Sorbonne Université

M1 Informatique - DAC
Compte rendu RITAL

Reconnaissance des locuteurs et analyse de sentiments des reviews

---

Etudiants :
Ghiles Ouhenia
Amayas Sadi

Encadrant :
Nicolas Thome

19 mars 2023

Table des matières

1 Introduction

Ce rapport de projet présente une campagne d'expériences menée pour résoudre deux problématiques de classification de documents textuels en utilisant des techniques de Machine Learning et de Traitement Automatique du Langage Naturel.

La première problématique consiste en la reconnaissance du locuteur à partir de discussions présidentielles de Chirac et Mitterrand, tandis que la seconde consiste à prédire les sentiments positifs ou négatifs des critiques de films.

Pour atteindre ces objectifs, une chaîne de traitement flexible a été mise en place pour entraîner des modèles d'apprentissage avec des paramètres optimaux.

2 Analyse des données

2.1 Problème de reconnaissance du locuteur

La base de données des locuteurs contient un total de 57 413 messages échangés entre les présidents Chirac et Mitterrand. Nous avons effectué une analyse des données pour extraire les informations clés.

Pour cela, nous avons créé des WordClouds pour les mots, les bi-grammes et les tri-grammes les plus fréquents dans les discussions. Ces WordClouds ont permis de visualiser les tendances dans les discours des deux présidents.

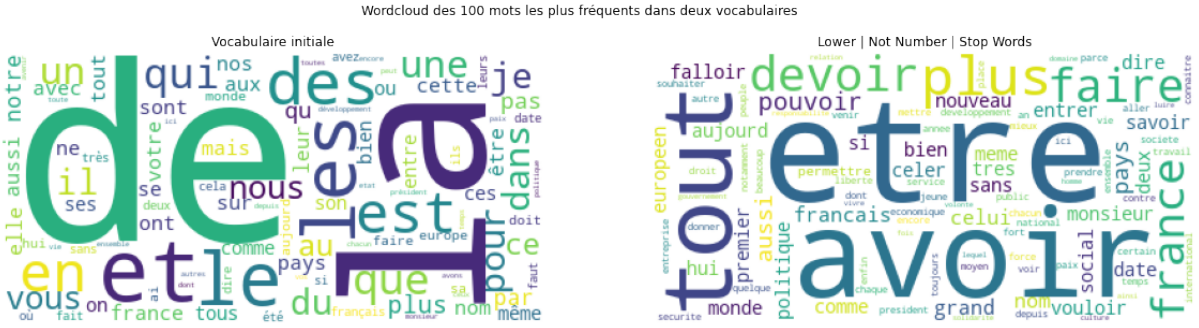

FIGURE 1 – Les mots les plus fréquents avant et aprés préprocessing

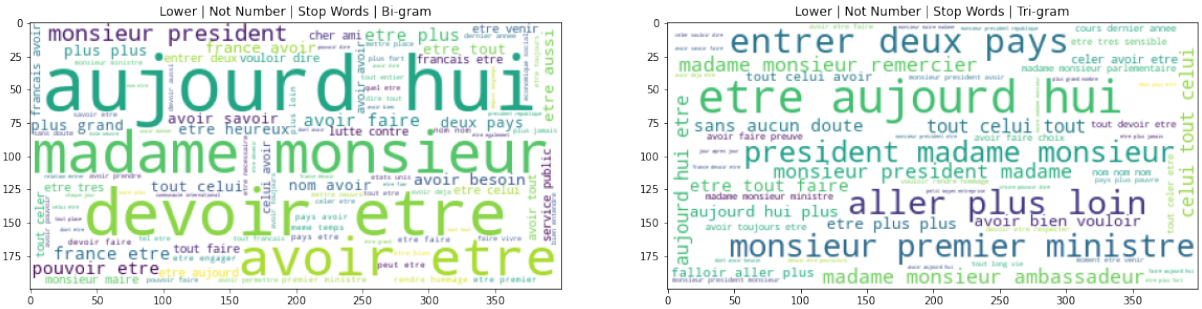

FIGURE 2 – Les bi-grammes et tri-grammes les plus fréquents

Nous avons également utilisé des techniques de statistiques pour identifier les mots les plus discriminants pour chaque locuteur, au sens de l'odds ratio. Les mots les plus discriminants étaient ceux qui apparaissaient fréquemment dans les messages d'un locuteur mais rarement dans ceux de l'autre locuteur.

