TP2

Apprentissage machine

Etienne Collin
20237904

Angela Dabilgou
20171112

Roman Gilles-Lesage
20175122

Intelligence artificielle: Introduction - IFT3335

Section A
Professeur Jian Yun Nie

Université de Montréal
Hiver 2024
À remettre le 27 Avril 2024 à 23:59

Table des matières

1. Introduction & Objectifs

Dans le cadre du cours d'introduction à l'intelligence artificielle, nous avons travaillé sur une multitude d'algorithmes d'apprentissage supervisé et non-supervisé. Ce rapport porte sur la comparaison de quelques de ces techniques lorsqu'appliquées à la tâche suivante: analyser une banque de tweets et déterminer automatiquement si chaque tweet est offensant ou non.

> **Note.** Le document a 8 pages de contenu, mais environ l'équivalent de 4.5-5 pages de texte sans les graphiques.

1.1. Méthodologie

À l'aide d'un script python, la collecte des résultats a été automatisée. C'est-à-dire, ont été générés automatiquement des résultats permettant de quantifier l'impact de techniques de prétraitement des données, ainsi que de comparer numériquement les performances de différents modèles d'apprentissage.

Voici les techniques de prétraitement des données qui ont été comparées:

- Standardisation des données ou non (`strip()` et convertion en minuscules)
- Filtrage des stopwords et des mots non alpha-numériques ou non
- Troncature des mots ou non
  - Pour la troncature: Lemmatization vs Stemming

Voici les modèles comparés. Pour chacun de ces modèles, plusieurs hyperparamètres ont été testés afin comparer leurs effets sur les performances des modèles.

- Naïve Bayes
- Arbre de décision
- Forêt aléatoire
- SVM
- Multi-Layer Perceptron
- Bert & DistilBert

Pour chaque modèle, les résultats des techniques de vectorisation suivantes ont été comparées:

- CountVectorizer
- TF-IDF Vectorizer

L'entrainement est effectué sur 70% des données, et le test sur les 30% restants.

2. Analyse exploratoire des données

Le corpus de données utilisé pour ce projet est un ensemble de 13240 tweets. Ces derniers sont annotés selon plusieurs catégories, dont « offensant » et « non offensant ». Ces 13240 tweets, environ 33% (4400) sont classés « offensant » et environ 66% (8840) sont classés « non-offensant ». Voici un exemple de tweet de chaque catégorie:

Offensant `@USER @USER Go home you're drunk!!! @USER #MAGA #Trump2020 :fist: :usa_flag: :fist: URL`
Non-offensant `@USER Buy more icecream!!!`

On remarque des « @USER » dans les tweets, qui sont des mentions d'utilisateurs et « URL » qui sont des liens. Ces éléments ne sont pas pertinents pour l'apprentissage et seront donc retirés lors du prétraitement des données. Les tags et emojis peuvent potentiellement être utiles pour déterminer si un tweet est offensant ou non.

3. Prétraitement des données

Les techniques de prétraitement des données sont essentielles pour nettoyer et transformer les données brutes en une forme plus adaptée à l'apprentissage automatique. Les techniques utilisées sont simples et efficaces pour améliorer les performances:

Standardisation des données Suppression des caractères/termes spéciaux et mise en minuscule pour réduire la complexité des données et améliorer la cohérence. Ces caractéstiques éliminées ne sont pas très utiles pour la classification d'un tweet comme étant offensant ou non.

Filtrage des stopwords et des mots non alpha-numériques Les stopwords sont des mots couramment utilisés qui n'apportent pas de valeur informative pour la classification. Les mots non alpha-numériques sont également supprimés car ils peuvent introduire du bruit dans les données.

Troncature des mots La troncature des mots est une technique qui consiste à réduire les mots à leur forme de base. Deux techniques de troncature ont été comparées: la lemmatisation et le stemming. La lemmatisation est une technique plus

avancée qui utilise des règles linguistiques pour réduire les mots à leur forme de base, tandis que le stemming est une technique plus simple qui utilise des heuristiques pour réduire les mots à leur racine.

