PROJET RITAL

RAPPORT

Reconnaissance de locuteurs et Analyse de sentiments

Étudiants :
Anyes TAFOUGHALT
Racha Nadine DJEGHALI

Encadré par :
Nicolas THOME

1er mars 2024

Table des matières

1 Introduction :

Le Traitement Automatique du Langage Naturel (TALN) offre des outils et des techniques essentiels pour analyser et comprendre le langage humain à travers des applications variées. Ce rapport se concentre sur deux problématiques fondamentales du TALN, chacune représentant un défi distinct et pertinent.

La première partie de ce projet se focalise sur le développement d'un détecteur de locuteur, visant à différencier les discours entre deux figures politiques : Jacques Chirac et François Mitterrand. Cette tâche implique la reconnaissance des caractéristiques distinctives du langage de chacun de ces locuteurs, offrant ainsi des perspectives précieuses sur leurs styles de discours respectifs et leur impact politique.

Dans la deuxième partie, notre projet explore le domaine de l'analyse de sentiment à travers l'examen des critiques de films. Notre objectif est de classer ces critiques en deux catégories principales : positives et négatives.

Pour aborder ces deux problématiques, nous avons adopté une approche méthodologique rigoureuse en utilisant plusieurs techniques de représentation de texte telles que le Bag-of-Words (BoW), le modèle TF-IDF et CountVectorizer. De plus, nous avons évalué la performance de différents algorithmes de classification, y compris le Naive Bayes, les Machines à Vecteurs de Support (SVM) et la Régression Logistique, afin de déterminer les modèles les plus adaptés à chaque tâche.

Ce rapport détaillera les méthodologies utilisées, les résultats obtenus ainsi que les analyses critiques de ces résultats.

2 Reconnaissance du locuteur (Chirac/Mitterrand) :

2.1 Analyse exploratoire des données

Dans cette section, nous procédons à une analyse exploratoire approfondie de nos données. Notre objectif est de comprendre la distribution des classes ainsi que les caractéristiques des discours prononcés par Jacques Chirac et François Mitterrand.

2.1.1 Distribution des classes

Nous avons examiné la répartition des classes dans notre ensemble de données. Les résultats montrent un déséquilibre significatif entre les deux classes, comme illustré dans le graphique ci-dessous :

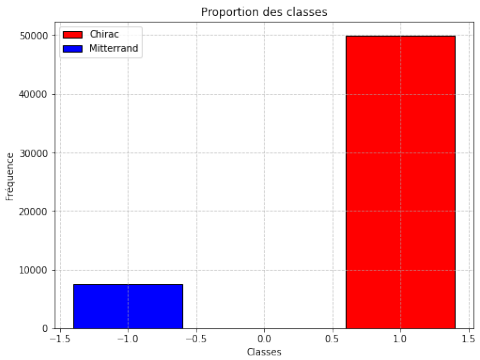

FIGURE 1 – Distribution des deux classes

On constate une disparité marquée entre les phrases attribuées à Chirac et celles attribuées à Mitterrand, avec un nombre beaucoup plus élevé de phrases attribuées à Chirac.

2.1.2 Distribution des longueurs des discours

Nous avons également analysé la distribution des longueurs des discours prononcés par les deux locuteurs. Les histogrammes suivants illustrent cette distribution :

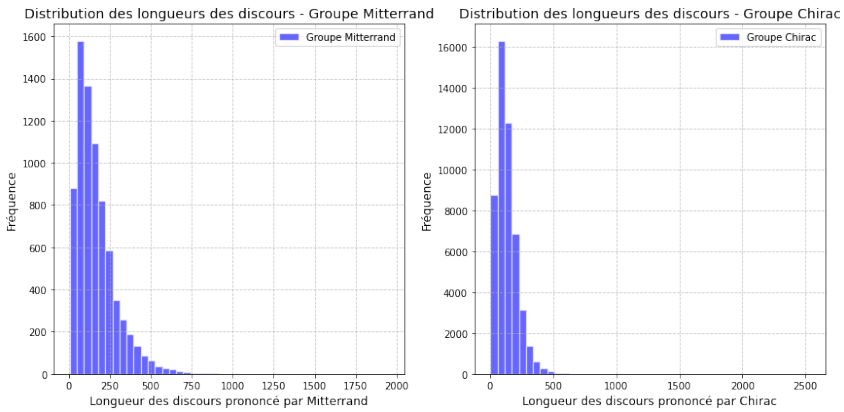

FIGURE 2 – Distribution des longueurs des discours de chacun des deux présidents

Les histogrammes montrent que les discours de Chirac ont tendance à être plus longs que ceux de Mitterrand, bien que les deux distributions présentent une certaine variation.

2.1.3 La plus longue phrase prononcée

Enfin, nous avons identifié la plus longue phrase prononcée dans notre ensemble de données, ainsi que son locuteur associé. Voici les détails :

— **Taille de la plus longue phrase prononcée :** 2530

— **Locuteur de la plus longue phrase :** Chirac

Cette analyse exploratoire nous offre un aperçu précieux de nos données et constitue une étape cruciale pour la suite de notre projet.

2.2 Transformation paramétrique du texte (pre-traitements)

Dans cette section, nous décrivons les différentes étapes de pré-traitement du texte que nous appliquerons avant d'effectuer toute analyse ultérieure. Ces étapes visent à nettoyer et préparer le texte pour une utilisation plus efficace dans le cadre de notre analyse.

2.2.1 Suppression de la ponctuation

La suppression de la ponctuation consiste à retirer tous les symboles de ponctuation du texte.

2.2.2 Remplacement des mots entièrement en majuscules

Pour conserver l'information sur les mots entièrement en majuscules, nous pouvons les remplacer par des marqueurs spécifiques pour les identifier plus tard dans l'analyse.

2.2.3 Suppression des chiffres

Pour supprimer les chiffres du texte, nous avons utilisé des expressions régulières et des fonctions de manipulation de chaînes. Cela permet de simplifier le texte en éliminant les nombres.

2.2.4 Suppression des balises HTML

Si le texte contient des balises HTML, il est utile de les supprimer pour ne garder que le contenu textuel (qui nous interésse réellement).

2.2.5 Stemming

Le stemming consiste à réduire les mots à leur racine (par exemple, "manger" devient "mang"). Cela permet de regrouper les variantes de mots qui ont la même racine, ce qui peut faciliter l'analyse.

2.2.6 Lemmatisation

La lemmatisation est similaire au stemming, mais elle tient compte du contexte et de la morphologie des mots pour les ramener à leur forme canonique ou "lemme" (par exemple, "manger" devient "mange"). Cela permet une normalisation plus précise des mots.

2.2.7 Conservation d'une partie du texte seulement

Il est parfois utile de ne conserver qu'une partie spécifique du texte, telle que la première ligne (titre) ou la dernière ligne (résumé). Cela peut être fait en extrayant la partie désirée du texte. On a donc implementé une fonction qui tient compte que du début ou de la fin d'un discours pour les tester plus tard.

2.2.8 Suppression des noms propres

Dans certaines analyses, il peut être pertinent de supprimer les noms propres du texte pour se concentrer sur le contenu général plutôt que sur les entités spécifiques.On a donc également implementé cette dernière afin de la tester sur ces problèmatiques.

2.3 Extraction du vocabulaire (BoW)

Dans cette section, nous allons explorer différentes caractéristiques du jeu de données, notamment la taille du vocabulaire, les mots les plus fréquents, les mots les plus discriminants, la distribution des mots, et les bigrammes/trigrammes les plus fréquents.

2.3.1 Exploration préliminaire des jeux de données

- **Taille d'origine du vocabulaire :** Nous commençons par déterminer la taille initiale du vocabulaire.
- **100 mots les plus fréquents :** Nous examinons quels sont les 100 mots les plus fréquents dans le corpus.
- **100 mots dont la fréquence documentaire est la plus grande :** Nous identifions les 100 mots ayant la fréquence documentaire la plus élevée.
- **100 mots les plus discriminants :** Nous calculons les odds ratios pour chaque mot afin de déterminer les 100 mots les plus discriminants entre différentes classes.
- **Distribution d'apparition des mots (Loi de Zipf) :** Nous étudions la distribution d'apparition des mots dans le corpus en utilisant la loi de Zipf.
- **100 bigrammes/trigrammes les plus fréquents :** Nous identifions les 100 bigrammes/trigrammes les plus fréquents dans le corpus.

2.3.2 Taille initiale du vocabulaire

Nous utilisons un *CountVectorizer* pour déterminer la taille initiale du vocabulaire qui est de 28524 mots.

2.3.3 100 mots les plus fréquents

Nous utilisons un histogramme pour visualiser les fréquences des 100 mots les plus fréquents.

FIGURE 3 – Tout le vocabulaire initial

FIGURE 4 – Word cloud des 100 mots les plus fréquents

2.3.4 100 mots les plus discriminants

Nous calculons les odds ratios pour chaque mot et identifions les 100 mots les plus discriminants. Nous avons utilisé la formule suivante pour nos calculs :
Odds ratio$(w) = \frac{\text{Fréquence de } w \text{ dans la classe M} * (1 - \text{Fréquence de } w \text{ dans la classe C})}{\text{Fréquence de } w \text{ dans la classe C} * (1 - \text{Fréquence de } w \text{ dans la classe M})}$

FIGURE 5 – 100 mots les plus discriminants

2.3.5 Distribution d'apparition des mots (Loi de Zipf)

Nous représentons graphiquement la distribution des fréquences des mots en utilisant la loi de Zipf.