FIGURE 3 – Les mots les plus discriminants au sens de odds ratio avant et aprés préprocessing

2.2 Anaylse des sentiments des critiques

Tout comme pour l'analyse des échanges, nous avons effectué une analyse des données afin d'extraire des vocabulaires que nous avons ensuite visualisés à l'aide de WordClouds.

FIGURE 4 – Les mots les plus fréquents avant et aprés préprocessing

FIGURE 5 – Les mots les plus discriminants au sens de odds ratio avant et aprés préprocessing

3 Chaîne de prétraitement

Plusieurs méthodes ont été développées pour traiter les données textuelles brutes, et elles peuvent être activées ou désactivées pour évaluer leur impact sur la qualité des prédictions.

Parmi nos pré-traitements effectués :

- Suppression des stop-words :
  - Pour cela, nous avons utilisé la bibliothèque NLTK.
- Suppression des mots très fréquents et très peu fréquents :
  - Nous avons utilisé directement les paramètres min_df et max_df de la méthode TF-IDF.
- Transformation TF-IDF
  -Il s'agit d'une transformation simple qui ne nécessite pas l'ajout de paramètres.
- Lemmatization
  - Nous avons utilisé la bibliothèque Spacy.
- Transformation en minuscule et découpage en mots

Nous avons opté pour la mise en œuvre de chaque stratégie sous la forme d'un dictionnaire, où chaque clé est associée à un paramètre particulier. À titre illustratif, voici un exemple :

```
strats.append({
        "name":"Lower | Not Number | Stop Words | TF",
        "ToLower": True,
        "DeleteNumbers": True,
        "stopwords": STOP_WORDS,
        "use_idf": False,
        "lemmatized": True,
      }
```

Cette approche nous permet de concevoir et tester autant de stratégies que nécessaire. Il suffit de créer un dictionnaire et de régler les paramètres souhaités. Un avantage supplémentaire est que nous pouvons stocker chaque modèle d'apprentissage entraîné et chaque BoW de la base d'entraînement en utilisant des dictionnaires, ce qui nous évite de devoir les recalculer à chaque fois.

4 Machine learning

Nos modèles de prédiction utilisés :

- Régression Logistique
- SVM
- RandomForest
- Naive Bayes

Pour lesquels nous avons optimisé les paramètres en testant plusieurs combinaisons à l'aide des techniques GridSearchCV et RandomizedSearchCV.

Pour chacun de nos modèles, ainsi que pour chaque combinaison de paramètres des stratégies de prétraitement, nous les avons évalués à l'aide des scores suivants :

- Score f1
- Accuracy
- Courbe ROC
- Précision et rappel

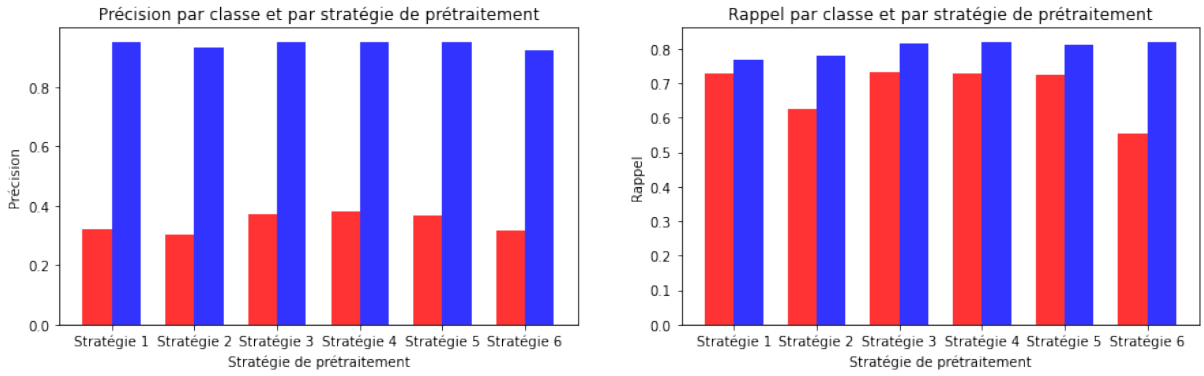

FIGURE 6 – Histogramme des précision et rappel de chaque classe pour chaque stratégie avec un model donné

