RITAL

Recherche d'information
et Traitement du Langage Naturel

Rapport classification de texte

Étudiants: — *Numéros:*
Balkis Bouthaina DIRAHOUI — 21113733
Miray Suzan SENYUZ — 3802805

Mars 2022

Table des matières

1 Introduction

Les problèmes de classification de texte sont des problemes recurrents depuis plusieurs années, en effet, plusieurs facteurs influencient les resultats de classification.

Nous serons ainsi amenés a étudier ce problème de a travers deux jeux de données différents, l'un étant sur un jeux de données desequilibré des discours de présidents (Chirac Mitterand) et l'autre de classfication de films.

Ce document décrit le travail réalisé dans le cadre d'un projet académique dans le but de concrétiser et améliorer nos connaissances pratiques et théoriques récoltées dans l'Unité d'Enseignement *"Recherche d'information et Traitement du Langage Naturel"* par le développement d'un programme qui s'intéresse a résoudre une problématique de classfication.

2 Modèles

Nous étudierons les performances des modèles SVM [1], Naive Bayes [2] et la régression logistique [3].

Nous appliquerons différents pré-traitement et différentes régularisation aux modèles afin d'étudier les performances de ces dernier.

3 Implementation et experimentation

3.1 Jeux de données (Datasets)

Nous testerons (comme precisé precedemment)sur les deux dataset Chirac et Mitterand, et Dataset de Films.

- **Dataset Chirac Mitterand** :

  Nous travaillerons sur un dataset contenat le discours de deux présidents, nous devrons classifier chaque texte. Nous pouvons voir que la dataset est enormement désiquiliberé avec mitterand etant la classe majoritaire.

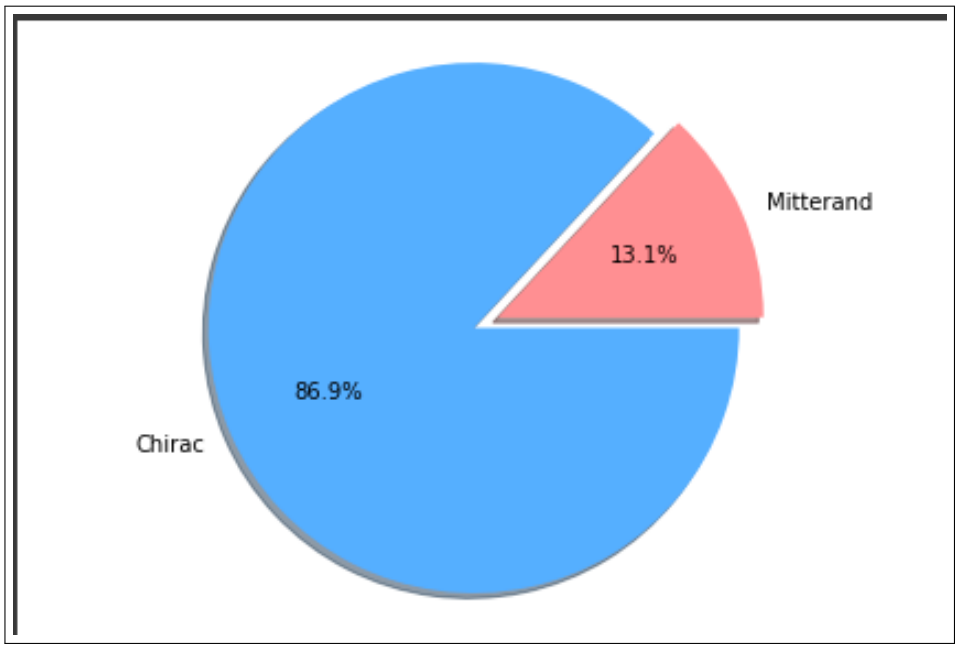


Figure 1: Equilibre Dataset Chirac Mitterand

– **Dataset Films :**

  Nous travaillerons sur un dataset contenat le script de films, nous devrons classifier le sentiment de chaque test. Nous pouvons voir que contrairement au premier dataset , celui-ci est equilibré.

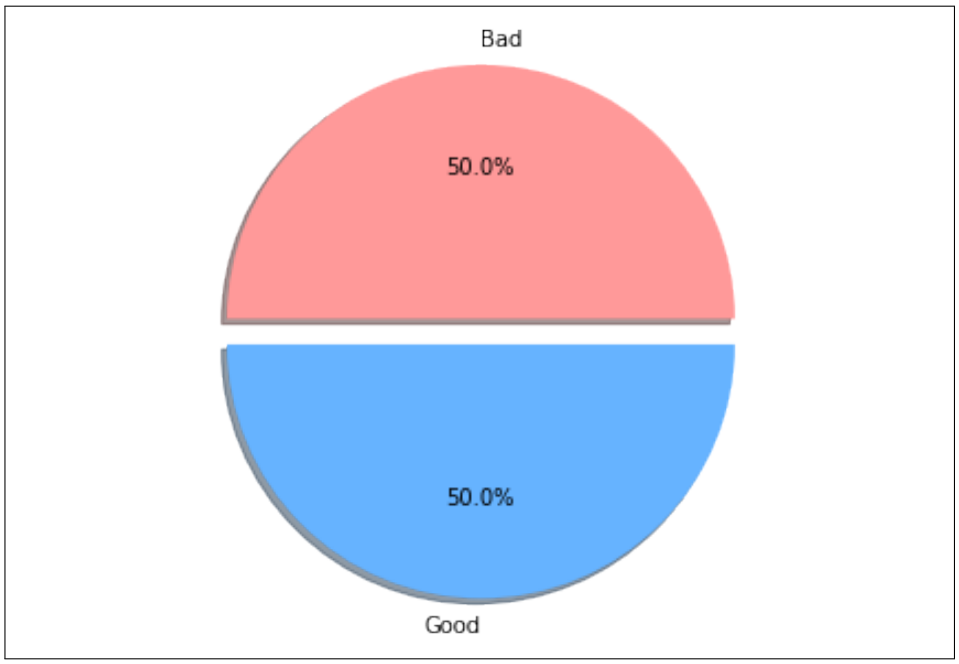


Figure 2: Equilibre Dataset Film

3.2 Métriques d'évaluation

Afin d'évaluer nos modèles, nous avons choisi la métrique $F - score$, qui se base sur le rappel et la précision.

