Cazeres Mathieu — n°22200082 — mathieu.cazeres@etu.umontpellier.fr
Martin-Chantereau Etienne — n°21909526 — etienne.martin-chantereau@etu.umontpellier.fr
Moreaux Victor — n°22200010 — victor.moreaux@etu.umontpellier.fr
Poiret Valentin — n°21609227 — valentin.poiret@etu.umontpellier.fr

Projet - Détection automatique des fake news à partir de données textuelles

HAI817 - Machine learning 1 (méthodes classiques)
2023-04-29

1 Introduction

L'objectif de ce projet est de trouver et d'ajuster des modèles performants qui visent à prédire la véracité d'articles de presse. A chaque texte des jeux de données utilisés est associé une notation déterminant si le contenu de l'article est considéré comme {*VRAI*}, {*FAUX*}, les deux ({*MIXTURE*}) ou autre ({*OTHER*}).

Les données qui nous ont été fournies sont issues de Politifact et Snopes. Ce sont tous deux des sites web de vérification de faits (fact-checking) americains qui vérifient la véracité des allégations de personnages public, discreditant rumeurs ainsi que théories du complot. Ce travail de fact-checking est une composante essentielle du journalisme déontologique.

Compte tenu de la quantité très importante d'articles de presse et de publications (réseaux sociaux) non fact-checkés produites par jour, il peut être intéressant de se pencher sur un algorithme de machine learning pouvant déterminer si un article publié est une fake-news ou non. L'intérêt est de permettre à la population d'avoir accès à de l'information vérifiée pour prendre ses décisions en toute connaissance de cause.

Notre objectif est dans un premier temps de travailler sur un prétraitement visant à nettoyer et normaliser les données pour qu'elles puissent être comprises par des modèles de classification.
Nous nous intéresserons ensuite à trois tâches de classifications où nous allons, pour chacune d'entre elles, tester plusieurs modèles et ajuster celui ou ceux qui donnent les meilleurs résultats. Enfin, nous discuterons de nos résultats en essayant de mon comprendre l'impact des 'features' sur la classification.

2 Ingénierie des données

La phase de prétraitement des données est une étape cruciale en ingénierie de données textuelles. Elle consiste à nettoyer et normaliser les données afin de les préparer pour la classification.

Le nettoyage des données textuelles peut comprendre la suppression des chiffres, de la ponctuation et des stopwords, qui sont des mots sans signification dans le contexte de la classification. Pour normaliser les données, nous pouvons utiliser le stemming ou le lemming. Le stemming consiste à raccourcir les mots de sorte qu'il ne reste que la racine des mots sans conjugaison ni accords. Le lemming quant à lui transforme chaque verbe en son infinitif. L'avantage du stemming sur le lemming est qu'il est plus rapide que le lemming. Par contre, le stemming à tendance à être moins précis que le lemming parce qu'il ne prend pas compte le sens du texte.

Il est également important de s'assurer que la distribution des classes est équilibrée. Concernant nos données, certaines classes sont sous-représentées ({Autre} par exemple) par rapport à d'autres. Dans ce cas, il peut être nécessaire d'utiliser des techniques de rééchantillonnage pour équilibrer les classes et améliorer les performances du modèle. Le downsampling (sous-échantillonnage) et le upsampling (sur-échantillonnage) sont deux techniques de rééchantillonnage qui nous avons utiliser pour s'assurer que nos données initiales soient équilibrées. De plus en utilisant la technique de stratification lors de la division des données en ensembles d'entraînement et de test, nous nous assurons que cet équilibre reste tout le long du processus.

Enfin, nous pouvons combiner le contenu du titre et du texte en une seule colonne pour fournir au modèle plus de contexte et d'informations lors de la classification des données.

Pour conclure, la préparation des données textuelles est une étape essentielle en ingénierie de données pour garantir des résultats précis et fiables dans les tâches de classification.

3 Tâches de classification

Dans cette partie, nous présentons pour chaque tâche de classification les processus de recherche de classification, les résultats des meilleurs modèles ainsi qu'une analyse de l'impact des hyperparamètres et des prétraitements sur les résultats des modèles.