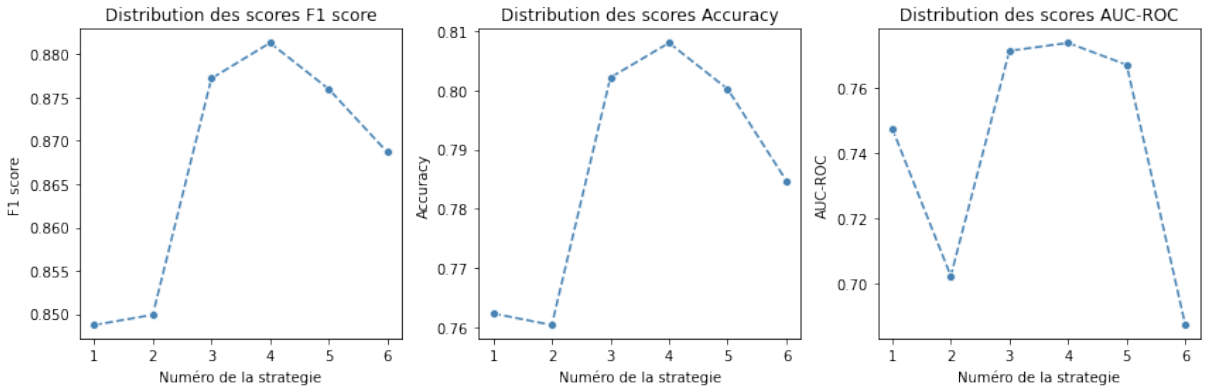

FIGURE 7 – Courbe des scores pour chaque stratégie avec un model donné

4.1 Déséquilibre de classes

La base de données des échanges entre les présidents présente un déséquilibre de classes important, avec une forte prédominance des messages de Chirac par rapport à ceux de Mitterrand (86% contre 13%). Ce déséquilibre peut biaiser l'apprentissage de modèles de classification et limiter leur capacité à généraliser correctement.

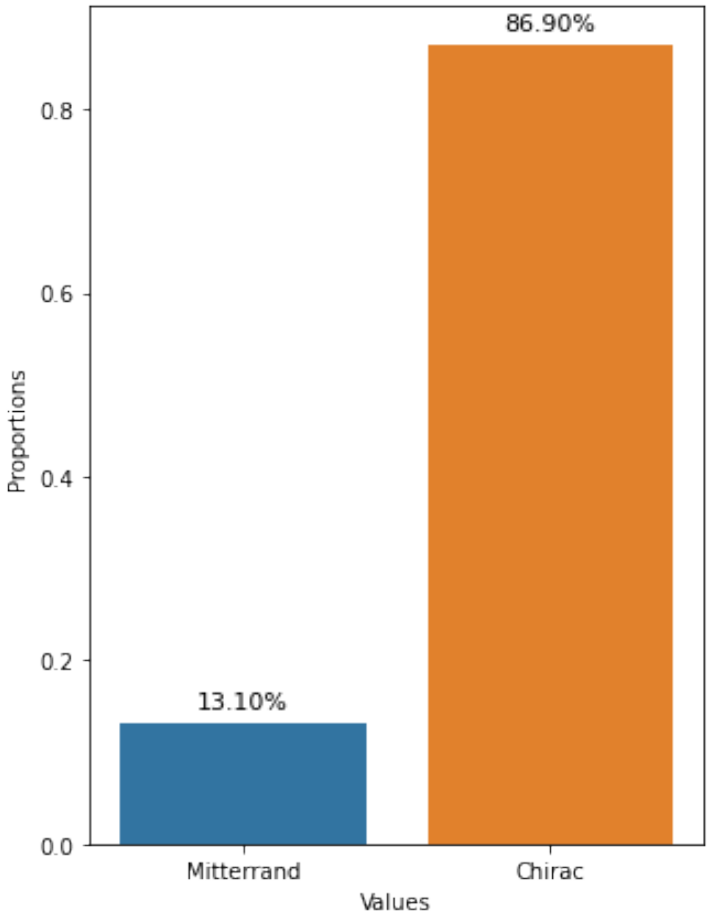

FIGURE 8 – Proportions de messages de chaque locuteur

Afin de remédier à cette situation, nous avons décidé d'utiliser des techniques de sous-échantillonnage telles que RandomUnderSampler provenant de la bibliothèque imblearn, ainsi que des techniques de sur-échantillonnage telles que SMOTE.

