République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université de Mohamed El Bachir El Ibrahimi de Borj Bou Arréridj
Faculté des Mathématiques et d'Informatique
Département d'informatique

MEMOIRE

Présenté en vue de l'obtention du diplôme

Master en informatique

Spécialité : Technologie de l'information et de la communication

THEME

Traiter le problème de déséquilibre de données en apprentissage automatique : le cas de la détection automatique des attitudes envers les rumeurs politiques en ligne

Présenté par :

M'HAMDI Nassima

AMARA Maroua

Soutenu publiquement le : 20/06/2023

Devant le jury composé de :

Président : SAIDANI Kaouthar

Examinateur : SAIFI Abdelhamid

Encadrante : MOHDEB Djamila

2022/2023

Dédicace

Je dédie ce travail :

À ma chère mère

À mon cher père

À ma grand-mère et à toutes mes tantes, surtout ***Warda***

À mes frères et mes soeurs

Pour leurs soutiens moraux tout au long de mes études, et qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

À mon cher mari, pour sa patience et son soutien

Preuve pour la durée de ce travail et pour qui je suis

Je tiens à exprimer mon affection et ma gratitude.

À mon fils ***Sanad***

À ma chère binôme ***Maroua***

À mes meilleurs amis ***Imane, Khaoula et Chaima***

À tous mes amis de promotion de 2eme année Master TIC

À toute personne qui occupe une place dans mon cœur

Nassima

Dédicace

Je dédie ce modeste travail

*À mes chers **parents**, il est difficile d'exprimer par des mots toute la gratitude, l'amour et l'appréciation que je ressens à votre égard, Vos sacrifices, et votre amour inconditionnel ont été les fondements de ma croissance et de mon bonheur. Je prie Dieu de vous protéger et de vous accorder santé et longue vie*

Insha'Allah.

À mes chères sœurs qui ont toujours étaient à mes cotés

*À mes chers neveu et nièce « **Basmala** et **Abdul Aleem** » Que Dieu vous guide et vous protège tout au long de votre vie*

*À mon ame sœur, ma compagne de vie « **Yakine** »*

*À ma chère binôme «**Nassima**»*

À mes Ames

À tous ceux qui me sont chers, à ma famille et à mes proches

Maroua

Remerciements

Tout d'abord, nous souhaitons exprimer notre gratitude à Dieu pour nous avoir donné la force et le courage de mener à bien ce travail.

Nous tenons à exprimer notre profonde reconnaissance envers Madame MOHDEB Djamila. Nous la remercions chaleureusement pour son encadrement exceptionnel, sa patience inébranlable, sa rigueur exemplaire et sa disponibilité constante tout au long de notre préparation de ce mémoire. Sa contribution précieuse a grandement contribué à la qualité et à la réussite de notre travail, et nous lui en sommes sincèrement reconnaissants.

Nous remercions chaleureusement les membres du jury d'avoir accepté de juger notre travail. Vos expertises et vos avis éclairés joueront un rôle essentiel dans l'évaluation et l'amélioration de notre projet.

En conclusion, nous exprimons notre gratitude à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce mémoire.

Résumé

Les médias sociaux jouent un rôle crucial dans la communication et le partage d'informations à notre époque. Cependant, la diffusion des données sur ces plateformes pose un défi majeur lorsqu'il s'agit de distinguer les informations véridiques des fausses. Les fausses nouvelles se propagent rapidement, causant ainsi des dommages considérables en propageant des rumeurs et des informations erronées.

Dans le cadre de ce projet, notre objectif principal était de développer un système de classification automatique des attitudes envers les rumeurs politiques en ligne, en tenant compte du défi commun de déséquilibre des données dans ce type de tâche de classification de texte. Nous avons mis en œuvre et évalué différentes approches de classification, telles que l'apprentissage par ensemble, le sur-échantillonnage (augmentation de texte), l'apprentissage sensible au coût et l'apprentissage mono-classe, afin de déterminer la plus adaptée à notre problématique.

Les résultats obtenus mettent en évidence les performances remarquables de l'approche de sur-échantillonnage par rapport aux autres approches concurrentes, notamment l'apprentissage par ensemble, l'apprentissage sensible au coût, et l'apprentissage mono-classe.

Mots-clés : fausses nouvelles, rumeurs, classification automatique, classification de texte, déséquilibre des données, apprentissage par ensemble, apprentissage sensible au coût, apprentissage mono-classe, apprentissage sur-échantillonnage.

Abstract

Social media plays a crucial role in communication and information sharing in our time. However, the dissemination of data on these platforms poses a major challenge when it comes to distinguishing true from false information. Fake news spreads quickly, causing massive damage by spreading rumors and misinformation.

In this project, our main objective was to develop an automatic classification system of attitudes towards online political rumors, taking into account the common challenge of data imbalance in this type of text classification task. We implemented and evaluated different classification approaches, such as set learning, oversampling (text augmentation), cost-aware learning, and single-class learning, to determine the most suitable to our problem.

The results obtained highlight the remarkable performance of the oversampling approach compared to other competing approaches, including ensemble learning, cost-sensitive learning, and single-class learning.

Keywords: fake news, rumors, automatic classification, text classification, data imbalance, ensemble learning, cost-sensitive learning, single-class learning, oversampling learning.

ملخص

تلعب وسائل التواصل الاجتماعي دورًا مهمًا في التواصل ومشاركة المعلومات في عصرنا. ومع ذلك، فإن نشر البيانات على هذه المنصات يشكل تحديًا كبيرًا عندما يتعلق الأمر بالتمييز بين المعلومات الصحيحة والخاطئة. تنتشر الأخبار الكاذبة بسرعة، مما يتسبب في أضرار جسيمة من خلال نشر الإشاعات والمعلومات المضللة.

في هذا المشروع، كان هدفنا الرئيسي هو تطوير نظام تصنيف آلي للمواقف تجاه الشائعات السياسية عبر الإنترنت، مع الأخذ في الاعتبار التحدي المشترك المتمثل في اختلال توازن البيانات في هذا النوع من مهام تصنيف النص. قمنا بتنفيذ وتقييم مناهج تصنيف مختلفة، مثل مجموعة التعلم، والاختزال (زيادة النص)، والتعلم المدرك للتكلفة، والتعلم من فئة واحدة، لتحديد الأنسب لمشكلتنا.

النتائج التي تم الحصول عليها تسلط الضوء على الأداء الرائع لنهج الإفراط في أخذ العينات مقارنة بالنهج المنافسة الأخرى، بما في ذلك التعلم الجماعي، والتعلم الحساس للتكلفة، والتعلم من فئة واحدة.

الكلمات المفتاحية: الأخبار الكاذبة، الشائعات، التصنيف الآلي، تصنيف النص، عدم توازن البيانات، التعلم الجماعي، التعلم الحساس للتكلفة، التعلم الفردي، التعلم المفرط.

Table des matières

Liste des abréviations

SMOTE	Synthetic Minority Over-sampling Technique
TF-IDF	Term Frequency-Inverse Document Frequency (TF-IDF)
SVM	Support Vector Machine
POS	Part-of-Speech
URL	Uniform Resource Locator
BERT	Bidirectional Encoder Representations from Transformers
ROC	Receiver Operating Characteristic

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction Générale

1. Contexte

Les médias sociaux offrent une abondance inépuisable d'informations, où les utilisateurs sont constamment submergés par une multitude de contenus. Cependant, parmi cette énorme quantité de données se cachent des rumeurs insidieuses qui se propagent rapidement et ont des effets profonds sur les communautés. Ces rumeurs, souvent dépourvues de véracité, alimentent la méfiance et la confusion parmi les utilisateurs, sapant ainsi la confiance mutuelle qui est essentielle pour maintenir une société bien informée. Les conséquences de ces rumeurs peuvent être dévastatrices, créant des divisions et exacerbant les tensions au sein des communautés. Il devient donc crucial pour les individus d'exercer un esprit critique et de vérifier les informations avant de les partager, afin de préserver l'intégrité des communautés en ligne et de promouvoir un échange d'informations fiable et constructif.

2. Problématique

L’apprentissage automatique, une partie de l’intelligence artificielle, fait face à un problème majeur appelé "déséquilibre des données." Ce problème survient lorsque les données utilisées pour l’apprentissage sont inégalement réparties, certains groupes étant sous-représentés comparativement à d’autres. Cela peut affecter les algorithmes d’apprentissage, qui peuvent avoir de la difficulté à identifier et à classer correctement les cas minoritaires lors de l’établissement de classifications. Le déséquilibre des données est crucial dans des domaines tels que la détection des attitudes envers les rumeurs, où cela peut conduire à un biais dans les résultats du modèle, avec des prédictions inexactes pour les groupes minoritaires. De plus, cela peut fausser les analyses statistiques et renforcer les biais actuels dans la collecte des données. Des mesures doivent être prises pour équilibrer les ensembles de données et assurer des prévisions exactes, comme le débordement de l’adoption de classe minoritaire ou majoritaire.

3. Objectif et contribution

L'objectif de ce projet d'étude est de développer un système de classification automatique des attitudes envers les rumeurs politiques en ligne, en tenant compte du problème courant de déséquilibre des données dans ce type de tâches de classification textuelle. Nous avons mis en place et évalué différentes approches de classification adaptées à cette problématique, notamment l'apprentissage par ensemble, le sur-échantillonnage (l'augmentation de texte), l'apprentissage sensible au coût et l'apprentissage mono-classe.

4. Structure du rapport

La structure de ce rapport comprend quatre chapitres qui détaillent la méthodologie utilisée dans ce projet :

- **Chapitre 01** : donne un aperçu de la détection de la position et de ses aspects liés tels que ses niveaux, son architecture de système, sa spécificité par rapport à la détection des sentiments, ses tâches et ses applications.
- **Chapitre 02** : se concentre sur le problème de déséquilibre de données, en définissant son importance dans les tâches de classification textuelle et de classification des positions, et présente ensuite les différentes techniques pour résoudre ce problème.
- **Chapitre 03 :** détaille la conception de notre système et présente les méthodes suivies dans ses différentes phases.
- **Chapitre 04** : décrit l'environnement matériel et logiciel utilisé pour mettre en place le système et discute les résultats expérimentaux obtenus.

En dernier lieu, une conclusion générale qui résume les points essentiels de ce travail.

Chapitre 01: Concepts de base

1.1 Introduction

La détection de position est l'extraction de la réaction d'un sujet à une affirmation faite par un acteur principal. Il s'agit d'un élément essentiel pour plusieurs approches qui sont destinées à l'évaluation des fausses informations.

Ce chapitre est un aperçu sur la détection de position et tout ce qui y est lié : ses niveaux, l'architecture de son système, sa spécificité par rapport à la détection des sentiments, ses tâches, et ses applications.

1.2 Définition de la position

Afin de comprendre la tâche de détection de position, nous fournissons d'abord des définitions de la position et du processus de prise de position.

Biber et Finegan définissent la position comme l'expression du point de vue et du jugement d'un locuteur envers une proposition donnée.

Une définition plus détaillée est celle de Du Bois (2007) qui a définit la position comme « un acte public d'un acteur social, accompli de manière dialogique par des moyens de communication manifestes, consistant à évaluer simultanément des objets, à positionner des sujets et à s'aligner sur d'autres sujets, par rapport à n'importe quel sujet »

Par conséquent, le processus de prise de position est affecté non seulement par des opinions personnelles, mais aussi par d'autres facteurs externes tels que les normes culturelles, les rôles dans l'institution de la famille ou dans la société [1].

La prise de position passe généralement par les étapes suivantes :

- Comprendre le sujet problématique de position
- Analyser et Evaluer les différentes points de vue autour du sujet problématique.
- Prendre une position évidente à l'égard du sujet problématique [1].

1.3 Définition de la détection de positions

La détection de position également connue sous le nom de détection des attitudes, classification de position, identification de position, prédiction de position, est un problème de type classification de texte qui vise à déterminer la position d'un auteur d'un texte vis-à-vis d'un sujet donné. Une position peut être explicitement ou implicitement une de ces trois catégories principales : {supporter, refuser, aucun}, Une position explicite est donnée à l'aide des expressions directes, comme le pouce vers le haut ou le pouce vers le bas. Une position implicite nécessite une étude plus attentive du texte [2].

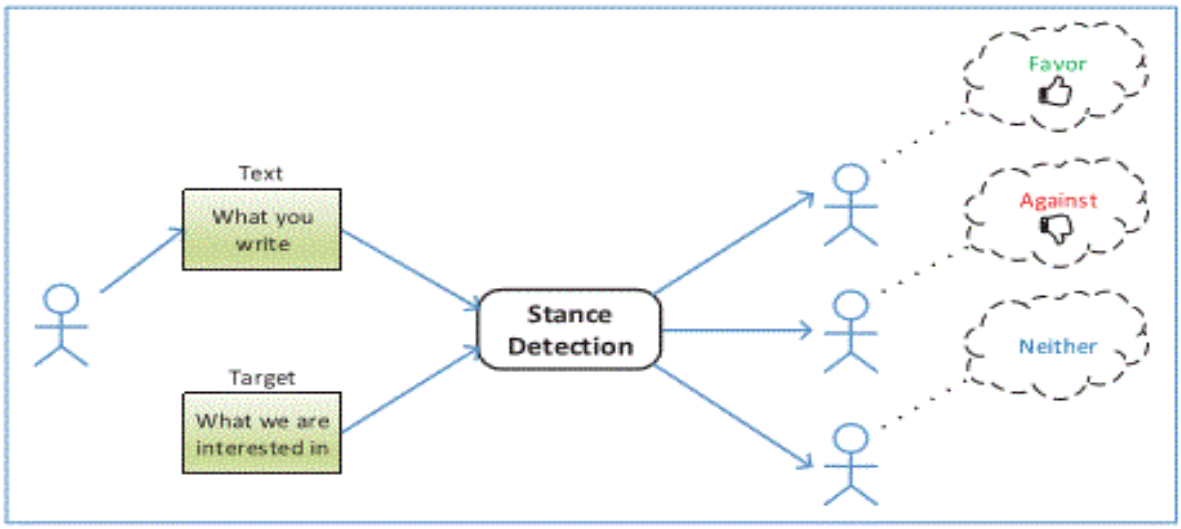


Figure 1.Réprésentation de la procédure de détection de position [2]

1.4 Niveaux de détection de positions

La classification des attitudes est appliquée à deux niveaux :

a. Le niveau d'énoncé

L'objectif est de prédire la position décrite dans un fragment de texte de taille donné en exploitant ses principaux attributs textuels extraits. Pour ce niveau, la tâche de classification a trois classes : **{supporter, refuser et aucun}** [3]**.**

b. Le niveau utilisateur

L'objectif est de prédire la position d'un utilisateur vis-à-vis d'un sujet donné où l'attention est accordée aux attributs de l'utilisateur qui dessinent ses réactions plutôt que son texte. Pour ce niveau, la tâche de classification comprends deux classes polarisées : **{supporter et refuser}** [3].

1.5 Différence entre la détection de positions et la détection des sentiments

La classification des positions et l'analyse des sentiments sont à la fois des tâches du traitement du langage naturel qui impliquent l'analyse du texte pour comprendre l'opinion ou la perspective de l'orateur ou de l'écrivain. Cependant, il existe des différences importantes entre les deux tâches.

L'analyse des sentiments est la tâche de déterminer si un texte donné exprime un sentiment positif, négatif ou neutre. Le but est d'identifier la réponse émotionnelle ou l'attitude de l'auteur envers le sujet discuté. L'analyse des sentiments peut être utilisée à diverses fins telles que la compréhension des commentaires des clients, l'analyse des publications sur les réseaux sociaux ou le suivi de l'opinion publique sur un sujet particulier.

La classification de la position, en revanche, est la tâche de déterminer la position ou la perspective d'un auteur sur un sujet ou un problème spécifique. L'objectif est de déterminer si l'auteur est en faveur ou contre un point de vue, une idée ou une politique particulière. La classification des positions est souvent utilisée dans le contexte des débats politiques ou sociaux, où il est important de comprendre les différentes positions et arguments [4].

Considérons les deux exemples suivants :

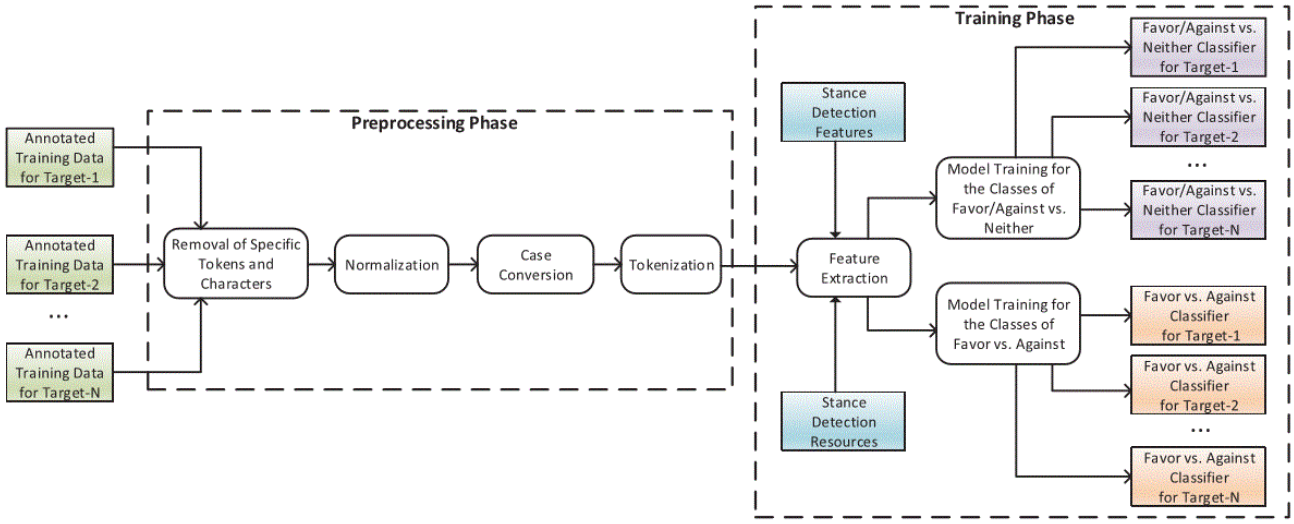
1. Analyse des sentiments : "J'ai vraiment adoré le film que j'ai vu hier soir. Le jeu des acteurs était superbe et l'histoire était tellement engageante.". Dans cet exemple, l'objectif est de déterminer la réaction émotionnelle de l'auteur au film, qui est clairement positive.