3.1 Tâche de classification {VRAI} vs {FAUX}

La première tâche de classification a pour objectif de classifier tous les textes labélisés {VRAI} ou {FAUX} du jeu de données. Comme exliqué précédement, les tâches de classification ont été testés sur le texte et sur les titres. Le graphique suivant montre que les résultats obtenus sur les textes seulement sont plus satisfaisants.

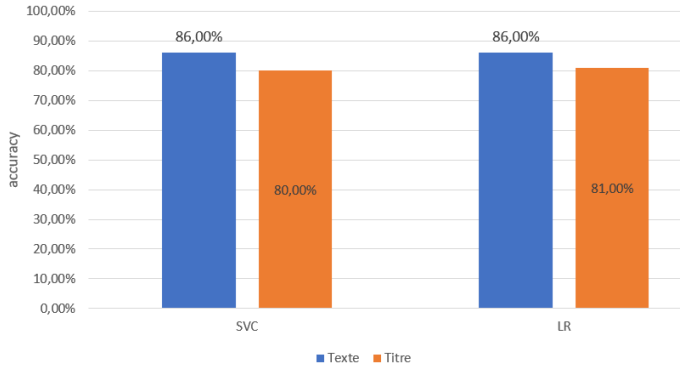


FIGURE 1 – Accuracy de deux classifieurs en fonction des données traitées.

3.1.1 Processus de classification

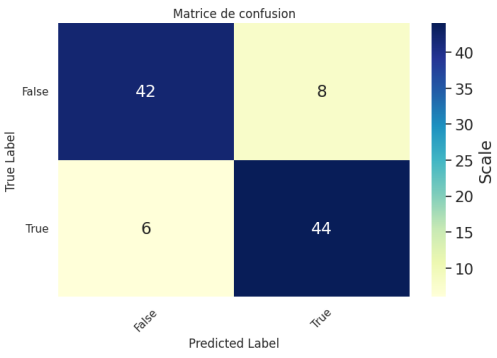
La proportion d'articles labélisés {VRAI} et {FAUX} étant assez grande, nous avons décidé d'utiliser un jeu de données équilibré contenant 500 articles de chaque classes. Ce jeu de données a été créé à l'aide d'une fonction d'équilibrage évoquée plus tôt.
Nous nous sommes focalisés sur les quatre modèles donnant les meilleurs résultats de classification : MultinomialNB (MNB), LogisticRegression (LR), RandomForestClassifier (RF) et SupportVectorClassifier (SVC). Pour chaque modèle, nous avons cherché les meilleurs de pré-traitement et les meilleurs hyper-paramètres du modèle. Nous avons également gardé en mémoire les résultats de tout les paramètrages possible afin de les analyser par la suite.

3.1.2 Résultats de la meilleur pipeline

Le résultat le plus satisfaisant pour cette tâche de classification a été obtenu par le modèle LR. La pipeline du meilleur modèle sauvegardé montre que les données ont été transformée à l'aide du TF-IDF sans retrait des stop words, ni des majuscules. Ce paramétrage n'effectue également aucun pré-traitement sur les textes du jeu de données. Le paramètre C du modèle de classification LogisticRegression a été réglé sur une valeur de 100. A noter que nous avons utilisé 90% du jeu de données pour l'apprentissage, et 10% pour le test du modèle.

Pipeline				
Pretraitement	*removedigit=False, getlemmatisation=False*			
TFIDF	*lowercase=True, stop_words=None*			
LR	*C=100, penalty='l2', solver='newton-cg'*			
	precision	**recall**	**f1-score**	***support***
false	0,875	0,84	0,85714	*50*
true	0,84615	0,88	0,86275	*50*
accuracy			0,86	*100*
macro avg	0,86058	0,86	0,85994	*100*
weighted avg	0,86058	0,86	0,85994	*100*

(a) Rapport de classification



(b) Matrice de confusion

FIGURE 2 – Resultats de classification du modèle LR

D'après le rapport de classification du modèle (figure 2a), le modèle a eu une précision de 86%, ce qui indique que le modèle est assez correct, comme le montre la matrice de confusion (figure 2b). Le f1-score, qui combine les résultats de précision et de recall du modèle, est élevé pour la classification des deux classes, confirmant le bon fonctionnement du modèle.