Dans les parties qui suivent, les résultats présentés concernent la base de données des présidents.

4.2 Régression Logistique

Nous avons essayé de maximiser la performance de chaque stratégie en ajustant les hyperparamètres 'C' et 'penalty' de la régression logistique à l'aide de GridSearchCV. Les valeurs possibles pour 'C' étaient [0.01, 0.1, 0.5, 1, 2], tandis que pour 'penalty', les options étaient 'l2' et 'None'.

Voici les résultats que nous avons obtenus pour la base des présidents :

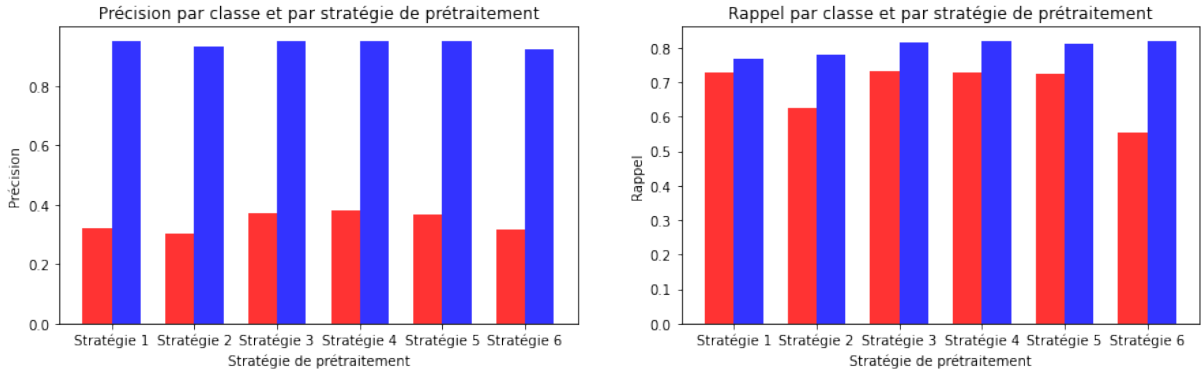

FIGURE 9 – Histogramme des précision et rappel de chaque classe pour chaque stratégie avec RegLog

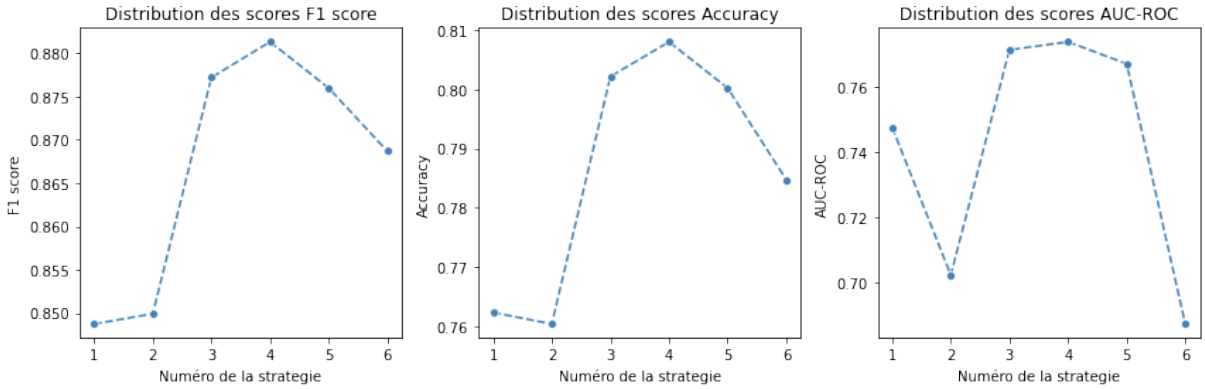

FIGURE 10 – Courbe des scores pour chaque stratégie avec RegLog

Nous remarquons que la stratégie 6 est la plus optimale avec un score F1 de 0,88. Toutefois, notre algorithme manque de précision en raison du déséquilibre des classes.

