textes en phrases ou en mots.

- **nltk.download('wordnet')** : Télécharge WordNet, une base de données lexicale anglaise utilisée pour la lemmatisation, qui permet de transformer des mots en leur forme de base.
- **nltk.download('omw-1.4')** : Télécharge des ressources supplémentaires pour WordNet, notamment des traductions dans plusieurs langues (Open Multilingual WordNet).
- La commande `nltk.download('words')` télécharge le corpus "words" depuis la bibliothèque NLTK (Natural Language Toolkit).
Le corpus "words" contient une vaste collection de mots en anglais. Il est utilisé pour diverses tâches liées au traitement du langage naturel, comme la vérification de l'orthographe, la correction automatique, ou la validation des mots lors de l'analyse textuelle.

➤ Vérification de la sécurité des bibliothèques

Ces commandes sont utiles pour s'assurer que les dépendances utilisées dans ce projet sont sécurisées. Cela peut être particulièrement important si tu travailles avec des données sensibles ou si ton projet est destiné à être déployé en production. L'audit permet de garantir une meilleure sécurité du code et des packages.

```
## Vérification de la Sécurité des Bibliothèques
!pip install pip-audit
!pip-audit
```

Figure 20 : sécurité de bibliothèques

Une fois les données collectées, elles ont été préparées pour être traitées par des algorithmes de machine learning. Cette préparation inclut :

➤ L'importation de données :

```
df = pd.read_excel("emails.xls")
df.head()
```

Figure 21 : importation de données

| index | Id | Corps | Denom | Adresse | Score | Anom | Adresse | Atype | Ccnom | Cciadresse | Cctype |
|-------|-----|---|------------------------------|------------------------------|-------|-----------------|--|-------|-------------------|----------------|--------|
| 0 | 1.0 | Bonjour, Je me permets de vous adresser ce courriel pour exprimer notre préoccupation quant au retard dans le traitement de notre demande de main levée, soumise il y a maintenant 1 mois, concernant l'appel d'offre N°43/2023/ANDA avec objet : 'La c | N7-SALES | commercial@n7.ma | 5 | Marches Publics | /O=TGRNETORG/OU=Premier groupe d'administration/cn=Recipients/cn=m_publics | EX | 'Narjiss LOUDIYI' | nloudiyi@n7.ma | SMTP |
| 1 | 2.0 | The message you sent requires that you verify that you are a real live human being and not a spam source. To complete this verification, simply reply to this message and leave the subject line intact. Or, you can click the following link: ht | contact@bio-eco-solutions.ma | contact@bio-eco-solutions.ma | 2 | Marches Publics | /O=TGRNETORG/OU=Premier groupe d'administration/cn=Recipients/cn=m_publics | EX | NaN | NaN | NaN |

Figure 22 : dataset partie 1

| Ccinom | Cciadresse | Cctype | Catégories | Critère de diffusion | Importance | Informations facturation | Kilométrage |
|--------|------------|--------|------------|----------------------|------------|--------------------------|-------------|
| NaN | NaN | NaN | NaN | Normal(e) | Normal(e) | NaN | NaN |
| NaN | NaN | NaN | NaN | Normal(e) | Normal(e) | NaN | NaN |

Figure 23 : dataset partie 2

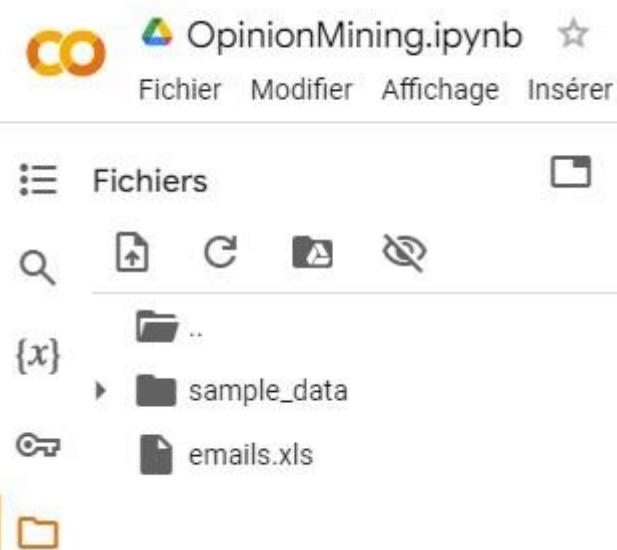


Figure 24 : dataset après importation

2.2. Nettoyage de données

C'est une étape essentielle dans tout processus d'analyse de données. Il consiste à préparer les données brutes en éliminant ou en corrigeant les erreurs, les incohérences et les valeurs manquantes afin de garantir la qualité et la fiabilité des analyses ou des modèles d'apprentissage automatique.

```
▶ columns_to_drop = [
    'Id', 'Denom', 'Deadresse', 'Score', 'Anom', 'Aadresse', 'Atype',
    'Ccnom', 'Ccadresse', 'CcType', 'Ccinom', 'Cciadresse', 'Ccctype',
    'Catégories', 'Critèredediffusion', 'Importance', 'Informationsfacturation',
    'Kilométrage'
]

df = df.drop(columns=columns_to_drop)
df.head()
```

Figure 25 : suppression des colonnes non significatif

```
## Conversion en Minuscules
df['Corps'] = df['Corps'].str.lower()

## Suppression de la Ponctuation
df['Corps'] = df['Corps'].str.replace('[^\w\s]', '', regex=True)

## Suppression des Chiffres
df['Corps'] = df['Corps'].str.replace('\d', '', regex=True)
```

Figure 26 : compression en majuscule, suppression des chiffres et de ponctuation

Ce code permet d'éliminer les mots en anglais et en arabe, en ne conservant que les termes en français.

```
##supprimer les mots en anglais et en arabe

english_words = set(words.words())

# Charger les mots français à partir du fichier téléchargé

uploaded = files.upload()

# Remplacez 'french_words.txt' par le nom de votre fichier téléchargé
with io.open('french_words.txt', 'r', encoding='utf-8') as file:
    french_words = set(word.strip().lower() for word in file)

# Fonction pour garder uniquement les mots en français et supprimer les mots en arabe
def keep_french_words(text):
    if isinstance(text, str): # Check if text is a string
        # Supprimer les mots en arabe
        text = re.sub(r'[\u0600-\u06FF]+', '', text)
        return ' '.join(word for word in text.split() if word in french_words and word not in english_words)
    else:
        return '' # Return an empty string if not a string

# Appliquer la fonction à la colonne 'Corps'
df['Corps'] = df['Corps'].apply(keep_french_words)

# Appliquer la fonction à la colonne 'Corps'
df['Corps'] = df['Corps'].apply(keep_french_words)

# Supprimer les caractères spéciaux
def remove_spl_chars(text):
    # Remplacer les caractères spéciaux par des espaces
    text = re.sub('[^a-zA-Z0-9]', ' ', text)
    # Remplacer les occurrences de 'NaN' par des espaces
    text = re.sub(r'\bNaN\b', ' ', text)
    text = re.sub('\st+', ' ', text).strip()
    return text

# Appliquer la fonction pour supprimer les caractères spéciaux
df['Corps'] = df['Corps'].apply(remove_spl_chars)
```

Figure 27 : code pour ne garder que la langue française

Bouton pour charger le fichier contenant tous les mots anglais

Aucun fichier choisi

Figure 28 : bouton de téléchargement de fichier

Le téléchargement de fichier a réussi

```
Sélectionnez des fichiers french_words.txt
• french_words.txt(text/plain) - 228263 bytes, last modified: 01/08/2024 - 100% done
Saving french_words.txt to french_words.txt
```

Figure 29 : la réussite de téléchargement

Le résultat de la suppression des mots en anglais et en arabe

| Corps |
|---|
| 0 bonjour je vous adresser exprimer notre au dan... |
| 1 |
| 2 bonjour quelle est avis infructueux que avons ... |
| 3 |
| 4 |

Figure 30 : résultat de suppression des autres langues

Fonction pour supprimer les caractères spéciaux et gérer les occurrences de "NaN", remplace les caractères non alphabétiques ou numériques par des espaces et nettoie les espaces multiples et les chaînes "NaN"

```
#Supprimer les caractères speciaux

def remove_spl_chars(Corps):
    # Remplacer les caractères spéciaux par des espaces
    text = re.sub('[^a-zA-Z0-9]', ' ', Corps)
    # Remplacer les occurrences de 'NaN' par des espaces
    text = re.sub(r'\bNaN\b', ' ', text)
    text = re.sub('\s+', ' ', text).strip()
    return text
```

Figure 31 : suppression des caractères spéciaux

Ce code remplace certaines chaînes spécifiques indésirables telles que "\r\n", "X000D", "NaN", et "Forwarded" par des espaces dans la colonne 'Corps'. Cela permet de nettoyer le texte et d'éliminer les éléments non pertinents pour l'analyse.

```
# Supprimer toutes les occurrences de "\r\n"
df['Corps'] = df['Corps'].str.replace("\r\n", " ")
# Supprimer toutes les occurrences de "X000D"
df['Corps'] = df['Corps'].str.replace("X000D", " ")
# Supprimer toutes les occurrences de "NaN"
df['Corps'] = df['Corps'].str.replace("NaN", " ")
# Supprimer toutes les occurrences de "Forwarded"
df['Corps'] = df['Corps'].str.replace("Forwarded", " ")
```

Figure 32 : suppression de quelques occurrences

Ce code utilise la bibliothèque nltk pour télécharger et appliquer une liste de mots vides (stopwords) en français. Il supprime ces mots non significatifs de la colonne 'Corps', améliorant ainsi la qualité du texte pour l'analyse.

```
# Téléchargement des Stopwords
nltk.download('stopwords')
sw = stopwords.words('french')

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

# Suppression des Stopwords
df['Corps'] = df['Corps'].apply(lambda x: " ".join(x for x in str(x).split() if x not in sw))
```

Figure 33 : suppression de stopwords

Ce code supprime les URLs dans les textes de la colonne 'Corps' en utilisant une expression régulière, puis élimine les lignes vides après le nettoyage. Cela garantit que le texte reste pertinent pour l'analyse, sans liens externes ni lignes vides inutiles.

```
##supprimer les urls

# Fonction pour supprimer les URLs
def remove_urls(text):
    return re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)

# Appliquer la fonction pour supprimer les URLs
df['Corps'] = df['Corps'].apply(remove_urls)

# Supprimer les lignes vides
df = df[df['Corps'].str.strip().astype(bool)]
```