3.1.3 Impact des paramètres sur la classification

Nous avons ensuite cherché à visualiser l'impact des paramétres du modèle sur la classification. En récupérant les données de paramétrages pour le modèle LR, il est possible de tracer un graphe représentant l'accuracy du modèle en fonction des paramétrages, classés du plus au moins performant (figure 3). Cette représentation graphique permet de visualiser deux ruptures, découpant ainsi l'étude des paramétrages en trois parties. Nous avons également tracé une courbe ROC pour les modèles situés aux extrémités des trois segments (figure 4), permettant de visualiser autrement l'efficacité des modèles.

FIGURE 3 – Visualisation de l'exactitude (accuracy) du modèle LR en fonction des paramètrages possibles.

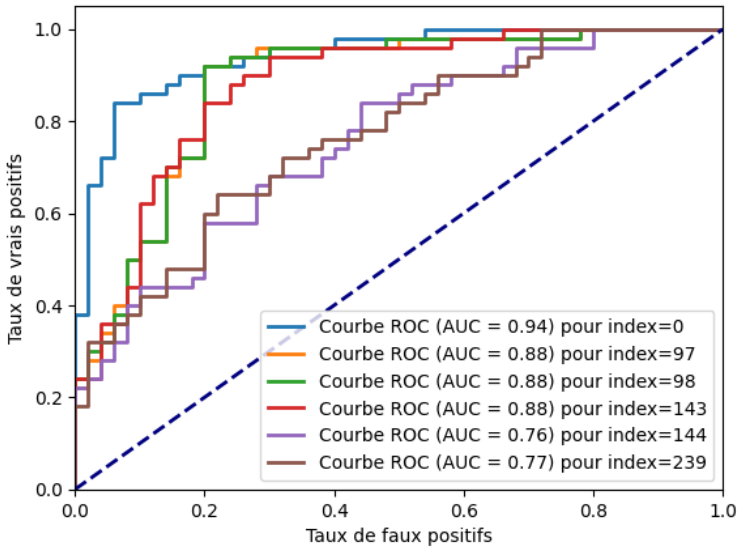
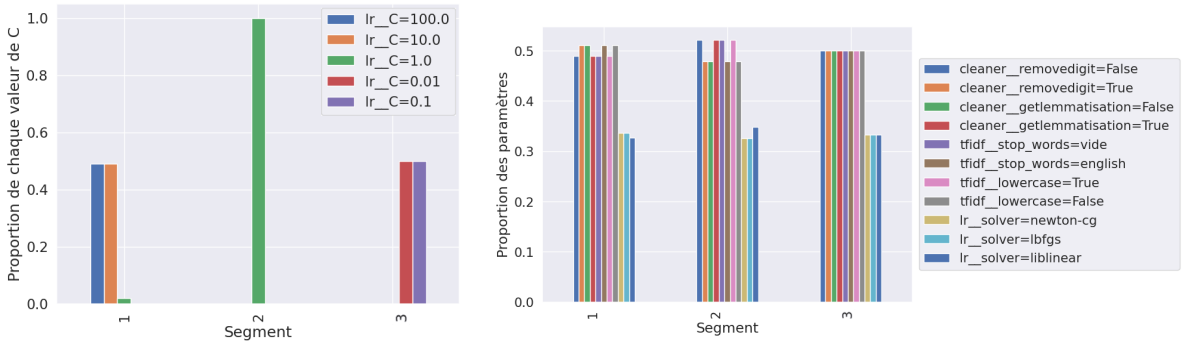


FIGURE 4 – Courbe de ROC pour les modèles situés aux extremes des trois segments

La représentation par segment des proportions des valeurs des différents paramètres du modèle nous a permis de conclure que pour ce cas de classification, le paramètre C du modèle LR était le seul paramètre qui influait sur les résultats de classification. La figure 5a montre que les meilleurs résultats du modèle sont obtenus pour de grandes valeurs de C, et surtout que chaque segment dégagé est spécifique à certaines valeurs de C. La répartition égale des autres paramètres (figure 5b) pour chaque segment montre qu'ils n'ont pas d'impact sur ce cas de classification.