4.3 SVM

Nous avons cherché à optimiser les performances de chaque stratégie en ajustant les hyperparamètres 'C' et 'penalty' du SVM à l'aide de RandomizedSearchCV. Cependant, ce processus était très chronophage, ce qui nous a amenés à opter pour l'utilisation d'un LinearSVC.

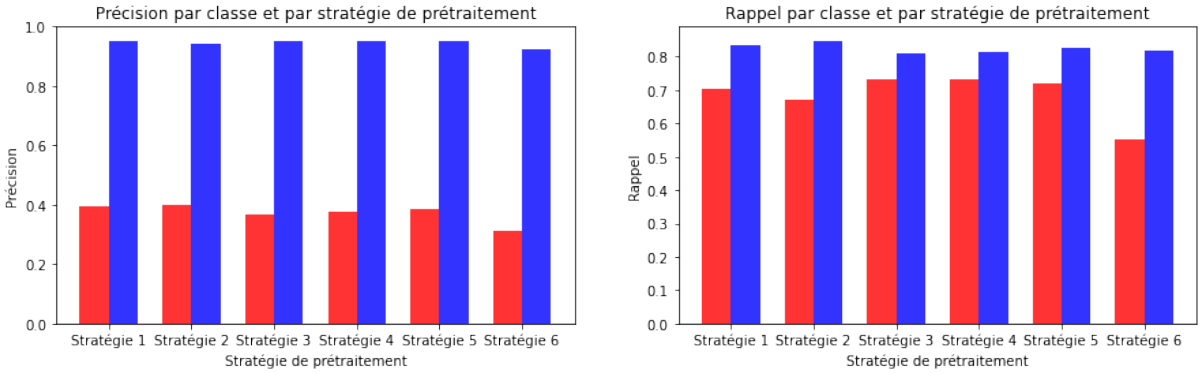

FIGURE 11 – Histogramme des précision et rappel de chaque classe pour chaque stratégie avec SVM

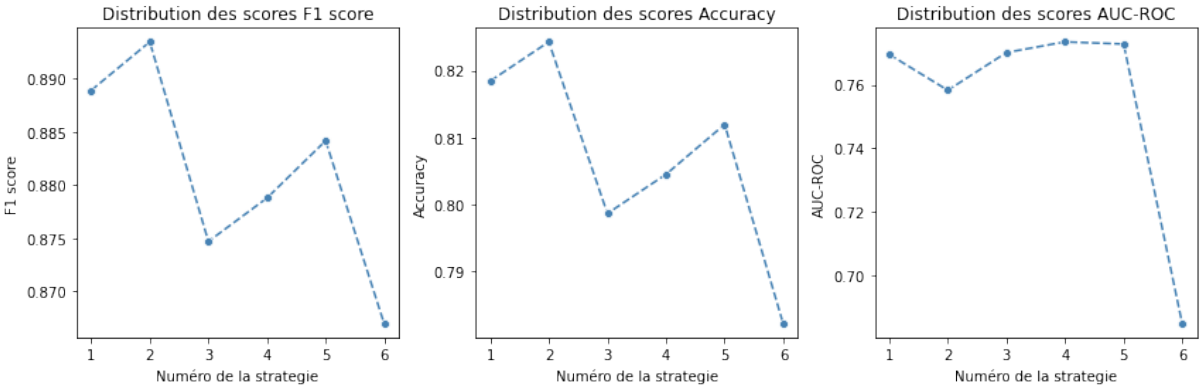

FIGURE 12 – Courbe des scores pour chaque stratégie avec SVM

4.4 RandomForest

Pour le model Random Forest, nous avons cherché à optimiser les performances du modèle en ajustant les hyperparamètres 'max_depth' et 'min_samples_split' à l'aide de GridSearchCV. 'max_depth' correspond à la profondeur maximale de chaque arbre dans la forêt, tandis que 'min_samples_split' représente le nombre minimal d'échantillons requis pour diviser un nœud interne de l'arbre en sous-groupes. Nous avons testé différentes valeurs pour 'max_depth' (5, 7 et 12) et pour 'min_samples_split' (5, 10, 20 et 30).