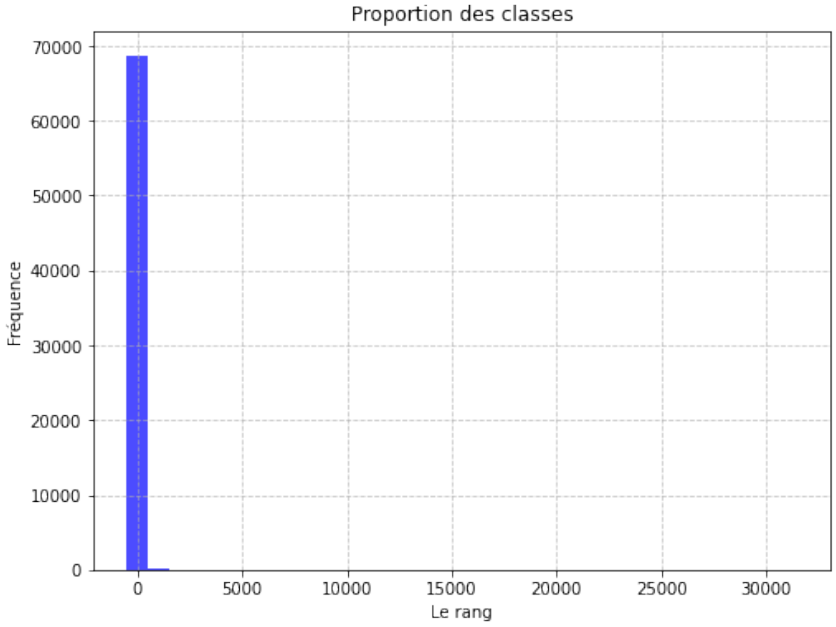

FIGURE 6 – Loi de Zipf

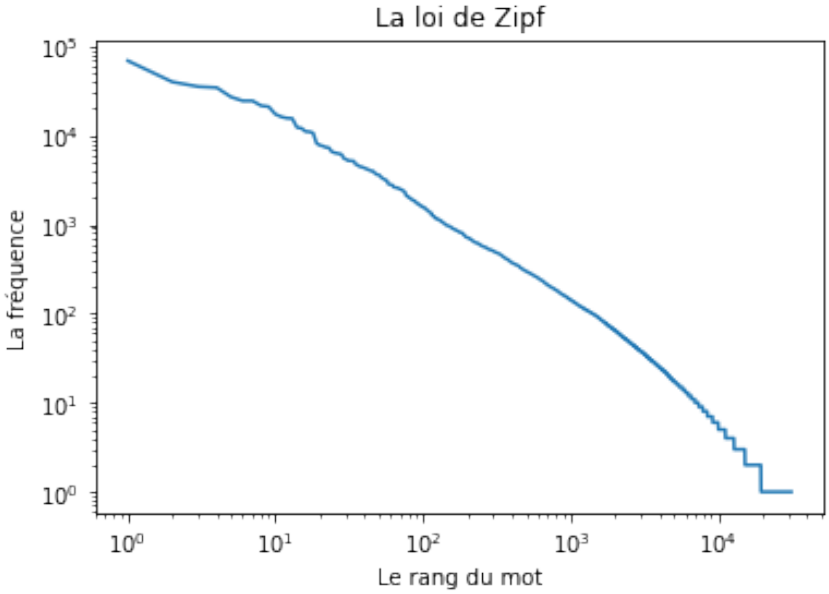

FIGURE 7 – Loi de Zipf log

Nous remarquons bien que la fréquence des mots décroit exponentiellement et on deduit que les mots sont les fréquences sont > 10000 ne sont pas discriminants et répresentent des stop words.

2.3.6 100 bigrammes/trigrammes les plus fréquents

Nous utilisons des bigrammes et trigrammes pour identifier les séquences de mots les plus fréquentes dans le corpus.

Bigrammes : Apès construction des bigrammes sans aucun pré-traitement la taille était de : 311838

FIGURE 8 – Les 100 bigrammes les plus fréquents

Trigrammes :Apès construction des bigrammes sans aucun pré-traitement la taille était de : 708080

FIGURE 9 – Les 100 trigrammes les plus fréquents

2.3.7 Stopwords

Comme on a remarqué que la plus part des mots qui sont les plus redondants que ça soit en unigrammes ou même en bi/trigrammes sont les mêmes ('de','la','une',....).

Nous utilisons une liste de stopwords pour filtrer les mots courants qui ne portent pas beaucoup d'information.

FIGURE 10 – Les 100 trigrammes les plus fréquents

2.4 Variantes de BoW

Nous avons exploré les variantes suivantes de BoW :

2.5 TF-IDF

TF-IDF est une méthode de vectorisation largement utilisée dans le traitement automatique du langage naturel . Nous avons évalué TF-IDF avec différentes configurations, notamment :

— Avec et sans pré-traitements tels que la suppression des balises HTML, la suppression de la ponctuation et la lemmatisation.

— Impact de la réduction de la taille du vocabulaire en utilisant des paramètres comme `min_df`, `max_df` et `max_features`.

— Utilisation de bi-grammes et tri-grammes pour capturer les relations entre les mots.

2.6 CountVectorizer

CountVectorizer est une autre méthode populaire de vectorisation de texte. Nous avons effectué une analyse similaire avec CountVectorizer, en évaluant différentes configurations et pré-traitements. Nous avons également comparé les performances de CountVectorizer avec celles de TF-IDF.

3 Fonction d'évaluation

Nous avons défini une fonction d'évaluation pour comparer les performances des modèles.

Nous avons divisé notre ensemble de données en un ensemble d'entraînement et un ensemble de test à l'aide d'une méthode de split. Cela nous permet d'évaluer les performances de nos modèles sur des données indépendantes de celles sur lesquelles ils ont été entraînés.

Étant donné que nos classes sont **déséquilibrées**, c'est-à-dire qu'il y a un nombre significativement plus élevé d'exemples dans une classe par rapport à une autre, nous avons pris des mesures pour corriger ce déséquilibre. Nous avons utilisé une technique appelée "undersampling", où nous supprimons aléatoirement des exemples de la classe majoritaire jusqu'à ce que le nombre d'exemples dans chaque classe soit plus équilibré. Cela permet d'éviter que le modèle ne soit biaisé vers la classe majoritaire lors de l'apprentissage.

Nous avons défini une fonction d'évaluation qui compare les performances des modèles. Cette fonction utilise plusieurs métriques telles que le F1-score, la précision, le recall et l'aire sous la courbe ROC. Le F1-score est particulièrement important dans notre cas car il tient compte à la fois de la précision et du recall, ce qui est crucial pour les classes minoritaires dans un problème de déséquilibre de classe.

Nous avons entraîné trois modèles de classification : Naïve Bayes, Régression Logistique et SVM linéaire. Ces modèles sont couramment utilisés pour la classification de textes et ont des caractéristiques différentes qui peuvent être explorées en fonction des données et du problème spécifique.

4 Résultats

Nos expériences ont montré que TF-IDF donne généralement de meilleures performances que CountVectorizer pour la classification de discours politiques. Nous avons également constaté que l'utilisation de bi-grammes peut améliorer les résultats, tandis que la suppression de la ponctuation n'a pas eu beaucoup d'impact. La réduction de la taille du vocabulaire peut être bénéfique pour accélérer le processus de vectorisation sans compromettre les performances du modèle.

TABLE 1 – Performance de TF-IDF avec différents pré-traitements en se focalisant ici sur le modèle de régression logistique

Pré-traitement	**F1-score**	**Precision**	**Recall**	**Roc auc**
Sans aucun pré-traitement	0.84	0.95	0.75	0.76
Avec suppression de balises	0.86	0.95	0.77	0.76
Sans ponctuation	0.85	0.95	0.77	0.77
Sans chiffres	0.86	0.84	0.87	0.81
Lemmatisation	0.85	0.95	0.77	0.77
Stemming	0.84	0.95	0.75	0.75
Réduction du vocabulaire	0.85	0.95	0.76	0.77
Binary BoW	0.85	0.95	0.76	0.76
Bi-grams	**0.86**	**0.96**	**0.78**	**0.77**
Tri-grams	0.85	0.94	0.77	0.72

TABLE 2 – Performance de CountVectorizer avec différents pré-traitements

Pré-traitement	**Accuracy**	**F1-score**	**Precision**	**Recall**
Sans aucun pré-traitement	0.83	0.94	0.73	0.73
Avec suppression de balises	0.84	0.95	0.75	0.75
Sans ponctuation	0.84	0.95	0.75	0.75
Lemmatisation	0.83	0.95	0.75	0.74
Stemming	0.82	0.94	0.73	0.72
Réduction du vocabulaire	0.83	0.95	0.74	0.74
Binary BoW	0.83	0.94	0.74	0.73
Bi-grams	0.86	0.94	0.79	0.75
Tri-grams	0.89	0.92	0.85	0.71

4.1 Évaluation des performances des classificateurs :

Dans cette étude, nous évaluons les performances de différents classificateurs en utilisant la technique de vectorisation de texte : TF-IDF et en utilisant comme pré-traitement la suppression des balises inutiles ainsi que la transformation de tous les mots majuscules en marquers spécifiques.

4.1.1 Division des données

Les données sont divisées en ensembles d'entraînement et de test pour évaluer les performances des modèles sur des données non vues.