Nous détaillons ces métriques ci-dessous :

– **Précision** : Nous utiliserons la précision afin d'évaluer le pourcentage de bonne classification de la classe positive. Par exemple,dans le premier *dataset*, nous évaluons le pourcentage de phrases correctement associées a Mitterrand parmi les phrases soumises. Sa formule est la suivante :

$$\text{Precision} = \frac{TP}{TP+FP}$$

Le rappel est une métrique qui quantifie le nombre de prédictions positives correctes faites de tous les positifs

– **Rappel** Nous utiliserons le rappel afin d'évaluer le pourcentagede prédictions positives correctes faites de tous les positifs . Par exemple,dans le premier *dataset*, nous évaluons le pourcentage de phrases correctement associées a Mitterrand parmi les phrase Mitterand réellement présentes dans le corpus. Sa formule est la suivante :

$$\text{Recall} = \frac{TP}{TP+FN}$$

– **F-score** La F-mesure correspond à un rapport entre la précision et le rappel donnant la performance du système.Sa formule est donnée comme suit :

$$\text{F1} = \frac{2*Precision*Recall}{Precision+Recall} = \frac{2*TP}{2*TP+FP+FN}$$

3.3 Pré traitement des données

Avant de lancer nos modelés, nous considérons le pré-traitement de nos données.

Le pré-traitement sera fait grâce a des opérations diverses, nous les citons ci-dessus :

– ***Stopwords***: Certains mots se retrouvent très fréquemment dans une langue.Ces derniers n'apportent pas très souvent de l'information et peuvent ensuite déstabiliser notre encodage (en Tf-IDF par exemple) , ces mots sont appelé en anglais les *Stop Words*.

– ***Stemming***: Le stemming a pour but de mettre un mot dans sa forme racine, et donc de regrouper plusieurs variantes d'un mot comme un même mot.

– ***Ponctuation***: Nous avons éliminer la ponctuation du textes dans un premier temps, dans un deuxième temps nous testons dans la section Résultats l'apport de cette de l'absence de la ponctuation sur les différents modelés.

– ***Chiffres***: Nous avons éliminer les chiffres du textes, ensuite, nous testons dans la section Résultats l'apport de cette opération.

– ***Tokenization***: Cette opération a pour but de transformer du texte en *tokens*, nous avons ensuite évaluer dans ce qui suit comment prendre les mots en n-gramme fait varier les performances des différents modèles.

– ***Vectorisation***:Afin d'encoder notre texte, nous avons choisi de comparer entre deux méthodes, la méthode de TF ou *Term Frequency* en anglais, qui est a vectorisation des mots en comptant combien de fois ils apparaissent dans les documents. La seconde approche que nous avons testé est l'approche $TF - IDF$ ou *Term Frequency — Inverse Document Frequency* en anglais , qui est une technique basée sur les sacs de mots pour encoder un texte.

3.4 Post-Processing

On a utilise le post processing pour les predictions des presidents. On avait remarque que les tests etait en forme bloque pour les classes. La fonction post_change les predictions si pour une cellule, tous les n cellules avant et apres sont tous egale, alors cette cellule va aussi etre egale.

Apres faisont des tests, on a trouve n=3, alors on cherche dans les cellules de longueur 7.

Le post processing etait utilise que dans les donnees des presidents et pas de movies.

3.5 Résultats

Dans les résultats qui suivent nous faisons varier les valeurs des paramètres de pré-traitement selon les natures indiqués dans le tableau suivant.

paramètre	*stopwords*	*stemming*	*miniscule*	*ngram*	*vecteur*
nature	*boolen*	*boolen*	*boolen*	(1,1),(1,2),(1,3)	Tf/Tfidf

3.5.1 Dataset Présidents

Nous testons les différentes combinaisons sur le dataset président.

- **SVM**
  - **Avec encodage tf ou tf idf**
    Nous représentons dans les tableaux ci-dessous les scores du modèles SVM sur le dataset avec une validation croisée en faisant varier plusieurs paramètres (avec les encodages tf et tfidf).

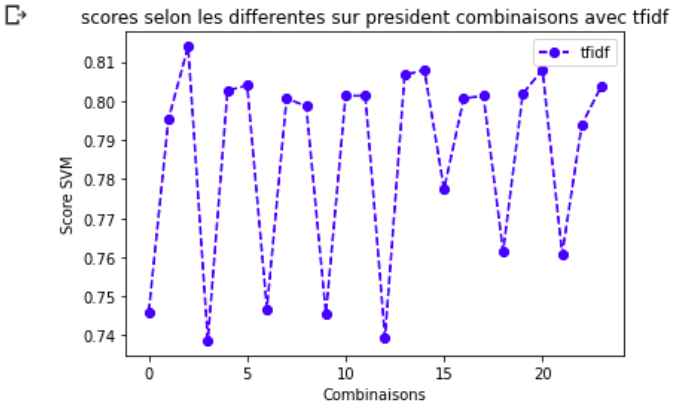
    Nous pouvons voir que la meilleure performance pour un codage tfidf est la combinaisons 2, en faisant le pre-processing et en prenant des n-gram en (1,3).
    En ce qui concerne le codage tf, nous pouvons remarquer que la meilleure combinaison est la combinaison numéro 16, ou on n'a presque pas de pré-processing.
    Les figures ci-dessous illustrent les résultats des tableaux.

Table 1: Tableau représentant les scores de l'encodage tfidf pour le modèle SVM

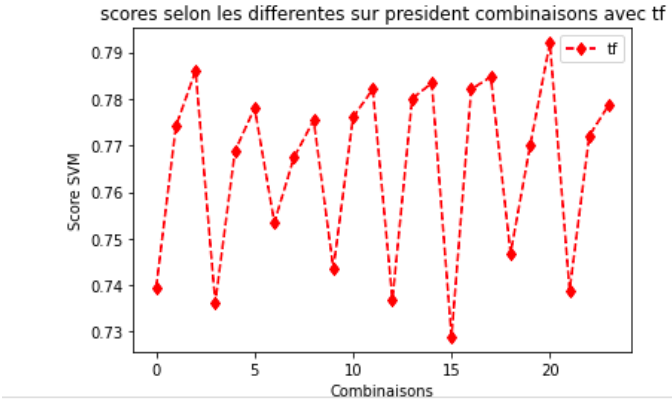
	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.746011
1	*True*	*True*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.795349
2	*True*	*True*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.813953
3	*True*	*True*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.738697
4	*True*	*True*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.802658
5	*True*	*True*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.803987
6	*True*	*False*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.746844
7	*True*	*False*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.800664
8	*True*	*False*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.798671
9	*True*	*False*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.745515
10	*True*	*False*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.801329
11	*True*	*False*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.801329
12	*False*	*True*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.739535
13	*False*	*True*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.806645
14	*False*	*True*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.807973
15	*False*	*True*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.777409
16	*False*	*True*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.800664
17	*False*	*True*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.801329
18	*False*	*False*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.761462
19	*False*	*False*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.801862
20	*False*	*False*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.807973
21	*False*	*False*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.760797
22	*False*	*False*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.79402
23	*False*	*False*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.803856

Table 2: Tableau représentant les scores de l'encodage tf pour le modèle SVM

	$stopword$	$stemming$	$miniscule$	$ngram$	$vecteur$	$chiffre$	$balancer$	$punct$	$score$
0	$True$	$True$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.739535
1	$True$	$True$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.774086
2	$True$	$True$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.786047
3	$True$	$True$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.736213
4	$True$	$True$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.768771
5	$True$	$True$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.778073
6	$True$	$False$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.753488
7	$True$	$False$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.767442
8	$True$	$False$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.775415
9	$True$	$False$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.743522
10	$True$	$False$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.77608
11	$True$	$False$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.78206
12	$False$	$True$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.736877
13	$False$	$True$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.780066
14	$False$	$True$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.783389
15	$False$	$True$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.728904
16	$False$	$True$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.78206
17	$False$	$True$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.784718
18	$False$	$False$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.746844
19	$False$	$False$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.7701
20	$False$	$False$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.792027
21	$False$	$False$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.73887
22	$False$	$False$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.772093
23	$False$	$False$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.778738



(a) SVM président avec codage tfidf



(b) SVM président avec codage tf

Figure 3: Modèle SVM selon les différentes comparaisons du tableau

– **Comparaison encodage tf et tf idf** La figure ci-dessous illustre la comparaison des performances des deux type d'encodage avec les differentes combinaisons (avec ou sans stop words..).