2. Détection des positions : "Je pense que le nouveau projet de loi sur les soins de santé est un pas dans la bonne direction. Il contribuera à fournir une couverture à davantage de personnes et à rendre les soins de santé plus abordables.". L'objectif dans cet exemple est de déterminer la position de l'auteur sur la facture de soins de santé, qui est en sa faveur.

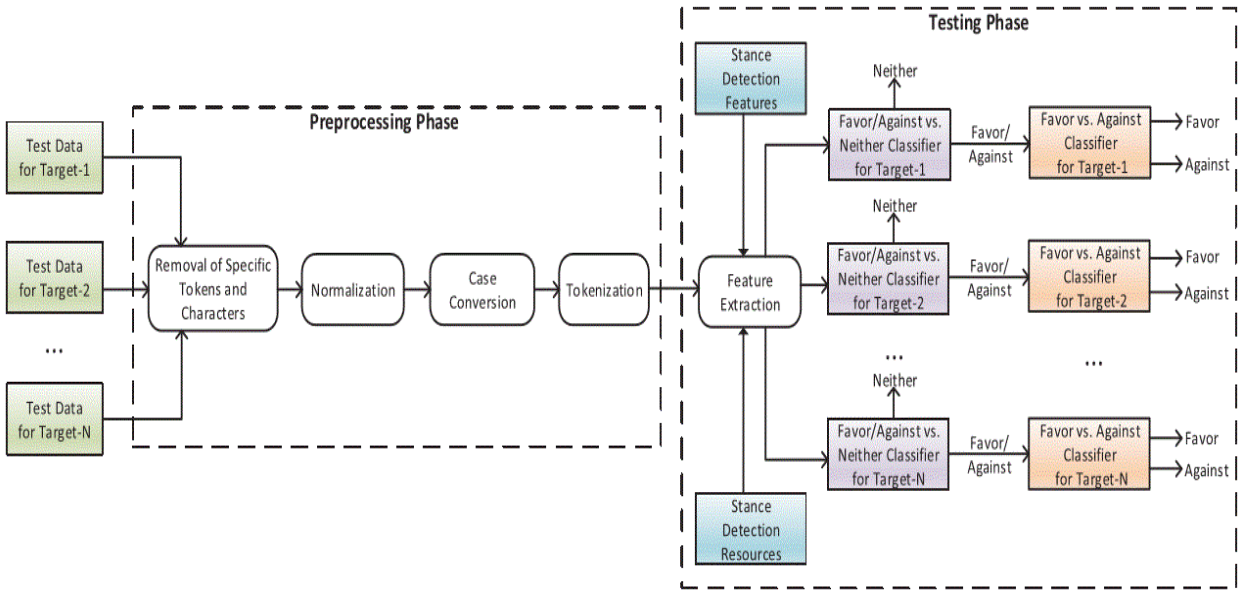
1.6 Architecture du système de détection de positions

Les deux figures ci-dessous (a) et (b) illustrent l'architecture générale qui reflète les caractéristiques communes des approches de détection de positions.

Les approches de détection de position présentées dans la littérature sont des systèmes basés sur l'apprentissage automatique qui comprend deux phases : la phase d'entraînement et la phase de test. Les deux phases sont accompagnées d'une phase de prétraitement du texte.



(a) Training phase.



(b) Testing phase.

Figure 2.Architecture de système de détection de position [2]

Phase de prétraitement du texte

Cette phase consiste à trier, nettoyer, transformer ou regrouper des données au préalable afin qu'elles soient traitées correctement, Les tâches les plus courantes qui sont effectuées pendant cette phase sont :

- **Nettoyage des données** : suppression ou correction des enregistrements contenant des valeurs corrompues ou non valides dans les données brutes, et suppression des enregistrements pour lesquels il manque un grand nombre de colonnes.
- Suppression de caractères spécifiques tels que les mots vides, les URL, les caractères spéciaux, les hashtags,
- La suppression de la ponctuation.
- La modification de la casse de la majuscule en minuscule.
- **La ségmentation (Tokenisation)** : découpage du texte en mots indépendants.
- **La racinisation (stemming)** : élimination des suffixes et de préfixes des mots.
- **La lemmatisation** : considérer la racine (lemme) des mots, afin de "fusionner" tous les mots de la même famille. Par exemple les mots enfant, enfance, enfantin sont tous transformé en la même racine [2].

Phase d'apprentissage (entraînement)

Au cours de la phase d'apprentissage, des fonctionnalités et des ressources de détection de position sont utilisés pour entraîner les modèles de classification. Voici les caractéristiques communes de la phase d'apprentissage d'un système de détection d'attitude, telles que rapportées dans la littérature :

- Pour les modèles d'apprentissage traditionnels basés sur les attributs (tels que Support Vector Machine (SVM) et les arbres de décision), des attributs prédéfinies (telles que des n-grammes, ainsi que des attributs basés sur les tags POS, des hashtags et des dictionnaires) sont utilisées pour entraîner les classificateurs [2].
- Un schéma de classification à deux étapes est adéquat pour une classification de positions à trois voies. Dans la première étape, le modèle de classification doit déterminer la pertinence, c'est-à-dire que le texte d'entrée est classé comme ayant une position (la classe «**supporter»** ou la classe «**refuser»**) ou non (la classe **ni supporter ni refuser**).
- Dans la deuxième étape, le texte d'entrée de la première phase classé comme ayant une position est finalement classé dans l'une des catégories «**supporte**r» ou «**refuser**» [2].

Phase du test

- Dans la phase de test, de la même manière, une étape de prétraitement textuel est effectuée sur l'ensemble de données de test, puis, pour chaque texte cible, deux classificateurs sont appliqués afin de produire la position {**Supporter, Refuser, Aucun**}[2].

1.7 Les applications de la détection de positions

La classification des positions est largement utilisée dans une variété d'applications, telles que la détection de fausses nouvelles, la compréhension de l'opinion publique et l'analyse des données des médias sociaux [3].

1) **Débats politiques** : la classification des positions peut être utilisée pour identifier automatiquement les positions des candidats et des partis dans les débats politiques. En analysant les transcriptions ou les données textuelles des médias sociaux, cela peut aider à

identifier les problèmes discutés, les différents points de vue et arguments, et la force de chaque position.

2) **Analyse des actualités** : la classification des positions peut être utilisée pour analyser des articles d'actualité afin de déterminer la position de l'auteur sur un événement ou un problème particulier. Cela peut aider à identifier les sources qui sont biaisées ou objectives et à suivre l'évolution de l'opinion publique au fil du temps.
3) **Analyse des réseaux sociaux** : la classification des positions peut être utilisée pour analyser les publications sur les réseaux sociaux afin d'identifier la position des utilisateurs sur divers sujets. Cela peut aider les entreprises à comprendre ce que les clients pensent de leurs produits ou services, ou ce que le public pense d'un problème ou d'un événement particulier.
4) **Service client** : la classification de la position peut être utilisée pour analyser les commentaires des clients afin d'identifier leur opinion à l'égard d'un produit ou d'un service. Il peut aider les entreprises à comprendre les enjeux et les problèmes rencontrés par leurs clients et à développer de meilleurs produits et services qui répondent à leurs besoins.
5) **Analyse juridique** : la classification des positions peut être utilisée dans l'analyse juridique pour identifier la position des parties dans une affaire. Cela peut aider les avocats à identifier les points forts et les points faibles de chaque argument et à préparer leurs propres arguments en conséquence.

1.8 Conclusion

Dans ce chapitre, notre objectif était de fournir des concepts essentiels et nécessaires à la compréhension de ce mémoire.

Le chapitre suivant, nous examinerons en détail le problème du déséquilibre des ensembles de données, ainsi que les techniques utilisées pour le résoudre.

Chapitre 02 : Les ensembles de données déséquilibrés et l’augmentation textuelle

2.1 Introduction

Compte tenu de l'importance de l'équilibre des données dans la création d'un bon classificateur, beaucoup des techniques ont été développées pour résoudre le problème du déséquilibre des classes pour la tâche de classification de texte.

Nous présentons dans ce chapitre tout ce qui concerne à ce problème, en particulier, sa description et son importance et les différentes techniques qui ont été proposées dans la litérature pour limiter ou éliminer son influence négative sur la performance des modèles de classification

2.2 Définition d’ensembles des données déséquilibres

Les ensembles de données déséquilibrés font référence à des ensembles de données où la distribution des observations entre différentes classes ou catégories est considérablement faussée ou disproportionnée. En d'autres termes, une ou plusieurs classes ont un nombre d'instances beaucoup plus petit par rapport aux autres classes. Ce déséquilibre peut créer des défis pour les algorithmes d'apprentissage automatique, car ils peuvent avoir du mal à apprendre avec précision des modèles et à faire des prédictions pour la ou les classes minoritaires en raison du manque d'exemples suffisants. Le traitement des données déséquilibrées nécessite des techniques spécialisées telles que le rééchantillonnage, la pondération des classes ou l'utilisation de méthodes d'ensemble pour garantir que le modèle peut capturer efficacement les modèles des classes majoritaires et minoritaires [5].

Figure 3.Exemple de distributions des données déséquilibrées [5]

2.3 Importance de problème de déséquilibre des ensembles de données

- Le déséquilibre des classes peut conduire à une performance biaisée du modèle : Lorsqu'un ensemble de données est déséquilibré, un modèle formé sur celui-ci peut être biaisé vers la classe majoritaire, et donner de mauvais résultats sur la classe minoritaire. Cela peut conduire à des prédictions incorrectes et à de mauvaises performances dans des scénarios réels.
- Le déséquilibre des classes affecte la généralisation du modèle : si un modèle est formé sur un ensemble de données déséquilibré, il peut ne pas bien se généraliser aux données nouvelles et invisibles. En effet, le modèle a appris à prédire en fonction de la distribution des données sur la formation, qui ne représentent peut-être pas exactement la distribution du monde réel.
- Le déséquilibre des classes peut affecter la prise de décision : Dans de nombreux scénarios réels, tels que le diagnostic médical ou la détection de fraude, une mauvaise classification des classes minoritaires peut avoir de graves conséquences. Un ensemble

de données déséquilibré peut entraîner une prise de décisions inexacte, ce qui peut avoir un impact négatif sur la vie des gens.

- Le déséquilibre des classes peut influer sur l'analyse des données : des données déséquilibrées peuvent fausser les analyses statistiques, ce qui entraîne des conclusions et des interprétations incorrectes. Par exemple, si un ensemble de données contient un grand nombre d'exemples négatifs et seulement quelques exemples positifs, les tests statistiques peuvent conclure à tort qu'il n'y a pas de différence significative entre les deux classes.
- Le déséquilibre des classes peut influer sur la collecte des données : Des ensembles de données déséquilibrés peuvent être un symptôme de biais sous-jacents dans le processus de collecte des données. Cela peut perpétuer les inégalités existantes et renforcer les stéréotypes, et peut conduire à des préoccupations éthiques concernant l'utilisation des données.

2.4 Techniques pour résoudre le problème de déséquilibre

2.4.1 Le sur-échantionnage (Over-sampling)

Le sur-échantillonnage est une technique utilisée pour résoudre les problèmes de déséquilibre dans les ensembles de données d'apprentissage automatique où la classe minoritaire a significativement moins d'instances que la classe majoritaire.

Le sur-échantillonnage consiste à augmenter le nombre d'instances dans la classe minoritaire en créant des exemples synthétiques similaires aux instances existantes [6].

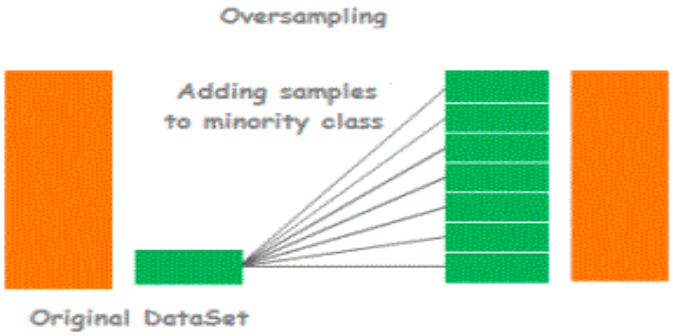


Figure 4.Réprésentation de la technique sur échantillonnage [6]

Dans le contexte où les données sont textuelles, le sur-échantillonnage s'effectue fréquemment en exploitant les techniques d'augmentation du texte. Il s'agit de générer de nouvelles données à partir des données existantes en appliquant diverses techniques comme le remplacement de synonymes, l'ajout ou la suppression de mots, le remaniement de phrases ou de paragraphes, la modification de l'ordre des mots... etc [7].

L'augmentation de texte peut être aussi effectuée avec les techniques suivantes :

- **La collecte de données** : recueillir plus des données textuelles pour les classes minoritaires.
- **La combinaison d'ensembles de données** : combiner plusieurs ensembles de données pour créer un ensemble de données plus grand. Cela peut être fait en fusionnant des ensembles de données avec des caractéristiques similaires ou en utilisant des données provenant de sources différentes.
- **L'utilisation de modèles linguistiques pré-formés** : utiliser des modèles linguistiques pré-formés comme BERT, GPT-2 ou RoBERTa pour générer des données textuelles. Ces modèles peuvent être affinés sur une tâche ou domaine spécifique pour générer du nouveau texte dans la catégorie des classes minoritaires.
- **La traduction des données** : traduire des données texte d'une langue à l'autre, puis le traduire à nouveau dans la langue originale. Cela peut aider à générer de nouvelles variations du texte.

Ces techniques peuvent nous aider à augmenter les données textuelles et à les rendre plus diversifiées, ce qui peut être utile pour former des modèles d'apprentissage automatique pour diverses tâches de traitement du langage naturel [7].

2.4.2 Le sous-échantionnage (under-sampling)

Le sous-échantillonnage est une technique utilisée dans l'apprentissage automatique pour résoudre le problème des classes déséquilibrées. Elle consiste à prélever des échantillons de la classe majoritaire afin qu'ils soient plus proches en taille de la classe minoritaire, c'est-à-dire qu'on cherche à réduire la taille de la classe majoritaire pour atténuer le déséquilibre de classes [6].

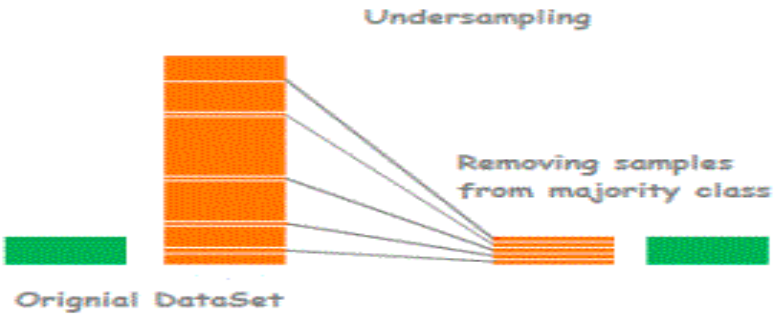


Figure 5.Réprésentation du technique sous-échantillonnage [6]

2.4.3 Apprentissage par ensemble (Ensemble Learning)

L'apprentissage par ensemble (Ensemble Learning en anglais) est une technique d'apprentissage automatique qui consiste à combiner plusieurs modèles de prévision pour améliorer les performances prédictives globales. Au lieu d'utiliser un seul modèle pour prédire des résultats, l'apprentissage par ensemble utilise plusieurs modèles et combine leurs prévisions pour obtenir une prédiction plus précise et plus fiable.

L'apprentissage par ensemble peut être très utile pour améliorer les performances prédictives, surtout lorsque les modèles individuels ont des performances médiocres ou se trompent souvent.

Cependant, il convient de noter que l'utilisation de l'apprentissage par ensemble peut rendre les modèles plus complexes et plus difficiles à interpréter [8].

Les exemples courants de techniques d'apprentissage par ensemble comprennent les méthodes de L'ensachage (Bagging)**,** Le renforcement (Boosting) et de L'empilement (le Stacking), L'ensemble de vote (le voting), Les forêts aléatoires (Random Forests).

- **L'ensachage (Bagging)**

Le Bagging signifiant « Bootstrap Aggregation », était l'une des premières méthodes proposées pour la création d'un ensemble de classifieurs. Il combine plusieurs modèles pour améliorer la précision et la stabilité des algorithmes d'apprentissage automatique [8].