4.1.2 Équilibrage des classes

Étant donné que les classes sont déséquilibrées, l'undersampling est utilisée pour équilibrer les classes dans l'ensemble d'entraînement.

4.1.3 Entraînement des classificateurs

Trois types de classificateurs sont utilisés : Naïve Bayes, Régression Logistique et SVM linéaire. Chacun de ces modèles est entraîné sur l'ensemble d'entraînement.

4.1.4 Évaluation des performances

Les performances des modèles sont évaluées à l'aide de diverses métriques telles que le F1-score, la précision, le recall, etc. Les performances sont comparées entre les différents classificateurs et techniques de prétraitement.

4.1.5 Résultats des comparaisons :

— **Régression logistique :** on constate que le F1 score de la classe Mitterand reste assez faible .

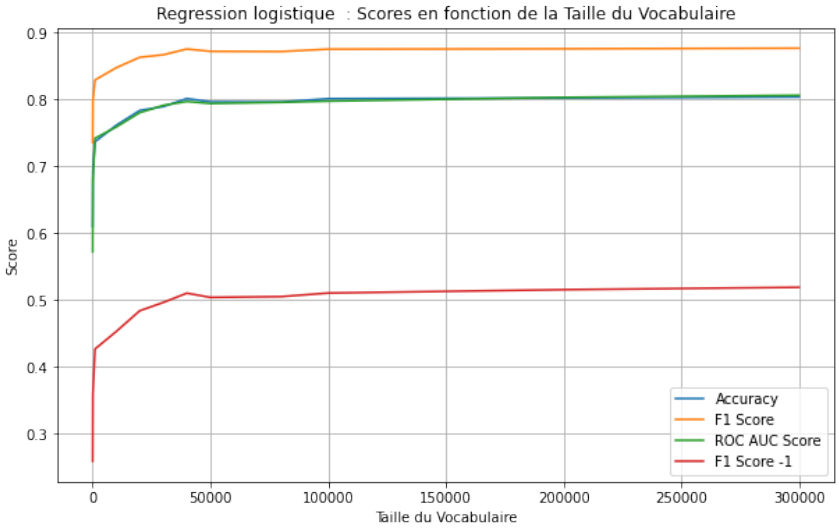

FIGURE 11 – Scores avec Régression logistique

— **Naîve Bayes :** on constate que les performances ont baissé par rapport a ce qui précède .

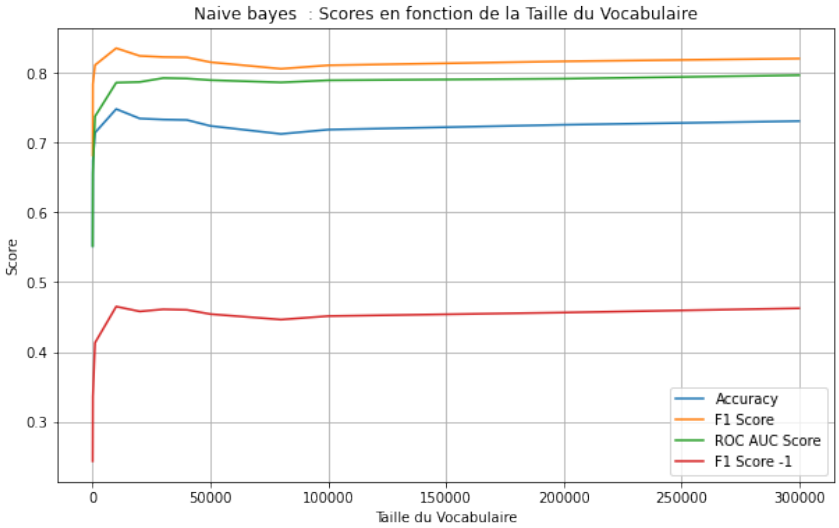

FIGURE 12 – Scores avec Naive Bayes

— **Naîve Bayes :** on constate que les résulats sont proches de ceux de la RL.

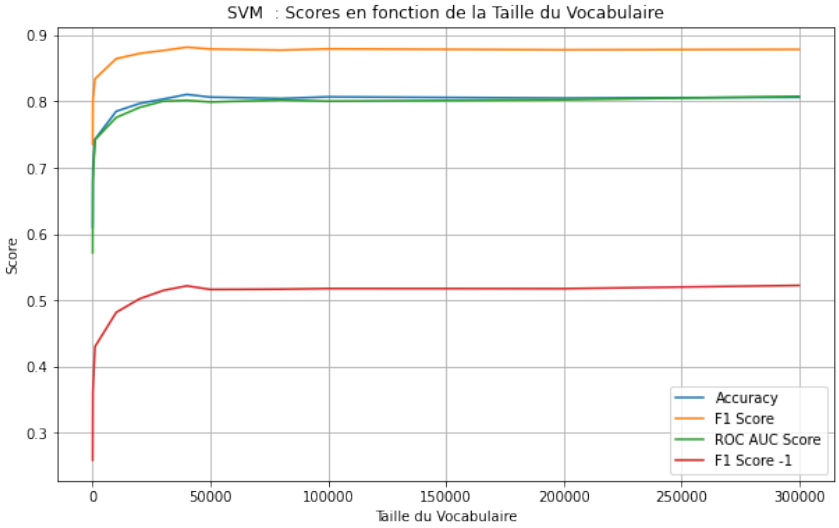

FIGURE 13 – Scores avec SVM lineaires

— **Comparaisons des performances 3 classifieurs :** on remarque que les svm et la RL donnent des résultats bien meilleurs que ceux avec Naive Bayes.

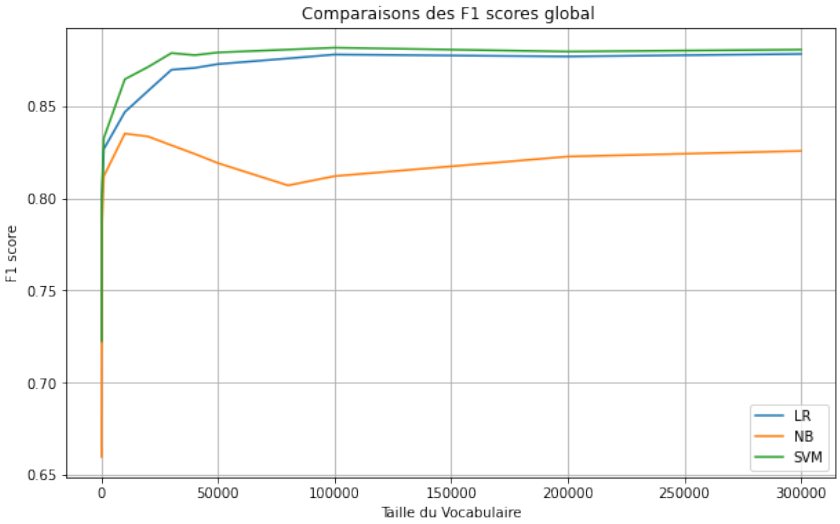

FIGURE 14 – Comparaisons des performances 3 classifieurs

— **Comparaisons des temps d'éxécution avec les 3 classifieurs :** on remarque que la regression lineaire prend plus de temps et que avec Naive Bayes est le plus rapide .

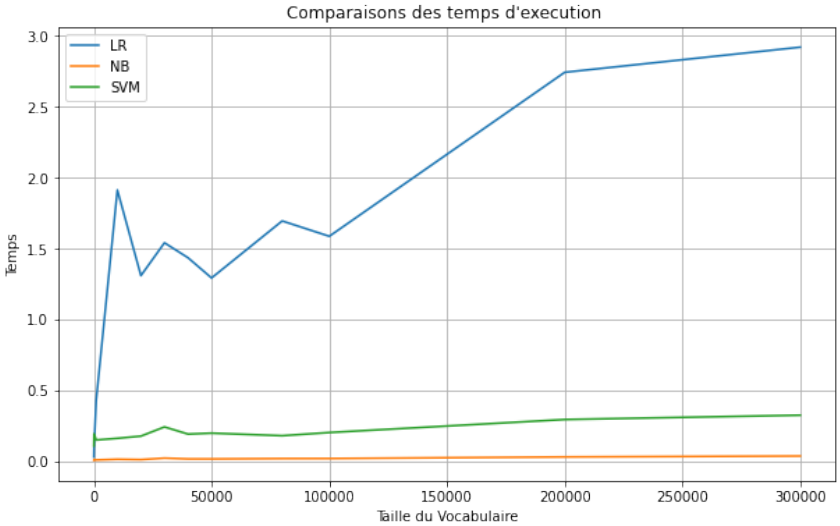

FIGURE 15 – Comparaisons des temps d'execution avec les 3 classifieurs

4.2 Importance de la validation croisée

4.2.1 Validation croisée

La validation croisée est une technique utilisée pour évaluer les performances d'un modèle sur des données non vues en partitionnant les données en ensembles d'entraînement et de validation à plusieurs reprises.Nous faisons ici des comparaisons afin de savoir s'il reste suffisant d'utiliser qu'un simple split.