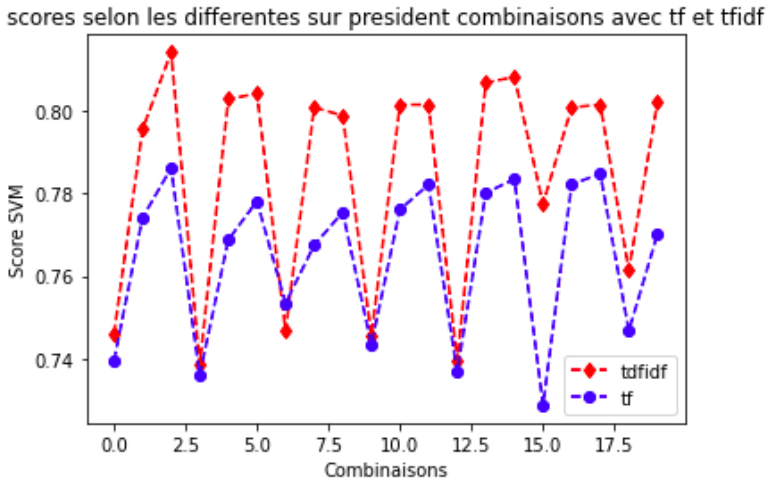


Figure 4: Comparaison des codges tf et tfidf

Naives Bayes

- **Avec encodage tf ou tf idf**

  Dans ces taleaux, nous avons les résultats des vecteur Tf etTfidf, nous pouvons alors remarquer que de la même façon qu'avec le modèle SVM, le Tfidf dépasse le Tf.

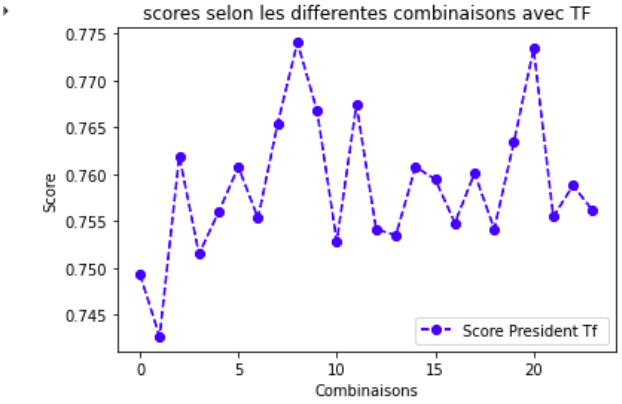
Table 3: Tableau représentant les scores de l'encodage tf pour le modèle Naive Bayes

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tf*	*True*	*True*	*False*	0.749335
1	*True*	*True*	*True*	(1, 2)	*tf*	*True*	*True*	*False*	0.742686
2	*True*	*True*	*True*	(1, 3)	*tf*	*True*	*True*	*False*	0.761968
3	*True*	*True*	*False*	(1, 1)	*tf*	*True*	*True*	*False*	0.751495
4	*True*	*True*	*False*	(1, 2)	*tf*	*True*	*True*	*False*	0.755984
5	*True*	*True*	*False*	(1, 3)	*tf*	*True*	*True*	*False*	0.760797
6	*True*	*False*	*True*	(1, 1)	*tf*	*True*	*True*	*False*	0.755319
7	*True*	*False*	*True*	(1, 2)	*tf*	*True*	*True*	*False*	0.765449
8	*True*	*False*	*True*	(1, 3)	*tf*	*True*	*True*	*False*	0.774086
9	*True*	*False*	*False*	(1, 1)	*tf*	*True*	*True*	*False*	0.766777
10	*True*	*False*	*False*	(1, 2)	*tf*	*True*	*True*	*False*	0.752824
11	*True*	*False*	*False*	(1, 3)	*tf*	*True*	*True*	*False*	0.767442
12	*False*	*True*	*True*	(1, 1)	*tf*	*True*	*True*	*False*	0.754153
13	*False*	*True*	*True*	(1, 2)	*tf*	*True*	*True*	*False*	0.753488
14	*False*	*True*	*True*	(1, 3)	*tf*	*True*	*True*	*False*	0.760797
15	*False*	*True*	*False*	(1, 1)	*tf*	*True*	*True*	*False*	0.759468
16	*False*	*True*	*False*	(1, 2)	*tf*	*True*	*True*	*False*	0.754817
17	*False*	*True*	*False*	(1, 3)	*tf*	*True*	*True*	*False*	0.760133
18	*False*	*False*	*True*	(1, 1)	*tf*	*True*	*True*	*False*	0.754153
19	*False*	*False*	*True*	(1, 2)	*tf*	*True*	*True*	*False*	0.763455
20	*False*	*False*	*True*	(1, 3)	*tf*	*True*	*True*	*False*	0.773422
21	*False*	*False*	*False*	(1, 1)	*tf*	*True*	*True*	*False*	0.755482
22	*False*	*False*	*False*	(1, 2)	*tf*	*True*	*True*	*False*	0.758804
23	*False*	*False*	*False*	(1, 3)	*tf*	*True*	*True*	*False*	0.756146

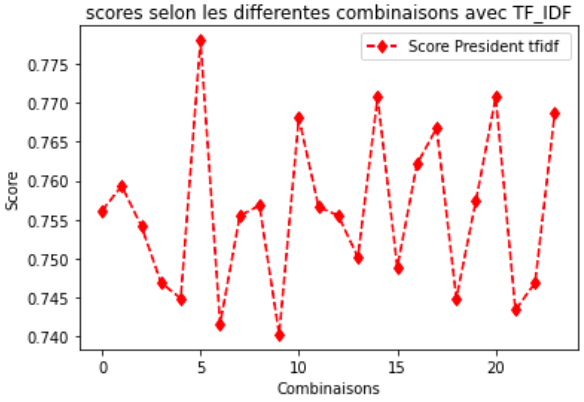
Table 4: Tableau repsentant les scores de l'encodage tfidf pour le modele Naives Bayes