Le Bagging est une technique qui implique la création de multiples sous-ensembles de données d'entraînement en échantillonnant aléatoirement les données originales avec remplacement. Chaque sous-ensemble est utilisé pour entraîner un modèle distinct en utilisant le même algorithme. Les prédictions de ces modèles sont ensuite combinées en prenant la moyenne (dans le cas de la régression) ou en effectuant un vote majoritaire (dans le cas de la classification). On distingue deux types d'algorithmes de Bagging : le SMOTE Bagging, qui utilise la technique SMOTE pour traiter les problèmes de déséquilibre de classe, et l'UnderBagging, qui utilise des sous-échantillons de la classe majoritaire pour résoudre Problème[8].

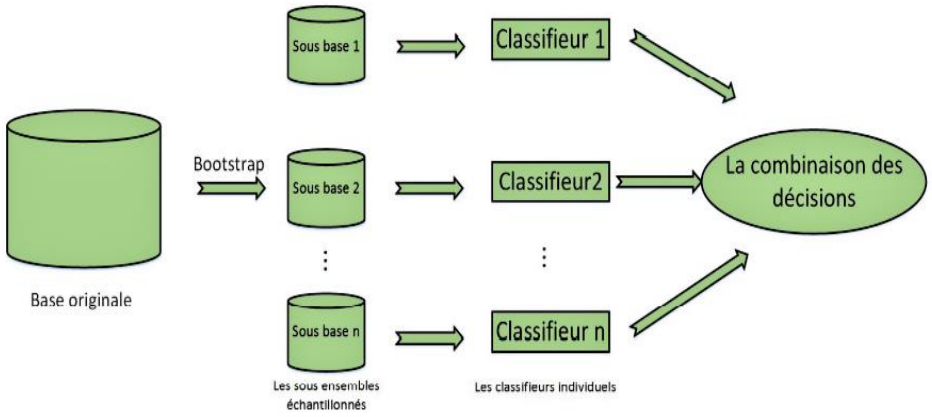


Figure 6.Principe de Bagging [8]

SMOTEBagging combine SMOTE et Bagging pour créer un ensemble de modèles formés sur différents sous-ensembles de données avec SMOTE appliqué à chaque sous-ensemble avant la formation.

Dans SMOTEBagging, la classe minoritaire est suréchantillonnée en utilisant SMOTE pour créer de nouveaux exemples synthétiques, qui sont ensuite utilisés pour créer les différents sous-ensembles de données pour Bagging. Chacun des modèles de l'ensemble est formé sur un sous-ensemble différent des données suréchantillonnées, puis combiné par la moyenne ou le vote. Il en résulte un modèle plus équilibré et plus précis qui donne de bons résultats tant pour la majorité que pour la minorité.

La deuxième technique est UnderBagging est une méthode d'ensemble qui combine sous-échantillonnage et Bagging. Elle consiste à créer un ensemble de modèles formés sur différents sous-ensembles de données avec sous-échantillonnage appliqué à chaque sous-ensemble avant la formation. Cette approche permet de réduire l'impact du déséquilibre des classes tout en préservant l'information dans la classe majoritaire [8].

- **Le renforcement (Boosting)**

Le Boosting est une technique d'apprentissage automatique visant à améliorer la précision d'un modèle en combinant les prédictions de plusieurs modèles plus faibles. Ces algorithmes fonctionnent de manière itérative en formant une séquence de modèles faibles. Chaque modèle suivant se focalise sur les points de données mal classés par les modèles précédents. En agrégeant les prédictions de ces modèles faibles, le modèle d'ensemble obtenu est souvent plus précis que chaque modèle individuel pris séparément [8].

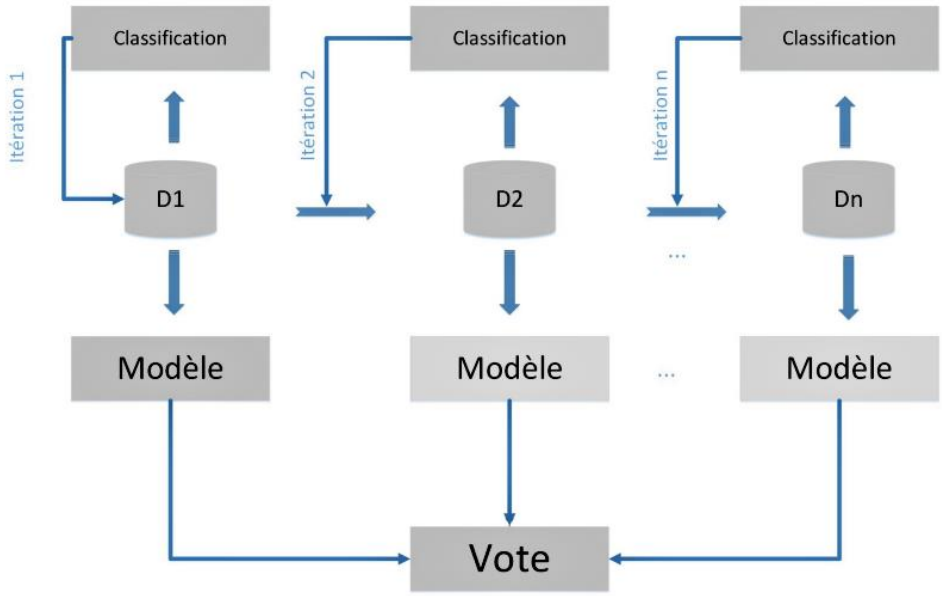


Figure 7.Principe de Boosting [8]

- **L'empilement (Stacking)**

Le stacking est une méthode d'ensemble pour l'apprentissage automatique supervisé qui combine les prédictions de plusieurs modèles de prédiction (ou de base learners) en un seul modèle pour améliorer la précision des prédictions. Contrairement aux méthodes d'ensemble telles que le voting ensemble ou le bagging, qui effectuent l'agrégation directe des prédictions des modèles de base, le stacking utilise une approche hiérarchique en ajoutant un modèle de méta-apprentissage pour combiner les prédictions des modèles de base.

Le processus de stacking consiste à diviser les données d'entraînement en plusieurs ensembles de données plus petits, puis à entraîner plusieurs modèles de base différents sur ces ensembles de données. Les prédictions de chaque modèle de base sont ensuite utilisées comme entrée pour un modèle de méta-apprentissage, qui apprend comment combiner les prédictions des modèles de base pour produire une prédiction finale plus précise. Le modèle de méta-apprentissage peut être entraîné sur un ensemble de données distinct de celui utilisé pour entraîner les modèles de base, Le schéma ci-dessous montre comment fonctionne le stacking [9].

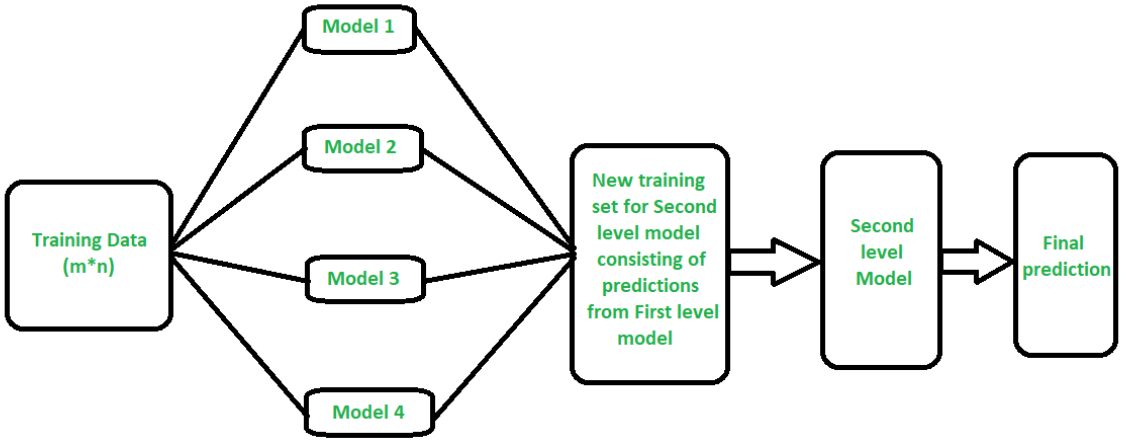


Figure 8.Principe de Stacking

- **L'ensemble de vote (voting)**

L'ensemble de vote (ou Voting Ensemble en anglais) est une méthode d'apprentissage par ensemble en apprentissage automatique, qui combine les prédictions de plusieurs modèles de prédiction pour produire une prédiction finale.

Dans le voting ensemble, chaque modèle de prédiction donne une prédiction pour une instance d'entrée donnée, et la prédiction finale est obtenue en agrégeant les votes de chaque modèle. Le modèle le plus couramment utilisé dans le voting ensemble est le classifieur binaire, où chaque modèle de prédiction peut prédire soit une classe positive, soit une classe négative [9].

Il existe deux types de voting ensemble :

1. Le vote majoritaire : Dans cette méthode, la prédiction finale est déterminée par la classe qui a reçu le plus de votes des modèles de prédiction. Si plusieurs classes ont le même nombre de votes, la prédiction finale peut être déterminée de manière aléatoire ou en utilisant des critères spécifiques tels que la confiance des modèles.

2. Le vote pondéré : Dans cette méthode, chaque modèle de prédiction est associé à un poids, qui peut être utilisé pour donner plus d'importance aux prédictions de certains modèles. La prédiction finale est déterminée en prenant une somme pondérée des prédictions des modèles, où

le poids de chaque modèle dépend de sa précision de prédiction On peut observer sur la figure ci-dessus comment l'ensemble de vote fonctionne :

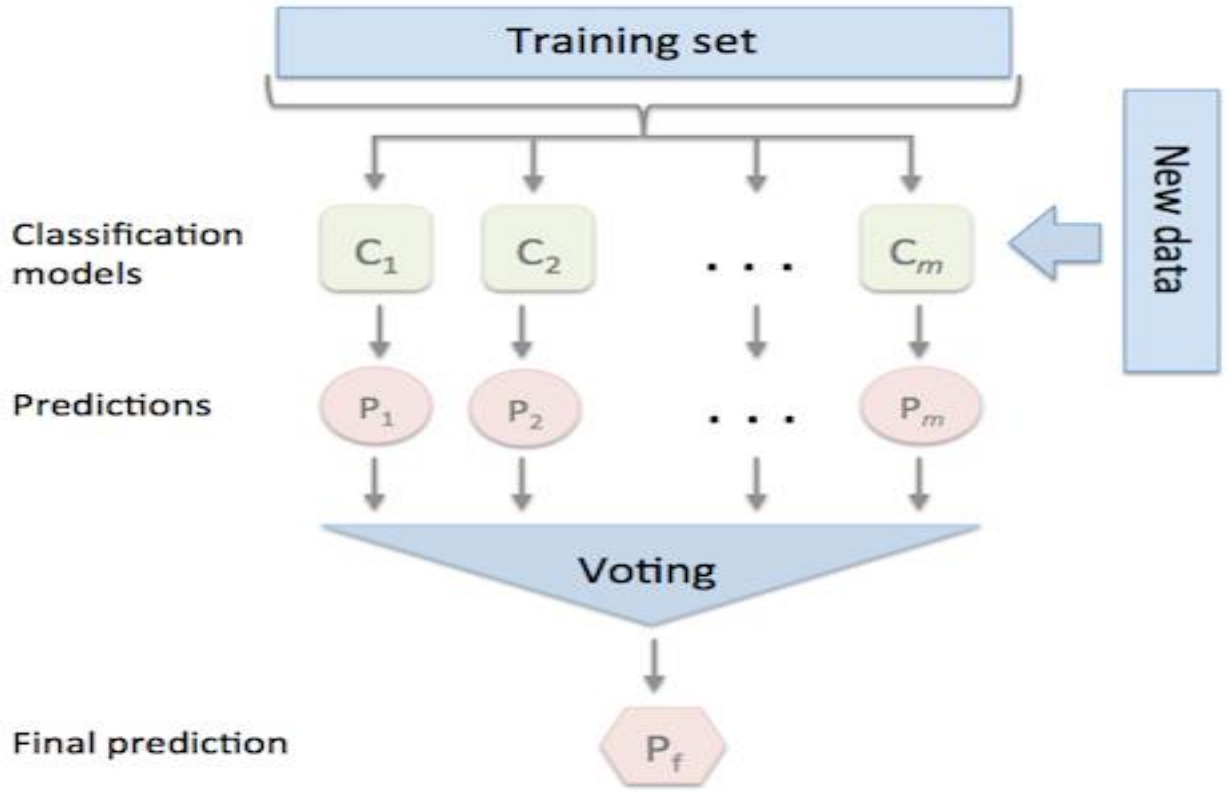


Figure 9.Principe de Voting Ensemble

- **Les forêts aléatoires (Random Forests)**

La méthode d'ensemble forêt aléatoire est une technique d'apprentissage automatique supervisé qui combine des arbres de décision aléatoires pour créer un modèle de prédiction.

La méthode forêt aléatoire fonctionne en créant un grand nombre d'arbres de décision aléatoires, où chaque arbre est construit en utilisant un sous-ensemble aléatoire de caractéristiques (variables) et un sous-ensemble aléatoire de données d'entraînement. Pour chaque arbre construit, le modèle utilise un critère de séparation de nœuds (par exemple, l'indice de Gini ou l'entropie) pour sélectionner la caractéristique qui permet de séparer les données d'entraînement de la manière la plus efficace possible.

Une fois que tous les arbres ont été construits, les prédictions finales sont calculées en agrégeant les prédictions de chaque arbre. Pour une classification binaire, par exemple, le modèle peut utiliser le vote majoritaire pour déterminer la classe prédite [9].

La figure ci-dessous illustre le fonctionnement de la forêt aléatoire

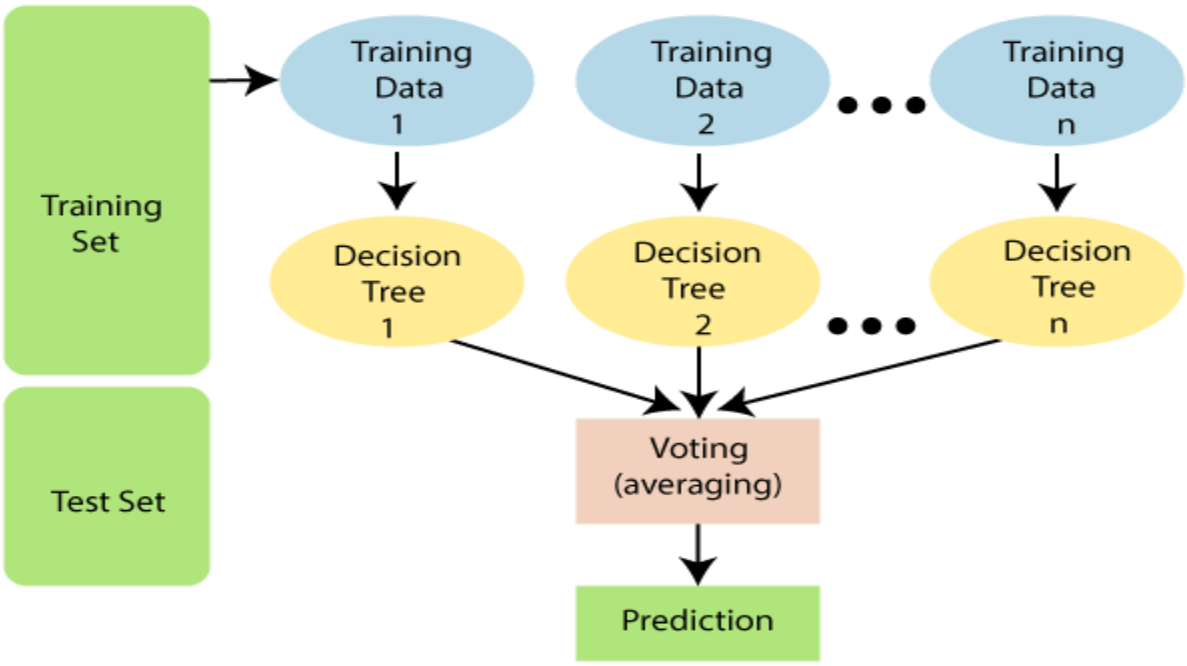


Figure 10.Principe de Foret aléatoire [12]

2.4.4 L'apprentissage sensible au coût (Cost sensitive learning)

L'apprentissage sensible au coût (ou Cost-sensitive learning en anglais) est une approche de l'apprentissage automatique supervisé qui prend en compte les coûts différents associés à la classification incorrecte des exemples dans un ensemble de données. Cette approche cherche à minimiser les coûts liés à des erreurs spécifiques, qui peuvent varier selon le contexte ou le domaine d'application [8].

Le principe de sensibilité de cout peut être appliqué à la plupart des algorithmes de classification tels que la régression logistique, SVM, les arbres binaires, le boosting ...etc.

Dans l'apprentissage sensible au coût, ces algorithmes sont modifiés pour prendre en compte les coûts différents associés aux erreurs de classification. Les poids de classe sont ajustés pour refléter ces coûts différents.