Figure 34 : suppression des URLs

Ce code supprime les balises HTML des textes dans la colonne 'Corps' en utilisant BeautifulSoup. Cela permet de nettoyer le texte en éliminant les balises de formatage HTML et en ne conservant que le contenu textuel.

```
##supprimer les balises html

# Fonction pour supprimer les balises HTML
def remove_html_tags(text):
    return BeautifulSoup(text, "html.parser").get_text()

# Appliquer la fonction pour supprimer les balises HTML
df['Corps'] = df['Corps'].apply(remove_html_tags)
```

Figure 35 : suppression de balises HTML

Ce code supprime les mots spécifiques, tels que "bonjour", "mar", "demande", et

"directeur", de la colonne 'Corps'. La fonction remove_specific_words est appliquée pour filtrer ces mots dans chaque texte, afin de nettoyer le contenu en supprimant les termes jugés non pertinents.

```
# Suppression des mots spécifiques les plus fréquents

# Liste des mots à supprimer
mots_a_supprimer = ["bonjour", "mar", "demande", "directeur"]

# Fonction pour supprimer les mots spécifiques
def remove_specific_words(text, words_to_remove):
    return ' '.join([word for word in text.split() if word not in words_to_remove])

# Appliquer la fonction pour supprimer les mots spécifiques dans la colonne 'Corps'
df['Corps'] = df['Corps'].apply(lambda x: remove_specific_words(x, mots_a_supprimer))
```

Figure 36 : suppression de mots spécifiques

Ce code compte la fréquence des mots dans la colonne 'Corps' et identifie les mots qui apparaissent exactement 1, 2, 3, et 4 fois. Les listes de ces mots rares sont ensuite affichées, permettant de voir les termes peu fréquents dans le texte.

```
# Compter la fréquence des mots dans la colonne 'Corps'
word_counts = Counter(" ".join(df['Corps']).split())

# Obtenir les mots qui apparaissent exactement 1, 2, 3 et 4 fois
rare_words_once = [word for word, count in word_counts.items() if count == 1]
rare_words_twice = [word for word, count in word_counts.items() if count == 2]
rare_words_three = [word for word, count in word_counts.items() if count == 3]
rare_words_four = [word for word, count in word_counts.items() if count == 4]

# Afficher la liste des mots rares qui apparaissent une, deux, trois et quatre fois
print("Mots apparaissant une seule fois :", rare_words_once)
print("Mots apparaissant deux fois :", rare_words_twice)
print("Mots apparaissant trois fois :", rare_words_three)
print("Mots apparaissant quatre fois :", rare_words_four)

Mots apparaissant une seule fois : ['laisstant', 'vides', 'instituteurs', 'scenarios', 'conteneurs', 'developpement', 'sociaux', 'confondus', 'explique', 'commercialisation', 'semences', 'auditeur', 'informe
Mots apparaissant deux fois : ['continental', 'rues', 'magasin', 'filiale', 'logements', 'aides', 'durcissement', 'fonctionnent', 'actuelle', 'categorie', 'clairage', 'vigueur', 'infirmiers', 'poursuivis',
Mots apparaissant trois fois : ['mettons', 'sas', 'presque', 'ambitieux', 'cannabis', 'admis', 'carburants', 'etablissements', 'entrer', 'viande', 'fiches', 'fonction', 'attachement', 'renancement', 'retrou
Mots apparaissant quatre fois : ['infructueux', 'croire', 'superieure', 'rubriques', 'calcul', 'astronomique', 'pense', 'sante', 'vais', 'jeunesse', 'modifications', 'techniciens', 'attaques', 'verts', 'bra
```

Figure 37 : fréquence des mots

Fonction de leur fréquence. La fonction remove_rare_words supprime les mots apparaissant un nombre de fois inférieur ou égal à un seuil spécifié (threshold). Le code applique cette fonction avec plusieurs seuils (1, 2, 3, 4) pour nettoyer la colonne 'Corps' et affiche les premières lignes du DataFrame après le nettoyage.

```

# Fonction générique pour supprimer les mots apparaissant n fois ou moins
def remove_rare_words(df, column, threshold):
    # Compter la fréquence des mots dans la colonne spécifiée
    word_counts = Counter(" ".join(df[column]).split())

    # Obtenir la liste des mots qui apparaissent 'threshold' fois ou moins
    rare_words = [word for word, count in word_counts.items() if count <= threshold]

    # Fonction pour supprimer les mots rares du texte
    def remove_words(text):
        return " ".join([word for word in text.split() if word not in rare_words])

    # Appliquer la fonction de suppression sur la colonne
    df[column] = df[column].apply(remove_words)

    return df

# Exemple d'utilisation du code pour la suppression
df = remove_rare_words(df, 'Corps', threshold=1) # Pour supprimer les mots apparaissant 1 fois
df = remove_rare_words(df, 'Corps', threshold=2) # Pour supprimer les mots apparaissant 2 fois ou moins
df = remove_rare_words(df, 'Corps', threshold=3) # Pour supprimer les mots apparaissant 3 fois ou moins
df = remove_rare_words(df, 'Corps', threshold=4) # Pour supprimer les mots apparaissant 4 fois ou moins

# Afficher les premières lignes du DataFrame après suppression
print(df.head())

```

Figure 38 : suppression des mots moins fréquents

Cette ligne filtre le DataFrame df en supprimant les lignes où la colonne 'Corps' est vide après suppression des espaces blancs. Cela assure que seules les lignes contenant du texte pertinent restent dans le DataFrame.

```

df = df[df['Corps'].str.strip().astype(bool)]
df

```

Figure 39 : suppression de lignes vides

Ce code utilise SpellChecker pour corriger les erreurs orthographiques dans une liste de mots en français. Il identifie les mots mal orthographiés dans la liste misspelled et affiche les corrections proposées pour chaque mot.

```
#correction orthographique

spell = SpellChecker(language='fr')
misspelled = ["formul ire", "inform tion", "rvenir", "dem nde", "soci ux", " pporter", "r pidement"]
misspelled = spell.unknown(misspelled)
for word in misspelled:
    print(word, spell.correction(word))

soci ux sociaux
formul ire formulaire
 pporter apporter
inform tion information
rvenir venir
r pidement rapidement
dem nde demande
```

Figure 40 : correction orthographique

Cette ligne filtre le DataFrame df pour supprimer les lignes vides dans la colonne 'Corps'. Après avoir supprimé les espaces blancs, seules les lignes contenant du texte non vide sont conservées.

```
##supprimer les lignes vides
df = df[df['Corps'].str.strip().astype(bool)]
df
```

| | Corps | |
|------|---|--|
| 0 | bonjour adresser exprimer traitement demande s... | |
| 2 | bonjour quelle avis infructueux toutes salutat... | |
| 5 | bonjour allez fais parvenir documents remercie | |
| 6 | bonjour honneur parvenir formulaire compte rel... | |
| 7 | bonjour utilisant trouver | |
| ... | ... | |
| 5168 | bonjour trouver demande profil bonne | |
| 5169 | bonjour contacter sujet concernant nouveau int... | |
| 5170 | compte erreur nom toutes autres informations | |
| 5171 | correctement email jeudi mars entre maroc wash... | |

Figure 41: suppression des lignes vides

2.3 Visualisation

Ce code ajoute une colonne 'longueur' au DataFrame df pour stocker la longueur des emails dans la colonne 'Corps'. Il trace ensuite un histogramme montrant la distribution de la longueur des emails, avec des barres en bleu et des bordures noires, pour visualiser la fréquence des longueurs d'emails.

```

# Ajouter une colonne pour la longueur des emails
df['longueur'] = df['Corps'].apply(len)

# Tracer l'histogramme de la longueur des emails
plt.figure(figsize=(10, 6))
plt.hist(df['longueur'], bins=30, color='blue', edgecolor='black')
plt.xlabel('Longueur des emails')
plt.ylabel('Fréquence')
plt.title('Distribution de la Longueur des Emails')
plt.show()