4.2.2 Stabilité des performances

Pour évaluer la stabilité des performances, la validation croisée est effectuée en variant le nombre de plis (folds) et en utilisant différentes valeurs de seed. Les résultats montrent que les performances commencent à se stabiliser à partir d'un nombre de plis d'environ 8. Ci-dessous les différents graphes qui nous ont permis de tirer cette conclusion

— **Stabilité de la F1 score :**

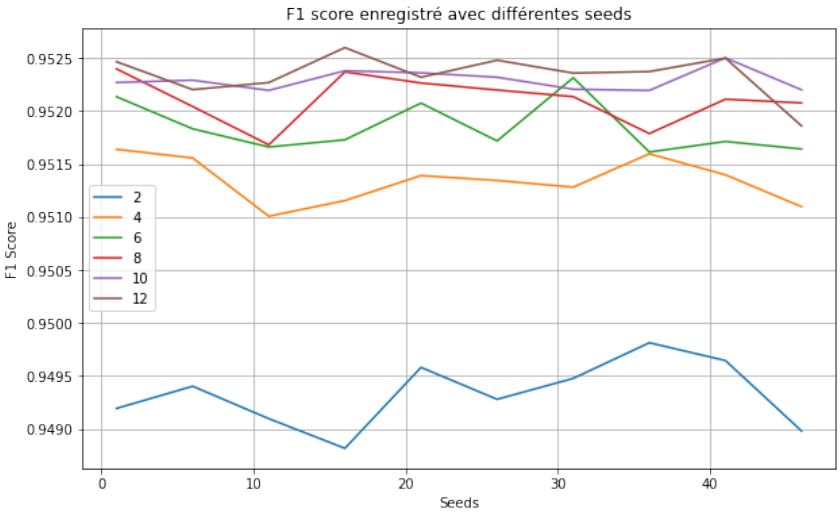

FIGURE 16 – Comparaisons stabilité du F1 score

— **Stabilité de la roc auc :**

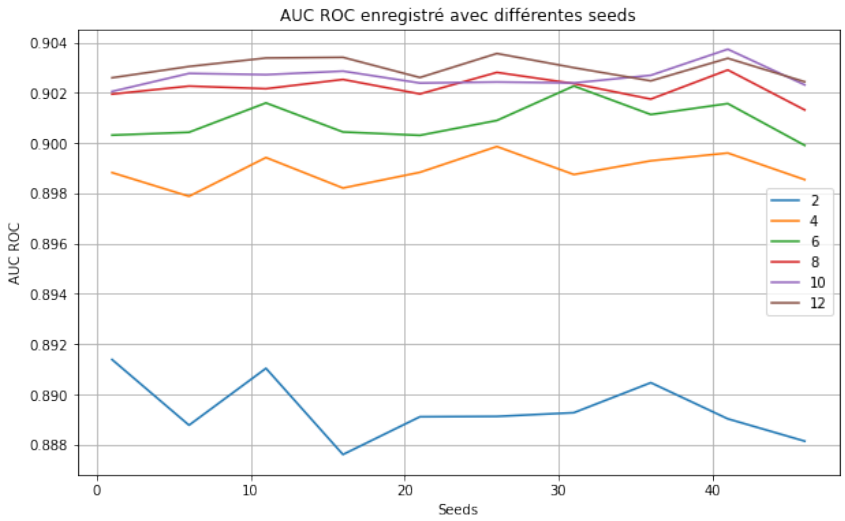

FIGURE 17 – Comparaisons stabilité de la roc auc

4.2.3 Visualisation des performances

Les performances sont visualisées à l'aide de graphiques tels que les courbes de scores en fonction du nombre de plis (ci dessus)et les violonplots pour montrer la distribution des scores.

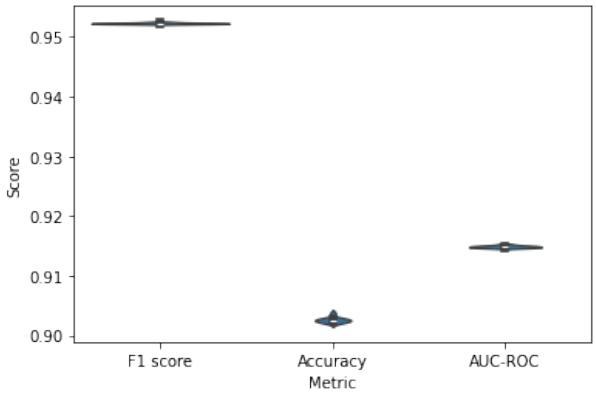

FIGURE 18 – Visualisations de la distribution suivies par les scores renvoyés

4.2.4 Conclusion

En conclusion, même si nous avons remarqué que dans des cas simple la validation croisée renvoie les même performance qu'avec un simple split cependant celle-ci devrait être préférée lors de l'évaluation des performances des modèles d'apprentissage automatique en raison de sa robustesse, de sa fiabilité et de sa précision accrue par rapport à un simple split des données(C'est pour cela que nous lors de nos test les plus exaustif nous utiliserons une grid search qui se base justement sur la validation croisée).

Pour une recherche plus exhaustive des paramètres optimaux nous avons utilisé :

4.3 Méthode de Grid Search

La méthode de grid search a été utilisée pour rechercher les meilleurs paramètres du modèle. Cette méthode consiste à définir une grille de paramètres à tester, puis à évaluer les performances du modèle pour chaque combinaison de paramètres dans cette grille. Le modèle est ensuite entraîné avec la meilleure combinaison de paramètres trouvée.

4.3.1 Paramètres à Optimiser

Les paramètres suivants ont été considérés pour l'optimisation du modèle :

- **max_df** : Valeur maximale pour filtrer les mots avec une fréquence de document supérieure à cette valeur.
- **min_df** : Valeur minimale pour filtrer les mots avec une fréquence de document inférieure à cette valeur.
- **ngram_range** : Plage de valeurs pour les n-grammes à extraire des textes.
- **binary** : Si les occurrences des mots dans le texte doivent être binaire (1 ou 0).
- **max_features** : Nombre maximal de caractéristiques à conserver.
- **class_weight** : Poids attribués aux classes pour compenser le déséquilibre de classe.
- **max_iter** : Nombre maximal d'itérations pour la convergence du modèle.
- **tol** : Tolérance pour la convergence du modèle.
- **penalty** : Type de régularisation à appliquer ('l1' ou 'l2').
- **C** : Paramètre de régularisation.

4.3.2 Tests et Résultats

Les résultats des différents tests sont présentés dans les sections suivantes. Dans cette phase de notre projet, nous avons adopté la méthode de Grid Search dans le but de déterminer la configuration optimale des paramètres afin d'optimiser les performances de notre modèle.

Pour ce faire, nous avons mis en place une pipeline intégrant le TF-IDF Vectorizer, accompagné de différentes fonctions de prétraitement spécifique visant à éliminer les balises HTML, la ponctuation, et à effectuer la lemmatisation. Les paramètres à optimiser englobaient divers aspects de la vectorisation tels que `max_df`, `min_df`, `ngram_range`, ainsi que les paramètres de régularisation de la régression logistique comme `C` et `penalty` ains que `class_weight` car comme cité précédemment le datasets est déséquilibré.

La procédure de Grid Search a été exécutée en utilisant une validation croisée. Nous avons évalué les performances du modèle en utilisant des métriques telles que le `score F1`, et l'`AUC ROC` . Les données ont été divisées en ensembles d'entraînement et de test dans un rapport de 70 :30.

Pour commencer, nous avons cherché à maximiser les performances en testant différents paramètres. Nous avons trouvé que `C=1` semblait être un choix adéquat lors de la maximisation de la **auc**, mais le score F1 n'était pas satisfaisant à ce stade. Par la suite, en visant à maximiser le score F1 de la classe minoritaire, nous avons identifié que `C=10` était plus approprié. Cette optimisation a également entraîné une augmentation de l'AUC.

4.3.3 Prétraitement des données

Dans le processus de prétraitement des données, nous avons supposé que l'utilisation de trigrammes pourrait aider à distinguer les différents styles de discours entre les deux présidents. Pour ce faire, nous avons entrepris plusieurs étapes cruciales. Tout d'abord, nous avons supprimé les balises HTML, les chiffres et la ponctuation, car ils semblaient ne pas être pertinents pour notre analyse. Ensuite, nous avons appliqué la lemmatisation, ce qui a permis de réduire la taille du vocabulaire en regroupant plusieurs mots similaires. De plus, la suppression des stopwords a considérablement aidé à éliminer le bruit dans nos données. Enfin, nous avons mis tous les mots en minuscules et ajouté des marqueurs pour les mots entiers, supposant que l'un des deux présidents utilise certains mots plus fréquemment que l'autre.

4.3.4 Optimisation des paramètres

L'optimisation des paramètres du modèle de régression a été réalisée en plusieurs étapes. Initialement, des tests simples avec un simple split ont été effectués. Par la suite, une approche plus robuste utilisant la recherche par grille (*grid search*) basée sur la validation croisée a été adoptée. Cela nous a permis d'explorer une gamme de prétraitements, y compris l'utilisation d'unigrammes et de valeurs de TF-IDF, ainsi que les paramètres spécifiques de la régression linéaire. Ces expérimentations nous ont aidés à déterminer les paramètres optimaux pour notre modèle.

4.3.5 Le paramètre de régularisation

Pour l'optimisation des paramètres, nous nous sommes principalement concentrés sur le paramètre de régularisation C. Après plusieurs expérimentations[1,10,100], nous avons constaté que $C = 10$ était le meilleur paramètre pour éviter le surapprentissage tout en maintenant une bonne performance sur les données d'apprentissage.