(a) Hyperparamètre C de LR (b) Autres paramètres de LR et paramètres de prétraitement

FIGURE 5 – Représentation pour chaque segment de la proportion des valeurs des paramètres du modèle LR

Il est bon de souligner que les meilleurs résultats obtenus pour cette classification avec le modèle LR utilisent des valeurs de C élevées. Ce paramètre est directement lié à la régularisation du modèle, qui permet de prévenir le surapprentissage. Une valeur de C trop élevée peut donc entraîner un surapprentissage et nuire à la capacité du modèle à généraliser à de nouvelles données.

3.2 Tâche de classification {VRAI/FAUX} vs {AUTRE}

La deuxième tâche de classification que nous avons traitée a pour objectif de classifier les textes labélisés {VRAI} ou {FAUX} et {OTHER}. Les textes {VRAI} ou {FAUX} on était labellisé {TRUE/FALSE}.

3.2.1 Processus de classification

Le processus de recherche de la meilleure classification a été globalement moins bon que la classification précédente {True} vs {False}. Il a fallu regrouper le rating True et False ensemble en True/False, pour que le modèle ne surapprenne pas une des valeurs nous avons pris avons appliqué le calcul suivant pour True et puis pour False : len(Other)/2. Pour {Other} nous avons fait du subsampling de l'ordre de 20%.

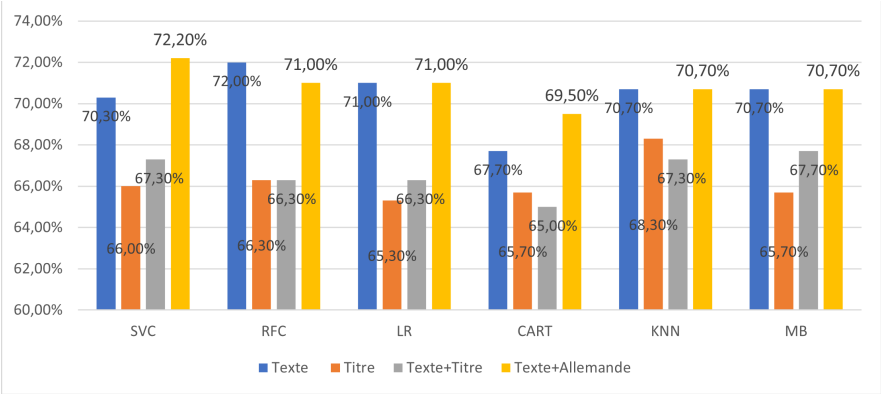


FIGURE 6 – Comparatif des résultats des meilleurs modèles pour le choix du classifieur

Après avoir comparé les 6 modèles et les 3 types de classement possible [Titre, Texte, Titre_Texte_concat], on se rend compte que la classification par le texte donne le meilleur résultat. Cela est dû à la quantité plus importante d'informations qu'ils contiennent par rapport au titre. À contrario, la concaténation vient noyer le titre dans le texte sur qui apporte une faible plus-value. Afin d'améliorer nos modèles nous avons enrichi les données à tester en convertissant les données allemandes du challenge classées other grâce à l'API deepl. Tout en faisant attention de garder les données TRUE_FALSE choisi précédemment lors de l'équilibrage.

À première vue sur le graphique il n'y a pas d'amélioration claire avec le rajout de données. Or Le gain d'un modèle ne sous joue pas uniquement sur le choix. Mais aussi lors de sa phase d'apprentissage où nous gagnons 30% d'accuracy. Cela est dû à l'augmentation des nombres de données sur lequel il peut apprendre. En effet avec données allemandes le nombre total de other étant de 205 nous n'avons pas opéré d'over sampling afin de rendre notre modèle plus pertinent.
Nous allons maintenant comparer les deux résultats du modèle entrainée avec et sans données allemandes.

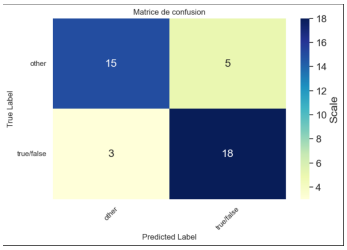
3.2.2 Résultat de la meilleure pipeline

Comme nous l'avons vu plus haut les résultats de la pipeline avec les données allemandes sont dans la globalités proches. Le résultat le plus satisfaisant a été obtenu par le modèle SVM avec les données allemandes. La principale différence entre la pipeline testée sur les textes et les données allemandes 7a et de base 7c est l'utilisation de la stemmatisation. Malgré le fait que deepl soit un bon traducteur et que nos données rajoutés sont traduites de l'allemand, la racination (stemmatisation) a permis aux modèles de mieux classer les mots. Ceci explique pourquoi la Pipeline avec les données allemandes active le paramètre Stemmatisation. Cela met donc en lumière la nécessité d'avoir des données de qualité et de correctement les traiter.