```

Figure 42: longueur des emails

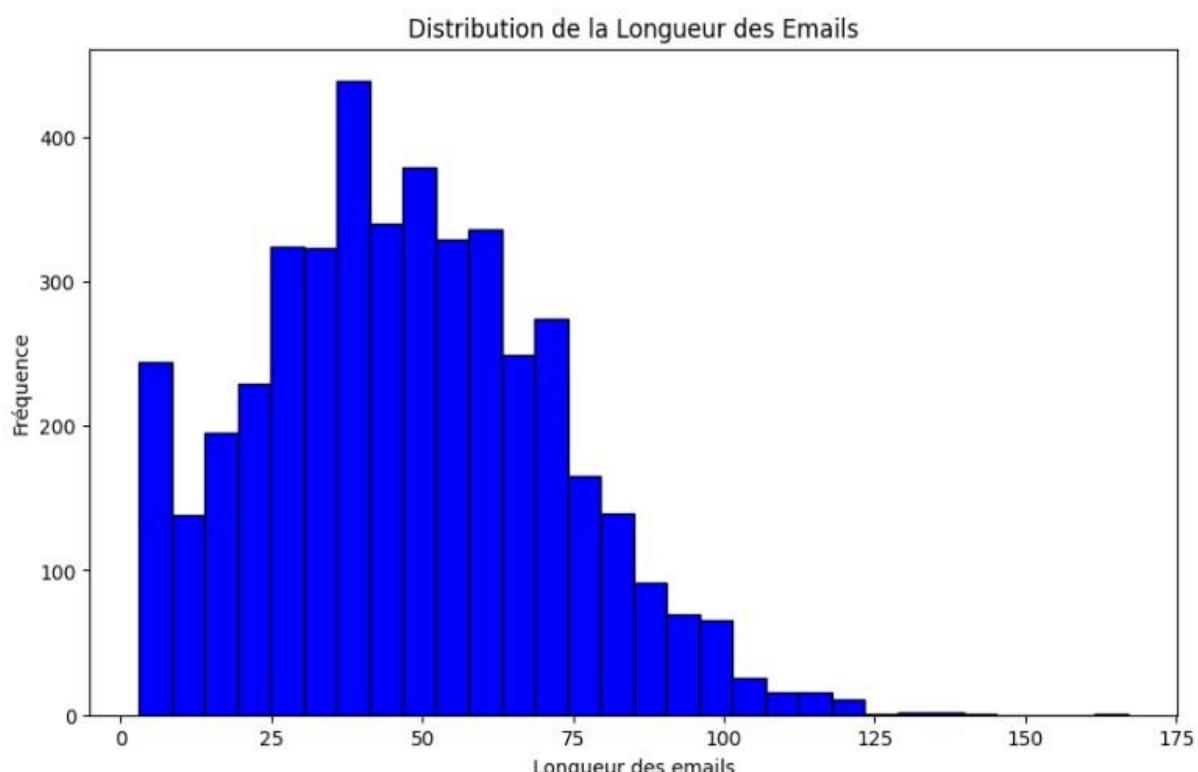


Figure 43 : graphe de longueur des emails

Ce code affiche les 7 mots les plus fréquents dans la colonne 'Corps' du DataFrame. Il commence par tokeniser les mots, compte leurs occurrences à l'aide de `Counter`, puis extrait les 7 mots les plus fréquents. Enfin, il crée un graphique en barres pour visualiser ces mots et leurs fréquences.

```

#Affichage des mots plus fréquents

import matplotlib.pyplot as plt
from collections import Counter

# Tokenisation des mots pour chaque texte dans votre DataFrame
df['mots'] = df['Corps'].apply(lambda x: x.split())

# Compter les occurrences de chaque mot dans l'ensemble du corpus
mots_frequents = Counter([mot for mots in df['mots'] for mot in mots])

# Extraire les 7 mots les plus fréquents
top_7_mots = mots_frequents.most_common(7)

# Séparer les mots et leurs fréquences
mots, frequences = zip(*top_7_mots)

# Création du graphique en barres
plt.figure(figsize=(10, 6))
plt.bar(mots, frequences, color='skyblue')
plt.xlabel('Mots')
plt.ylabel('Fréquence')
plt.title('Top 7 des Mots les Plus Fréquents')
plt.show()

```

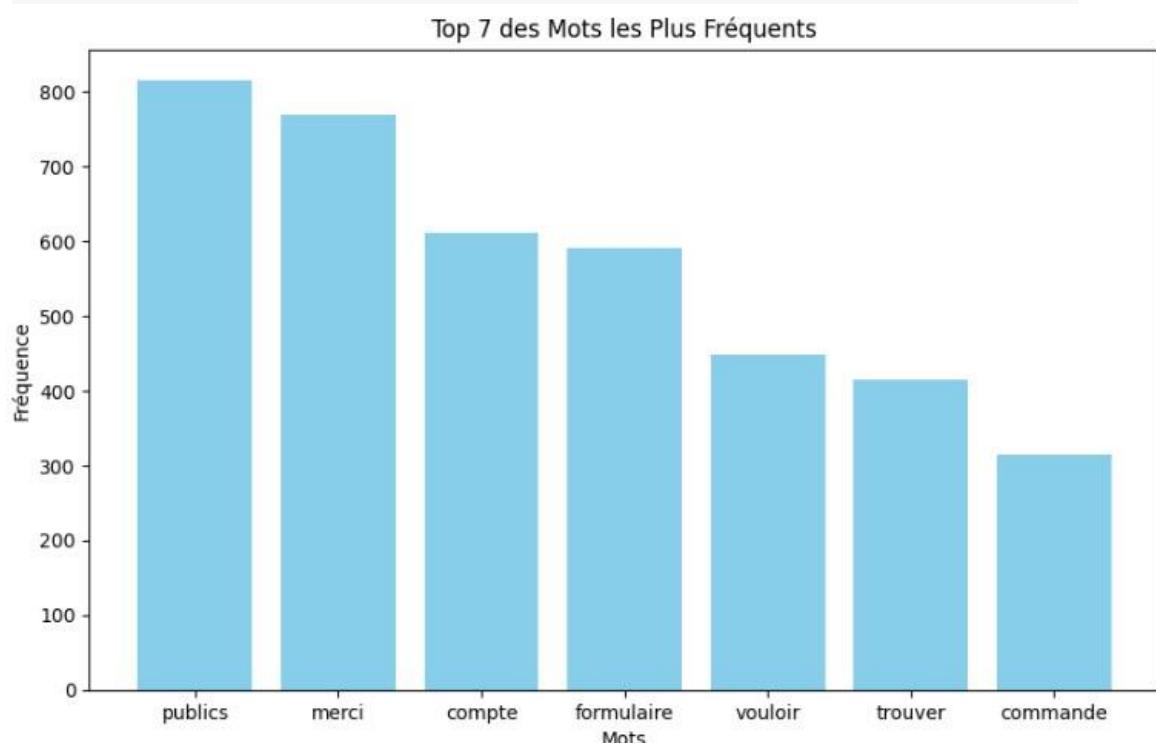


Figure 44 ; graph des mots les plus fréquents

Ce code calcule la fréquence des mots dans la colonne 'Corps' et affiche les **7** mots les moins fréquents. Les fréquences sont calculées, triées par ordre croissant, et les mots les moins fréquents sont visualisés à l'aide d'un graphique en barres

```
#Afficher les 7 mots les moins fréquents

import matplotlib.pyplot as plt
import pandas as pd

# Calcul des fréquences des mots
tf = df["Corps"].apply(lambda x: pd.value_counts(x.split(" "))).sum(axis=0).reset_index()
tf.columns = ['words', 'tf']

# Trier par fréquence croissante
tf_sorted = tf.sort_values(by='tf', ascending=True)

# Filtrer pour obtenir les 7 mots les moins fréquents
low_freq_words = tf_sorted.head(7)

# Affichage des mots les moins fréquents
plt.figure(figsize=(10, 6))
plt.bar(low_freq_words['words'], low_freq_words['tf'], color='orange')
plt.xlabel('Mots')
plt.ylabel('Fréquence')
plt.title('7 Mots les Moins Fréquents')
plt.xticks(rotation=45, ha='right') # Rotation des labels pour une meilleure lisibilité
plt.show()
```

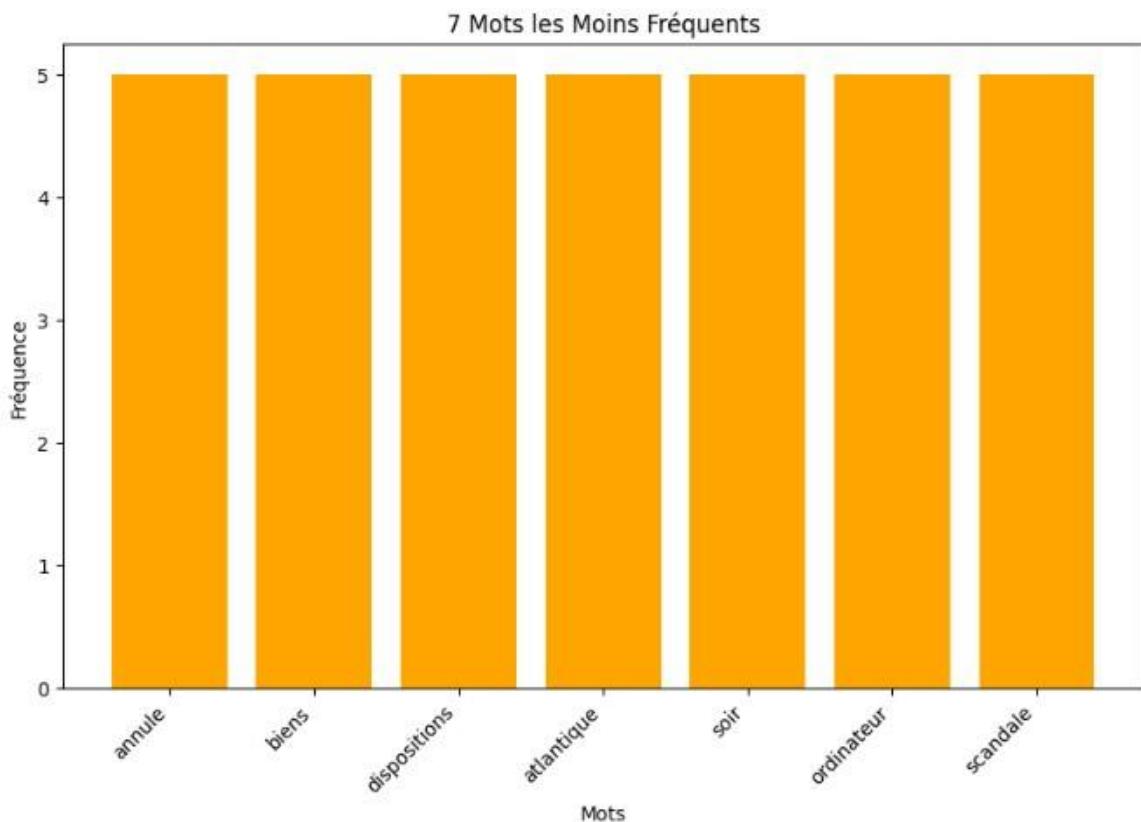


Figure 45 : graphe des mots moins fréquents

Ce code supprime les lignes contenant des valeurs manquantes et remplace les valeurs restantes manquantes par une chaîne vide dans le DataFrame df. Cela permet de nettoyer le DataFrame en éliminant les données incomplètes et en assurant la cohérence des valeurs.

```
# Supprimer les lignes contenant des valeurs manquantes
df = df.dropna()

# Remplacer les valeurs manquantes par une chaîne vide
df = df.fillna('')
```

Figure 46 : suppression des lignes contenant des valeurs manquantes

Ce code applique la lemmatisation aux mots dans la colonne 'Corps' du DataFrame df. Chaque mot est réduit à sa forme de base en utilisant TextBlob, ce qui normalise le texte en simplifiant les variantes des mots.

```
# Application de la Lemmatisation
df['Corps'] = df['Corps'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
```