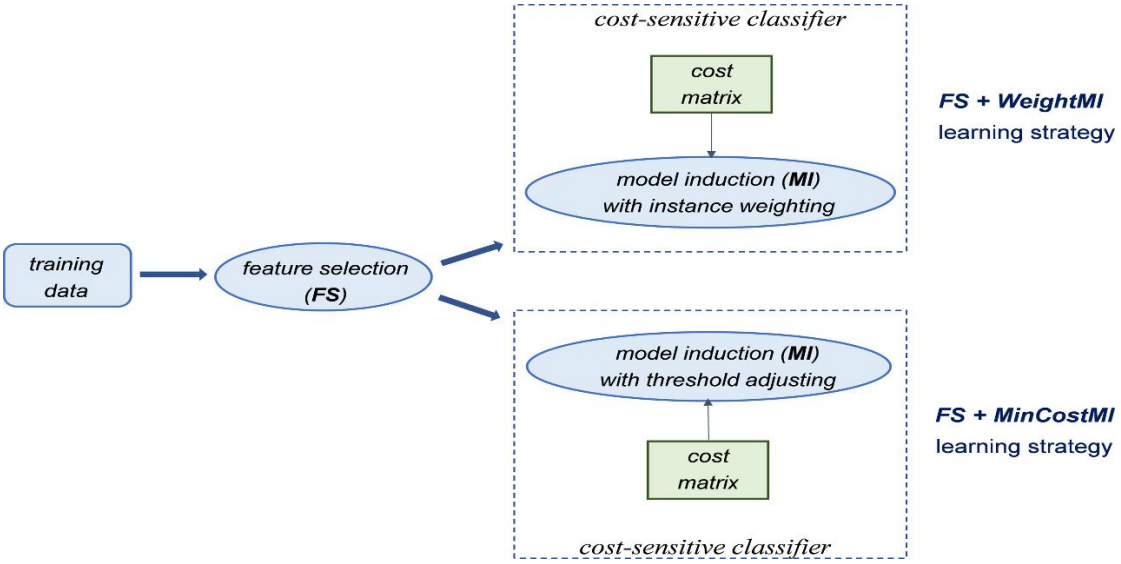


Figure 11.Principe du l'apprentissage sensible au cout [13]

2.4.5 L'apprentissage mono-classe

L'apprentissage mono-classe (ou One-class Learning en anglais) est une technique d'apprentissage automatique supervisé qui vise à modéliser une classe d'objets sans référence à d'autres classes. Contrairement à l'apprentissage multi-classes où l'on cherche à identifier les relations entre différentes classes, l'apprentissage mono-classe vise à identifier les caractéristiques qui définissent une seule classe d'objets.

L'objectif de l'apprentissage mono-classe est de construire un modèle qui peut reconnaître les objets appartenant à la classe spécifiée et rejeter les objets qui n'appartiennent pas à cette classe. Cette technique est souvent utilisée dans des applications telles que la détection de fraudes, la détection d'intrusions, la détection de spam ou la détection de défauts dans des systèmes.

Pour entraîner un modèle d'apprentissage mono-classe, on utilise des exemples d'objets qui appartiennent à la classe spécifiée. Le modèle apprend ensuite à caractériser cette classe d'objets en se basant sur des propriétés telles que la densité, la distance ou la séparation par rapport aux autres objets [8] .

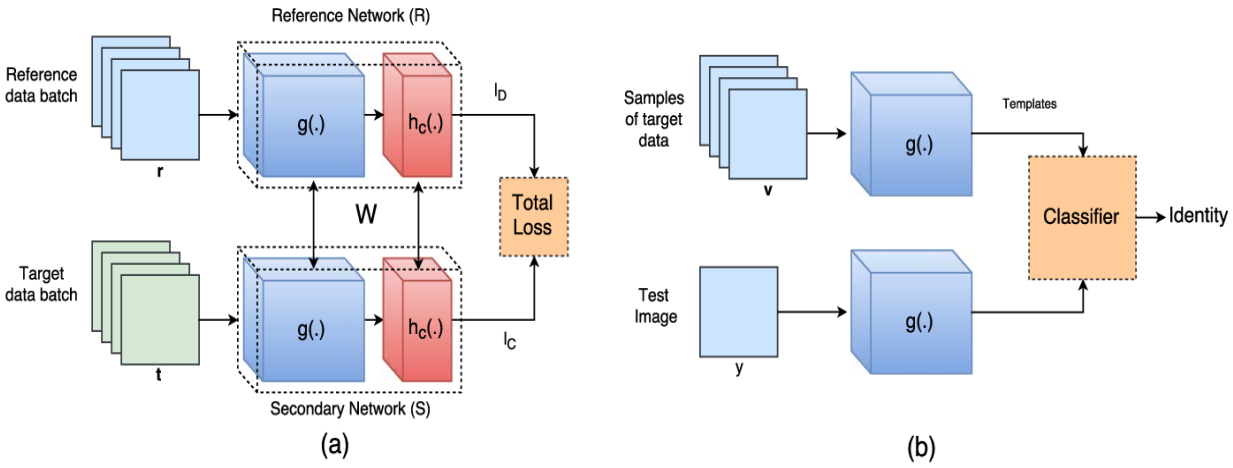


Figure 12.Principe du l'apprentissage mono-classe [14]

2.5 Conclusion

Au cours de ce chapitre, nous avons examiné les aspects les plus importants des différentes approches utilisées pour surmonter le problème de déséquilibre de données, notamment les méthodes d'échantillonnage et les méthodes d'ensemble, ainsi que l'apprentissage sensible au coût et l'apprentissage mono-classe.

Dans le chapitre suivant, nous présenterons les différentes étapes de conception et de réalisation de notre application.

Chapitre 03: Conception & réalisation

3.1 Introduction

Dans ce chapitre nous aborderons une description générale de notre système, en mettant en évidence son côté conceptuel et méthodologique. Nous détaillerons ensuite chaque étape de projet en citant les principaux algorithmes et techniques utilisées dans chacune des phases, et en précisant les différentes métriques d'évaluation utilisées pour déterminer leurs performances.

3.2 Description du projet

Notre objectif en lançant ce projet était la classification automatique des attitudes envers les rumeurs politiques en ligne tout en prenant en compte le problème de déséquilibre de données très commun pour cette catégorie des tâches de classification du texte. Nous avons implémenté et évalué les approches de classification les plus appropriées pour cette problématique, y compris l'apprentissage par ensemble, le sur-échantillonnage (l'augmentation de texte), l'apprentissage sensible au coût et là l'apprentissage mono-classe.

Ces approches de classification ont été appliquées sur un ensemble déséquilibré de données qui ont été recueillies pour le projet de Master intitulé « Propagation de la rumeur sur les médias sociaux» [15] [15] réalisé par RIGHI Mohamed el Manar et BOUSSAHEL Djalal Eddine lors de l'année académique 2020/2021, et publié ensuite dans les proceddings de la 5ème IEEE Conférence Internationale sur les Aspects Avancées de l'Ingénierie des Logiciels ICAASE'22 (The 5th IEEE International Conference on Advanced Aspects of Software Engineering, Constantine) [15].

Les données comprennent une collection de 3314 commentaires provenant de 11 vidéos publiées sur You Tube et recueillis à l'aide de l'API de You Tube. Ces commentaires ont exprimé les attitudes ou les positions des internautes algériens envers les rumeurs qui se sont déclenchées suite à l'absence prolongée du président algérien Abdelmadjid Tebboune, contaminé par la COVID-19, entre octobre 2020 et mars 2021.

3.3 Description de la méthodologie de conception

Notre projet a impliqué une série d'étapes pour arriver à l'application finale. La figure ci-dessous présente les étapes clés et importantes de notre travail :

a. Collection de données.

b. Annotation (étiquetage) manuelle des données collectées.

c. Prétraitement et préparation de données pour l'étape suivante.

d. Classification de texte avec différents méthodes d'apprentissage.

e. Evaluation des résultats obtenus.

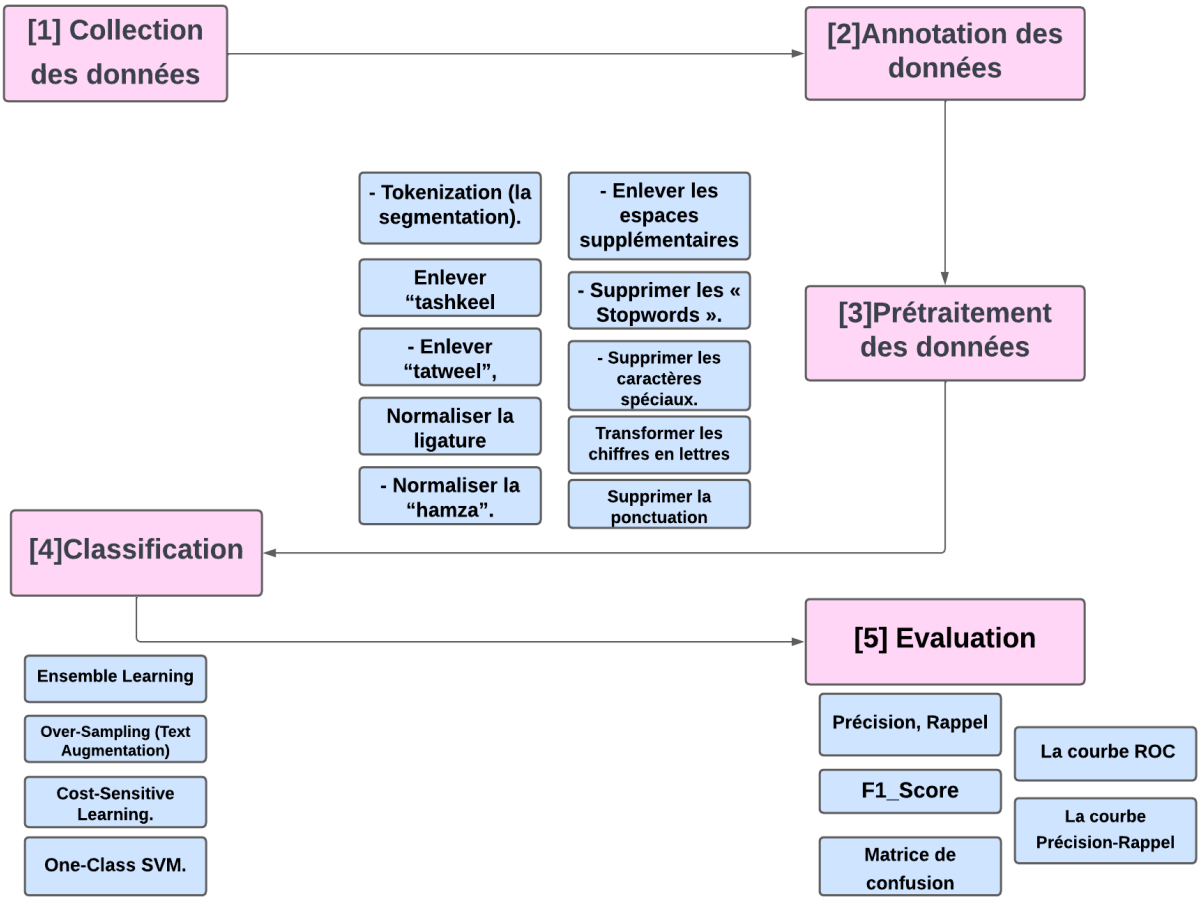


Figure 13.Architecture de système proposé

3.3.1 Collection et annotation de données

Le jeu de données comprend plus de 3314 commentaires collectés depuis la plateforme You Tube en utilisant l'API You Tube puis nettoyé et préparé pour l'annotation manuelle.

Nous avons entrepris d'annoter chaque commentaire en fonction de la position exprimée par le commentateur à l'égard du contenu de la vidéo et du diffuseur de la vidéo. Les commentaires ont été classifiés selon les six étiquettes suivantes :

- **Supporter** : Commentaire qui appuie une idée, un argument ou une opinion exprimée dans la vérité de la rumeur.
- **Refuser** : Commentaire rejetant une idée, un argument ou une opinion exprimée au sujet de la rumeur.
- **Requête** : Le commentaire pose des questions, demande des précisions ou des renseignements supplémentaires sur la vérité des rumeurs.
- **Commentaire :** l'auteur de la réponse fait son propre commentaire sans contribuer clairement à apprécier la véracité de la rumeur.
- **Commentaire positif :** qui désigne les commentaires où l'auteur ne soutient pas la véracité de la rumeur, mais soutient le diffuseur de la rumeur ou attaque ses détracteurs.
- **Commentaire négatif :** qui désigne les commentaires où l'auteur ne soutient pas la véracité de la rumeur et critique le diffuseur de la rumeur en personne ou attaque ses fans.

3.3.2 Prétraitement de données

Le processus de prétraitement de données est crucial dans la classification de texte car il vise à éliminer tout bruit présent dans le texte étudié, tel que les mots inutiles et redondants, les chiffres, les préfixes et suffixes, etc. Un prétraitement efficace des données peut améliorer les résultats de classification, indépendamment de la capacité du classificateur utilisé. Les principales étapes du prétraitement incluent :

- **Enlever les espaces supplémentaires** : une fonction qui permet de supprimer tous les espaces en trop ou inutiles dans un texte.

- **Supprimer les "stopwords"** : une fonction qui permet d'éliminer les mots courants qui ne contribuent pas au sens du texte, comme "le", "la", "de", "un", "une", etc. Toutefois, il est important de noter que la liste des stopwords peut varier selon la langue utilisée. En arabe, les stopwords courants incluent des mots tels que "إلى", "من", "على", "في", "و", etc.

- **Supprimer les caractères spéciaux** : une fonction qui permet de supprimer tous les caractères spéciaux tels que les symboles de ponctuation, les parenthèses, les guillemets, les tirets, les points d'interrogation, etc.

- **Transformer les chiffres en lettres** : une fonction qui permet de convertir tous les chiffres présents dans le texte en leur équivalent en lettres.

- **Supprimer les URLs** : une fonction qui permet de supprimer tous les liens URL présents dans le texte.

- **Supprimer les mentions (@)** : une fonction qui permet de supprimer toutes les mentions d'utilisateurs Twitter ou d'autres réseaux sociaux, qui sont souvent précédées du symbole "@".

- **Supprimer les hashtags** : une fonction qui permet de supprimer tous les hashtags présents dans le texte, qui sont souvent précédés du symbole "#" et utilisés pour identifier un sujet ou une thématique spécifique.

- **Supprimer les Emojis** : une fonction qui permet de supprimer tous les emojis présents dans le texte, qui sont souvent utilisés pour exprimer des émotions ou des sentiments dans les messages sur les réseaux sociaux.

- **Supprimer la ponctuation** : c'est une fonction qui supprime tous les signes de ponctuation d'une chaîne de caractères, tels que les virgules, les points, les points-virgules, les points d'exclamation, etc.

- **Enlever "tashkeel"** : cette fonction consiste à supprimer les diacritiques (points, traits, etc.) qui sont souvent utilisés pour indiquer la prononciation correcte des mots dans certaines langues telles que l'arabe.

- **Enlever "tatweel"** : cette fonction vise à supprimer les caractères qui étirent une lettre ou un mot, souvent utilisés pour des raisons stylistiques ou esthétiques dans certaines langues telles que l'arabe.

- **Normaliser la ligature** : cette fonction consiste à remplacer les lettres qui sont normalement liées dans certaines langues (comme le français ou l'italien) par leur forme séparée, par exemple en français remplacer le "œ" par "oe" et en arabe remplacer la lettre "ل" lorsqu'elle est liée à la lettre "ا" par la lettre "لا" par sa forme séparée "ل" suivie de "ا". De même, remplacer la lettre "ل" lorsqu'elle est liée à la lettre "ي" par la lettre "ليـ" par sa forme séparée "ل" suivie de "ي".

- **Normaliser la "hamza"** : cette fonction vise à normaliser les différentes formes de la lettre "hamza" qui est utilisée dans certaines langues telles que l'arabe et qui peut avoir différentes formes selon sa position dans un mot ou selon la manière dont elle est utilisée.

- **Tokenization (la segmentation)** : c'est une fonction qui découpe une chaîne de caractères en mots individuels, ou "tokens", afin de faciliter leur traitement et leur analyse. Cette fonction peut prendre en compte différents critères pour déterminer les limites entre les mots, tels que les espaces, les signes de ponctuation, les caractères spéciaux, etc.

3.3.3 Extraction des caractéristiques

Après avoir effectué le prétraitement des données, nous avons procédé à la vectorisation du texte en utilisant la technique de la TF-IDF.

Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF est une technique couramment utilisée en traitement de texte pour la vectorisation du texte. Elle permet de quantifier l'importance d'un terme dans un document en fonction de sa fréquence d'apparition et de sa rareté dans l'ensemble des documents. La formule de TF-IDF est donnée par [16]:

TF-IDF = TF * IDF

Où :

- **TF (Term Frequency)** représente la fréquence du terme dans le document donné. Elle est calculée en divisant le nombre d'occurrences du terme dans le document par le nombre total de termes dans le document.

- **IDF (Inverse Document Frequency)** mesure l'importance du terme dans l'ensemble de la collection de documents. Elle est calculée en prenant le logarithme inverse de la fraction du nombre total de documents sur le nombre de documents contenant le terme.

3.4 Méthodes utilisées pour la classification du texte

3.4.1 Méthodes de base

- **Machine à vecteurs de support (SVM)**

Le principe de base des SVM est de trouver une frontière de décision qui sépare les exemples d'une classe de ceux d'une autre classe. Cette frontière de décision est choisie de manière à maximiser la marge, c'est-à-dire la distance entre la frontière de décision et les exemples les plus proches de chaque classe, Le pseudo-algorithme de SVM est donné ci-dessous :

Algorithme 1:Pseudo-code de la méthode SVM

1. représenter chaque point de données par un vecteur caractéristique dans un espace n-dimensionnel.
2. ajuster les caractéristiques pour qu'elles aient des plages similaires.
3. choisir un hyperplan aléatoire pour commencer l'optimisation.
4. identifier les points de données les plus proches de l'hyperplan, appelés vecteurs de support, et calculer la marge.
5. effectuer un processus d'optimisation pour trouver l'hyperplan optimal en minimisant l'erreur de classification et maximisant la marge.
6. utiliser le truc du noyau pour mapper les données dans un espace de dimensions supérieures si elles ne sont pas linéairement séparables.
7. choisir une fonction de noyau appropriée, telle que linéaire, polynomiale, RBF ou sigmoïde.
8. évaluer les performances du modèle en utilisant des mesures telles que l'exactitude, la précision, le rappel et le score F1.