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.756146
1	*True*	*True*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.759309
2	*True*	*True*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.754153
3	*True*	*True*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.746844
4	*True*	*True*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.74485
5	*True*	*True*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.778073
6	*True*	*False*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.741528
7	*True*	*False*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.755482
8	*True*	*False*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.756811
9	*True*	*False*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.740199
10	*True*	*False*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.768106
11	*True*	*False*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.756649
12	*False*	*True*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.755482
13	*False*	*True*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.750166
14	*False*	*True*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.770764
15	*False*	*True*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.748837
16	*False*	*True*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.762126
17	*False*	*True*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.766777
18	*False*	*False*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.74485
19	*False*	*False*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.757475
20	*False*	*False*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.770764
21	*False*	*False*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.743351
22	*False*	*False*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.746844
23	*False*	*False*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.768771

Nous pouvons visualiser le tableau dans la figure ci-dessous :



(a) Nb president avec codge tfidf



(b) NB président avec codage tf

Figure 5: Modèle NB selon les différentes comparaisons du tableau

- comparaison encodage tf et tf idf

  Nous pouvons ainsi mieux voir que, encore une fois la méthode de l'encodage tfidf a généré de meilleurs scores.

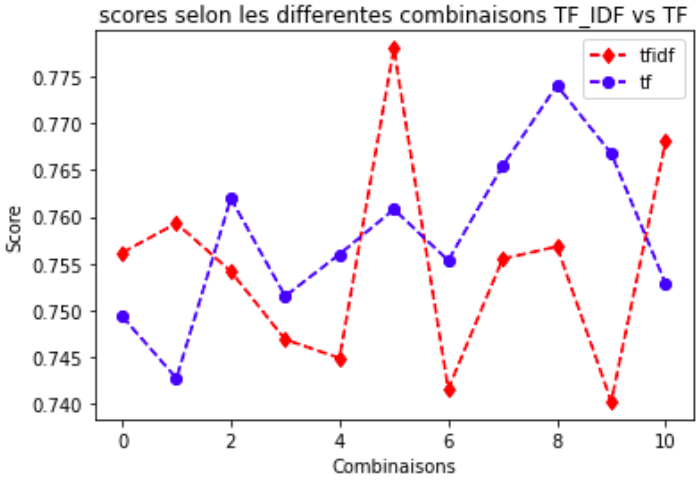


Figure 6: Comparaison des codages tf et tfidf

Régression logistique

Nous avons enfin les tableaux pour le modèle de la régression logistique, nous pouvons voir que cette dernière a de meilleurs performances avec la méthode de l'encodage tfidf.

- avec encodage tf ou tf idf

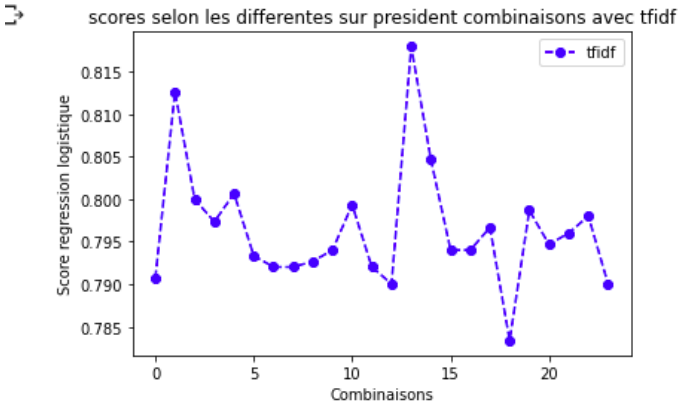
Table 5: Tableau représentant les scores de l'encodage tf pour le modèle de la Régression logistique

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	$True$	$True$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.774086
1	$True$	$True$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.788564
2	$True$	$True$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.791362
3	$True$	$True$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.780731
4	$True$	$True$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.791362
5	$True$	$True$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.788704
6	$True$	$False$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.781395
7	$True$	$False$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.800664
8	$True$	$False$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.801329
9	$True$	$False$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.77992
10	$True$	$False$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.793218
11	$True$	$False$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.801993
12	$False$	$True$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.77992
13	$False$	$True$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.799336
14	$False$	$True$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.795349
15	$False$	$True$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.784053
16	$False$	$True$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.787375
17	$False$	$True$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.784053
18	$False$	$False$	$True$	$(1,1)$	tf	$True$	$True$	$False$	0.794684
19	$False$	$False$	$True$	$(1,2)$	tf	$True$	$True$	$False$	0.796678
20	$False$	$False$	$True$	$(1,3)$	tf	$True$	$True$	$False$	0.79402
21	$False$	$False$	$False$	$(1,1)$	tf	$True$	$True$	$False$	0.779402
22	$False$	$False$	$False$	$(1,2)$	tf	$True$	$True$	$False$	0.803987
23	$False$	$False$	$False$	$(1,3)$	tf	$True$	$True$	$False$	0.802658

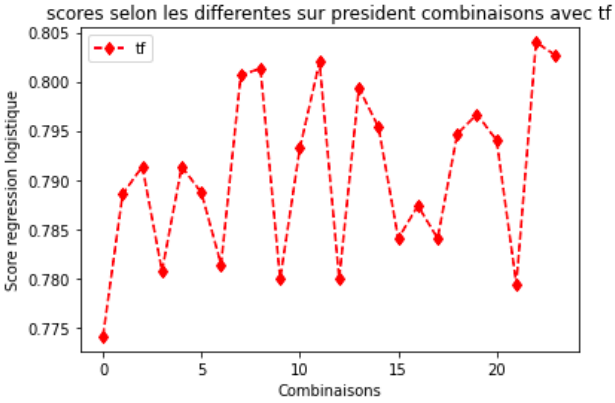
Table 6: Tableau représentant les scores de l'encodage tfidf pour le modèle Régression logistique

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.790698
1	*True*	*True*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.812625
2	*True*	*True*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.8
3	*True*	*True*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.797342
4	*True*	*True*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.800664
5	*True*	*True*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.793355
6	*True*	*False*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.792027
7	*True*	*False*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.792027
8	*True*	*False*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.792691
9	*True*	*False*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.79402
10	*True*	*False*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.799336
11	*True*	*False*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.792027
12	*False*	*True*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.790033
13	*False*	*True*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.81794
14	*False*	*True*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.804651
15	*False*	*True*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.79402
16	*False*	*True*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.79402
17	*False*	*True*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.796678
18	*False*	*False*	*True*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.783389
19	*False*	*False*	*True*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.798671
20	*False*	*False*	*True*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.794684
21	*False*	*False*	*False*	(1, 1)	*tfidf*	*True*	*True*	*False*	0.796013
22	*False*	*False*	*False*	(1, 2)	*tfidf*	*True*	*True*	*False*	0.798007
23	*False*	*False*	*False*	(1, 3)	*tfidf*	*True*	*True*	*False*	0.790033

La figure correspondante est donnée comme suit :