Pipeline	
Pretraitement	removedigit=False, getlemmatisation=True, getstemmer=True
TFIDF	lowercase=True, stop_words=None
SVM	C=10, gamma=0,1, kernel='rfbf', probability=True

	precision	recall	f1-score	support
other	0,833333	0,75	0,78947	20
true/false	0,78261	0,85714	0,81818	21
accuracy			0,80488	41
macro avg	0,80797	0,80357	0,80383	41
weighted avg	0,80797	0,80357	0,80383	41

(a) Rapport de classification allemandes

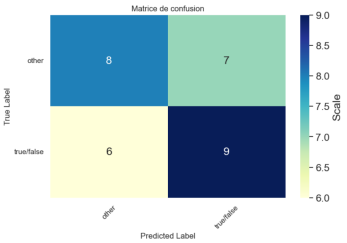


(b) Matrice de confusion allemandes

Pipeline	
Pretraitement	removedigit=True, getlemmatisation=True, getstemmer=False
TFIDF	lowercase=False, stop_words=None
SVM	C=10, gamma=1, kernel='rfbf', probability=True

	precision	recall	f1-score	support
other	0,57143	0,53	0,55172	15
true/false	0,5625	0,6	0,58065	15
accuracy			0,56667	30
macro avg	0,56696	0,56667	0,56618	30
weighted avg	0,56696	0,56667	0,56618	30

(c) Rapport de classification sans ajout

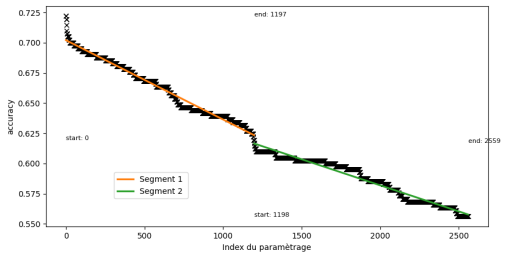


(d) Matrice de confusion sans ajout

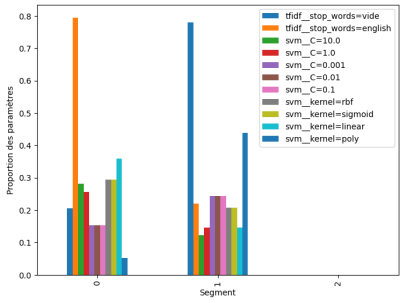
FIGURE 7 – Comparatif résultats de classification du modèle SVM avec et sans ajout des données allemandes

3.2.3 Impact des paramètres sur la classification

La représentation par segment des proportions des valeurs des différents paramètres du modèle nous indique à contrario des autres classification que l'évolution de la qualité du modèle a été assez linéaire.En effet il n'y a eu qu'une rupture avec un faible gain. Donc que seul un ou deux paramètre ont légèrement impacté la classification.



(a) Evolution de l'accuracy



(b) Evolution des paramètres impactant

FIGURE 8 – Evolution des paramètres et accuracy au cours de l'ajustement du modèlesR

Nous pouvons contaster sur la figure qu'en effet c'est l'application des stop words vides, qui a eu un impact significatif sur l'accuracy de la classification. Cela est dû à l'utilisation de données d'entrée traduites de l'allemand, où la traduction peut différer d'un anglais bilingue, ce qui peut affecter la pertinence des stops words. En ne les retirant pas, le modèle a pu mieux capturer les relations entre les mots et améliorer la précision de la classification. De plus le deuxième paramètre impactant est l'application du kernel polynomial, qui permet de transformer les vecteur d'entrée TF-IDF dans un espace de dimension supérieure. Pour capturer des relations non linéaires entre les mots qui ne seraient pas détectées dans l'espace de dimension initiale. Au

regard de la richesse et longueur des textes il évidemment qu'il ressort comme meilleur kernel et impact donc la classification.