4.3.6 Gestion du déséquilibre des classes

Pour aborder le problème du déséquilibre des classes dans nos données, plusieurs approches ont été explorées. Outre la définition de poids de pénalisation pour les classes lors de l'entraînement du modèle de régression linéaire, nous avons également testé des méthodes de rééquilibrage telles que l'undersampling et l'oversampling pour génerer de nouvelles données ou en supprimer . Initialement, nous avons attribué un poids de pénalisation de 5 pour la classe des "Mitterand" et de 1 pour celle de "Chirac". Après plusieurs essais, il a été observé que l'option de pondération *balanced* était la plus appropriée, ce qui a permis de mieux traiter le déséquilibre des classes.

4.3.7 Post-traitement des données

Enfin, après l'entraînement du modèle de régression, des mesures de post-traitement ont été entreprises pour améliorer davantage les performances. Il a été observé que des phrases successives étaient souvent associées au même locuteur, ce qui pouvait influencer les résultats. Pour atténuer cet effet, un lissage gaussien a été appliqué avec différentes valeurs de sigma. Cette technique a considérablement amélioré les performances du modèle en tenant compte de la structure des données.

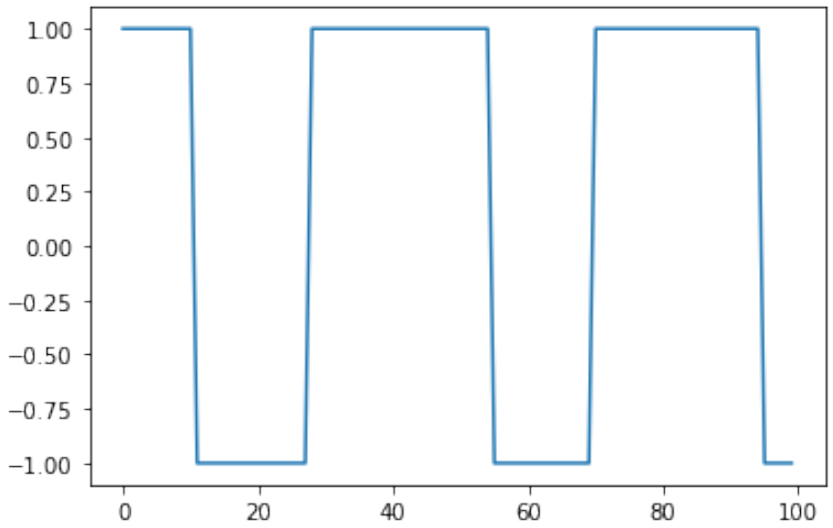

FIGURE 19 – Cette figure montre sur les 100 premier discours a quelle classe ils appartiennet

4.3.8 Résultat et paramétres du meilleur modèle :

Les meilleurs paramètres trouvés après cette étape de lissage étaient : Les performances du modèle ont été optimisées en utilisant les paramètres suivants :

Prétraitement	**Détails**
Suppression des balises HTML	Effectuée
Suppression de la ponctuation	Effectuée
Suppression des chiffres	Effectuée
Conversion en minuscules	Effectuée
Majuscules en marqueurs	Effectuée
Suppression des stopwords en français	Effectuée
Lemmatization	Effectuée

TABLE 3 – Détails du prétraitement des données

Les meilleurs paramètres trouvés après l'optimisation étaient :

Paramètre	**Valeur**
tfidf_ngram_range	(1, 3)
tfidf_min_df	5
tfidf_max_df	0.5
tfidf_max_features	None
tfidf_binary	True
C	10
Tolerance	1e-8
class_weight	balanced

TABLE 4 – Meilleurs paramètres trouvés

Les performances après cette optimisation ont été résumées comme suit :

Métrique	**Valeur**
F1 Score	**80.1120**
AUC	97.2702

TABLE 5 – Performances après l'optimisation

Toutes les expérimentations ainsi que les resultats sont enregistrés dans le notebook **"grid_search.ipynb"**.

5 Analyse de sentiments (Movies Review)

5.1 Analyse exploratoire des données

Dans cette partie, nous avons réalisé la même analyse afin d'explorer de manière approfondie nos données sur les revues de films.

5.1.1 Distribution des classes

Nous avons étudié la distribution des classes dans notre ensemble de données sur les revues des films. Les résultats indiquent que les deux classes concerant les avis positifs et négatifs sont équilibré, comme le montre le graphique ci-dessous.

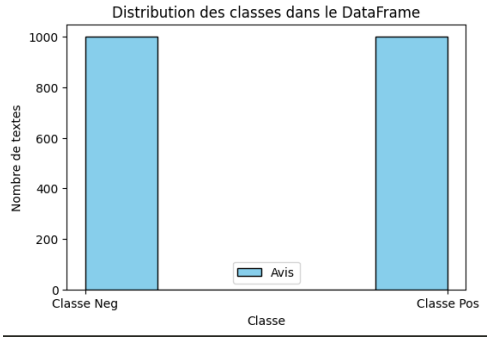

FIGURE 20 – Distribution des deux classes

5.1.2 Distribution des longueur des revues

Nous avons également examiné la répartition des longueurs de toutes les critiques de films. L'histogramme ci-dessous illustre cette distribution :

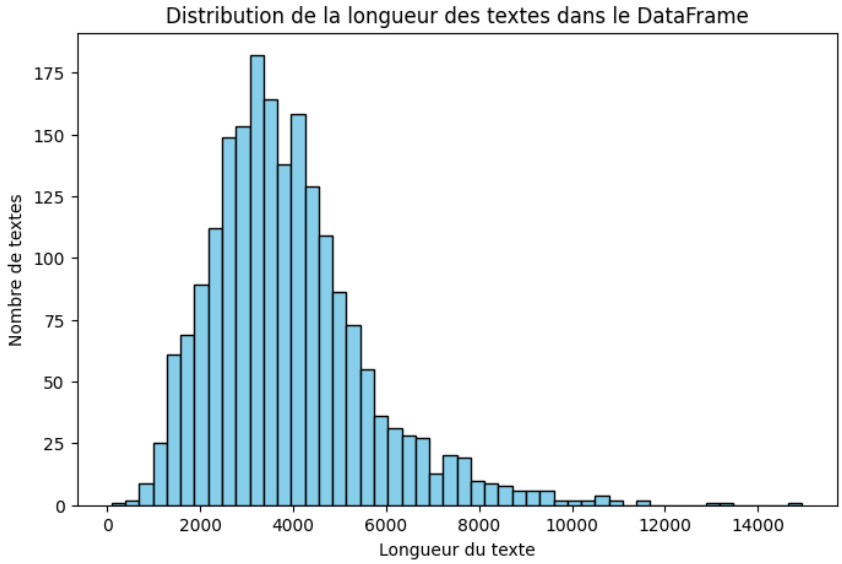

FIGURE 21 – Distribution des longueurs des revues des film

Cet histogramme révèle que la majorité des critiques ont une longueur comprise entre 2000 et 6000 mots.

Ensuite, nous avons analysé la distribution des longueurs dans chaque classe et obtenons les histogrammes suivants :

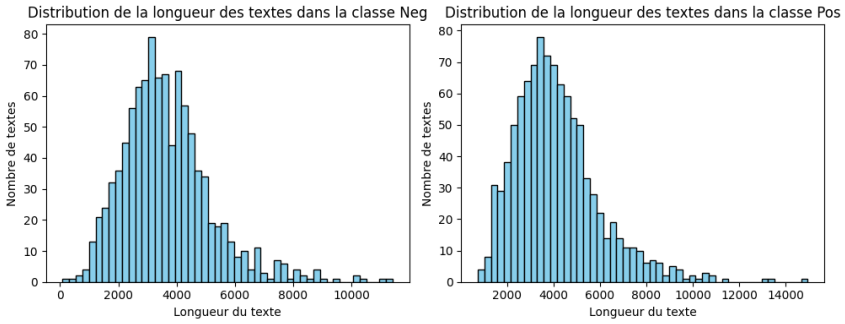

FIGURE 22 – Distribution des longueurs des revues des film par classes

D'après ces histogrammes, on peut conclure que la distribution des longueurs des textes est similaire pour les deux classes.

5.2 Transformation paramétrique du texte (pre-traitements)

Les prétraitements effectués dans cette partie du projet sont similaires à celles de la section précédente. Certaines adaptations ont été apportées : le corpus utilisé dans la première partie est en français, tandis que celui de la seconde est en anglais.

5.3 Extraction du vocabulaire (BoW)

Nous allons présenté les diverses caractéristiques du jeu de données concernant les revues de films.

5.3.1 Taille initiale du vocabulaire

Le vocabulaire initial comporte 39659 mots différents

5.3.2 Las 100 mots les plus fréquents

En observant le nuage de mots ci-dessous, il est notable que la plupart des termes fréquents sont des stop words (comme "the", "and", "you") et que la ponctuation est également abondante.