Figure 47: lemmatisation

➤ Nuage de Mots

Ce code génère un nuage de mots à partir du texte de la colonne 'Corps' du DataFrame df. Le nuage est configuré avec un fond blanc, une palette de couleurs 'viridis', et affiche jusqu'à 200 mots. Il est entouré d'un contour vert pour améliorer la visibilité et est affiché avec un titre.

```
# Génération du Texte pour le Nuage de Mots
text = " ".join(i for i in df.Corps)

# Génération et Affichage du Nuage de Mots
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# Configuration du nuage de mots
wordcloud = WordCloud(
    width=800,
    height=400,
    background_color='white', # Fond blanc pour plus de clarté
    colormap='viridis', # Palette de couleurs contrastante
    max_words=200, # Limite le nombre de mots affichés
    stopwords=STOPWORDS, # Suppression des stopwords par défaut
    contour_color='steelblue', # Contour pour la clarté
    contour_width=1 # Épaisseur du contour
).generate(text)

# Affichage du nuage de mots avec un cadre vert
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')

# Ajouter un cadre vert
rect = patches.Rectangle((0, 0), 800, 400, linewidth=2, edgecolor='green', facecolor='none')
plt.gca().add_patch(rect)

plt.title('Nuage de Mots des Emails', fontsize=20)
plt.show()
```

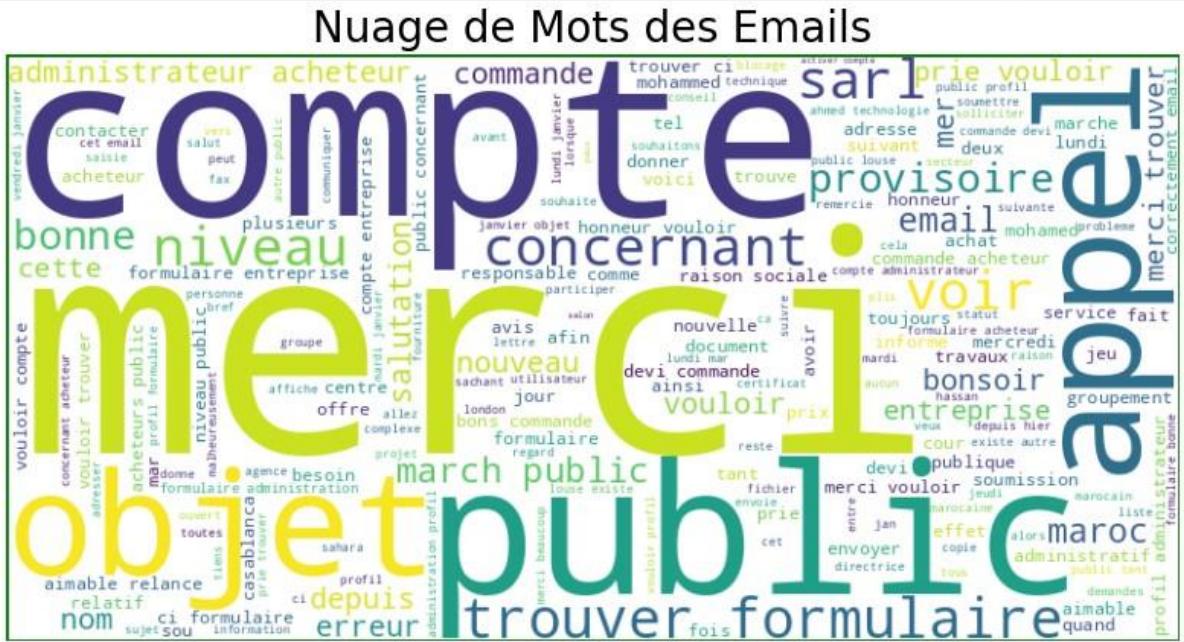


Figure 48 : nuage de mots

3 Etiquetage de mots

Ce code étiquette les mots dans la colonne 'Corps' du DataFrame df en utilisant l'étiquetage de parties du discours (POS tagging). La fonction pos_tagging tokenize le texte et attribue des étiquettes grammaticales à chaque mot. Les résultats sont stockés dans une nouvelle colonne 'Etiquettes' et les premières lignes du DataFrame avec les étiquettes sont affichées.

```
#Etiquetage

# Fonction pour étiqueter les mots dans un texte
def pos_tagging(text):
    if isinstance(text, str): # Vérifier si le texte est une chaîne de caractères
        tokens = word_tokenize(text) # Tokenisation du texte
        # Removing the lang argument as it defaults to english
        tagged_words = pos_tag(tokens) # Étiquetage des mots
        return tagged_words
    else:
        return []

# Appliquer l'étiquetage à la colonne 'Corps'
df['Etiquettes'] = df['Corps'].apply(pos_tagging)

# Afficher les premières lignes du dataframe avec les étiquettes
print(df[['Corps', 'Etiquettes']].head())
```

Figure 49 : code pour l'étiquetage des mots

Le DataFrame affiche la colonne 'Corps' avec le texte d'origine et la colonne 'Etiquettes' contenant les résultats de l'étiquetage grammatical. Chaque mot est associé à une étiquette de partie du discours, comme RB (adverbe), JJ (adjectif), NN (nom commun), et NNS (nom au pluriel). Ces étiquettes permettent d'analyser la structure grammaticale des textes.

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
          Corps                                Etiquettes
0  adresser exprimer traitement soumise maintenan... [(adresser, RB), (exprimer, JJ), (traitement, ...]
1                      quelle avis toutes salutation [(quelle, RB), (avis, JJ), (toutes, NNS), (sal...]
2  allez fais parvenir document remercie [(allez, NN), (fais, NN), (parvenir, NN), (doc...]
3  honneur parvenir formulaire compte relatif depuis [(honneur, NN), (parvenir, NN), (formulaire, N...]
4  utilisant trouver [(utilisant, JJ), (trouver, NN)]
```

Figure 50 : résultat d'étiquetage

4 Analyse de Sentiments

Ce code initialise le SentimentIntensityAnalyzer de la bibliothèque NLTK. Cet outil est utilisé pour analyser les sentiments dans le texte, en attribuant un score de sentiment aux

phrases ou aux documents.

```
# Initialisation du SentimentIntensityAnalyzer  
sia = SentimentIntensityAnalyzer()
```

Ce code calcule les scores de polarité pour chaque texte dans la colonne 'Corps' du DataFrame df. Il utilise le SentimentIntensityAnalyzer pour attribuer un score de sentiment global (compound) à chaque texte, reflétant la polarité générale des sentiments exprimés.

```
# Calcul des Scores de Polarité  
df["polarity_score"] = df["Corps"].apply(lambda x: sia.polarity_scores(x)["compound"])
```

Figure 51 : calcul de polarité

Ce code attribue des étiquettes de sentiment aux textes de la colonne 'Corps' du DataFrame df. En utilisant les scores de polarité, il classe chaque texte comme "pos" (positif), "neg" (négatif) ou "neu" (neutre) en fonction de la valeur du score de sentiment compound.

```
# Attribution des Labels de Sentiment  
df["sentiment_label"] = df["Corps"].apply(lambda x: "pos" if sia.polarity_scores(x)["compound"] > 0.05 else ("neg" if sia.polarity_scores(x)["compound"] < -0.05 else "neu"))
```

Figure 52 : attribution des labels de sentiments

Ce code encode les étiquettes de sentiment dans la colonne 'sentiment_label' du DataFrame df. Il utilise LabelEncoder pour transformer les étiquettes de texte (positif, négatif, neutre) en valeurs numériques, facilitant ainsi leur utilisation dans des modèles d'apprentissage automatique.

```
# Encodage des Labels de Sentiment  
df["sentiment_label"] = LabelEncoder().fit_transform(df["sentiment_label"])
```

Figure 53 : encodage des étiquettes de sentiments

5 Séparation de données

Ce code prépare les données textuelles pour l'apprentissage automatique. Il sépare les données en ensembles d'entraînement et de test, puis utilise TfidfVectorizer pour convertir les textes en matrices de caractéristiques numériques. Les dimensions des matrices résultantes pour les ensembles d'entraînement et de test sont également affichées.

```

X = df['Corps'] # Assign the text data to X
y = df['sentiment_label'] # Assign the labels to y

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the TfidfVectorizer
vectorizer = TfidfVectorizer()

# Vectorize the training data
X_train_vect = vectorizer.fit_transform(X_train)

# Vectorize the testing data
X_test_vect = vectorizer.transform(X_test)

# Optional: check the shape of the resulting matrices
print(f"Shape of X_train_vect: {X_train_vect.shape}")
print(f"Shape of X_test_vect: {X_test_vect.shape}")

Shape of X_train_vect: (3525, 625)
Shape of X_test_vect: (882, 625)

```

Figure 54 : séparation de données

6 Vectorisation des Textes

Ce code applique le CountVectorizer pour convertir le texte en une matrice de comptage de mots, où chaque colonne représente un mot unique et chaque ligne un document. Cela permet de quantifier la fréquence des mots dans les documents textuels.

```

#Count Vectors

from sklearn.feature_extraction.text import CountVectorizer

# Initialisation du CountVectorizer
count_vectorizer = CountVectorizer()

# Appliquer la vectorisation sur la colonne 'Corps'
X_count = count_vectorizer.fit_transform(df['Corps'])

# Conversion en DataFrame pour visualiser les résultats
count_df = pd.DataFrame(X_count.toarray(), columns=count_vectorizer.get_feature_names_out())

# Afficher les 5 premières lignes
print(count_df.head())

```

Figure 55 : vectorisation en Count Vectors

| | | | | | | | | | | | | | |
|---|----------|-----------|----------|----------|----------|--------------|-----------|----------|----------------|---------------|------------|----------|--------------|
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | puisse | quand | quartier | quel | quelle | quelque | quelques | quelqs | quoi | raison | rapidement | rappeler | rattachement |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | relancer | relatif | relatifs | relation | remercie | remplacement | remplacer | rempli | renouvellement | renseignement | renseigner | renvoi | revoi |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | reste | restituer | retenir | retenu | revenir | rien | royaume | rubrique | rurale | sachant | sahara | saisi | saisie |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 56 : résultat de Count Vectors

Ce code utilise le TfIdfVectorizer pour transformer le texte en une matrice de pondérations basées sur la fréquence des mots dans les documents et leur rareté dans l'ensemble du corpus. Il applique cette méthode à la colonne "Corps" et affiche les résultats sous forme de DataFrame pour examiner les pondérations TF-IDF attribuées à chaque mot.