- **Régression logistique**

La régression logistique est un algorithme de classification qui utilise une fonction logistique pour calculer la probabilité d'appartenance d'un échantillon à une classe donnée. La fonction logistique est une fonction en S qui prend une valeur d'entrée x et produit une sortie comprise entre 0 et 1. Ci-dessous le pseudo-algorithme de la régression logistique :

Algorithme 2: Pseudo-code de la méthode régression logistique

1. Préparer l'ensemble de données étiqueté avec les caractéristiques et les étiquettes de classe binaire.
2. Mettre les caractéristiques à l'échelle pour s'assurer qu'elles ont des plages similaires.
3. Utiliser la fonction sigmoïde pour transformer la sortie en une valeur de probabilité entre 0 et 1.
4. Modéliser la probabilité du résultat binaire à l'aide de la fonction d'hypothèse qui applique la fonction sigmoïde à la combinaison linéaire de valeurs et de poids des caractéristiques.
5. Choisir une fonction de perte appropriée, comme la perte logistique, pour quantifier la différence entre les probabilités prévues et les étiquettes de classe réelles.
6. Estimer les valeurs optimales pour le vecteur de poids et le terme de biais en minimisant la fonction de perte à l'aide d'algorithmes d'optimisation comme la descente en gradient ou la méthode de Newton.
7. Établir une limite décisionnelle qui sépare les deux classes de l'espace de présentation.
8. Prédire la probabilité du résultat binaire pour les nouvelles données en appliquant les poids et le biais formés à la fonction de l'hypothèse et en classant en fonction d'un seuil choisi.

- **Naïve de Bayes**

Le Naïve Bayes est un algorithme de classification basé sur la théorie de Bayes. Il utilise les probabilités conditionnelles des caractéristiques (ou attributs) d'un échantillon pour déterminer la classe à laquelle il appartient. Le terme "naïf" fait référence à l'hypothèse simplificatrice selon laquelle les caractéristiques sont indépendantes les unes des autres, ce qui peut ne pas être vrai dans la pratique, mais qui permet des calculs plus simples et plus rapides, Voici le pseudo-code de Naïve Bayes :

Algorithme 3: Pseudo-code de la méthode Naïve de bayes

1. Recueillir un ensemble de données étiqueté avec les caractéristiques et les étiquettes de classe correspondantes.
2. Calculer la probabilité antérieure de chaque classe dans l'ensemble de données.
3. Pour chaque caractéristique, calculer la probabilité d'observer cette caractéristique, compte tenu de chaque classe de l'ensemble de données.
4. Calculer la probabilité conditionnelle de chaque classe compte tenu des caractéristiques observées à l'aide du théorème de Bayes.
5. Faire l'hypothèse naïve que les caractéristiques sont conditionnellement indépendantes compte tenu de la classe. Bien que cette hypothèse soit souvent violée dans la réalité, Bayes naïve peut encore bien fonctionner dans la pratique.
6. Calculer la probabilité conjointe des caractéristiques observées pour chaque classe en multipliant les probabilités conditionnelles.
7. Calculer la probabilité postérieure de chaque classe en multipliant la probabilité conjointe par la probabilité antérieure de la classe.
8. Attribuer l'étiquette de classe ayant la probabilité postérieure la plus élevée comme classe prédite pour les nouvelles instances.

3.4.2 Apprentissage par ensemble (Ensemble Learning)

L'idée principale derrière l'apprentissage par ensemble est que la combinaison de plusieurs modèles peut réduire les biais, la variance et les erreurs de chaque modèle individuel. Ci-dessous les pseudo-codes des méthodes d'apprentissage par ensemble que nous avons implémentés dans notre application.

- **Le Bagging**

Algorithme 4:Pseudo-code de l'algorithme Bagging

1. Préparer un ensemble de données étiqueté avec les caractéristiques et les étiquettes de classe ou les valeurs cibles correspondantes.
2. Sélectionner au hasard des sous-ensembles, avec remplacement, à partir de l'ensemble de données original pour créer plusieurs ensembles de données de formation.
3. Former un modèle de base sur chaque échantillon bootstrap indépendamment.
4. Pour les tâches de classification, combiner les prédictions par vote majoritaire; pour les tâches de régression, les prédictions agrégées par moyenne ou par une autre méthode appropriée.

5. En combinant les prévisions de plusieurs modèles, l'ensachage réduit la variance et améliore la précision et la robustesse globales.
6. Agréger les prévisions des modèles de base à l'aide d'une méthode appropriée comme le vote majoritaire ou la moyenne.
7. Optimiser le rendement de l'ensemble d'ensachage en réglant des hyperparamètres comme le nombre de modèles de base ou la taille des échantillons bootstrap.
8. Utiliser l'ensemble d'ensachage formé pour faire des prédictions pour de nouvelles données invisibles en agrégeant les prédictions des modèles de base.

- **Le Boosting**

Algorithme 5: Pseudo-code de l'algorithme Boosting

1. Préparation des données étiquetées.
2. Attribution de poids égaux à toutes les instances de l'ensemble de données de formation.
3. Formation d'un modèle de base, souvent un apprenant faible, sur l'ensemble de données de formation.
4. Utilisation de l'apprenant faible pour faire des prédictions sur l'ensemble de données de formation et calcul de l'erreur de prédiction.
5. Augmentation des poids des instances mal classées pour leur donner une plus grande influence lors des étapes suivantes.
6. Combinaison des prédictions de tous les apprenants faibles en utilisant le vote pondéré ou la moyenne pondérée en fonction de leurs performances.
7. Répétition des étapes 3 à 6 pour un nombre prédéfini d'itérations ou jusqu'à ce qu'un seuil de performance soit atteint, en ajustant les poids des instances à chaque itération.
8. Obtention de la prédiction finale en agrégeant les prédictions de tous les apprenants faibles en fonction de leurs poids respectifs.

- **Le Stacking**

Algorithme 6:Pseudo-code de l'algorithme Stacking

1. Préparation des données étiquetées.
2. Formation de plusieurs modèles de base.
3. Utilisation des modèles de base pour générer de nouvelles caractéristiques.
4. Formation d'un méta-modèle à partir des nouvelles caractéristiques.
5. Combinaison des prédictions des modèles de base par le méta-modèle.
6. Répétition du processus pour plusieurs couches si nécessaire.
7. Utilisation de l'ensemble empilé pour faire des prédictions sur de nouvelles données.

- **Ensemble de vote**

Algorithme 7:Pseudo-code de l'algorithme ensemble de vote

1. Préparer un ensemble de données étiqueté avec les caractéristiques et les étiquettes de classe ou les valeurs cibles correspondantes.
2. Former plusieurs modèles individuels en utilisant différents algorithmes ou configurations sur l'ensemble de données de formation.
3. Utiliser les modèles formés pour faire des prédictions sur l'ensemble de données de validation ou d'essai.
4. Chaque modèle apporte sa prédiction à l'ensemble.
5. Déterminer la prédiction finale en choisissant l'étiquette de classe ou le résultat qui reçoit la majorité des votes des modèles individuels.
6. Dans le cas d'une égalité, vous pouvez utiliser différentes stratégies de bris d'égalité, comme le choix de l'étiquette de la classe ayant la cote de confiance la plus élevée ou le recours à la prédiction d'un modèle particulier.
7. Évaluer le rendement de l'ensemble de vote en comparant ses prévisions avec les véritables étiquettes de classe ou les valeurs cibles.
8. L'ensemble de vote peut être appliqué aux problèmes de classification et de régression, avec différentes stratégies de vote, comme le vote ferme (vote majoritaire) ou le vote mou (moyenne pondérée des probabilités ou des scores).

- **Arbre de décision**

Algorithme 8: Pseudo-code de l'algorithme Arbre de décision

1. Préparation des données étiquetées avec les caractéristiques et les étiquettes de classe ou les valeurs cibles correspondantes.
2. Sélection de la meilleure fonction (caractéristique) dans l'ensemble de données pour servir de nœud racine de l'arbre de décision, basée sur des critères tels que le gain d'information ou l'impureté de Gini.
3. Division de l'ensemble de données en sous-ensembles en fonction de la fonction sélectionnée, chaque sous-ensemble représentant une valeur ou une plage unique de cette fonction.
4. Répétition récursive des étapes 2 et 3 pour chaque sous-ensemble, en choisissant la meilleure fonction à chaque niveau et en créant des sous-ensembles plus petits jusqu'à ce qu'un critère d'arrêt soit satisfait (par exemple, une limite de profondeur ou un nombre minimum d'échantillons dans un nœud).
5. Attribution d'une étiquette de classe ou d'une valeur prédite à chaque nœud de feuille en fonction de la classe majoritaire ou de la valeur moyenne des échantillons dans ce nœud.
6. Construction d'un arbre de décision complet qui peut être utilisé pour faire des

prédictions sur de nouvelles données en parcourant l'arbre à partir de la racine en fonction des valeurs des caractéristiques, en suivant les branches appropriées jusqu'à atteindre un nœud de feuille et en retournant l'étiquette de classe ou la valeur prédite associée.

- **Les forêts aléatoires**

Algorithme 9: Pseudo-code de la méthode foret aléatoire

1. Préparer un ensemble de données étiqueté avec les caractéristiques et les étiquettes de classe ou les valeurs cibles correspondantes.
2. Sélectionner au hasard des sous-ensembles de l'ensemble de données (avec remplacement) pour créer plusieurs ensembles de données de formation, appelés échantillons bootstrap.
3. Construire un arbre de décision sur chaque échantillon bootstrap en utilisant un sous-ensemble aléatoire de caractéristiques à chaque nœud.
4. Pour les tâches de classification, chaque arbre prédit indépendamment l'étiquette de classe, et la classe avec la majorité des votes devient la prédiction finale. Pour les tâches de régression, les arbres prédisent une valeur continue, et la prédiction finale est souvent la moyenne ou la médiane des prédictions de chaque arbre.
5. Combiner les prédictions de tous les arbres pour faire une prédiction finale.
6. Calculer l'importance de chaque caractéristique en fonction de la mesure dans laquelle la précision diminue lorsque la caractéristique est permutée au hasard.
7. Optimiser le rendement du modèle de forêt aléatoire en réglant des hyperparamètres comme le nombre d'arbres, la profondeur des arbres et le nombre de caractéristiques considérées à chaque split.
8. Utiliser le modèle de forêt aléatoire pour faire des prédictions pour de nouvelles données invisibles en agrégeant les prédictions de tous les arbres.

3.4.3 Le sur--échantillonnage (Over-Sampling)

Pour augmenter le nombre des échantillons des classes minoritaires dans notre corpus, nous avons appliqué les techniques de suppression aléatoire de mots, insertion de caractères aléatoires, transposition de mots, remplacement par synonyme, et la randomisation.

- **La suppression aléatoire de mots** : est une technique d'augmentation de données en traitement de texte qui consiste à supprimer des mots aléatoirement dans les phrases du corpus d'entraînement pour créer de nouveaux exemples d'entraînement.

- **L'insertion de caractères aléatoires :** Cette technique consiste à ajouter des caractères aléatoires (par exemple des lettres, des chiffres, des signes de ponctuation, etc.) à des endroits aléatoires dans le texte original, afin de créer des exemples d'apprentissage supplémentaires qui sont similaires, mais légèrement différents des exemples d'origine.
- **Transposition de mots :** est une technique d'augmentation de texte qui consiste à échanger l'ordre de deux mots consécutifs dans une phrase. Cette technique permet de générer de nouvelles variantes de phrases qui ont une signification similaire à la phrase d'origine. Par exemple, pour la phrase "Le chat noir dort sur le tapis", la transposition de mots peut donner "Le noir chat dort sur le tapis" ou "Le chat dort noir sur le tapis"
- **Augmentation par synonyme :** La technique d'augmentation de texte par synonymes consiste à remplacer certains mots dans une phrase par des synonymes. L'objectif est de générer de nouvelles phrases qui ont une signification similaire à la phrase d'origine, mais avec une variété de vocabulaire.
- **Randomisation :** La technique d'augmentation de texte par randomisation consiste à remplacer certains mots dans une phrase par des mots choisis au hasard dans un dictionnaire ou une liste de mots préétablie. L'objectif est de générer de nouvelles phrase qui ont une structure grammaticale similaire à la phrase d'origine, mais avec des mots différents.

3.4.4 L'apprentissage sensible au coût (Cost-Sensitive Learning)

Dans cette catégorie, nous avons implémenté les méthodes de classification suivantes : la régression logistique, SVM, les arbres de décision, et Gradient Boosting avec XGBoost.
Voici un pseudo-code étape par étape du fonctionnement de l'algorithme d'apprentissage sensible aux coûts :

Algorithme 10 : Pseudo-code de l'approche apprentissage sensible au cout

1. Préparer un ensemble de données étiqueté avec les caractéristiques et les étiquettes de classe ou les valeurs cibles correspondantes.
2. Définir les coûts de défaut de classification pour différentes catégories ou instances. Ces coûts peuvent être attribués en fonction de la connaissance du domaine ou de l'importance de prévoir correctement chaque classe.
3. Choisir un algorithme d'apprentissage de base, comme les arbres de décision, la

régression logistique ou SVM, qui peut être modifié pour intégrer l'approche sensible aux coûts.
4. Ajuster l'algorithme d'apprentissage pour tenir compte des coûts de classification erronée pendant le processus de formation. Cela peut se faire en modifiant la fonction objective ou en ajustant le seuil de décision.
5. Former le modèle à l'ensemble de données sensibles aux coûts, où les coûts de classification erronée sont pris en compte au cours du processus d'apprentissage. Le modèle apprend à minimiser le coût total des erreurs de classification.
6. Évaluer le rendement du modèle sensible aux coûts à l'aide de mesures d'évaluation appropriées, comme l'exactitude, la précision, le rappel ou des mesures fondées sur les coûts.
7. Ajuster les coûts de classification erronée au besoin, en fonction du rendement du modèle. L'ajustement des coûts peut aider à atteindre un meilleur équilibre entre les différents types d'erreurs.
8. Utiliser le modèle adapté aux coûts pour faire des prévisions sur de nouvelles données invisibles. Le modèle tient compte des coûts de classification erronée pendant la phase de prévision pour prendre des décisions qui minimisent le coût global.

3.4.5 L'apprentissage mono-classe

Le pseudocode suivant illustre le principe de l'apprentissage mono-classe. Dans cette catégorie de méthodes de classification, nous avons choisi les SVM mono-classe pour l'objectif de notre projet.

Algorithme 11.Pseudo-code de l'approche apprentissage mono-classe

1. Préparer un jeu de données contenant uniquement des instances positives appartenant à la classe cible. Le jeu de données ne contient pas d'instances provenant d'autres classes.
2. Choisir un algorithme d'apprentissage approprié à une seule classe, comme les machines vectorielles de soutien à une classe (SVM), les forêts aléatoires à une seule classe ou les approches basées sur les voisins les plus proches.
3. Former le modèle d'apprentissage en classe unique en utilisant les instances positives. Le modèle apprend les caractéristiques et les limites de la classe cible.
4. Pendant la formation, l'algorithme vise à créer une limite de décision ou une représentation de la classe cible qui encapsule la majorité des cas positifs tout en minimisant l'influence des valeurs aberrantes ou des cas provenant d'autres classes.
5. Évaluer le rendement du modèle d'apprentissage à classe unique à l'aide de mesures appropriées, comme la précision, le rappel, le score F1 ou le secteur sous la courbe ROC (AUC).
6. Affiner le modèle si nécessaire en ajustant les hyper paramètres ou en explorant différents algorithmes pour améliorer ses performances.
7. Utiliser le modèle d'apprentissage en classe unique pour faire des prédictions sur de nouvelles données invisibles. Le modèle détermine si les instances appartiennent à la

classe cible en fonction de leur similarité avec la représentation apprise.

3.5 Mesures des performances

Les métriques d'évaluation souvent utilisées pour mesurer la performance d'un modèle de classification de texte sont les suivants :

- **La précision** : c'est la proportion de prédictions positives correctes par rapport à toutes les prédictions positives. Elle mesure la capacité du modèle à identifier correctement les exemples positifs peut être représentée par la formule [17] :
  - ✓ **Précision = True Positives / (True Positives + False Positives)** ou
  - ✓ $Précision = \frac{TP}{TP+FP}$

- **Le rappel :** (en anglais "recall") c'est la proportion de prédictions positives correctes par rapport à toutes les instances positives réelles. Il mesure la capacité du modèle à trouver tous les exemples positifs peut être représentée par la formule [17] :
  - ✓ **Recall = True Positives / (True Positives + False Negatives)** ou
  - ✓ $Rappel = \frac{TP}{TP+FN}$

- **Le F1_score** : c'est une mesure harmonique de la précision et du rappel, qui donne une valeur unique qui combine les deux métriques peut être représentée par la formule [17] :
  - ✓ $F1\text{−}Score = 2* (Precision*Recall)/(Precision+Recall)$ ou
  - ✓ $F1\text{−}Score = \frac{2*P*R}{P+R}$
- **La matrice de confusion** : elle est utilisée pour visualiser la performance d'un modèle de classification. Elle montre le nombre de vrais positifs, de vrais négatifs, de faux positifs et de faux négatifs peut être représentée comme ce tableau [17] :

Tableau 1.Matrice de confusion

	Classe Positive (réelle)	**Classe Négative (réelle)**
Classe Positive (prédite)	True Positives (TP)	False Positives (FP)
Classe Négative (prédite)	False Negatives (FN)	True Negatives (TN)

- **La courbe ROC :** elle est utilisée pour évaluer la performance d'un modèle de classification binaire en traçant le taux de vrais positifs par rapport au taux de faux positifs pour différents seuils de classification peut être représentée par la formule [17] :
  - ✓ **TPR(True Positive Rate)=** $\frac{TP}{TP+FN}$
  - ✓ **FPR(False Positive Rate) =** $\frac{FP}{TN+FP}$
- **La courbe Précision-Rappel** : elle est utilisée pour évaluer la performance d'un modèle de classification binaire en traçant la précision par rapport au rappel pour différents seuils de classification. Elle est utile lorsque la distribution des classes n'est pas équilibrée [17].