5.3.3 Les 100 mots les plus discriminants

Lorsqu'on parle de mots discriminants en termes d'odds ratio, on fait généralement référence à des mots qui sont fortement associés à une certaine catégorie ou classe par rapport à une autre. Dans ce contexte, nous comparons l'occurrence d'un mot dans une catégorie par rapport à son occurrence dans une autre catégorie.

$$\text{Odds ratio}(w) = \frac{\text{Fréquence de } w \text{ dans la classe } 1 * (1 - \text{Fréquence de } w \text{ dans la classe } 0)}{\text{Fréquence de } w \text{ dans la classe } 0 * (1 - \text{Fréquence de } w \text{ dans la classe } 1)}$$

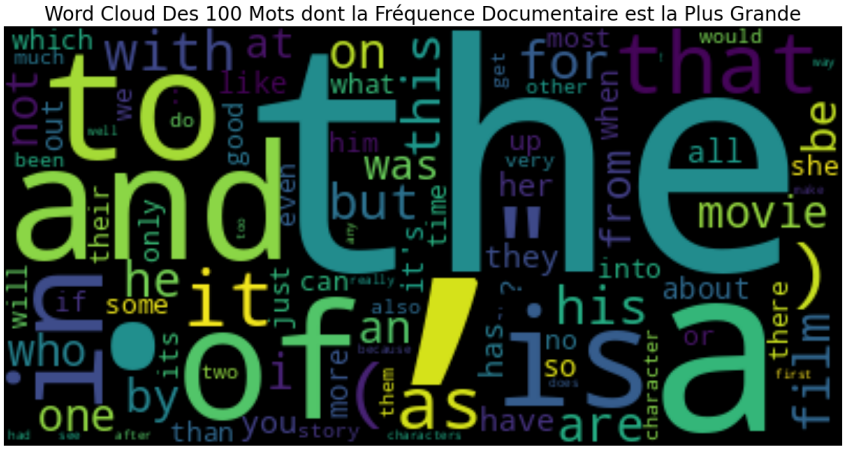

FIGURE 23 – Word cloud des 100 mots les plus fréquents

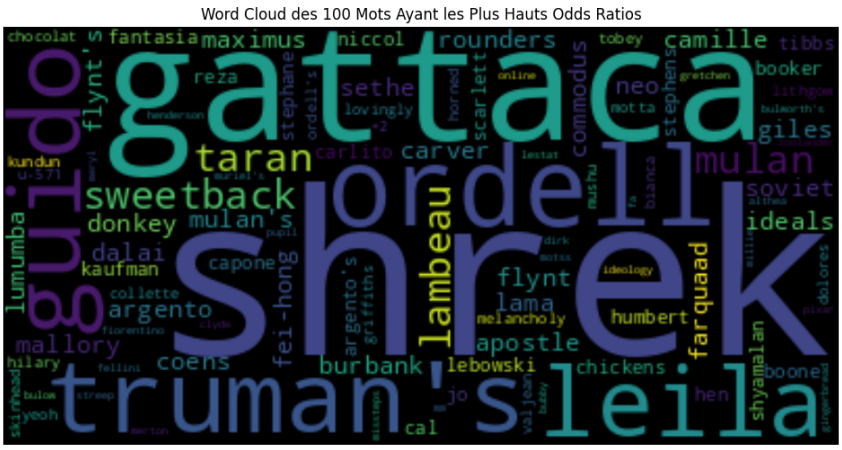

FIGURE 24 – Word cloud des 100 mots les plus discriminants

5.3.4 Distribution d'apparition des mots (Loi de Zipf)

La Loi de Zipf est une loi empirique de la distribution des fréquences des mots dans un langage naturel. Elle stipule que, dans une grande quantité de texte, la fréquence d'un mot est inversement proportionnelle à son rang dans l'ordre de fréquence.

D'aprés le graphe ci-dessus, on remarque que les fréquences des mots diminuent rapidement avec leur rang. L'utilisation d'une échelle logarithmique dans ce graphe, nous aidera à mieux visualiser la distribution des mots et à vérifier si les données suivent la loi de Zipf.

5.3.5 Les 100 bigrammes les plus fréquents

Parmi les 100 bigrammes les plus fréquents, les paires de mots suivantes se démarquent :

— *of the* avec 8778 occurrences
— *in the* avec 5807 occurrences
— *the film* avec 4544 occurrences
— *to the* avec 2778 occurrences
— *to be* avec 2745 occurrences

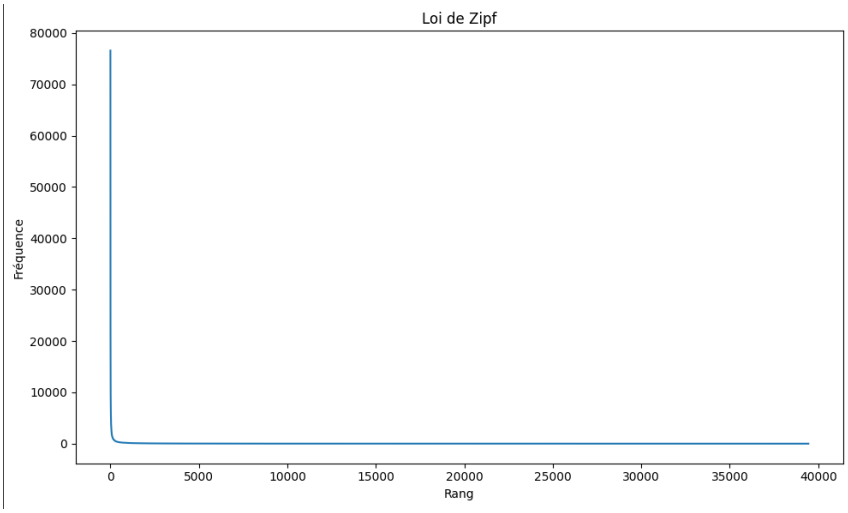

FIGURE 25 – Loi de Zipf

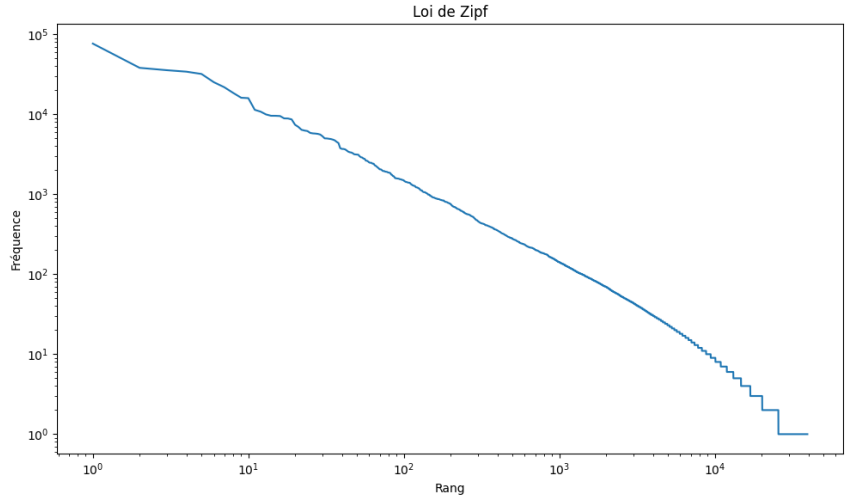

FIGURE 26 – Loi de Zipf

— ...

Ces bigrammes fournissent un aperçu des combinaisons de mots couramment utilisées dans les critiques de films, mettant en évidence des expressions fréquemment employées pour discuter du contenu, du scénario, ou de l'expérience du spectateur.

5.3.6 Les 100 trigrammes les plus fréquents

L'analyse des trigrammes les plus fréquents révèle les groupes de trois mots qui se répètent le plus souvent dans nos données :

— *one of the* avec 1026 occurrences
— *of the film* avec 886 occurrences
— *in the film* avec 585 occurrences
— *the film is* avec 543 occurrences
— *of the movie* avec 375 occurrences

— ...

Ces trigrammes offrent des indications sur les structures linguistiques récurrentes dans les critiques de films, mettant en evidence des phrases et des expressions souvent utilisées pour exprimer des opinions.

5.4 Analyse comparative de variantes BoW

Nous avons appliqué la méthode Bag-of-Words (BoW) pour représenter les textes sous forme de vecteurs numériques, en utilisant à la fois CountVectorizer et tf-idfVectorizer. Chaque variation de BoW a été associée à différents prétraitements textuels pour former des ensembles de données distincts. Ensuite, nous avons entraîné et évalué trois modèles de classification : Naïve Bayes, Régression Logistique et SVM. Les performances des modèles ont été mesurées en termes d'accuracy, de score F1, de précision, de rappel et de l'aire sous la courbe ROC (ROC AUC). Nous avons également enregistré le temps d'exécution de chaque modèle.

Pour tf-idfVectorizer, les performances du modèle de Régression Logistique étaient les suivantes :

TABLE 6 – Performance de tf-idfVectorizer avec différents pré-traitements pour le modèle de Régression Logistique

Pré-traitement	**Accuracy**	**F1-score**	**Precision**	**Recall**	**ROC AUC**
Sans aucun pré-traitement	0.83	0.83	0.81	0.85	0.83
Suppression de ponctuation	0.83	0.83	0.81	0.84	0.83
Suppression de chiffres	0.84	0.83	0.80	0.85	0.83
Suppression de Stops Words	0.84	0.83	0.82	0.85	0.84
Tout en miniscule	0.83	0.83	0.81	0.85	0.83
Stemming	0.83	0.83	0.82	0.84	0.83
Lemmatisation	0.85	0.84	0.84	0.85	0.85
Réduction du vocabulaire	0.83	0.83	0.80	0.86	0.83
Binary BoW	0.87	0.87	0.85	0.89	0.87
Bi-grams	0.82	0.82	0.81	0.83	0.82
Tri-grams	0.78	0.78	0.76	0.80	0.78
Binary Bow et Lemmatisation	0.86	0.86	0.84	0.89	0.86

Pour CountVectorizer, les performances du modèle de Régression Logistique étaient les suivantes :

TABLE 7 – Performance de CountVectorizer avec différents pré-traitements

Pré-traitement	Accuracy	F1-score	Precision	Recall	ROC AUC
Sans aucun pré-traitement	0.82	0.81	0.81	0.82	0.82
Suppression de ponctuation	0.82	0.82	0.81	0.83	0.82
Suppression de chiffres	0.82	0.82	0.81	0.83	0.82
Suppression de Stops Words	0.83	0.82	0.82	0.83	0.83
Tout en miniscule	0.82	0.81	0.81	0.82	0.82
Stemming	0.81	0.80	0.82	0.77	0.81
Lemmatisation	0.83	0.82	0.83	0.81	0.83
Réduction du vocabulaire	0.83	0.83	0.82	0.84	0.83
Binary BoW	0.86	0.86	0.85	0.86	0.86
Bi-grams	0.79	0.78	0.79	0.77	0.79
Tri-grams	0.72	0.70	0.75	0.66	0.72
Binary Bow et Lemmatisation	0.86	0.85	0.84	0.87	0.86

Voici quelques conclusions tirées de notre analyse :

— Tf-Idf donne de meilleurs résultats que CountVectorizer.