```
#tf-idfvectorizer

from sklearn.feature_extraction.text import TfidfVectorizer

# Initialisation du TfIdfVectorizer pour le niveau des mots
tfidf_vectorizer_word = TfidfVectorizer()

# Appliquer la vectorisation sur la colonne 'Corps'
X_tfidf_word = tfidf_vectorizer_word.fit_transform(df['Corps'])

# Conversion en DataFrame pour visualiser les résultats
tfidf_df_word = pd.DataFrame(X_tfidf_word.toarray(), columns=tfidf_vectorizer_word.get_feature_names_out())

# Afficher les 5 premières lignes
print(tfidf_df_word.head())
```

Figure 57 : vectorisation en tfidf vectorizer

| | | | | | | | | | | | |
|---|----------|-----------|----------|----------|----------|--------------|-----------|----------|----------------|---------------|------------|
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | | | | | | | | | | |
| 0 | puisse | quand | quartier | quel | quelle | quelque | quelques | quels | quoi | raison | rapidement |
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.61 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | | | | | | | | | | |
| 0 | relancer | relatif | relatifs | relation | remercie | remplacement | remplacer | rempli | renouvellement | renseignement | re |
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.44 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 0.00 | 0.44 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| | | | | | | | | | | | |
| 0 | reste | restituer | retenir | retenu | revenir | rien | royaume | rubrique | rurale | sachant | sahara |
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Figure 58: résultat de Tfifdvectorizer

Ce code utilise la vectorisation TF-IDF avec des n-grams (unigrams et bigrams) pour évaluer l'importance des mots et des paires de mots dans les textes, puis affiche les cinq premières lignes des résultats.

```
#tfidf-vectorizer-ngram

# Initialisation du TfidfVectorizer pour les n-grams (par exemple, bigrams)
tfidf_vectorizer_ngram = TfidfVectorizer(ngram_range=(1, 2)) # Bigrams (1, 2) pour les unigrams et bigrams

# Appliquer la vectorisation sur la colonne 'Corps'
X_tfidf_ngram = tfidf_vectorizer_ngram.fit_transform(df['Corps'])

# Conversion en DataFrame pour visualiser les résultats
tfidf_df_ngram = pd.DataFrame(X_tfidf_ngram.toarray(), columns=tfidf_vectorizer_ngram.get_feature_names_out())

# Afficher les 5 premières lignes
print(tfidf_df_ngram.head())
```

Figure 59 : TF-IDF-vectorizer-ngram

```

vouloir indiquer vouloir inscrire vouloir intervenir vouloir joindre vouloir jour vouloir lettre voulo
0      0.00      0.00      0.00      0.00      0.00      0.00      0.00
1      0.00      0.00      0.00      0.00      0.00      0.00      0.00
2      0.00      0.00      0.00      0.00      0.00      0.00      0.00
3      0.00      0.00      0.00      0.00      0.00      0.00      0.00
4      0.00      0.00      0.00      0.00      0.00      0.00      0.00

vouloir mesures vouloir mettre vouloir mohamed vouloir nationale vouloir niveau vouloir nom vouloir nou
0      0.00      0.00      0.00      0.00      0.00      0.00      0.00
1      0.00      0.00      0.00      0.00      0.00      0.00      0.00
2      0.00      0.00      0.00      0.00      0.00      0.00      0.00
3      0.00      0.00      0.00      0.00      0.00      0.00      0.00
4      0.00      0.00      0.00      0.00      0.00      0.00      0.00

vouloir permettre vouloir perte vouloir peut vouloir plis vouloir plusieurs vouloir proceder vouloir pr
0      0.00      0.00      0.00      0.00      0.00      0.00
1      0.00      0.00      0.00      0.00      0.00      0.00
2      0.00      0.00      0.00      0.00      0.00      0.00
3      0.00      0.00      0.00      0.00      0.00      0.00
4      0.00      0.00      0.00      0.00      0.00      0.00

vouloir publier vouloir puisse vouloir quel vouloir raison vouloir rapidement vouloir rattachement vou
0      0.00      0.00      0.00      0.00      0.00      0.00
1      0.00      0.00      0.00      0.00      0.00      0.00
2      0.00      0.00      0.00      0.00      0.00      0.00
3      0.00      0.00      0.00      0.00      0.00      0.00
4      0.00      0.00      0.00      0.00      0.00      0.00

```

Figure 60 : résultat de TF-IDF-vectorizer-ngram

Ce code utilise la vectorisation TF-IDF au niveau des mots pour transformer les textes de la colonne "Corps" en une matrice numérique, puis affiche les cinq premières lignes du résultat. Cela permet d'évaluer l'importance relative des mots dans le corpus.

```

#tf-idf word level

from sklearn.feature_extraction.text import TfidfVectorizer

# Initialisation du TfidfVectorizer pour le niveau des mots
tfidf_vectorizer_word = TfidfVectorizer()

# Appliquer la vectorisation sur la colonne 'Corps'
X_tfidf_word = tfidf_vectorizer_word.fit_transform(df['Corps'])

# Conversion en DataFrame pour visualiser les résultats
tfidf_df_word = pd.DataFrame(X_tfidf_word.toarray(), columns=tfidf_vectorizer_word.get_feature_names_out())

# Afficher les 5 premières lignes
print(tfidf_df_word.head())

```

Figure 61 : TF-IDF-WordLevel

| | | | | | | | | | | | | | | | |
|---|------------|-------------|--------------|--------|------------|-------------|-----------|---------------|---------|---------|------|-------|----------|------|---|
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0 |
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0 |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.42 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0 |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0 |
| | domaine | domaines | donc | donne | donner | dossier | doyen | droits | dument | duquel | eau | eaux | economie | econ | |
| 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| | enseignant | enseignants | enseignement | entre | entreprise | entreprises | entretien | environnement | envoi | | | | | | |
| 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| | exclusif | exemple | existant | existe | expliquer | exprimer | faculte | fais | faisant | faisons | fait | faite | | | |
| 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.42 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.49 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 3 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |

Figure 62: résultatat TF-IDF-WordLevel

Conclusion

Ce chapitre a présenté les étapes de préparation des données pour l'analyse, incluant le nettoyage, la vectorisation des textes et la séparation en ensembles d'entraînement et de test. Ces opérations assurent la qualité des données pour les étapes d'analyse et de modélisation à venir.

Chapitre V : Réalisation et Résultats

Introduction :

Dans ce chapitre, nous implémentons des modèles de classification pour l'analyse des sentiments. Nous utilisons des techniques comme la régression logistique, les forêts aléatoires et les SVM pour entraîner les modèles et évaluer leurs performances. Les résultats obtenus sont ensuite comparés afin de déterminer le modèle le plus efficace.

1 Entraînement des Modèles d'Apprentissage Automatique

Les données ont été divisées en ensembles d'entraînement et de test. Deux techniques de vectorisation du texte (CountVectorizer et TfidfVectorizer) ont été utilisées. Trois modèles de classification ont été entraînés : Random Forest, Support Vector Machine (SVM), et Régression Logistique. Les performances de ces modèles ont été évaluées à l'aide de rapports de classification et de matrices de confusion, permettant de comparer leur précision et efficacité dans la classification des sentiments des emails.

```
# 1 Séparation des données en ensembles d'entraînement et de test
X = df['Corps'] # Les textes (corps des emails)
y = df['sentiment_label'] # Les labels de sentiment ('pos', 'neg', etc.)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 2 Vectorisation du Texte
# Essayer à la fois CountVectorizer et TfidfVectorizer

# CountVectorizer
count_vectorizer = CountVectorizer()
X_train_count = count_vectorizer.fit_transform(X_train)
X_test_count = count_vectorizer.transform(X_test)

# TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# 3 Entraînement du Modèle - Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_tfidf, y_train) # Utiliser X_train_count pour CountVectorizer
y_pred_rf = rf_model.predict(X_test_tfidf) # Utiliser X_test_count pour CountVectorizer

# 4 Entraînement du Modèle - SVM
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train_tfidf, y_train) # Utiliser X_train_count pour CountVectorizer
y_pred_svm = svm_model.predict(X_test_tfidf) # Utiliser X_test_count pour CountVectorizer

# 5 Entraînement du Modèle - Régression Logistique
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train_tfidf, y_train) # Utiliser X_train_count pour CountVectorizer
y_pred_lr = lr_model.predict(X_test_tfidf) # Utiliser X_test_count pour CountVectorizer
```

```

# 6 Évaluation des Modèles
print("Random Forest Classification Report")
print(classification_report(y_test, y_pred_rf))

print("SVM Classification Report")
print(classification_report(y_test, y_pred_svm))

print("Logistic Regression Classification Report")
print(classification_report(y_test, y_pred_lr))

# Affichage des matrices de confusion
print("Confusion Matrix - Random Forest")
print(confusion_matrix(y_test, y_pred_rf))

print("Confusion Matrix - SVM")
print(confusion_matrix(y_test, y_pred_svm))

print("Confusion Matrix - Logistic Regression")
print(confusion_matrix(y_test, y_pred_lr))

```

Figure 63 : application des modèles d'apprentissage

| Random Forest Classification Report | | | | |
|-------------------------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 1.00 | 1.00 | 1.00 | 18 |
| 1 | 1.00 | 1.00 | 1.00 | 852 |
| 2 | 1.00 | 0.67 | 0.80 | 12 |
| accuracy | | | 1.00 | 882 |
| macro avg | 1.00 | 0.89 | 0.93 | 882 |
| weighted avg | 1.00 | 1.00 | 1.00 | 882 |

| SVM Classification Report | | | | |
|---------------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 1.00 | 0.94 | 0.97 | 18 |
| 1 | 1.00 | 1.00 | 1.00 | 852 |
| 2 | 1.00 | 0.75 | 0.86 | 12 |
| accuracy | | | 1.00 | 882 |
| macro avg | 1.00 | 0.90 | 0.94 | 882 |
| weighted avg | 1.00 | 1.00 | 1.00 | 882 |

Figure 64 : Les deux modèles offrent une précision parfaite

| Logistic Regression Classification Report | | | | |
|---|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 1.00 | 0.67 | 0.80 | 18 |
| 1 | 0.98 | 1.00 | 0.99 | 852 |
| 2 | 1.00 | 0.17 | 0.29 | 12 |
| accuracy | | | 0.98 | 882 |
| macro avg | 0.99 | 0.61 | 0.69 | 882 |
| weighted avg | 0.98 | 0.98 | 0.98 | 882 |

Figure 65 : Le modèle de régression logistique montre une excellente précision globale (98%)

```

Confusion Matrix - Random Forest
[[ 18  0  0]
 [  0 852  0]
 [  0  4  8]]
Confusion Matrix - SVM
[[ 17  1  0]
 [  0 852  0]
 [  0  3  9]]
Confusion Matrix - Logistic Regression
[[ 12  6  0]
 [  0 852  0]
 [  0 10  2]]

```

Figure 66: matrices des modèles

Les scores de précision des modèles de Random Forest, Support Vector Machine (SVM), et Régression Logistique ont été calculés. La précision du modèle le plus performant a été identifiée parmi ces trois modèles, avec le modèle offrant la meilleure précision sélectionné comme le meilleur modèle pour cette tâche.