(a) régression logistique président avec codge tfidf



(b) régression logistique avec codge tf

Figure 7: Modèle de la régression logistique selon les différentes combinaisons du tableau

- comparaison encodage tf et tf idf

  Nous pouvons également visualiser le contraste du TF vs TF-IDF:

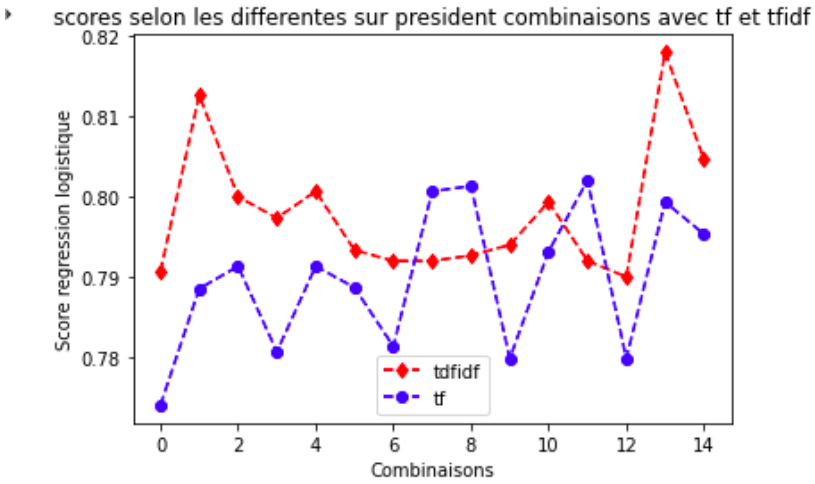


Figure 8: Comparaison des codges tf et tfidf

Aussi, en général,Nous remarquons qu'enlever les stop words et le stemming peut donner de meilleurs performances, et cela grâce a la diminution de la perte de l'information

Comparaison des performances des modèles

Comme vu précédemment, l'encodage tf idf a en moyenne de meilleurs performances que l'encodage tf pour tous les modèles. Nous comparons alors les trois modèles entre eux et nous obtenus la figure ci-dessous.

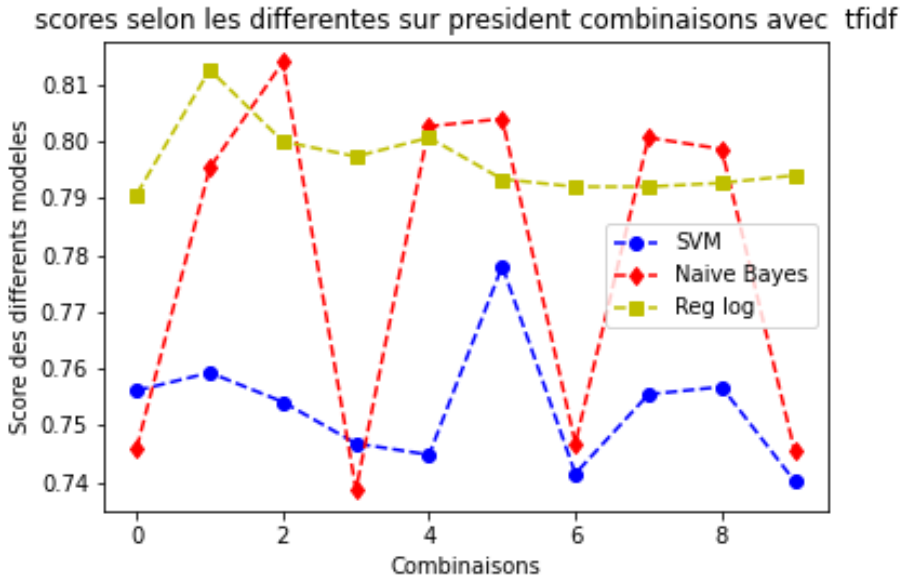


Figure 9: Comparaison des codages tf et tfidf

Nous pouvons alors remarquer que la régression logistique a en général de meilleurs performances que les autres modèles, et permet donc de les classifier d'une meilleur façon.

3.5.2 Dataset Films

En ce qui concerne le dataset des films, nous avons fixer la colonne "balance" a False (le dataset etant deja balancé) et avons fait varier l'utilisation des stopwords , stemming et ngrams pour voir leur impact sur chaque modele.

Nous passons maintenant au Dataset de films, un dataset qui est déjà balancé, et donc qui n'a pas besoin d'avoir la colonne "balance" a True.

- **SVM**
  - avec encodage tf ou tf idf

    Les tableaux ci-dessous montrent les performances des tf-idf vs tf, nous pouvons voir que contrairement au premier dataset, l'encodage tf a de meilleurs performances pour quelques combinaisons, mais

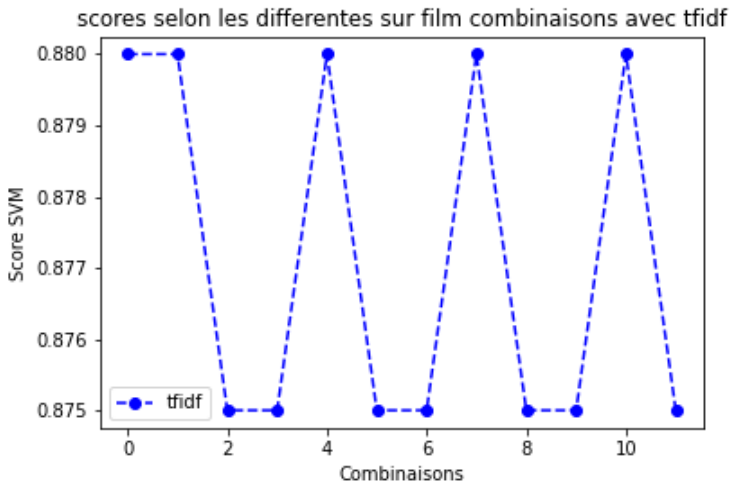
Table 7: Tableau représentant les scores de l'encodage tfidf pour le modèle SVM

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	$True$	$True$	$True$	$(1,1)$	$tfidf$	$True$	$False$	$True$	0.88
1	$True$	$True$	$True$	$(1,2)$	$tfidf$	$True$	$False$	$True$	0.88
2	$True$	$True$	$True$	$(1,3)$	$tfidf$	$True$	$False$	$True$	0.875
3	$True$	$False$	$True$	$(1,1)$	$tfidf$	$True$	$False$	$True$	0.875
4	$True$	$False$	$True$	$(1,2)$	$tfidf$	$True$	$False$	$True$	0.88
5	$True$	$False$	$True$	$(1,3)$	$tfidf$	$True$	$False$	$True$	0.875
6	$False$	$True$	$True$	$(1,1)$	$tfidf$	$True$	$False$	$True$	0.875
7	$False$	$True$	$True$	$(1,2)$	$tfidf$	$True$	$False$	$True$	0.88
8	$False$	$True$	$True$	$(1,3)$	$tfidf$	$True$	$False$	$True$	0.875
9	$False$	$False$	$True$	$(1,1)$	$tfidf$	$True$	$False$	$True$	0.875
10	$False$	$False$	$True$	$(1,2)$	$tfidf$	$True$	$False$	$True$	0.88
11	$False$	$False$	$True$	$(1,3)$	$tfidf$	$True$	$False$	$True$	0.875