3.6 Conclusion

Au cours de ce chapitre, nous avons exposé les objectifs de notre projet ainsi que la conception de notre système. Nous avons également présenté les méthodes implémentées dans ce projet dans ses différentes phases, ainsi que les métriques d'évaluation de la qualité de classification.

Dans le prochain chapitre, nous approfondirons les aspects techniques de la mise en œuvre et présenterons les résultats obtenus, ainsi que les perspectives pour l'avenir.

Chapitre 04 : Implémentation & Expérimentations

4.1 Introduction

Ce chapitre se compose de deux parties. Tout d'abord, nous exposons l'environnement de développement et les outils qui ont été employés pour la création du système de classification, ainsi que les bibliothèques intégrées. La seconde partie est dédiée à la présentation et la discussion des résultats obtenus.

4.2 Environnement et outils d'implémentation

4.2.1 Matériel

Tableau 2.Caractéristiques du matériels utilisé

Caractéristiques	**Poste de travail N°01**	**Poste de travail N°02**
PC	Lenovo	Lenovo
Système d'exploitation	Windows 10 Professionnel	Windows 10 Professionnel
Processeur	Intel(R) Core(TM) i5-2540M CPU @ 2.60GHz 2.60 GHz	Intel(R) Pentium(R) CPU B950 @2.10 GHz 2.10 GHz
RAM	4,00 Go	4,00 Go
Type de système	SE 64 bits	SE 64 bits

4.2.2 Langage de programmation Python

Nous avons opté pour le langage de programmation Python pour la réalisation de ce projet. Python est un langage polyvalent et interprété, reconnu pour sa syntaxe claire et lisible, ce qui facilite la compréhension du code. Il est largement utilisé dans divers domaines tels que le développement web, l'analyse de données, l'intelligence artificielle et l'automatisation des tâches. Python bénéficie d'une vaste bibliothèque standard qui offre une multitude de fonctionnalités prêtes à l'emploi. Sa popularité croissante, sa communauté active et les nombreuses ressources en ligne disponibles en font un choix privilégié pour les développeurs du monde entier [18].

4.2.3 Environnement de programmation

Nous avons travaillé sur deux environnements de programmation pour développer ce projet :

- **Jupyter notebook :** Jupyter notebook est un environnement de programmation interactif basé sur le Web qui permet aux utilisateurs de créer des documents contenant du code, des textes, des images, des graphiques et des équations mathématiques. Les documents Jupyter Notebook sont créés à l'aide de cellules, où chaque cellule peut contenir du code exécutable, du texte formaté en markdown, ou des médias. Les utilisateurs peuvent exécuter les cellules séparément et voir immédiatement les résultats affichés dans le document. Jupyter Notebook prend en charge plusieurs langages de programmation, y compris Python, R et Julia, ainsi que de nombreuses bibliothèques populaires pour l'analyse de données, l'apprentissage automatique, la visualisation de données et bien plus encore [19].
- **Google Colaboratory** : également connu sous le nom de Google Colab, est un environnement de développement basé sur le cloud qui permet aux utilisateurs d'exécuter du code Python dans leur navigateur. Il offre une plateforme gratuite pour l'exécution de code Python, en fournissant des ressources de calcul et d'accès aux bibliothèques populaires telles que TensorFlow, PyTorch, et Pandas.

  En utilisant Google Colaboratory, les utilisateurs peuvent créer et exécuter des notebooks Jupyter, collaborer avec d'autres utilisateurs en temps réel et profiter des avantages de l'intégration avec d'autres services Google tels que Google Drive et GitHub. Cela permet aux utilisateurs de travailler sur des projets de machine learning, de data science et d'autres tâches de développement en bénéficiant de la puissance de calcul et des fonctionnalités offertes par le cloud de Google [20].

Figure 14. Jupyter et Google Colaboratory

4.2.4 Les principaux packages Python utilisés

Parmi les packages que nous avons utilisés pour mener à bien ce projet, il y avait :

- **nlpaug** : est une bibliothèque open-source pour Python qui fournit des outils d'augmentation de données pour le traitement naturel du langage (NLP). Cette bibliothèque peut être utilisée pour augmenter des données de texte pour diverses tâches de NLP telles que la classification de texte, l'analyse des sentiments, la traduction automatique, la génération de texte, etc. nlpaug propose plusieurs techniques d'augmentation de données pour NLP, telles que la substitution de mots, l'insertion de mots, la suppression de mots, la permutation de mots, la transformation de clavier, la correction orthographique, etc [21] .
- **Pandas** : est une bibliothèque open-source pour Python qui propose des outils et des structures de données pour l'analyse de données, spécifiquement pour la manipulation de données tabulaires et des séries temporelles. Cette bibliothèque fournit des fonctionnalités pour le traitement et l'analyse des données, comme la sélection, le filtrage, le tri, la fusion, la transformation et l'agrégation des données [22].
- **NLTK (Natural Language Toolkit)** : est une bibliothèque open-source pour le langage de programmation Python, qui fournit des outils pour le traitement automatique du langage naturel (NLP). NLTK contient des modules pour effectuer diverses tâches de NLP, telles que le traitement de texte brut, la tokenisation, la lemmatisation, l'analyse de la syntaxe, l'analyse sémantique et la classification de texte. Cette bibliothèque est largement utilisée dans la recherche en NLP, l'enseignement et l'industrie pour développer des applications basées sur le traitement du langage naturel [23].
- **PyArabic** : est une bibliothèque open-source pour le langage de programmation Python, qui fournit des outils pour le traitement automatique de la langue arabe. PyArabic permet de manipuler facilement des textes arabes, notamment la segmentation de mots, la normalisation de texte, la translittération, la dérivation de racines, la génération de formes fléchies et la correction orthographique. Cette bibliothèque est largement utilisée dans les domaines de la recherche en linguistique arabe, l'enseignement et le développement d'applications basées sur le traitement de texte arabe [24].

- **Arabic-Stopwords :** est une bibliothèque open-source pour le langage de programmation Python, qui fournit une liste de mots vides pour la langue arabe. Les mots vides, également connus sous le nom de stopwords, sont des mots courants qui sont souvent exclus lors de l'analyse de texte, car ils n'apportent généralement pas de valeur sémantique importante. Arabic-Stopwords fournit une liste complète de mots vides pour l'arabe, qui peuvent être utilisés pour nettoyer et prétraiter des textes arabes avant de les analyser avec d'autres outils de traitement de texte [25].
- **Gensim** : est une bibliothèque open-source pour le langage de programmation Python, qui fournit des outils pour le traitement automatique du langage naturel, tels que la modélisation de sujets, la vectorisation de texte et la similarité sémantique. Gensim est particulièrement utile pour l'analyse de grands corpus de texte, et propose des algorithmes de modélisation de sujets tels que LSI (Latent Semantic Indexing) et LDA (Latent Dirichlet Allocation) [26].
- **Scikit-learn (ou sklearn)** : est une bibliothèque open-source pour le langage de programmation Python, qui fournit des outils pour le machine learning et le traitement automatique du langage naturel. Scikit-learn propose une grande variété d'algorithmes de machine learning pour la classification, la régression et le clustering, ainsi que des outils pour la préparation et la transformation des données. Cette bibliothèque est largement utilisée dans la recherche et l'industrie pour le développement d'applications basées sur la machine learning [27].

4.3 Informations générales sur le jeu de données

Les caractéristiques du jeu de données final que nous avons utilisé, nommé "Arabic_Rumor_Data", sont présentées dans le tableau ci-dessous :

Tableau 3.Caractéristiques de l'ensemble de données

Colonne	**Type**	**Description**
ID_Commentaire	String	Identifient du commentaire
ID_Video	String	Identifient de la vidéo
ID_Chaine	String	Identifient de la chaine YouTube
Name	String	Nom d'utilisateur
Comment	String	Texte du commentaire
Label	String	Attitude exprimée envers les rumeurs : ▪ Supporter ▪ Refuser ▪ Requête ▪ Commentaire ▪ Commentaire Positif ▪ Commentaire Négatif
Time	String	La date et l'heure de la publication
likes	String	Le nombre de mentions « J'aime »
Reply Count	String	Le nombre de réponses

4.4 Distribution générale de classes

L'ensemble de données "Arabic_Rumor_Data" comprend 3314 commentaires, répartis en différentes classes telles que "Supporter", "Refuser", "Requête", "Commentaire", "Commentaire Positif" et "Commentaire Négatif". Cette répartition est présentée dans le Tableau 4 :

Tableau 4.Distribution des classes de l'ensemble de données

Label	**Type**	**Nombre**
Commentaire	**String**	**1726**
Commentaire Négatif	**String**	**613**
Commentaire Positif	**String**	**269**
Supporter	**String**	**331**
Requête	**String**	**164**
Refuser	**String**	**211**
Total		**3314**

4.5 Les termes les plus fréquents (nuage de mots)

Le nuage de mots (ou Word Cloud) est une représentation visuelle des mots les plus fréquents dans un ensemble de données textuelles .Les mots les plus fréquents apparaissent en général en plus gros caractères et en avant-plan, tandis que les mots moins fréquents apparaissent en plus petits caractères et en arrière-plan.

La figure 15 est une représentation visuelle détaillée des mots les plus courants dans notre corpus. Elle nous offre un aperçu visuel précieux des principaux thèmes abordés dans le discours public en Algérie, tels que "امير" ,"العصابة" ,"مات" ,"تبون", etc. Cette représentation met en évidence les préoccupations nationales, les acteurs politiques et les enjeux géopolitiques pertinents.

Figure 15.les mot les plus fréquents dans l'ensemble de données

4.6 Préparation de données pour la classification

Le prétraitement des données pour nettoyer et transformer le texte arabe, la vectorisation des données, et la division des données en deux ensembles : ensemble d'apprentissage et ensemble de test : sont les premiers étapes de la préparation des données pour la tâche de classification du texte. Ces étapes jettent les bases des étapes suivantes du processus de classification de texte, permettant le développement de modèles de classification robustes et précis.

Dans notre application :

- Le prétraitement du texte arabe a été effectué à l'aide du package PyArabic.
- La représentation numérique des données textuelles a été effectuée à l'aide de la méthode TF-IDF.
- Une proportion de 80% de données a été allouée pour l'ensemble d'apprentissage tandis que la partie restante a été utilisée pour tester et évaluer les performances des modèles de classification appliqués.

Les fonctions utilisées pour préparer et nettoyer les données brutes ont été mentionnées précédemment dans le chapitre précédent. Ces fonctions comprennent l'espacement

supplémentaire, la suppression de caractères spéciaux, la suppression des "stopwords", la conversion des chiffres en lettres, la suppression des URLs, la suppression des hashtags, la suppression des Emojis, la suppression du "tashkeel", la suppression du "tatweel", la normalisation de la ligature et la normalisation de la "hamza". De plus, les tableaux ci-dessus nous fournissent un aperçu des résultats obtenus pour certains de ces processus :

Tableau 5 : Exemples des commentaires après l'application des fonctions de prétraitement

Text brute	**Text après l'application des fonctions**
هل صحيح ما تقول؟؟؟؟. كرهنا الخرطي	صحيح تقول كرهنا الخرطي
دعوي شعب لي راهي تخرج فيهم؟؟؟فمابالوك 40 مليون ضحية	دعوي شعب تخرج فمابالوك مليون ضحيه
يوم 12 12 الششششششششششششششعب قال كلمتو وقالهم مكانش إنتخابات مع العصابات	يوم الشعب قال كلمتو وقالهم انتخابات العصابات
@Omer Dida تكذب	Omer Dida تكذب
😎😎 ظهر الحق وزهق الباطل	ظهر الحق وزهق الباطل

4.7 Processus de sélection des attributs (Paramétrage des méthodes)

4.7.1 Les Méthodes de base

- **Naïve de Bayes**

Nous avons paramétré la méthode Naïve de Bayes, en particulier GaussianNB, par les valeurs par défaut suivants :

- **GaussianNB** (priors, var_smoothing)
- **GaussianNB**(priors=None, var_smoothing=1e-09)

- **Priors =None :** Probabilités a priori des classes. Si spécifié, les a-priori ne sont pas ajustés en fonction des données.

- **Var_smoothing : float, default=1e-9** : Valeur ajoutée à la variance des variables pour la stabilité numérique.
    - **Régression Logistique**

Nous avons paramétré la régression logistique par les valeurs par défaut suivants :

- **LogisticRegression**(penalty, tol, C, random_state)
- **penalty=l2 :** Précise la norme de la peine. Par défaut, la régularisation L2 est utilisée.
- **tol=1e-4 :** Tolérance pour les critères d'arrêt.
- **C=1.0 :** Inverse de la force de régularisation. Une valeur plus petite de C indique une régularisation plus forte.
- **random_state=None** : Graine utilisée par le générateur de nombres aléatoires pour l'initialisation du solveur.
    - **SVM**

Nous avons paramétré la méthode SVM par les valeurs par défaut suivants:

- **SVC**(C, kernel, gamma, coef, random_stat)
- **C=1.0**: Paramètre de régularisation. La force de la régularisation est inversement proportionnelle à C. Doit être strictement positive. La pénalité est une pénalité de l2 au carré.
- **Kernel = linear :** spécifie que le noyau utilisé pour le modèle SVM est linéaire.
- **Gamma = scale :** Coefficient de noyau qui contrôle l'influence des exemples d'entraînement sur la frontière de décision.
- **Coef0 =0.0 :** Terme indépendant dans la fonction noyau.
- **random_state = none :** Graine aléatoire pour la reproductibilité des résultats.
    - **La forêt aléatoire et l'arbre de décision**

Les modèles de forêt aléatoire (Random Forest) et d'arbre de décision (Decision Tree) partagent plusieurs paramètres similaires :

- **RandomForestClassifier**(n_estimators , max_depth, min_samples_split , **min_samples_leaf,** max_features, random state)
- **DecisionTreeRegressor**(criterion,max_depth,min_samples_leaf, max_features, min_samples_split, random state)

- **Max_depth = none :** Profondeur maximale des arbres. Si None, les arbres sont développés jusqu'à ce que toutes les feuilles soient pures ou que le nombre minimal d'échantillons requis pour une scission soit atteint.
- **min_samples_split = 2 :** nombre minimum d'échantillons requis pour diviser un nœud.
- **min_samples_leaf = 1 :** nombre minimum d'échantillons requis dans chaque feuille.
- **max_features = sqrt :** nombre maximum de variables à considérer pour chaque split.
- **Random state = none :** Contrôle le caractère aléatoire de l'estimateur.

La forêt aléatoire

- **n_estimators** = 100
- **Max_depth** = none
- **max_features** = sqrt

Arbre de décision

- **criterion = squared_error :** Le critère utilisé pour mesurer la qualité de la scission lors de la construction de l'arbre de décision.
- **Max_depth** = none
- **min_samples_leaf** =1
- **min_samples_split** = 2
- **max_features** = none

4.7.2 Apprentissage par ensemble (Ensemble Learning)

Les méthodes d'ensemble partagent des paramètres similaires comme :

- **n_estimators :** définit le nombre d'estimateurs à créer.
- **Estimateurs :** Estimateurs de base qui seront empilés ensemble. Chaque élément de la liste est défini comme un tuple de chaîne (c'est-à-dire un nom) et une instance d'estimateur.
- **Random_state** : Cela définit la graine aléatoire pour la reproductibilité des résultats

    - **Le Bagging**

BaggingClassifier(estimator, n_estimators, boostrap, random_state)

- **Base_estimator :** L'estimateur de base à ajuster sur des sous-ensembles aléatoires de l'ensemble de données. Dans ce cas, il s'agit d'un DecisionTreeClassifier.
- **n_estimators** = 10
- **Bootstrap :** détermine si un échantillon bootstrap est utilisé pour entraîner chaque estimateur.
- **Random_state** = 42

  - **Le Stacking**

StackingClassifier (estimators, final_estimator)

- **Estimateurs** : dans notre code nous utilisons deux estimateurs sont :
  - ✓ RandomForestClassifier(n_estimators=10, random_state=42)
  - ✓ LogisticRegression(random_state=42)

- **Final_estimator** : Un classificateur qui sera utilisé pour combiner les estimateurs de base. Le classificateur qui nous utilisons est **DecisionTree**.

  - **Ensemble de vote**

Voting Classifier (estimators, voting='soft')

- **Estimators :** nous utilisons trois estimators : régression logistique (**log_clf**), arbre de décision (**tree_clf**) et k-plus proches voisins (**knn_clf**).