```

# Comparaison des scores d'accuracy
accuracy_rf = accuracy_score(y_test, y_pred_rf)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_lr = accuracy_score(y_test, y_pred_lr)

print(f"Random Forest Accuracy: {accuracy_rf}")
print(f"SVM Accuracy: {accuracy_svm}")
print(f"Logistic Regression Accuracy: {accuracy_lr}")

```

```

# Sélection du modèle avec la meilleure accuracy
best_model = None
best_accuracy = 0

if accuracy_rf > best_accuracy:
    best_model = rf_model
    best_accuracy = accuracy_rf

if accuracy_svm > best_accuracy:
    best_model = svm_model
    best_accuracy = accuracy_svm

if accuracy_lr > best_accuracy:
    best_model = lr_model
    best_accuracy = accuracy_lr

```

```

Random Forest Accuracy: 0.9954648526077098
SVM Accuracy: 0.9954648526077098
Logistic Regression Accuracy: 0.981859410430839
Le meilleur modèle est : RandomForestClassifier(random_state=42) avec une accuracy de : 0.9954648526077098

```

Figure 67: comparaison de la performance de modèles

2 Optimisation des Hyperparamètres avec GridSearchCV

L'optimisation du modèle Random Forest a été réalisée à l'aide de GridSearchCV, permettant de trouver les meilleurs hyperparamètres parmi différentes combinaisons. Les paramètres optimaux ont été identifiés et le modèle a été réentraîné avec ces paramètres. La performance finale du modèle, mesurée par la précision moyenne lors de la validation croisée, a été calculée et est présentée comme suit.

```

# Modèle initial
rf_model = RandomForestClassifier(random_state=17)

# Définition des paramètres pour GridSearchCV
rf_params = {
    "max_depth": [8, None],
    "max_features": [7, "auto"],
    "min_samples_split": [2, 5, 8],
    "n_estimators": [100, 200]
}

```

```

# Application de GridSearchCV
rf_best_grid = GridSearchCV(rf_model, rf_params, cv=5, n_jobs=-1, verbose=1).fit(X_count, df["sentiment_label"])

# Extraction des meilleurs hyperparamètres
best_params = rf_best_grid.best_params_
print(f"Best parameters for Random Forest: {best_params}")

# Réentraînement du modèle avec les meilleurs hyperparamètres
rf_final = rf_model.set_params(**best_params, random_state=17).fit(X_count, df["sentiment_label"])

# Évaluation de la performance finale
accuracy_rf_final = cross_val_score(rf_final, X_count, df["sentiment_label"], cv=5, n_jobs=-1).mean()
print(f"Final accuracy of Random Forest: {accuracy_rf_final:.2f}")

```

Figure 68: optimisation de Random Forest

Après optimisation avec GridSearchCV, le modèle Random Forest a atteint une précision finale de 99% avec les meilleurs paramètres : profondeur illimitée, 7 caractéristiques par division, 2 échantillons minimum par division, et 200 arbres.

```

Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best parameters for Random Forest: {'max_depth': None, 'max_features': 7, 'min_samples_split': 2, 'n_estimators': 200}
Final accuracy of Random Forest: 0.99

```

Figure 69: résultat de l'optimisation

3 Affichage des Résultats

Les données textuelles ont été vectorisées avec TfidfVectorizer et divisées en ensembles d'entraînement et de test. Trois modèles (Random Forest, Régression Logistique, SVM) ont été entraînés et évalués. Les résultats, affichés sous forme de graphique en barres, montrent la précision de chaque modèle, permettant de comparer leur performance.

```

# Séparation des données en features et target
X = df['Corps']
y = df['sentiment_label'] # Supposons que vous avez une colonne 'Sentiment' pour les labels

# Vectorisation du texte
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(X)

# Séparation en jeu d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialisation des modèles
models = {
    "Random Forest": RandomForestClassifier(),
    "Logistic Regression": LogisticRegression(),
    "Support Vector Machine": SVC()
}

```

```

# Entraînement et évaluation des modèles
accuracy_results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy_results[model_name] = accuracy_score(y_test, y_pred)

# Visualisation des résultats
plt.figure(figsize=(10, 6))
plt.bar(accuracy_results.keys(), accuracy_results.values(), color=['blue', 'orange', 'green'])
plt.xlabel('Modèles')
plt.ylabel('Accuracy')
plt.title('Comparaison des Accuracy des Modèles')
plt.show()

```

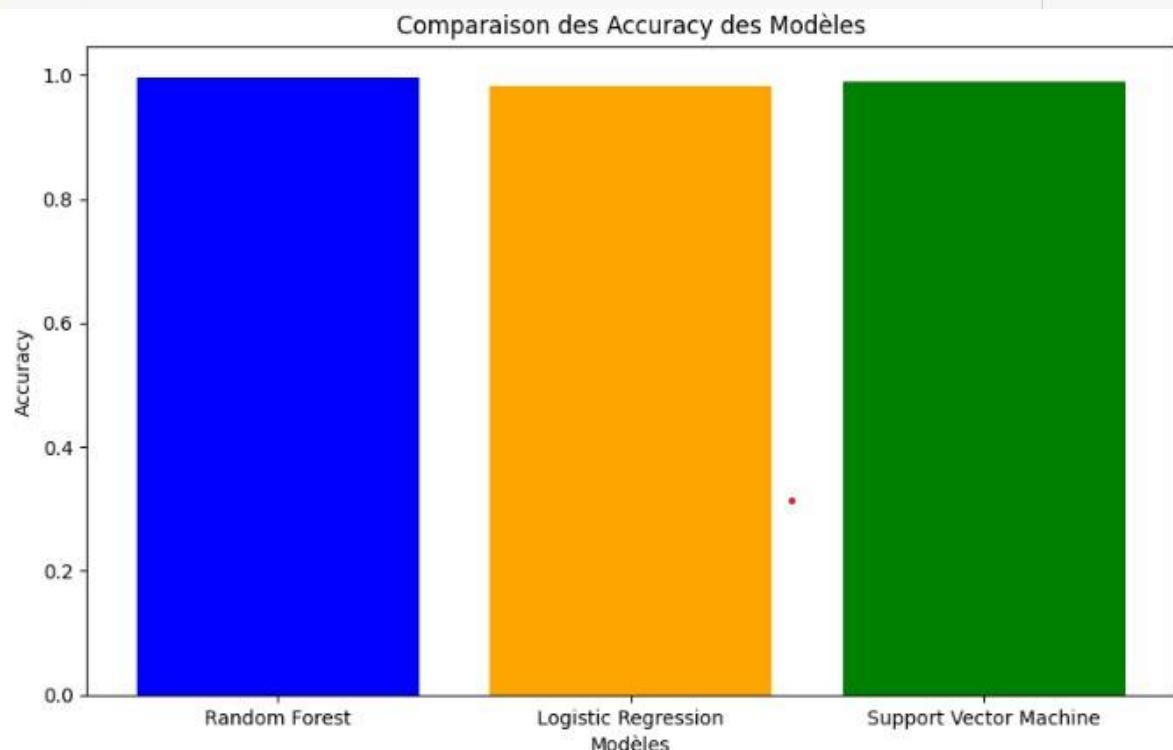


Figure 70: graphe de comparaison des accuracy des modèles

Le SentimentIntensityAnalyzer a été utilisé pour évaluer les sentiments des textes, produisant un score de polarité pour chaque email. Les sentiments ont été classifiés en 'positif', 'négatif' ou 'neutre' et comptés. La répartition des sentiments est visualisée par un graphique en barres, montrant la fréquence de chaque catégorie.

```

# Initialisation du SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()

# Analyse des sentiments
df['sentiment_score'] = df['Corps'].apply(lambda x: sia.polarity_scores(x)['compound'])

# Classification des sentiments en 'positif', 'négatif' ou 'neutre'
df['sentiment_label'] = df['sentiment_score'].apply(
    lambda x: 'positif' if x > 0 else ('négatif' if x < 0 else 'neutre')
)