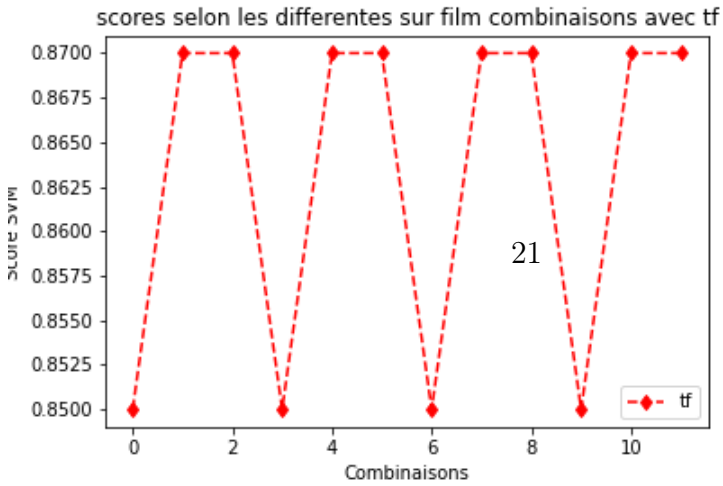
le tfidf l'emporte quand meme dans plusieurs combinaisons de paramètres.

Table 8: Tableau représentant les scores de l'encodage tf pour le modèle SVM

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	$(1,1)$	*tf*	*True*	*False*	*True*	0.85
1	*True*	*True*	*True*	$(1,2)$	*tf*	*True*	*False*	*True*	0.87
2	*True*	*True*	*True*	$(1,3)$	*tf*	*True*	*False*	*True*	0.87
3	*True*	*False*	*True*	$(1,1)$	*tf*	*True*	*False*	*True*	0.85
4	*True*	*False*	*True*	$(1,2)$	*tf*	*True*	*False*	*True*	0.87
5	*True*	*False*	*True*	$(1,3)$	*tf*	*True*	*False*	*True*	0.87
6	*False*	*True*	*True*	$(1,1)$	*tf*	*True*	*False*	*True*	0.85
7	*False*	*True*	*True*	$(1,2)$	*tf*	*True*	*False*	*True*	0.87
8	*False*	*True*	*True*	$(1,3)$	*tf*	*True*	*False*	*True*	0.87
9	*False*	*False*	*True*	$(1,1)$	*tf*	*True*	*False*	*True*	0.85
10	*False*	*False*	*True*	$(1,2)$	*tf*	*True*	*False*	*True*	0.87
11	*False*	*False*	*True*	$(1,3)$	*tf*	*True*	*False*	*True*	0.87



(a) SVM président avec codage tfidf



– comparaison encodage tf et tf idf

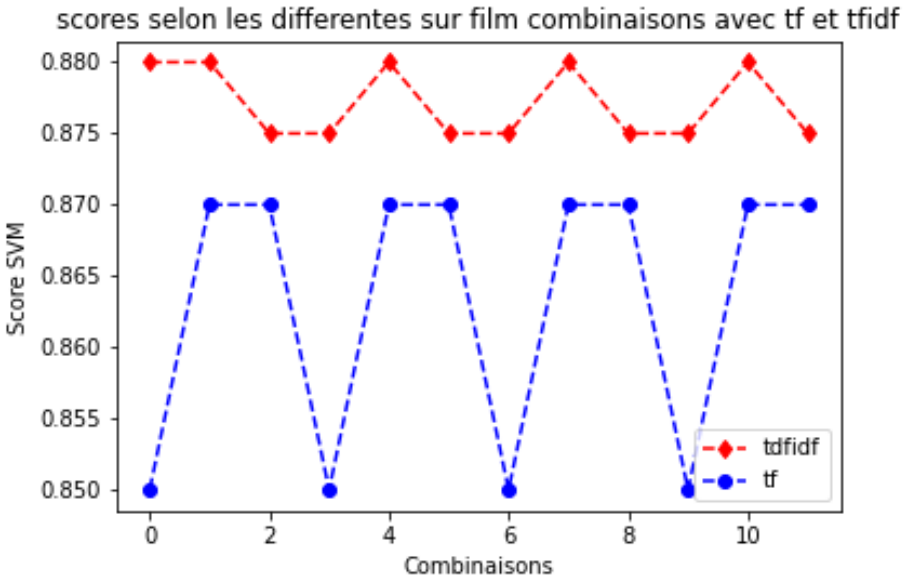
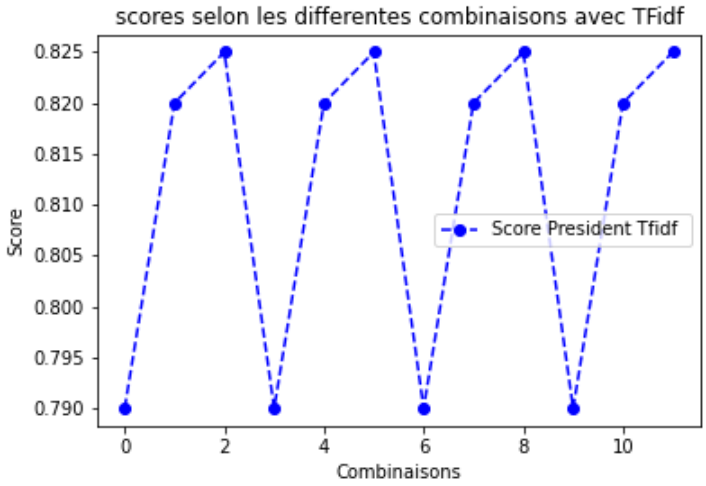


Figure 11: Comparaison des codages tf et tfidf

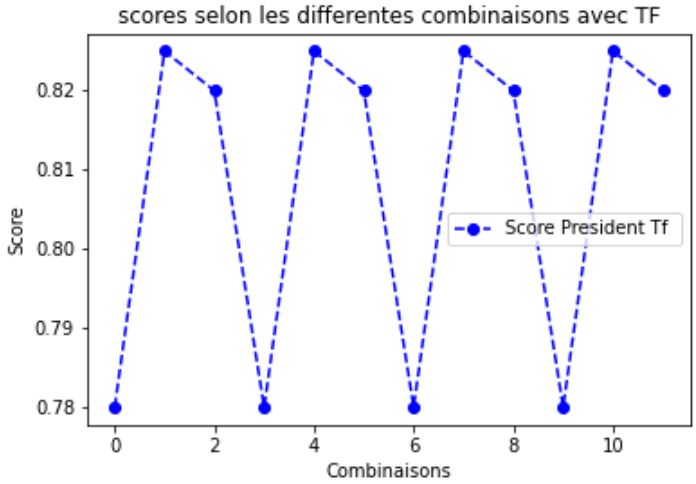
Naives Bayes

- avec encodage tf ou tf idf

  Selon les figures ci-dessous, le modèle Naive Bayes a également de meilleurs performances tf-idf.