- **Voting** : Si « difficile », utilise les étiquettes de classe prédites pour le vote selon la règle de la majorité. Sinon, si "soft", prédit l'étiquette de classe en fonction de l'argmax des sommes des probabilités prédites, ce qui est recommandé pour un ensemble de classificateurs bien calibrés et nous utilisons voting = soft.

  - **Le boosting avec AdaBoost**

AdaBoostClassifier(estimators=50, random_state)

- **n_estimators =**100

- **Random_state =**42

4.7.3 L’apprentissage sensible au coût (Cost sensitive learning)

Dans le paramétrage de l'approche de la sensibilité aux coûts, nous avons utilisé les mêmes paramètres que ceux de la première approche, qui comprennent les algorithmes SVM, régression logistique et arbre de décision. Cependant, nous avons ajouté un nouveau paramètre, **`class_weight="balanced"`** ce paramètre permet de gérer les déséquilibres de classe dans les données en ajustant automatiquement les poids des classes lors de l'apprentissage du modèle. En utilisant `class_weight="balanced"`, le modèle sera plus sensible aux classes minoritaires, ce qui peut aider à améliorer les performances dans les scénarios où les classes sont déséquilibrées.

4.7.4 L’apprentissage mono classe

Dans le cadre de l'apprentissage d'un classificateur SVM à classe unique, il est courant d'utiliser des paramètres similaires à ceux utilisés pour le modèle SVM standard.

4.8 Les Résultats obtenus

4.8.1 Méthodes de bases

Les performances des classifieurs de base sur notre corpus de données sont résumées dans le tableau ci-dessous, en utilisant les métriques de précision, rappel (recall) et score F1, ainsi que l'exactitude (accuracy)

Tableau 6: Comparaison des résultats de performance des méthodes de bases

Métriques d'évaluation	Accuracy	Precision	Recall	F1-score
Naïve de Bayes	0.46	0.49	0.46	0.47
SVM	**0.60**	**0.63**	**0.60**	**0.52**
régression logistique	0.58	0.68	0.58	0.48
Arbre de décision	0.53	0.49	0.53	0.50
Forêt Aléatoire	**0.59**	**0.58**	**0.59**	**0.52**

La figure 16 mettent en évidence la performance des méthodes de base mesurée par la Courbe ROC et la courbe PRC (courbe de la Précision –Rappel).

Il est observé visuellement que l'évaluation de la courbe ROC révèle une valeur élevée de AUC=0,89 (Aire sous la courbe) avec la méthode SVM linéaire, tandis que la mesure de la courbe PRC indique une valeur de précision moyenne qui est égale à 0,41 pour les méthodes SVM et régression logistique.

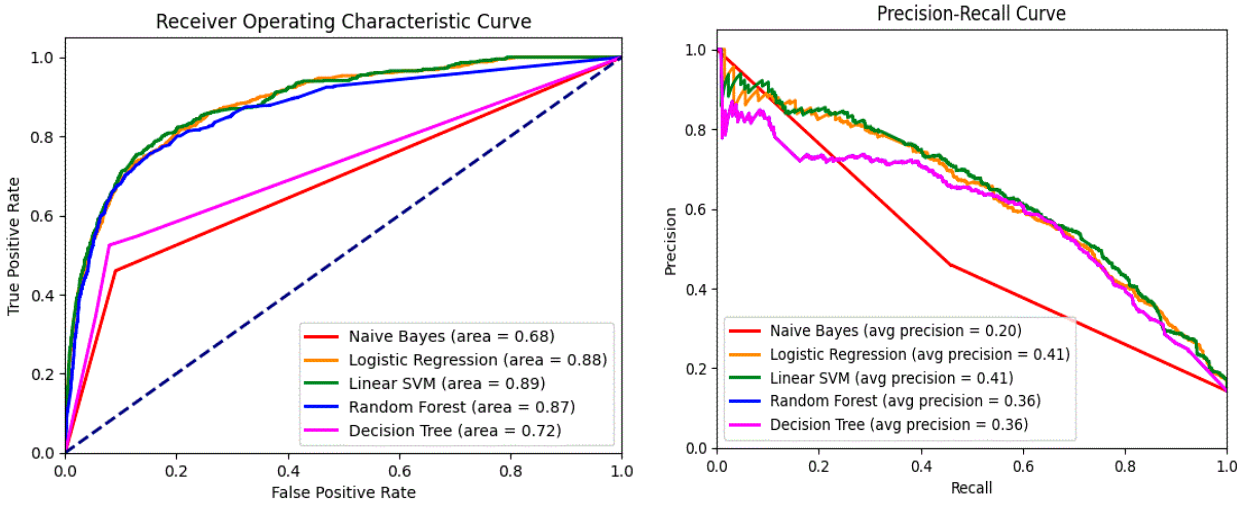


Figure 16. Courbe ROC (gauche) et courbe PRC (droite) des méthodes de base

La Figure 17 présente la matrice de confusion du modèle SVM, la méthode la plus performante dans l'approche de classification de base :

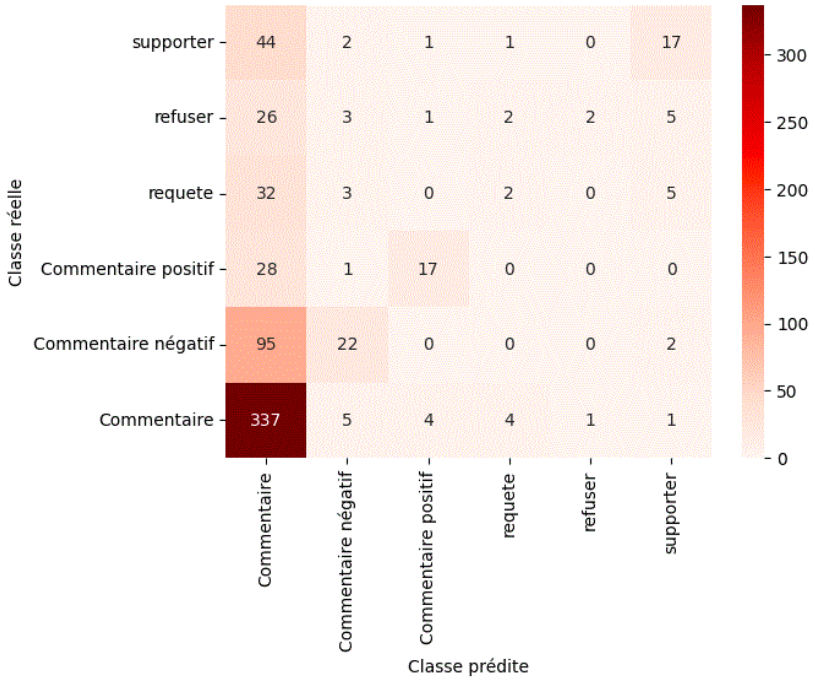


Figure 17. La matrice de confusion du SVM

4.8.2 Apprentissage par ensemble (Ensemble Learning)

Le Tableau 10 présente les performances des classificateurs de l'apprentissage par ensemble, évaluées à l'aide des métriques de précision, rappel (recall), score F1 et exactitude (accuracy) :

Tableau 7: Comparaison de résultats de performance des classificateurs ensemblistes

Métriques d'évaluation	**Accuracy**	**Precision**	**Recall**	**F1-score**
Bagging	**0.58**	**0.55**	**0.58**	**0.52**
Boosting	0.56	0.51	0.56	0.45
Stacking	0.45	0.47	0.45	0.45
Voting	0.58	0.68	0.58	0.47

La figure 18 mettent en évidence la performance des méthodes ensemblistes mesurée par la Courbe ROC et la courbe PRC (courbe de la Précision –Rappel).

Il est observé visuellement que l'évaluation de la courbe ROC révèle une valeur élevée de AUC=0,87 (Aire sous la courbe) avec la méthode Voting, tandis que la mesure de la courbe PRC indique une valeur de précision moyenne qui est égale à 0,36 pour la même méthode.

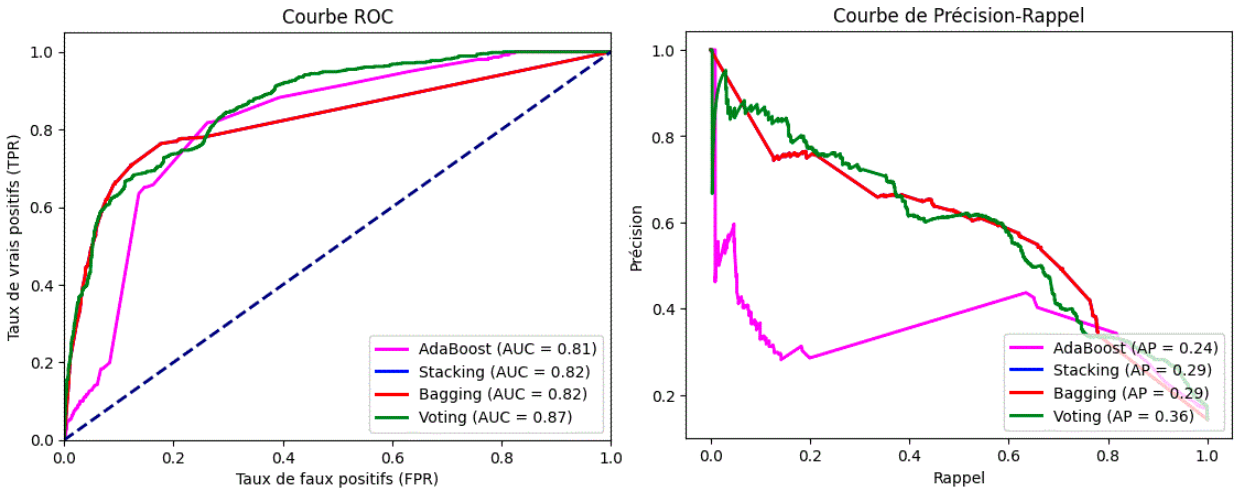


Figure 18.Courbe ROC (gauche) et courbe PRC des méthodes ensemblistes

La figure suivante représente la matrice de confusion de la méthode de Bagging, le meilleur algorithme parmi les méthodes ensemblistes.

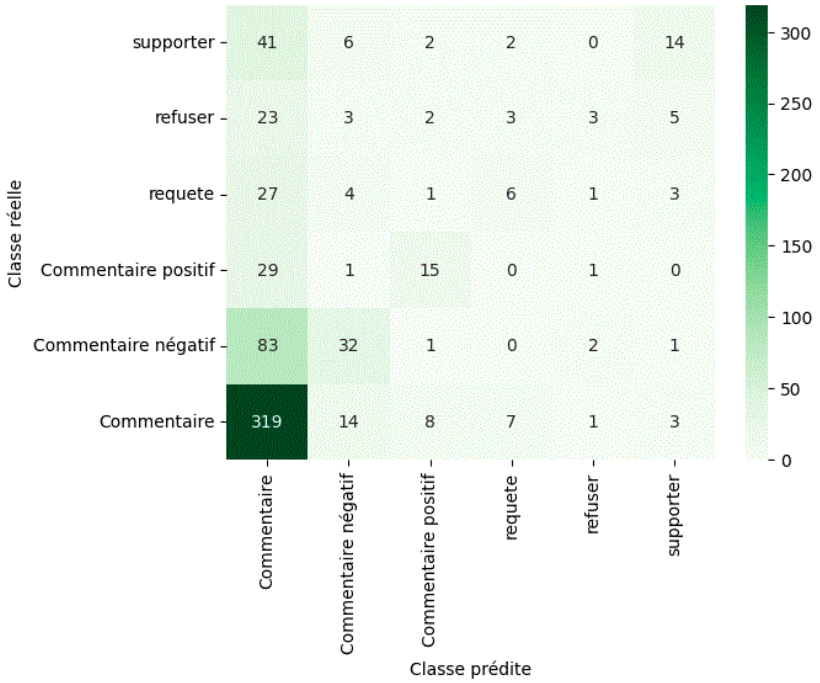


Figure 19. La matrice de confusion du classificateur ensembliste Bagging

4.8.3 L’apprentissage sensible au coût (Cost-Sensitive Learning)

Les performances de l'approche d'apprentissage avec prise en compte des coûts pour les classificateurs de base SVM, régression logistique et arbre de décision sur l'ensemble de données sont récapitulées dans le tableau ci-dessous, en utilisant les métriques de précision, rappel (recall), score F1 et exactitude (accuracy) :

Tableau 8 : Comparaisob de résultats de performance des classificateurs sensibles au coût

Métriques devaluation	**Accuracy**	**Precision**	**Recall**	**F1-score**
Cost-Sensitive logistic regression	**0.60**	**0.58**	**0.60**	**0.59**
Cost-sensitive decision Tree	0.52	0.50	0.52	0.51
Cost-sensitive SVM	0.48	0.56	0.48	0.46

La figure 20 présente les métriques d'évaluation (courbe ROC, courbe PRC) de l'approche d'apprentissage avec prise en compte des coûts. Nous avons remarqué que leurs performances, mesurées par l'AUC, sont très élevées, avec des valeurs allant de 0,90 (régression logistique), 0.87 (SVM) et à 0,71 (arbre de décision) pour la courbe ROC. En ce qui concerne la courbe PRC, nous observons une valeur de mAP=0,42 pour la méthode régression logistique, mAP 0.39 pour SVM, tandis que l'arbre de décision présente une valeur de mAP=0,21.

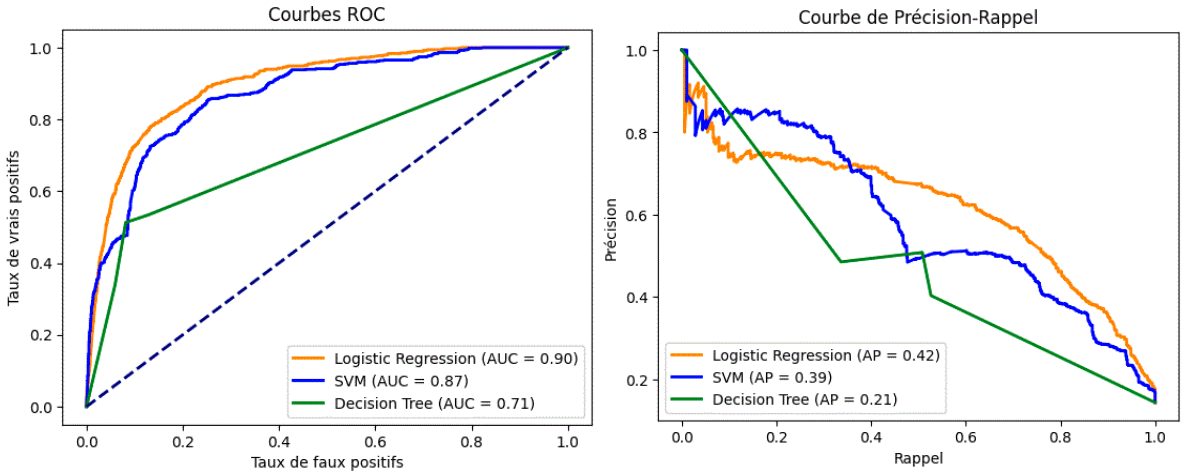


Figure 20. Courbes ROC (gauche) et les courbes PRC des classificateurs sensibles aux coûts

La figure suivante représente la matrice de confusion de la régression logistique sensible au coût, le meilleur algorithme parmi les méthodes de l'approche sensible au coût.

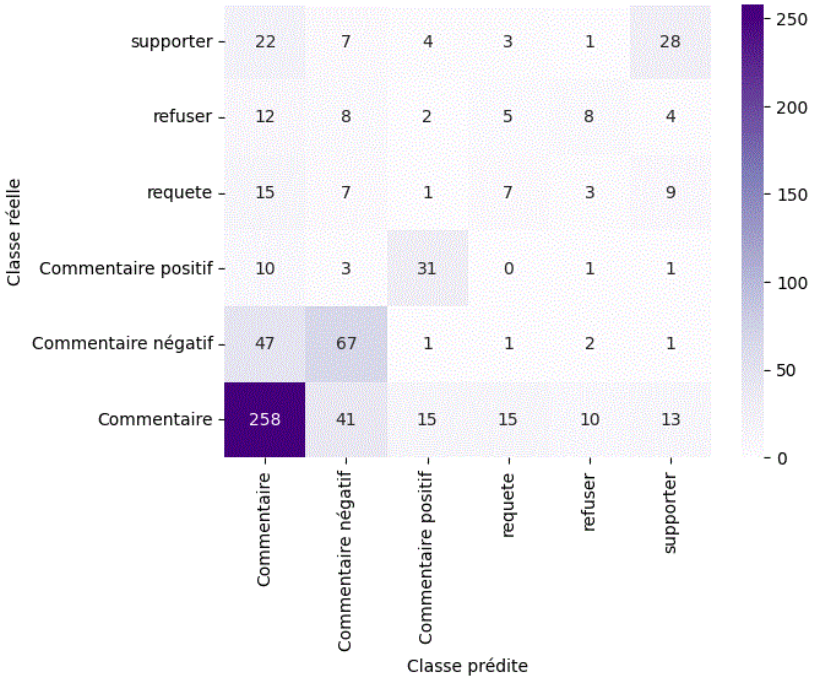


Figure 21. La matrice de confusion de la régression logistique sensible au coût

4.8.4 L'apprentissage mono classe

Le tableau 8 présente les résultats de performance du classificateur SVM de l'apprentissage mono classe, évaluée à l'aide des métriques de précision, rappel (recall), score F1 et accuracy :

Tableau 9: la performance du SVM mono-classe

Métriques d'évaluation	Accuracy	Precision	Recall	F1-score
SVM mono-classe	**0.01**	**0.01**	**0.01**	**0.01**

La figure suivante présente le rapport de classification du SVM mono-classe

	precision	recall	f1-score	support
Commentaire	0.00	0.00	0.00	119
Commentaire négatif	0.10	0.20	0.14	46
Commentaire positif	0.00	0.00	0.00	42
refuser	0.00	0.00	0.00	65
requete	0.00	0.00	0.00	352
supporter	0.05	0.69	0.09	39
accuracy			0.05	663
macro avg	0.03	0.15	0.04	663
weighted avg	0.01	0.05	0.01	663

Figure 22 le rapport de classification du SVM mono-classe

4.8.5 Le sur-échantillonage (Over-Sampling)

Le Tableau 9 présente les performances des classificateurs de base après l'application de techniques d'augmentation de données, consistant à ajouter de nouvelles données à la classe minoritaire ("refuser", "supporter" et "requête") jusqu'à ce que les classes soient équilibrées dans notre corpus. Les performances ont été évaluées à l'aide des métriques de précision, rappel (recall), score F1 et exactitude (accuracy) :

Tableau 10: Comparaison de résultats de performance des classificateurs après l'application du sur-échantillonage

Métriques d'évaluation	Accuracy	Precision	Recall	F1-score
Naïve bayes	0.69	0.76	0.69	0.72
SVM	**0.73**	**0.78**	**0.73**	**0.72**
Logistique régression	0.71	0.77	0.71	0.69
Décision tree	0.67	0.70	0.67	0.67
Random Forest	0.72	0.77	0.72	0.72

En analysant les courbes ROC (figure 23), nous avons identifié les meilleurs classificateurs : le SVM linéaire avec area=0,94. De plus, les résultats de la courbe PRC (figure 27) ont également indiqué de meilleures performances pour la même méthode, avec une valeur de précision moyenne AP= 0,64.