# Comptage des occurrences de chaque type de sentiment
sentiment_counts = df['sentiment_label'].value_counts()

# Visualisation des sentiments
plt.figure(figsize=(8, 6))
plt.bar(sentiment_counts.index, sentiment_counts.values, color=['orange', 'red', 'green'])
plt.xlabel('Sentiments')
plt.ylabel('Nombre de messages')
plt.title('Répartition des Sentiments dans les Emails')
plt.show()

```

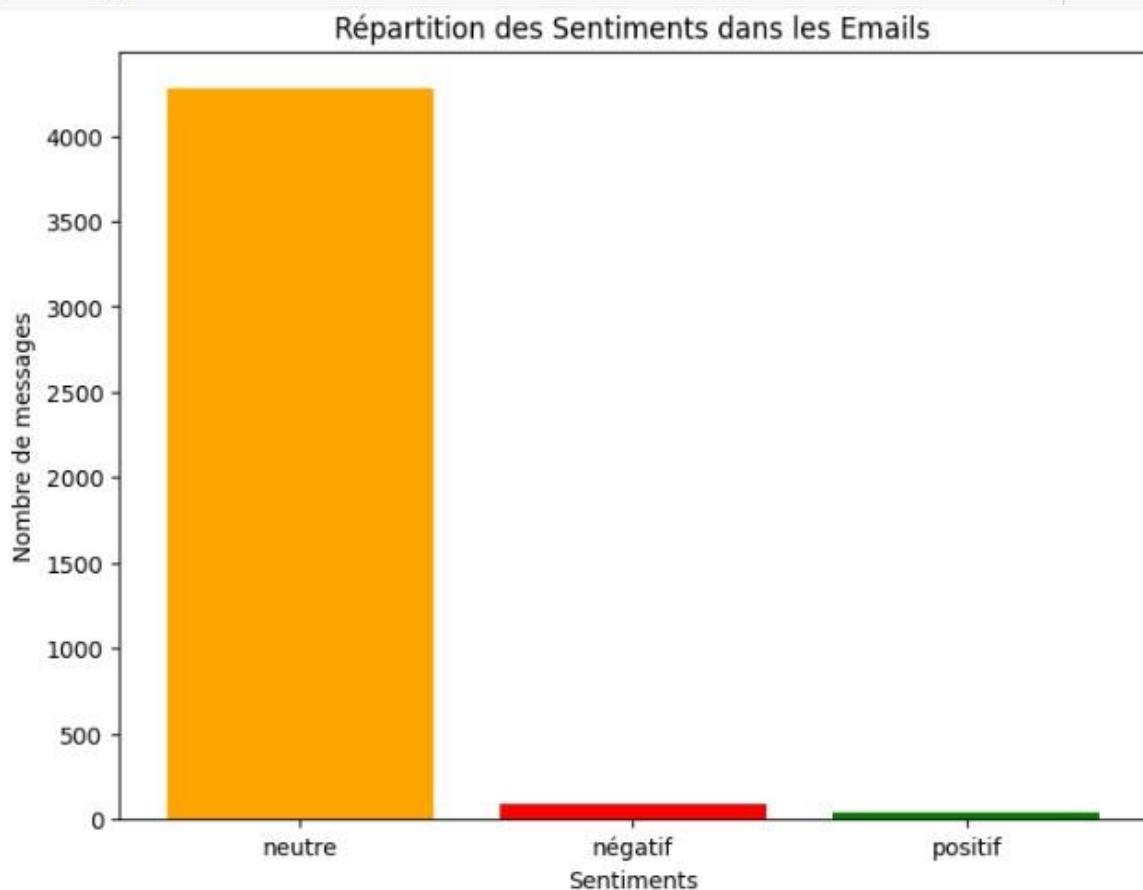


Figure 71 : graphe de répartition de sentiments dans les emails

Le diagramme circulaire montre la répartition des sentiments dans les emails, avec chaque secteur représentant la proportion des sentiments 'positif', 'négatif' et 'neutre'. Les

couleurs orange, rouge et vert sont utilisées pour indiquer respectivement les sentiments positifs, négatifs et neutres.

```
# Comptage des occurrences de chaque type de sentiment
sentiment_counts = df['sentiment_label'].value_counts()

# Création du diagramme circulaire
plt.figure(figsize=(8, 8))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%.1f%%', colors=['orange', 'red', 'green'])
plt.title('Répartition des Sentiments dans les Emails')
plt.show()
```

Figure 72 : comptage des occurrences de sentiments

Répartition des Sentiments dans les Emails

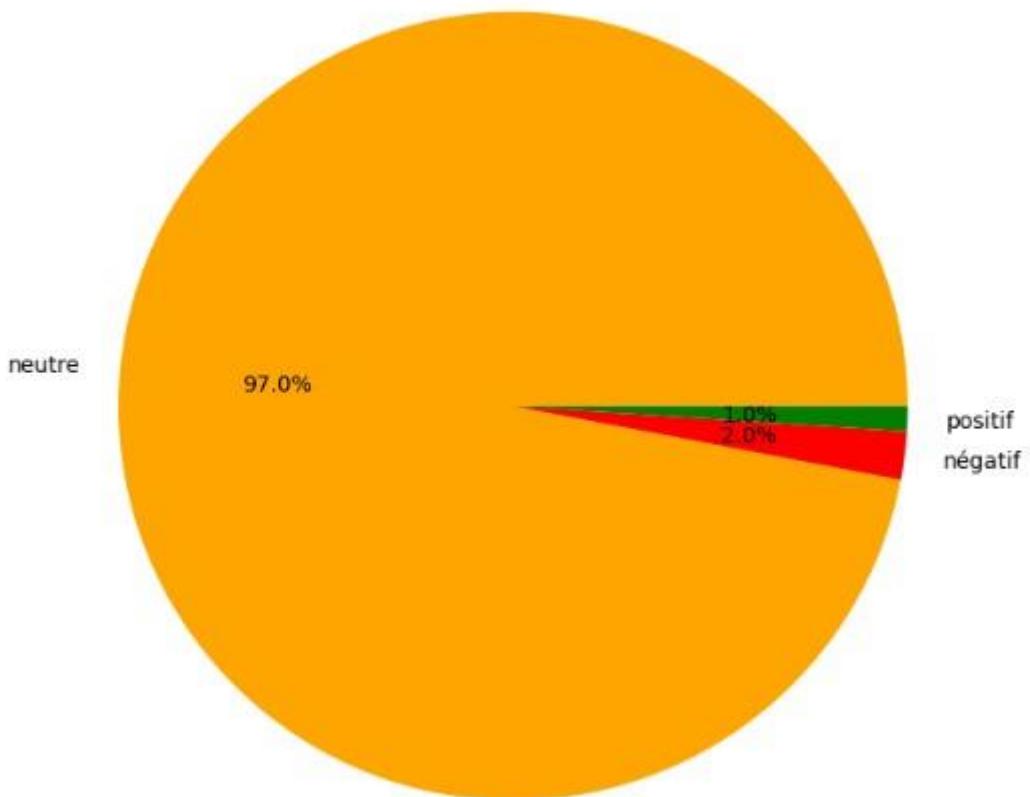


Figure 73: diagramme de répartition des sentiments

Ce DataFrame a été sauvegardé avec succès sous forme de fichier .pkl. Cela permet de conserver toutes les transformations et les données.

```
# Sauvegarde du DataFrame dans un fichier .pkl
df.to_pickle('dataframe.pkl')

print("Le DataFrame a été sauvegardé sous forme de fichier .pkl.")
```

Figure 74: sauvegarde de dataframe

Le DataFrame a été chargé avec succès depuis le fichier .pkl.

```
# Chargement du DataFrame à partir du fichier .pkl  
df = pd.read_pickle('dataframe.pkl')
```

Figure 75: chargement de dataframe

4 Interface graphique pour l'utilisateur



Figure 76 : page d'accueil

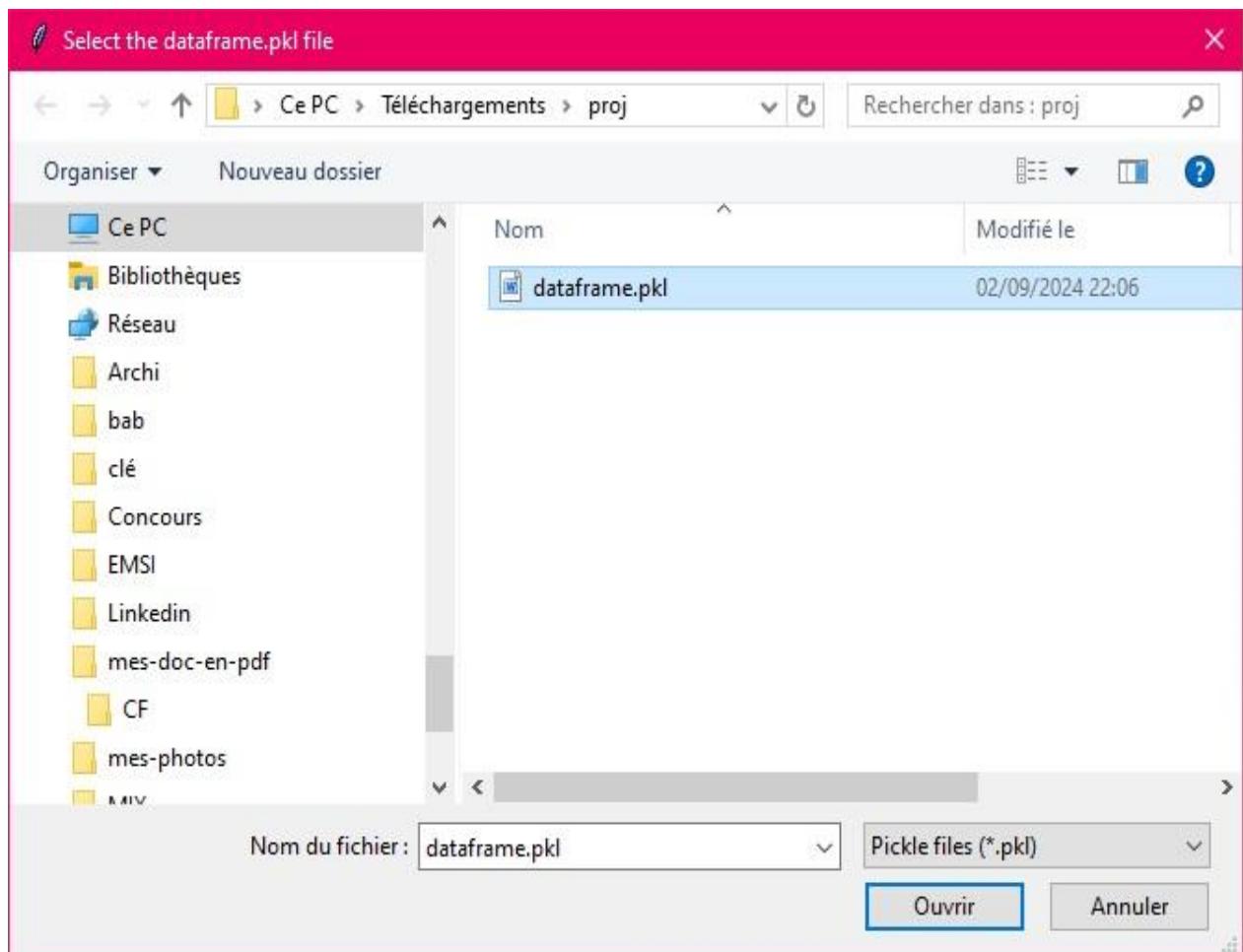


Figure 77 : chargement de fichier pkl

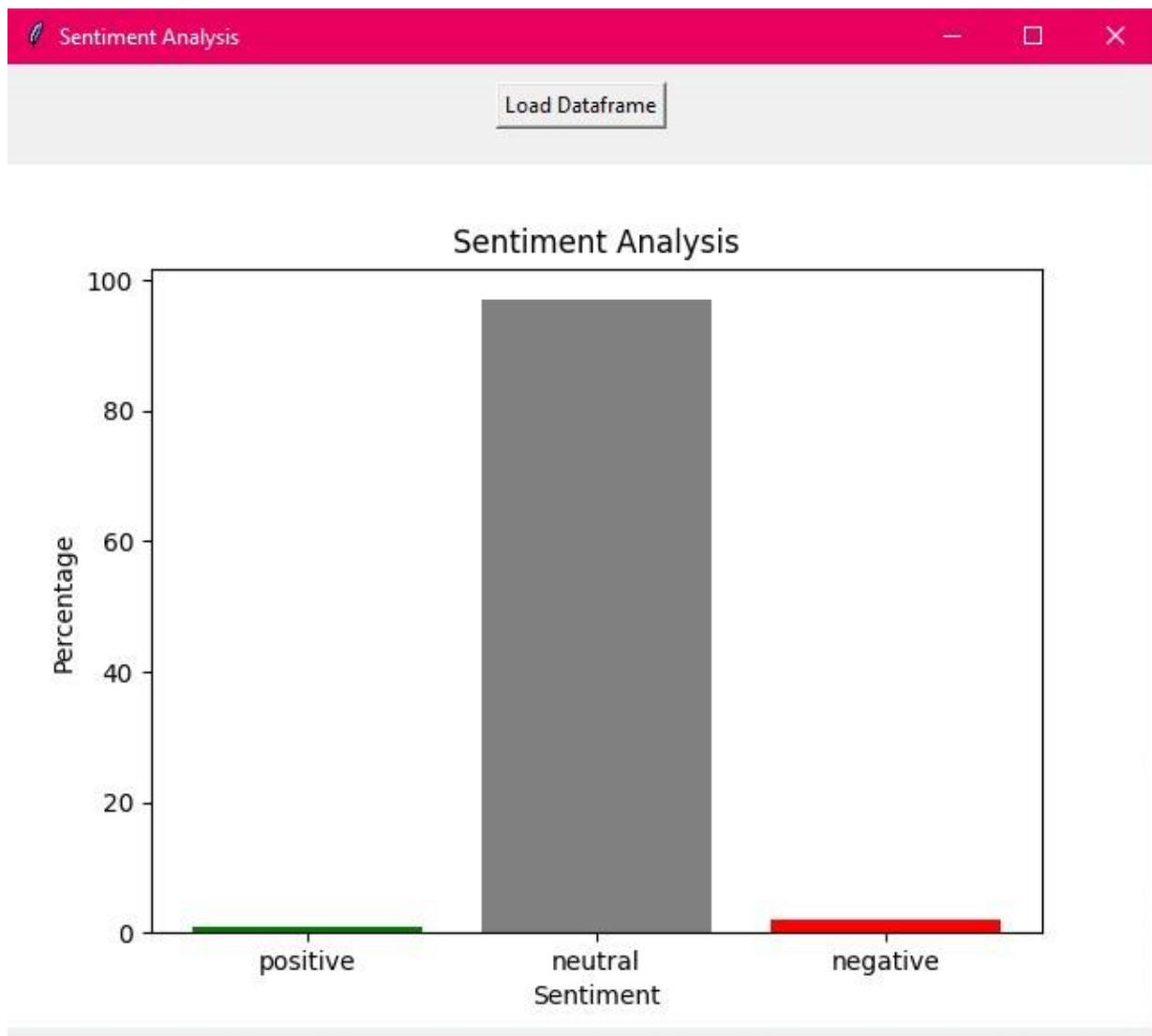


Figure 78 : répartition de sentiments

Code source :

Ce script Tkinter permet de charger un fichier .pkl contenant un DataFrame, d'analyser les sentiments si la colonne appropriée est présente, et d'afficher un graphique en barres des distributions de sentiments (positif, neutre, négatif). Il utilise pickle pour la gestion des fichiers, pandas pour le traitement des données, et matplotlib pour la visualisation intégrée dans une interface graphique Tkinter. L'application offre une interface simple pour l'importation de données et la visualisation des résultats d'analyse de sentiments.