(a) Nb film avec codage tfidf



(b) NB film avec codage tf

Figure 12: Modèle NB selon les différentes comparaisons du tableau

- comparaison encodage tf et tf idf

  Les tableaux ci-dessous décrivent les résultats des figures.

Table 9: Tableau représentant les scores de l'encodage tf pour le modèle Naive Bayes

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.78
1	*True*	*True*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.825
2	*True*	*True*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.82
3	*True*	*False*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.78
4	*True*	*False*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.825
5	*True*	*False*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.82
6	*False*	*True*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.78
7	*False*	*True*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.825
8	*False*	*True*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.82
9	*False*	*False*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.78
10	*False*	*False*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.825
11	*False*	*False*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.82

Table 10: Tableau représentant les scores de l'encodage tfidf pour le modèle Naives Bayes

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.79
1	*True*	*True*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.82
2	*True*	*True*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.825
3	*True*	*False*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.79
4	*True*	*False*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.82
5	*True*	*False*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.825
6	*False*	*True*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.79
7	*False*	*True*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.82
8	*False*	*True*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.825
9	*False*	*False*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.79
10	*False*	*False*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.82
11	*False*	*False*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.825

Nous pouvons comparer les performances alors ci-dessous:

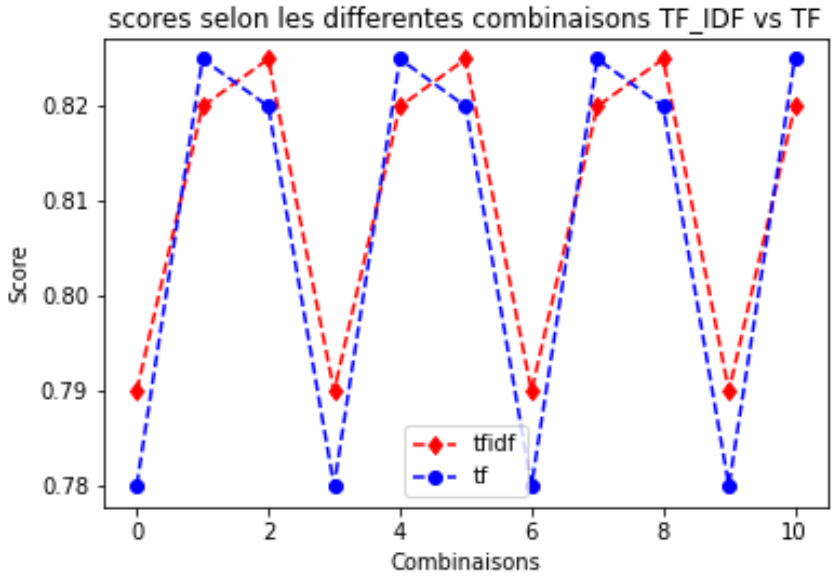


Figure 13: Comparaison des codages tf et tfidf

Nous pouvons voir que les combinaisons sans stop words tendent a donner de meilleur performances.

Régression logistique

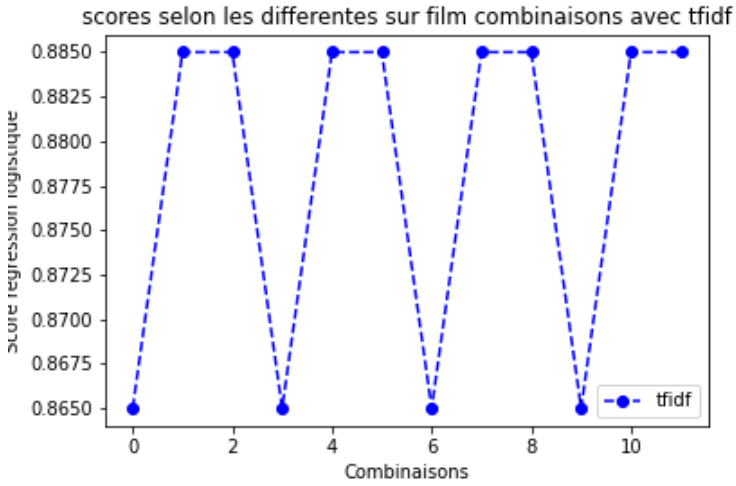
- avec encodage tf ou tf idf
  Enfin, nous avons les tableaux de la régression logistique.

Table 11: Tableau représentant les scores de l'encodage tf pour le modèle de la Répression logistique

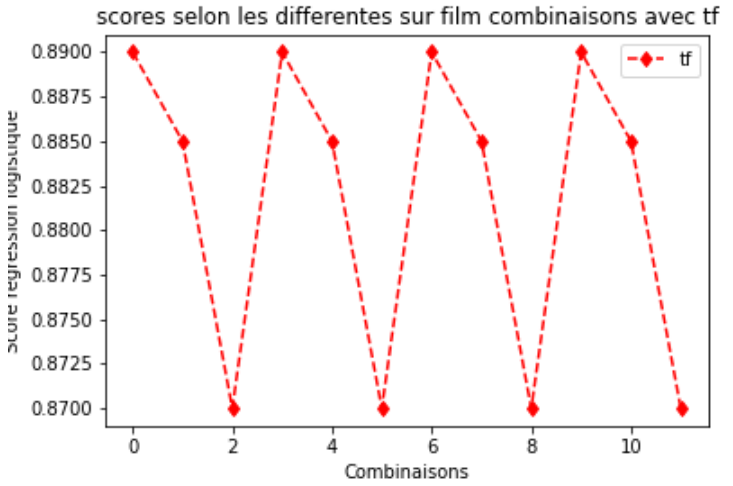
	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.89
1	*True*	*True*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.885
2	*True*	*True*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.87
3	*True*	*False*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.89
4	*True*	*False*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.885
5	*True*	*False*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.87
6	*False*	*True*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.89
7	*False*	*True*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.885
8	*False*	*True*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.87
9	*False*	*False*	*True*	(1, 1)	*tf*	*True*	*False*	*True*	0.89
10	*False*	*False*	*True*	(1, 2)	*tf*	*True*	*False*	*True*	0.885
11	*False*	*False*	*True*	(1, 3)	*tf*	*True*	*False*	*True*	0.87