— Le meilleur prétraitement trouvé jusqu'à présent comprend la suppression des mots vides (stop words), la lemmatisation, l'utilisation d'un schéma binaire (BoW binaire), ainsi que des paramètres min_df=5, max_df=10000 et max_features=20000.

5.5 Analyse comparative des performances et des temps d'exécution des divers modèles

Afin de faire l'analyse comparative entre les differennts modéles, nous avons choisi d'utilisé tf_idfVectorizer pour cette partie étant donné qu'il donne de meilleure résultats comparant à CountVectorizer, et pour faire cette comparaison nous avons choisi les étapes suivantes :

— **Prétraitement des données** : Avant l'entraînement des modèles, les données textuelles subissent un prétraitement. Celui-ci englobe la suppression des balises HTML, des chiffres et de la ponctuation, la mise en minuscules, la lemmatisation, etc. Ces étapes visent à normaliser et à nettoyer les données textuelles pour améliorer l'efficacité des modèles.

— **Division des données** : Les données sont divisées en ensembles d'entraînement et de test à l'aide de la fonction train_test_split. Cette division permet d'évaluer les modèles sur des données indépendantes de celles sur lesquelles ils ont été formés.

— **Vectorisation des données** : Les textes sont convertis en vecteurs numériques grâce à la méthodes de vectorisation TfidfVectorizer

— **Entraînement des modèles** : Trois modèles de classification sont entraînés sur les données d'entraînement : Naïve Bayes, Régression Logistique et SVM linéaire. Chaque modèle est entraîné sur les différentes variantes de données obtenues après vectorisation et prétraitement.

— **Évaluation des modèles** : Les modèles entraînés sont évalués sur les données de test à l'aide de plusieurs métriques de performance, telles que l'accuracy, le score

F1, la précision, le rappel et l'aire sous la courbe ROC. Ces métriques permettent d'évaluer la qualité des prédictions des modèles sur les données de test.

5.5.1 Comparaisons des scores

Les graphes ci-dessous révèlent que, pour chaque modèle, les valeurs d'accuracy, de score F1 et d'aire sous la courbe ROC sont les même. Cepandant les métriques du modele Naive Bayes restent nférieures à celles de la Régression Linéaire et du SVM.

— **Régression Logistique**

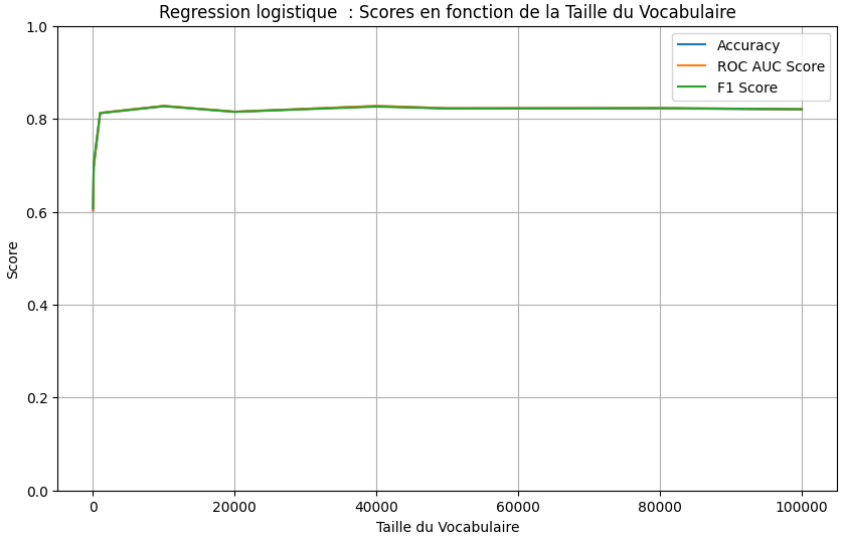

FIGURE 27 – Scores avec Régression logistique

— **Naive Bayes**

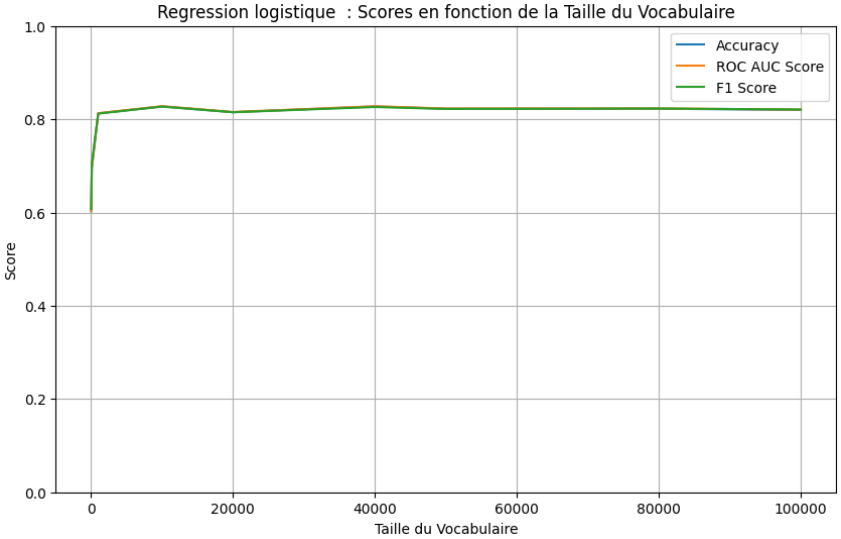

FIGURE 28 – Scores avecNaive Bayes

— **SVM**

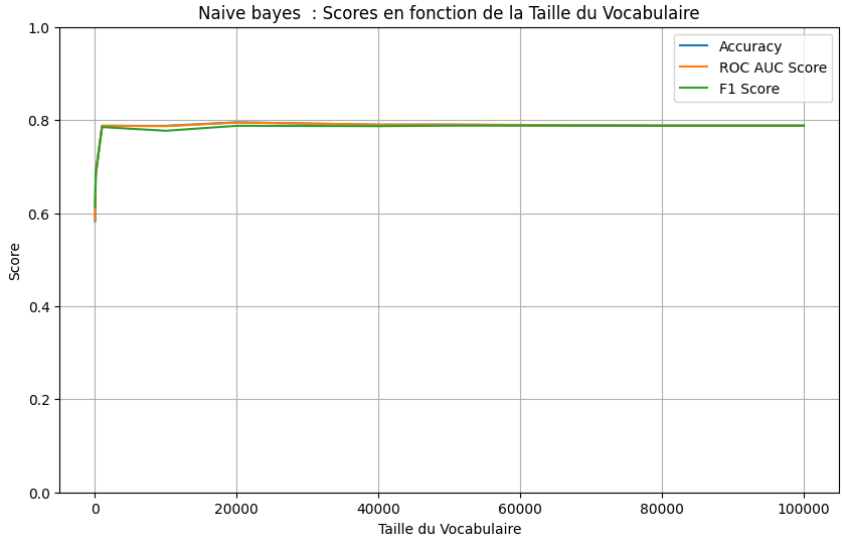

FIGURE 29 – Scores avec SVM

5.5.2 Comparaisons des temps d'execution

D'aprés ce graphe, il est clair que la régression linéaire nécessite davantage de temps en comparaison avec les modèles SVM et Naive Bayes. Notamment, le modèle Naive Bayes se distingue par sa rapidité significative.

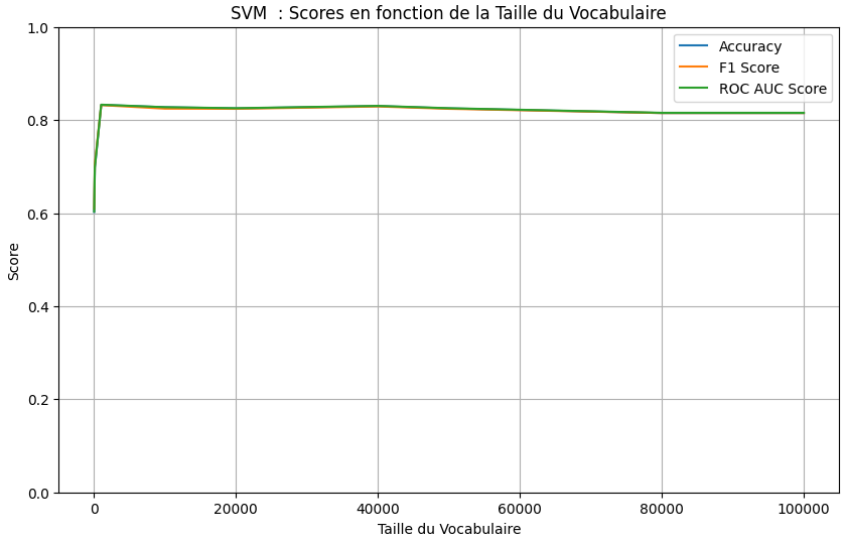

FIGURE 30 – Comparaisons des temps d'execution

5.5.3 Comparaison des performances

On constate que la regression linéaire et les SVM donnent de meilleurs performances par rapport à Naive Bayes.