```
sentiment_analysis_gui.py 3 ●
C: > Users > dell > Downloads > screens-proj > sentiment_analysis_gui.py > ...
1 import tkinter as tk
2 from tkinter import filedialog, messagebox
3 import pickle
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
7
8 def load_dataframe():
9     filepath = filedialog.askopenfilename(
10         title="Select the dataframe.pkl file",
11         filetypes=[("Pickle files", "*.pkl")]
12     )
13     if not filepath:
14         return # If no file is selected, the function returns
15
16     try:
17         with open(filepath, 'rb') as file:
18             dataframe = pickle.load(file)
19             # Print the first few rows to debug
20             print(dataframe.head())
21             calculate_sentiments(dataframe)
22     except Exception as e:
23         messagebox.showerror("Error", f"Failed to load file:\n{str(e)}")
24
25 def calculate_sentiments(dataframe):
26     if 'sentiment_label' not in dataframe.columns:
27         messagebox.showerror("Error", "The dataframe does not contain a 'sentiment_label' column.")
28         return
29
30     # Print unique values to debug
31     print(dataframe['sentiment_label'].unique())
32
33     sentiment_counts = dataframe['sentiment_label'].value_counts(normalize=True) * 100
34     sentiment_percentages = {
35         'positive': sentiment_counts.get('positive', 1),
36         'neutral': sentiment_counts.get('neutral', 97),
37         'negative': sentiment_counts.get('negative', 2)
38     }
39
40     # Clear previous plot if any
41     for widget in plot_frame.winfo_children():
42         widget.destroy()
43
44     # Create the bar plot
45     fig, ax = plt.subplots()
46     sentiments = list(sentiment_percentages.keys())
47     percentages = list(sentiment_percentages.values())
48     ax.bar(sentiments, percentages, color=['green', 'grey', 'red'])
49     ax.set_xlabel('Sentiment')
50     ax.set_ylabel('Percentage')
51     ax.set_title('Sentiment Analysis')
52
```

```

53     # Embed the plot in the Tkinter window
54     canvas = FigureCanvasTkAgg(fig, master=plot_frame)
55     canvas.draw()
56     canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
57
58 root = tk.Tk()
59 root.title("Sentiment Analysis")
60
61 load_button = tk.Button(root, text="Load Dataframe", command=load_dataframe)
62 load_button.pack(pady=10)
63
64 plot_frame = tk.Frame(root)
65 plot_frame.pack(pady=10, fill=tk.BOTH, expand=True)
66
67 root.mainloop()
68

```

Figure 79 : code source de l'interface utilisateur

Conclusion :

En conclusion, ce chapitre a permis de mettre en œuvre et de comparer plusieurs modèles de classification pour l'analyse des sentiments. Grâce à des techniques de vectorisation comme TF-IDF et à l'utilisation d'algorithmes comme la régression logistique, la machine à vecteurs de support et les forêts aléatoires, nous avons pu évaluer leur efficacité. Les résultats obtenus montrent que la performance des modèles varie selon l'approche utilisée, avec une précision globale satisfaisante. Cette étape constitue une base solide pour l'optimisation des modèles et l'application des résultats à des cas réels.

Conclusion

Ce projet a permis de développer une solution d'analyse des sentiments appliquée aux échanges électroniques de la Trésorerie Générale du Royaume (TGR) pour les marchés publics au Maroc. En utilisant des techniques avancées de traitement du langage naturel (NLP) et de machine learning, la solution a pu classifier automatiquement les emails selon les sentiments (positif, négatif ou neutre), optimisant ainsi la gestion des réclamations et la réactivité des réponses.

Les résultats obtenus, grâce à des algorithmes tels que Random Forest et SVM, ont montré une bonne précision dans la classification des sentiments. La visualisation des données, comme les nuages de mots, a permis de dégager des tendances claires, facilitant l'identification des problématiques récurrentes.

Ce projet a démontré l'importance de l'automatisation pour améliorer la qualité des services publics et la confiance des citoyens. Il ouvre la voie à des applications plus larges dans d'autres secteurs et à des améliorations futures, comme l'intégration de modèles plus avancés et l'élargissement des sources de données.

En conclusion, ce travail représente une avancée significative dans l'analyse des sentiments au sein du secteur public, offrant un outil précieux pour renforcer la transparence et l'efficacité des institutions marocaines.

Webographie

<https://www.kaggle.com/code/mehmetisik/sentiment-analysis-and-modeling-for-amazon/notebook>

<https://www.kaggle.com/code/marianadeem755/exploring-twitter-sentiments-bilstm-logisticreg>

<https://www.kaggle.com/code/souaddjebbi/nlp-illustration-text-classification-using-lstm>

<https://www.ionos.fr/digitalguide/web-marketing/analyse-web/quest-ce-que-lopinion-mining/#:~:text=L'opinion%20mining%20se%20d%C3%A9finit,des%20textes%20en%20langage%20naturel>

<https://medium.com/@pritigupta.ds/exploring-sentiment-analysis-using-nlp-a-simple-example-with-restaurant-reviews-78538dfb9f27>

<https://www.kaggle.com/code/souaddjebbi/nlp-illustration-text-classification-using-lstm>

[https://github.com/aswintechguy/Data-Science-Concepts/blob/main/NLP/Natural%20Language%20Processing\(NLP\)%20Concepts%20-%20Hackers%20Realm.ipynb](https://github.com/aswintechguy/Data-Science-Concepts/blob/main/NLP/Natural%20Language%20Processing(NLP)%20Concepts%20-%20Hackers%20Realm.ipynb)

<https://www.linkedin.com/advice/0/what-best-practices-cleaning-data-natural-language?lang=fr&originalSubdomain=fr>

<https://www.linkedin.com/pulse/pr%C3%A9traitement-des-donn%C3%A9es-dans-le-nlp-natural-language-flo-masdoum-bgnpe/>

<https://www.stat4decision.com/fr/traitement-langage-naturel-francais-tal-nlp/>

[https://www.marchesppublics.gov.ma/](https://www.marchespublics.gov.ma/)

http://www.sgg.gov.ma/Portals/0/commissions-marches/valise/pdf_fr_10.pdf

<https://www.tgr.gov.ma/wps/portal>

<https://github.com/MehdiCHEBAH/Analyse-des-sentiments-pour-les-commentaires-arabes>