Table 12: Tableau représentant les scores de l'encodage tfidf pour le modèle de la Régression logistique

	stopword	*stemming*	*miniscule*	*ngram*	*vecteur*	*chiffre*	*balancer*	*punct*	*score*
0	*True*	*True*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.865
1	*True*	*True*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.885
2	*True*	*True*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.885
3	*True*	*False*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.865
4	*True*	*False*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.885
5	*True*	*False*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.885
6	*False*	*True*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.865
7	*False*	*True*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.885
8	*False*	*True*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.885
9	*False*	*False*	*True*	(1, 1)	*tfidf*	*True*	*False*	*True*	0.865
10	*False*	*False*	*True*	(1, 2)	*tfidf*	*True*	*False*	*True*	0.885
11	*False*	*False*	*True*	(1, 3)	*tfidf*	*True*	*False*	*True*	0.885



(a) Modèle de la régression logistique sur film avec codage tfidf



(b) Modèle de la régression logistique sur film avec codage tf

Figure 14: Modèle de la régression logistique sur selon les différentes combinaisons du tableau

- comparaison encodage tf et tf idf

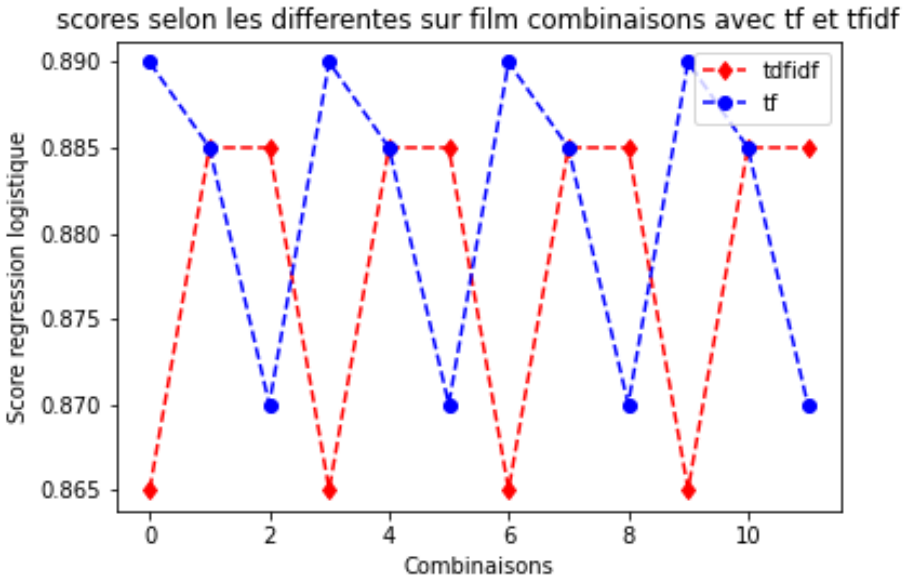


Figure 15: Comparaison des codages tf et tfidf

Nous pouvons voir ainsi, dans la figure et les tableaux, que la régression logistique ici répond mieux a l'encodage tfidf.

Comparaisons des modèles

Comme vu précédemment, l'encodage tfidf a en moyenne de meilleurs performances que l'encodage tf pour tous les modèles. Nous comparons alors les trois modèles entre eux et nous obtenons la figure ci-dessous.

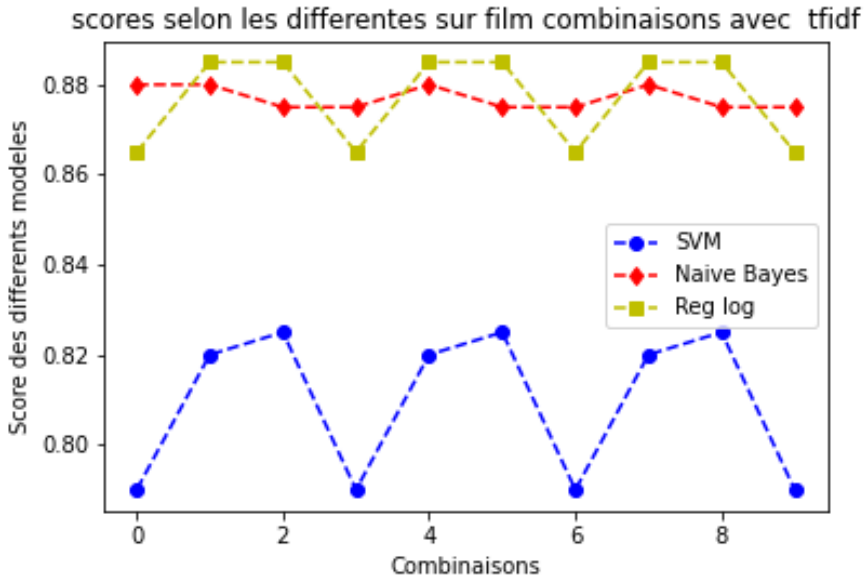


Figure 16: Comparaison des codages tf et tfidf

Nous pouvons alors remarquer que la régression logistique a en général de meilleurs performances que les autres modèles, et permet donc de les classifier d'une meilleur façon.

4 Exploration

4.1 Grid Search

Afin d'avoir un modèle d'apprentissage automatique robuste, il faut sélectionner le bon algorithme d'apprentissage automatique avec la bonne combinaison d'hyperparamètres et de pré-processing comme nous avons pu le voir.

Nous avons fait un premier filtrage et avons fait des études sur les différentes combinaisons, mais, nous pouvons explore le module "*Grid Search* de la bibliothèque python sickit-learn[1] . .

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

Cela nous permet aussi de faire une validation croisée mais qui nous permettra de choisir le meilleure modèle selon les paramétrés (ici ils peuvent être l'utilisation des stop words, la variation des n-grams.. comme vu précédemment).

Avec le gridsearch, on a trouve que la meuilleure modele pour movies et president etait la regression logistique avec les donnees equilibres avec les poids de weighted class avec max df et min df.

4.2 Movies

- c: 2
- tfidf use idf: True
- 'vect ngram range': (1, 1)
- 'vect stop words': None
- accuracy score : 0.7983333333333333

4.3 Presidents

- c: 1
- class weight: 1:9, -1:1
- 'vect ngram range': (1, 2)
- 'vect stop words': None
- 'vect max df': 0.1
- f1 score : 0.55845736

5 Conclusion

Nous avons étudié dans ce travail les performances des algorithmes de machine learning développés pour une tache de classification de texte en variant différents paramètres dans le but de trouver la meilleure méthode de classification,ainsi,nous pouvons conclure que bien que, la partie pré-traitement est

relativement importante, cette dernière peut apporter un important perte d'information , et donc , faire une étude sur son dataset peut améliorer significativement les performances.

6 Bibliographie

[1] Nello Cristianini and Elisa Ricci. 2008. Support Vector Machines. In Encyclopedia of Algorithms, Ming-Yang Kao (ed.). Springer US, Boston, MA, 928–932.

[2] David J. Hand and Keming Yu. 2001. Idiot's Bayes: Not So Stupid after All? International Statistical Review / Revue Internationale de Statistique 69, 3 (2001), 385–398.

[3] Gary King and Langche Zeng. 2001. Logistic Regression in Rare Events Data. Polit. anal. 9, 2 (2001), 137–163.