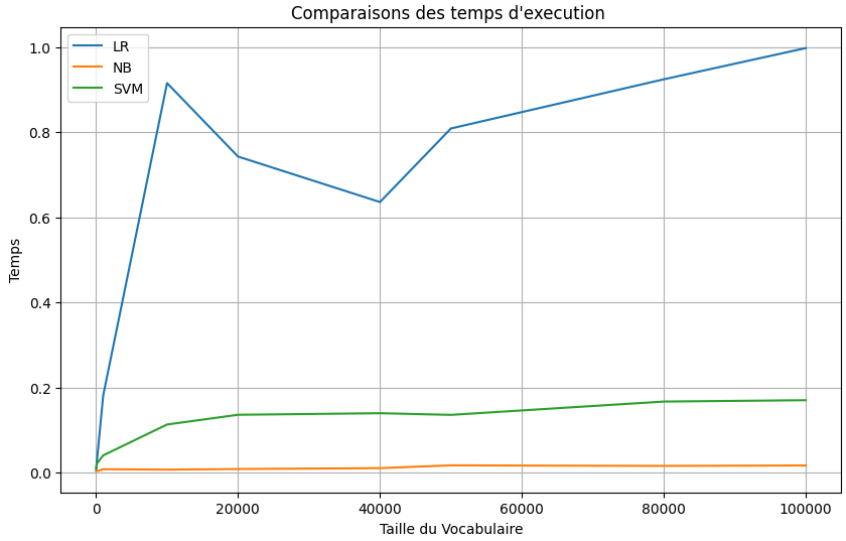

FIGURE 31 – Comparaisons des F1 scores global

6 Validation Croisée

6.1 Stabilité des performances

Pour évaluer la constance des performances, nous avons réalisé une validation croisée en variant le nombre de plis et en utilisant différentes valeurs de graine aléatoire. Les résultats indiquent que les performances commencent à se stabiliser à partir d'environ 6 ou 8 plis. Les graphiques ci-dessous illustrent cette conclusion :

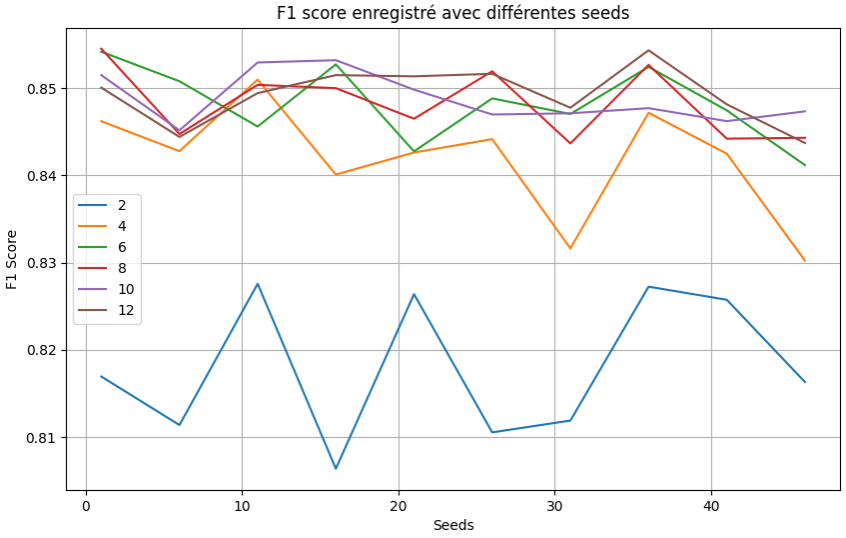

FIGURE 32 – Comparaisons stabilité du F1 score

Les performances sont également visualisées à l'aide de graphiques, tels que les courbes de scores en fonction du nombre de plis, ainsi que les violonplots pour représenter la distribution des scores.

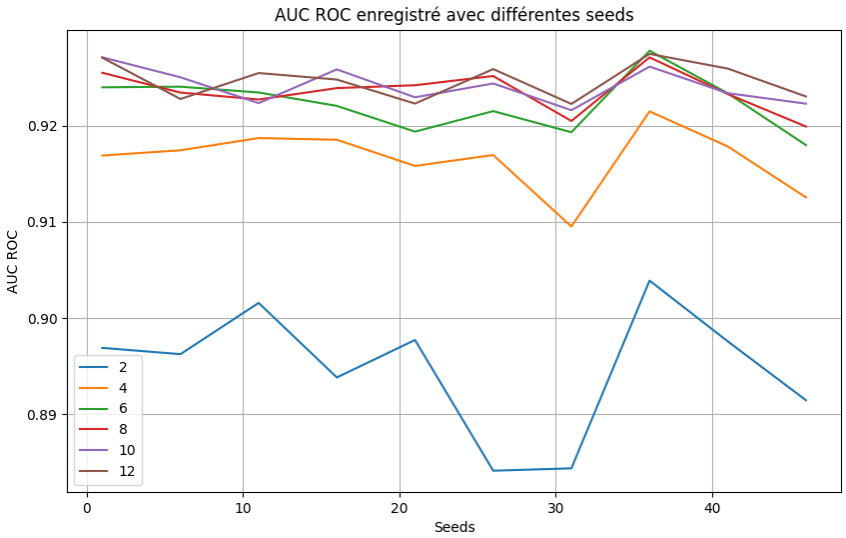

FIGURE 33 – Comparaisons stabilité de la courbe ROC

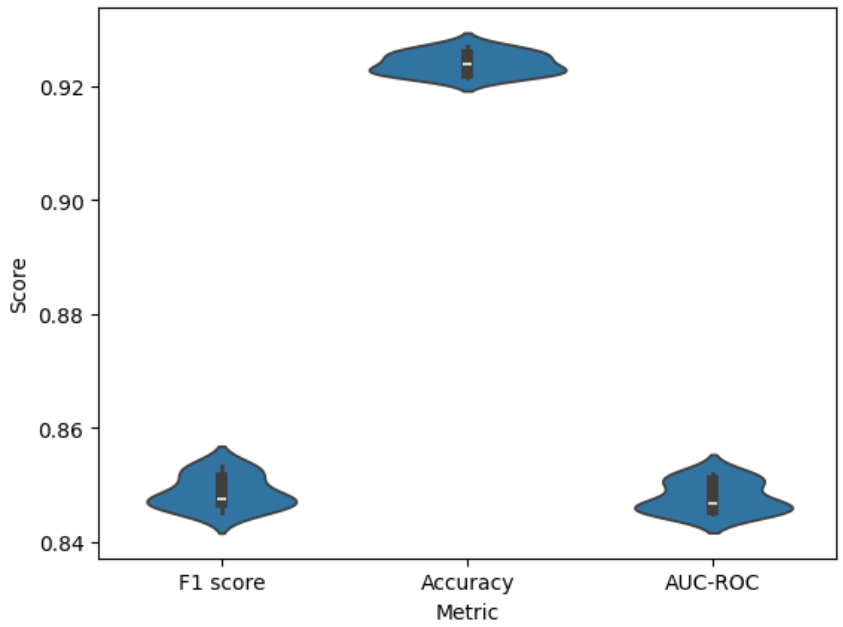

FIGURE 34 – Visualisations de la distribution suivies par les scores renvoyés

En conclusion, bien que dans des cas simples, une simple séparation des données donne des performances similaires à celles de la validation croisée, cette dernière devrait être privilégiée lors de l'évaluation des performances des modèles d'apprentissage automatique en raison de sa robustesse, de sa fiabilité et de sa précision accrue. C'est pourquoi, lors de nos tests les plus approfondis, nous utiliserons une recherche sur grille, qui repose précisément sur la validation croisée.

6.2 Méthode de Grid Search

Dans cette phase de notre projet, nous avons adopté la méthode de Grid Search dans le but de déterminer la configuration optimale des paramètres afin d'optimiser les performances de notre modèle.

Pour ce faire, nous avons mis en place une pipeline intégrant le TF-IDF Vectorizer, accompagné d'une fonction de prétraitement spécifique visant à éliminer les balises HTML, les chiffres, la ponctuation, et à effectuer la lemmatisation. Les paramètres à optimiser englobaient divers aspects de la vectorisation tels que max_df, min_df, ngram_range, ainsi que les paramètres de régularisation de la régression logistique comme C et penalty.

La procédure de Grid Search a été exécutée en utilisant une validation croisée. Nous avons évalué les performances du modèle en utilisant des métriques telles que l'accuracy, le score F1 et l'AUC ROC. Les données ont été divisées en ensembles d'entraînement et de test dans un rapport de 80 :20.

Cette analyse exhaustive a permis d'identifier les paramètres les plus performants pour notre modèle de classification, qui se résument comme suit :

- **tfidf_use_idf** : True.
- **tfidf_sublinear_tf** : True.
- **tfidf_ngram_range** : (1, 2).
- **tfidf_min_df** : 5.
- **tfidf_max_features** : 20000.
- **tfidf_max_df** : 0.3.
- **tfidf_lowercase** : False.
- **tfidf_binary** : True.
- **penalty** : l2.
- **C** : 100.