3.3 Tâche de classification {VRAI} vs {FAUX} vs {AUTRE} vs {MIXTE}

La troisième tache de classification que nous avons traitée est la tache à quatre classes, qui a pour objectif d'essayer de classifier au mieux les articles entre {VRAI}, {FAUX}, {MIXTE} et {AUTRE}.

3.3.1 Processus de classification

Le processus de recherche de la meilleure classification a été pour cette classification à quatre classes plus complexes et moins bonnes dans sa globalité que dans les classifications précédant. Globalement, l'ensemble des résultats ne sont pas satisfaisants (fig.9). Dans cette figure, la comparaison se fait en fonction des meilleurs résultats obtenus, c'est-à-dire que les paramètres et prétraitement peuvent être différents. Par la suite non étudierons plus précisément l'impact des paramètres sur un des modèles.

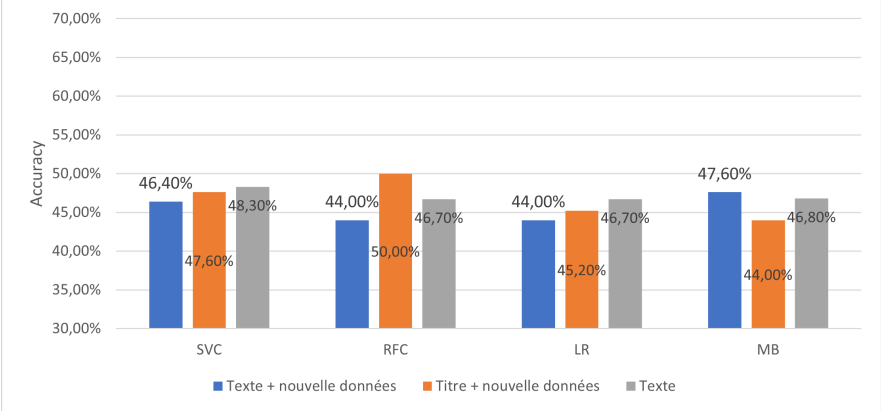


FIGURE 9 – Comparatif des résultats des meilleurs modèles optimisés appliqué sur les textes et sur les titres.

Tout d'abord, le manque de donnée pour le sous-échantillon AUTRE a été un frein dans notre recherche de bonne classification. En effet, peu de données peuvent induire un entraînement insuffisant des modèles (sprint 1). Par la suite, la seconde étape a été d'utiliser un nouveau jeu de donnée traduit de l'allemand en anglais pour obtenir quelques données supplémentaires, les analyses ont été réalisées sur les textes puis les titres (sprint 2 et 3). Enfin, nous avons essayé une dernière classification plus complexe qui se déroule en deux temps : la première classification permet de classifier les textes entre 2 classes {VRAI & FAUX} contre {MIXTE & AUTRE} puis dans un second temps, pour chacune des deux regroupements de classes réaliser une seconde classification ({VRAI} contre {FAUX}) et ({MIXTE} contre {AUTRE}) cette étape donne des résultats assez similaires, en effet la succession des étapes de classification a pour conséquence une accumulation des erreurs (sprint 4).

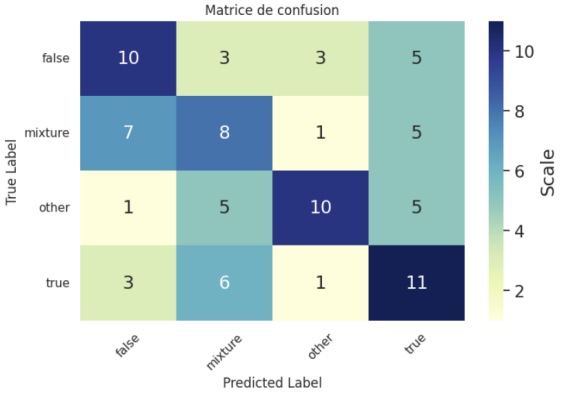
3.3.2 Résultats de la meilleur pipeline

Comme vu précédemment les résultats sont très proche dans l'ensemble, par la suite nous allons nous pencher sur l'un d'entre eux et plus particulièrement sur les resultats du modèle SVM du sprint 2(text + nouvelles données). Ce résultat est loin d'être optimal cependant, l'ensemble des sprints qui ont montré des résultats intéressants sont regroupés dans le notebook.
La pipeline (fig.10a) utilise la méthode TF-IDF pour représenter les textes et le modèle SVM pour la classification. Les paramètres du modèle SVM sont : un paramètre de régularisation C de 10 et un noyau RBF avec un paramètre gamma de 0,1. Les étapes de prétraitement du texte, tel que la suppression des chiffres et la lemmatisation, n'ont pas été appliqué.