Pour la valeur de max_depth, nous avons conclu qu'il est préférable d'utiliser une valeur inférieure à 16, car les arbres résultants auraient plus de 2^{16} feuilles, ce qui est supérieur au nombre d'exemples dans la base de données et peut conduire à un sur-apprentissage

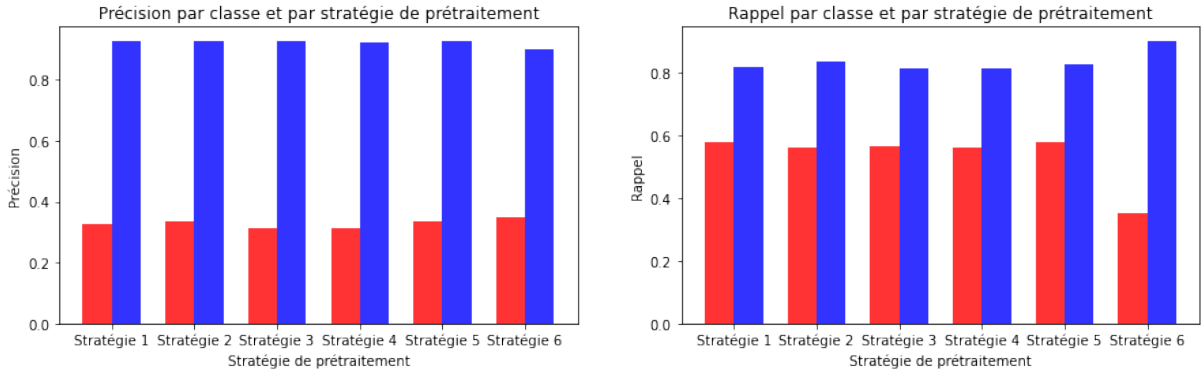

FIGURE 13 – Histogramme des précision et rappel de chaque classe pour chaque stratégie avec RandomForest

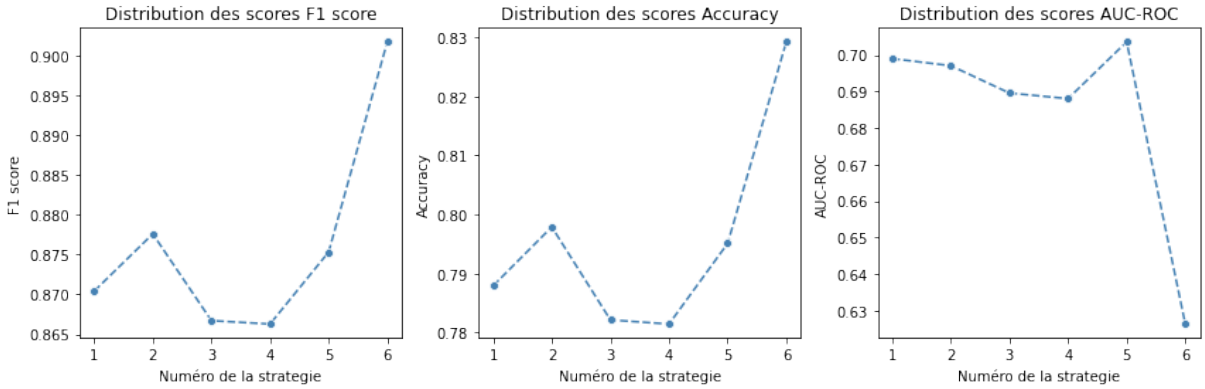

FIGURE 14 – Courbe des scores pour chaque stratégie avec RandomForest

4.5 Naive Bayes

Concernant Naive Bayes, nous avons décidé de ne pas optimiser le seul paramètre existant, qui est alpha.