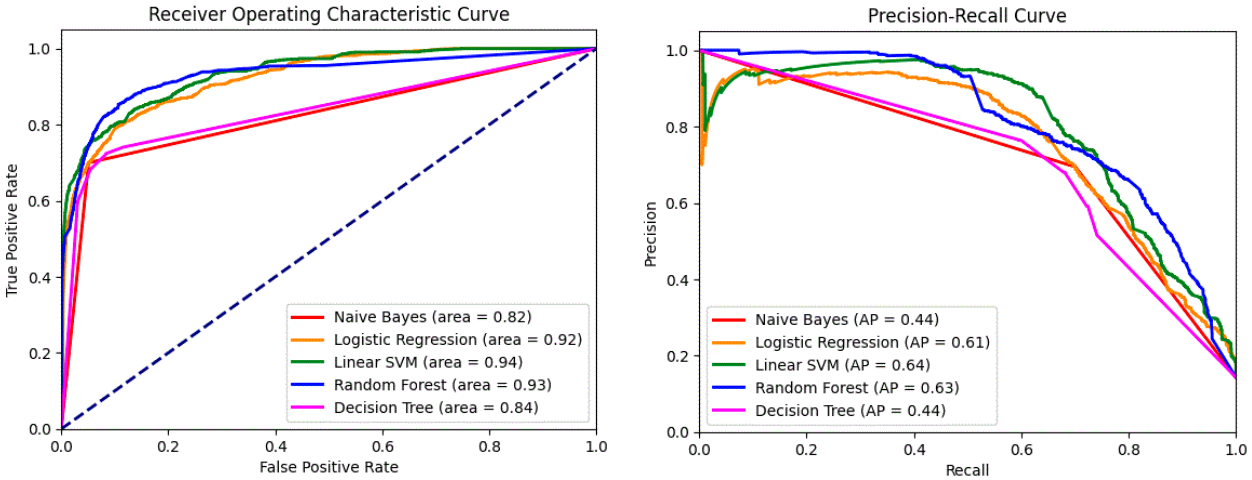


Figure 23. Courbes ROC (gauche) et courbes PRC (droite) des classificateurs de base après le sur-échantillonnage

La figure suivante représente la matrice de confusion du SVM, le meilleur algorithme parmi les méthodes de classification de base après l'augmentation de données.

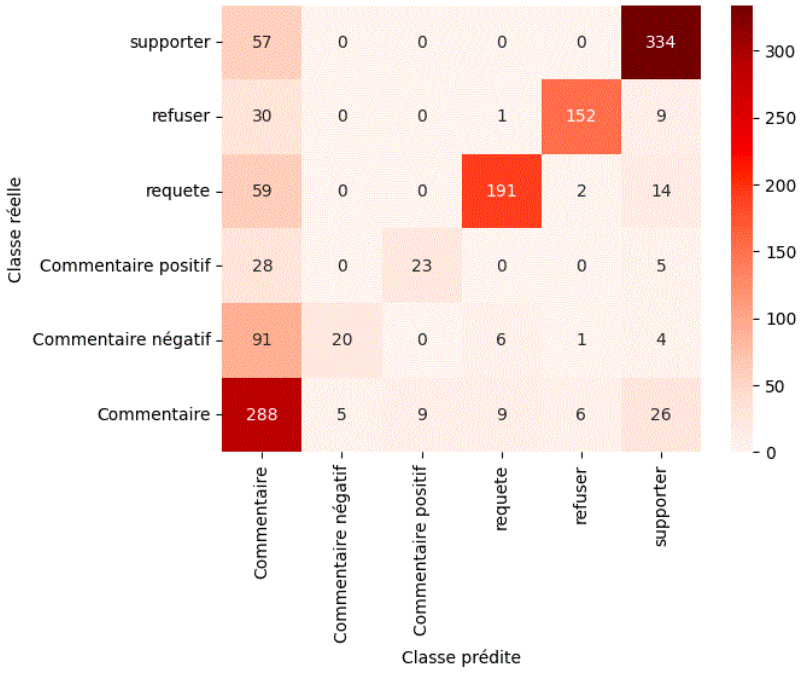


Figure 24. La matrice de confusion de l'algorithme SVM après l'application du sur-échantillonnage

4.9 Discussion des résultats

En examinant les résultats présentés dans les tableaux 9, 10, 11, 12 et 13, nous avons cherché à déterminer l'approche la plus efficace pour la classification des attitudes envers les rumeurs en prenant en considération la nature déséquilibrée des données de ce type de tâches. Il est important de noter que la technique de sur échantillonnage a montré des performances remarquables, en particulier lorsqu'elle est associée au classificateur SVM, par rapport à l'apprentissage d'ensemble, à l'apprentissage sensible aux coûts et à l'apprentissage à classe unique. Cela indique qu'en résolvant le problème de déséquilibre des données par le sur échantillonnage, la capacité des classifieurs de base à classer avec précision les instances a été considérablement amélioré, ce qui a entraîné une amélioration globale des performances.

En revanche, les autres approches, y compris l'apprentissage par ensemble, l'apprentissage sensible aux coûts et l'apprentissage à classe unique, ont démontré une efficacité relativement plus faible. Examinons individuellement chacune de ces approches pour mieux comprendre leurs limites et défis dans la détection des situations déséquilibrées dont le cas de la détection des attitudes envers les rumeurs.

- **Approche d'apprentissage d'ensemble** : Bien que les méthodes d'ensemble puissent améliorer les performances globales, elles peuvent ne pas être le choix optimal pour traiter le déséquilibre des classes dans le contexte de la détection des attitudes envers les rumeurs. Les méthodes ensemblistes reposent sur divers modèles de base pour améliorer les performances, mais dans un ensemble de données déséquilibré, la classe majoritaire a tendance à dominer, ce qui conduit à des modèles de base biaisés qui favorisent les prédictions de la classe majoritaire. Cette limitation diminue l'efficacité de l'apprentissage d'ensemble à capturer les nuances et les modèles de la classe minoritaire, ce qui entraîne une baisse des performances dans la détection des situations déséquilibrées [28].
- **Approche d'apprentissage sensible aux coûts** : Bien que l'apprentissage sensible aux coûts puisse être bénéfique dans certains scénarios, ce n'est peut-être pas l'option la plus efficace pour détecter les situations déséquilibrées. Attribuer des coûts plus élevés à une classification erronée de la classe minoritaire peut donner la priorité à sa détection, mais cela ne résout pas directement le problème de la représentation limitée et du manque de diversité au sein de la classe minoritaire. Cela peut conduire à un classificateur trop conservateur qui favorise la classe minoritaire, entraînant des taux de faux positifs plus élevés pour la classe majoritaire et des performances globales inférieures [29].
- **Approche d'apprentissage mono classe** : l'apprentissage à classe unique consiste à entraîner le modèle uniquement sur des instances d'une seule classe, sans tenir compte des autres classes. Apparemment, cette approche n'est pas adaptée pour détecter les situations déséquilibrées car elle ignore complètement les précieuses informations présentes dans la classe majoritaire. En se concentrant uniquement sur la classe minoritaire, l'apprentissage à classe unique ne parvient pas à saisir les relations et les modèles complexes entre les différentes classes, ce qui conduit à une généralisabilité limitée et à de mauvaises performances par rapport à des données invisibles [30].

En conclusion, nous pouvons affirmer que le sur-échantillonnage est le meilleur choix pour appliquer des tâches de classification de texte, tout en prenant en compte le problème courant de déséquilibre des données.

.

4.10 Conclusion

Ce chapitre présente et discute les résultats de l'étude expérimentale menée pour évaluer les performances du système de classification proposé pour classifier les attitudes envers les rumeurs en tenant compte le problème de déséquilibre de données . Les résultats obtenus ont montré l'efficacité de l'approche de sur-échantillonage par rapport aux autres approches concurrentes telles que l'apprentissage par ensemble, l'apprentissage sensible au coût et l'apprentissage mono-classe.

Conclusion générale

L'essor des réseaux sociaux et des plateformes en ligne pour évoquer les sujets politiques a généré une multiplication des fausses informations et des rumeurs en ligne. La détection automatique des attitudes envers les rumeurs politiques en ligne est un domaine de recherche important pour mieux appréhender l'influence de ces phénomènes sur l'opinion publique.

Dans ce contexte, nous avons élaboré un système de détection automatique des positions envers les rumeurs qui tient compte du problème courant de déséquilibre des données dans cette catégorie de tâches de classification du texte. Nous avons implémenté et évalué les approches de classification les plus appropriées pour résoudre cette problématique, telles que l'apprentissage en ensemble, le sur-échantillonnage du texte, l'apprentissage sensible au coût et l'apprentissage mono-classe. Les résultats obtenus ont montré l'efficacité du suréchantillonage à réduire l'influence négative de déséquilibre de données sur les performances des algorithmes de classification. En suréchantillonnant la classe minoritaire, on peut résoudre ce problème en augmentant sa représentation dans l'ensemble de données, fournissant ainsi au modèle un ensemble plus équilibré de données d'apprentissage.

Il existe diverses limitations au système de détection des attitudes implémenté. L'un des défis majeurs de cette application concerne le style linguistique utilisé dans les textes collectés. Les utilisateurs de YouTube ont tendance à écrire en utilisant des dialectes et des formes linguistiques spécifiques à leur région, ce qui rend difficile la compréhension de ces textes par les algorithmes de traitement de texte utilisés pour la classification automatique des positions des commentateurs envers les rumeurs. Ce style linguistique incompréhensible peut affecter la qualité de l'annotation manuelle des données. En conséquence, notre base de données contient un nombre très élevé de commentaires non pertinents de type "Commentaire" par rapport aux textes informatifs qui expriment clairement la position du commentateur, tels que les commentaires de type "Supporter", "Requête" ou "Refuser". Une autre limitation que nous avons rencontrée pour le calcul de la courbe ROC et de l'AUC n'est pas applicable au modèle One-Class SVM. La courbe ROC est basée sur la comparaison entre les scores de décision positifs et négatifs, ce qui n'est pas applicable dans le cas d'un modèle One-Class SVM et il n'est pas possible de générer une matrice de confusion car il n'y a qu'une seul classe (la classe positive) .Il est important de

noter que cela peut avoir un impact sur la précision globale du système d'annotation.

Ce projet a été l'occasion d'une expérience enrichissante, nous permettant d'approfondir nos connaissances et nos compétences dans la divulgation spontanée des attitudes envers les rumeurs politiques en ligne, en particulier en tenant compte le problème du déséquilibre des données. À l'avenir, nous prévoyons mettre en œuvre d'autres approches, les tester en utilisant d'autres types de classifications dans le cadre de nos recherches scientifiques en informatique.

Les références

[1] M. Hardalov, A. Arora, P. Nakov, and I. Augenstein, “A Survey on Stance Detection for Mis- and Disinformation Identification,” Feb. 2021, [Online]. Available: http://arxiv.org/abs/2103.00242

[2] D. Küçük and F. Can, “Stance Detection,” *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–37, Jan. 2021, doi: 10.1145/3369026.

[3] A. ALDayel and W. Magdy, “Stance detection on social media: State of the art and trends,” *Inf. Process. Manag.*, vol. 58, no. 4, p. 102597, Jul. 2021, doi: 10.1016/j.ipm.2021.102597.

[4] O. Habimana, Y. Li, R. Li, X. Gu, and G. Yu, “Sentiment analysis using deep learning approaches: an overview,” *Sci. China Inf. Sci.*, vol. 63, no. 1, p. 111102, Jan. 2020, doi: 10.1007/s11432-018-9941-6.

[5] C. Y. Huang and H. L. Dai, “Learning from class-imbalanced data: review of data driven methods and algorithm driven methods,” *Data Sci. Financ. Econ.*, vol. 1, no. 1, pp. 21–36, 2021, doi: 10.3934/DSFE.2021002.

[6] R. Mohammed, J. Rawashdeh, and M. Abdullah, “Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results,” in *2020 11th International Conference on Information and Communication Systems (ICICS)*, IEEE, Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.

[7] C. Shorten, T. M. Khoshgoftaar, and B. Furht, “Text Data Augmentation for Deep Learning,” *J. Big Data*, vol. 8, no. 1, p. 101, Dec. 2021, doi: 10.1186/s40537-021-00492-0.

[8] L. Wang, M. Han, X. Li, N. Zhang, and H. Cheng, “Review of Classification Methods on Unbalanced Data Sets,” *IEEE Access*, vol. 9, pp. 64606–64628, 2021, doi: 10.1109/ACCESS.2021.3074243.

[9] T. Hastie, J. Friedman, and R. Tibshirani, *The Elements of Statistical Learning*. New

York, NY: Springer New York, 2001. doi: 10.1007/978-0-387-21606-5.

[10] GeeksforGeeks. (n.d.), “Stacking in Machine Learning. Retrieved May 18, 2023, from https://www.geeksforgeeks.org/stacking-in-machine-learning/.”

[11] S. (n. d. . Raschka, “EnsembleVoteClassifier. Retrieved May 18, 2023, from https://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/”.

[12] JavaTpoint. (n.d.), “Random Forest Algorithm. Retrieved May 18, 2023, from https://www.javatpoint.com/machine-learning-random-forest-algorithm.”

[13] D. Luengo, F. Herrera, and C. Márquez, “Cost-sensitive learning strategies for high-dimensional and imbalanced data. PeerJ Computer Science, 5, e832. From https://peerj.com/articles/cs-832/,” 2019.

[14] P. Perera and V. M. Patel, “Learning Deep Features for One-Class Classification,” *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5450–5463, Nov. 2019, doi: 10.1109/TIP.2019.2917862.

[15] M. E. M. . & B. D. E. RIGHI, “Propagation de la rumeur sur les médias sociaux,” 2021.

[16] TF-IDF.(n.d.), “In Wikipedia. Retrieved May 18, 2023, from https://fr.wikipedia.org/wiki/TF-IDF.”

[17] S. Bird, E. Klein, and E. Loper, “Natural language processing with Python: analyzing text with the natural language toolkit. ‘ O’Reilly Media, Inc.,’” 2009.

[18] Python.(n.d.),“InWikipedia.RetrievedMay18,2023,from https://fr.wikipedia.org/wiki/Python_(langage).”

[19] Jupyter.(n.d.),“InWikipedia.RetrievedMay18,2023,from https://fr.wikipedia.org/wiki/Jupyter.”

[20] “GoogleColaboratory.(n.d.).RetrievedMay18,2023,from https://colab.research.google.com/.”

[21] C.(n.d.).nlpau.Mak,“RetrievedMay18,2023,from https://github.com/makcedward/nlpaug.”

[22] Pandas. (n.d.), “ Retrieved May 18, 2023, from https://pandas.pydata.org/.”

[23] “Natural Language Toolkit. (n.d.). Retrieved May 18, 2023, from https://www.nltk.org/.”

[24] “PyArabic. (n.d.). Retrieved May 18, 2023, from https://pypi.org/project/PyArabic/”

[25] “Arabic-stopwords. (n.d.). Retrieved May 18, 2023, from https://pypi.org/project/Arabic-Stopwords/.”

[26] “Gensim.(n.d.).RetrievedMay18, 2023, from https://radimrehurek.com/gensim/intro.html”

[27] “Scikitlearn.(n.d.).RetrievedMay18,2023,fromhttps://scikitlearn.org/stable/modules/ensemble.html.”

[28] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study. Intelligent data analysis,” pp. 429–449, 2002

[29] Y. Ma and H. He, “Imbalanced learning: foundations, algorithms, and applications,” 2013

[30] D. M. J. Tax, “One-class classification: Concept learning in the absence of counter-examples,” 2002