Pipeline	
Pretraitement	*removedigit=False, getlemmatisation=False*
TFIDF	*lowercase=False, stop_words=None*
SVM	*C=10, gamma=0,1, kernel='rbf'*

	precision	recall	f1-score	*support*
false	0,47619	0,47619	0,47619	*21*
mixture	0,36364	0,38095	0,37209	*21*
other	0,66667	0,47619	0,55556	*21*
true	0,42308	0,52381	0,46809	*21*
accuracy			0,46429	*84*
macro avg	0,48239	0,46429	0,46429	*84*
weighted avg	0,48239	0,46429	0,46429	*84*

(a) Rapport de classification



(b) Matrice de confusion

FIGURE 10 – Resultats de classification du modèle SVM

Le rapport de classification (fig.10a) révèle que le modèle SVM a obtenu une précision globale de 46,4%, indiquant que le modèle n'a correctement classé qu'environ la moitié des données de test. Les résultats individuels pour chaque classe montrent des performances supérieures pour la classe {AUTRE} et des performances inférieures pour les classes {MIXTE}, VRAI} et {FAUX}. Cela peut suggérer que le modèle a des difficultés à distinguer entre ces trois classes. Dans l'ensemble, bien que la performance globale du modèle soit faible, l'analyse des textes peut montrer qu'il est nécessaire d'utiliser des approches de modélisation plus performantes pour améliorer la précision de la classification (méthodes d'embedding...).

3.3.3 Impact des paramètres sur la classification

L'objectif de cette partie est d'étudier l'impact des paramètres sur les performances du modèle. Le principe de la 'learning curve' en fonction des paramètres (fig.11) est de représenter graphiquement l'évolution de la performance d'un modèle en fonction de la variation de ses paramètres, tout en gardant la quantité de données d'entraînement constante.

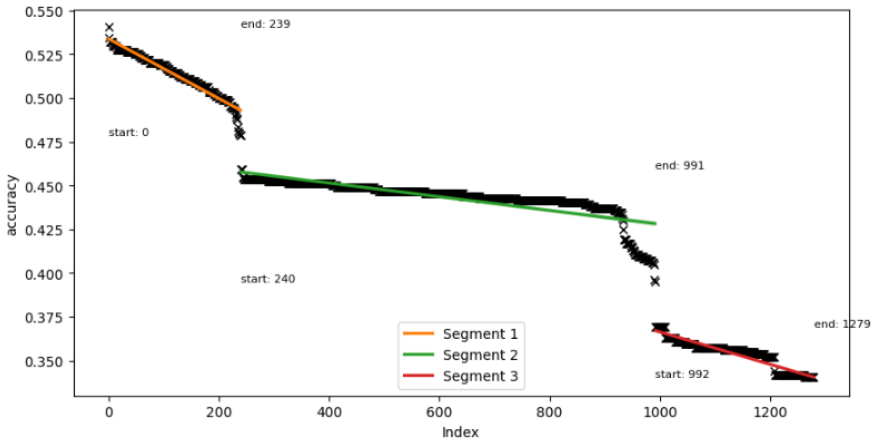


FIGURE 11 – Evolution de 'accuracy' au cours de l'ajustement du modèle.

La figure 10a nous montre la présente de 2 ruptures dans la courbe, nous avons donc mis en avant les segments, par la suite l'objectif a été de visualiser la répartition dans chaque segment des paramètres afin de visualiser les ceux qui ont un fort impact (fig.10a).