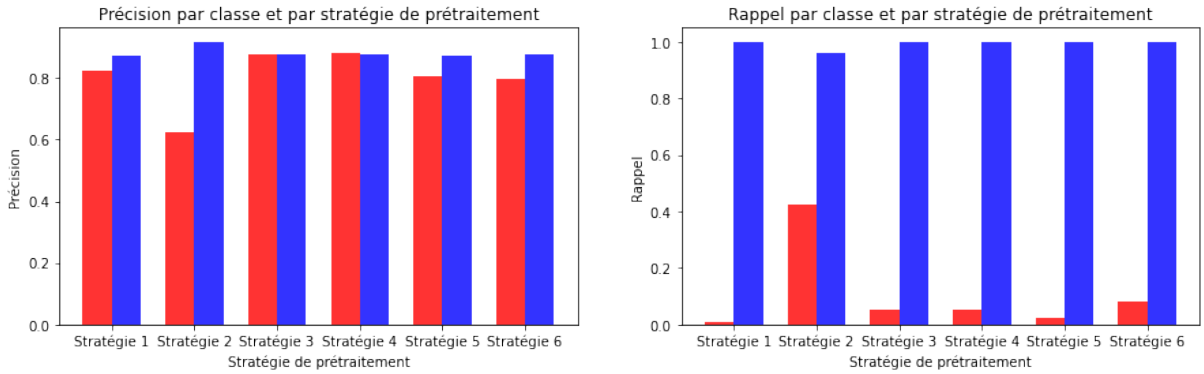

FIGURE 15 – Histogramme des précision et rappel de chaque classe pour chaque stratégie avec Naive Bayes

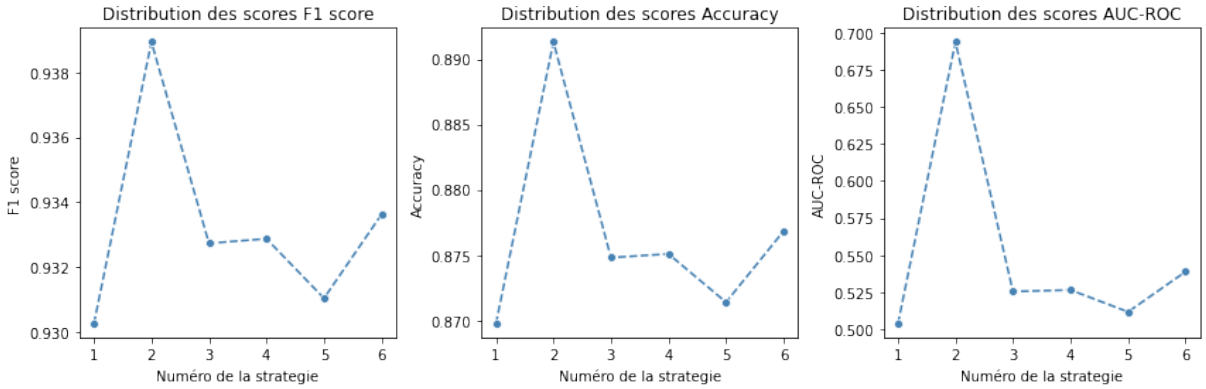

FIGURE 16 – Courbe des scores pour chaque stratégie avec Naive Bayes

On remarque que la précision pour la classe minoritaire avec Naive Bayes est élevée.

5 Conclusion

En conclusion de cette campagne d'expériences, nous pouvons dire que le pré-processing des données textuelles ne garantit pas toujours un meilleur score. Parfois, cela peut simplement conduire à la perte d'informations. Pour trouver la version optimale de la chaîne de traitement, il est intéressant d'essayer toutes les combinaisons possibles de prétraitement et de visualiser les données induites pour pouvoir interpréter et expliquer les différences.

Nous avons utilisé différentes stratégies d'optimisation pour ajuster les hyperparamètres de chaque algorithme et nous avons constaté que les performances varient en fonction de l'algorithme et de la base de données

En fin de compte, cette expérience nous a permis de mieux comprendre l'importance du prétraitement des données ainsi que l'optimisation des modèles pour améliorer les scores de classification de textes. Le choix de l'algorithme et de la stratégie d'optimisation dépendent des caractéristiques de la base de données et de l'objectif de la classification. Il est donc important d'évaluer différentes options et de comparer les performances avant de choisir l'algorithme final.