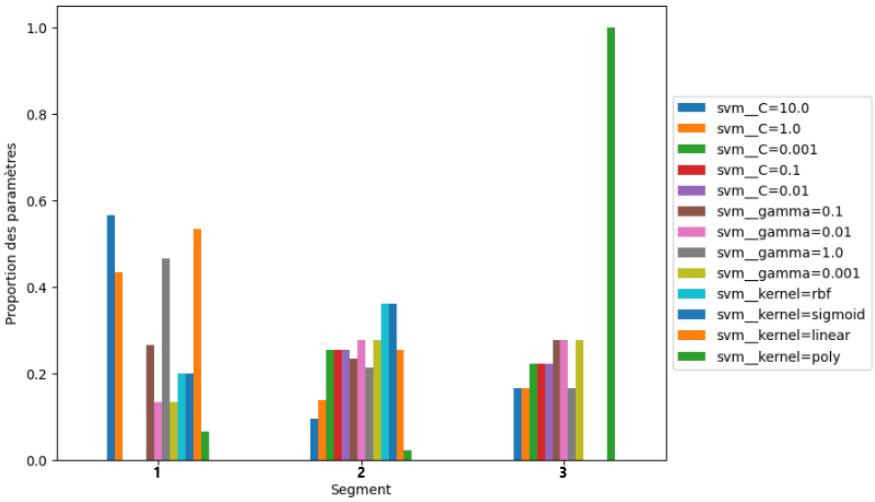


FIGURE 12 – Diagramme en baton qui represente pour chaque segment la proportion de chaque paramètre.

La figure 12 fournit des informations importantes sur l'influence des hyperparamètres sur la qualité de la classification. Nous pouvons constater que les hyperparamètres jouent un rôle majeur dans la qualité de la classification. Le segment 3, qui est le moins performant, est composé à 100% de modèles SVM avec un noyau polynomial, ce qui représente également la quasi-totalité des modèles avec ce noyau. Le segment 1, qui est le plus performant, est composé de modèles SVM avec des valeurs de C=10,1, un noyau linéaire et un gamma=1 majoritairement. Les deux autres segments (2 et 3) ont des répartitions plus homogènes. Dans l'ensemble, il semble que les paramètres C, gamma et kernel aient un impact significatif sur la qualité de la classification.

4 Discussion et conclusion

En général, les hyperparamètres d'un modèle, tels que les paramètres C, gamma et kernel pour SVM, peuvent avoir un impact significatif sur la performance de la classification. D'autres modèles (LR, MNB, RBF etc...) peuvent avoir des hyperparamètres différents. Ces hyperparamètres déterminent la façon dont le modèle est entraîné et comment il prend des décisions de classification, et peuvent donc avoir un impact significatif sur la performance de la classification. En revanche, les paramètres de prétraitement, tels que la lemmatisation ou la suppression des stop words, ont un impact moins important sur la performance des modèles car ils ne modifient pas fondamentalement la structure de l'ensemble de données. Bien que ces paramètres puissent améliorer la performance de la classification dans certains cas, ils ne permettent pas de modéliser des relations plus complexes entre les données.

Enfin, l'utilisation des méthodes de Topic Modeling n'a pas donné des valeurs de cohérence suffisamment élevées, avec des scores se situant entre 0,25 et 0,5 maximum. En conséquence, nous avons décidé de ne pas poursuivre la classification avec cette technique (fig.13). Bien que la modélisation de sujets puisse être utile dans certaines situations, les résultats que nous avons obtenus dans ce cas particulier ont montré qu'elle ne sembait pas utile pour améliorer nos résultats.

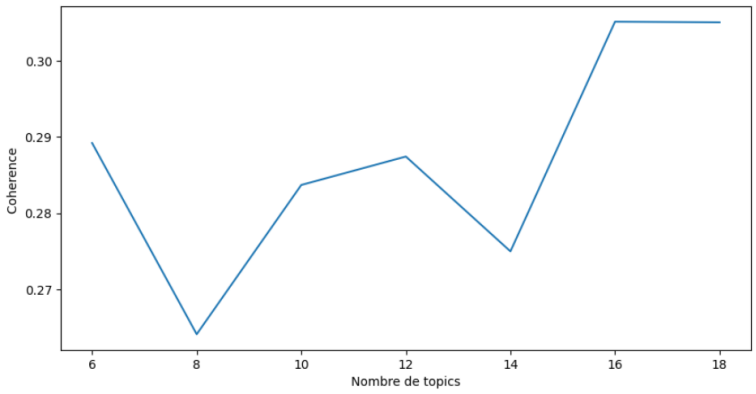


FIGURE 13 – Evolution de la 'coherence' en fonction du nombres de thèmes.