- Exemple de lemmatisation: « running » → « run », « ran » → « run »
- Exemple de stemming: « running » → « run », « ran » → « ran »

Des résultats ont été générés pour chaque combinaison de ces techniques de prétraitement:

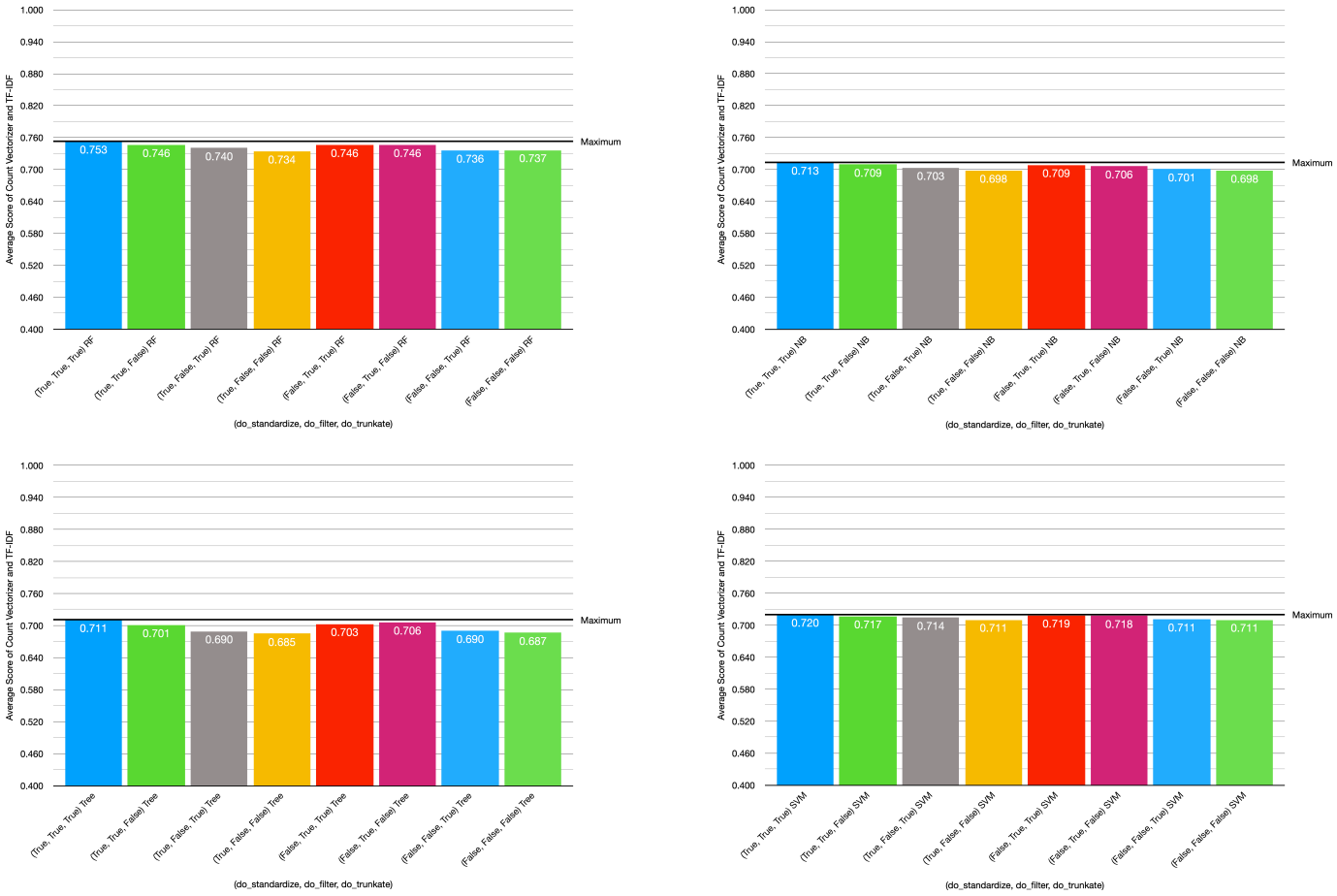

Fig. 1. – De gauche à droite et de haut en bas: comparaison des performances des différentes combinaisons de techniques de prétraitement pour les modèles de forêt aléatoire, de Naïve Bayes, d'arbre de décision et de SVM.

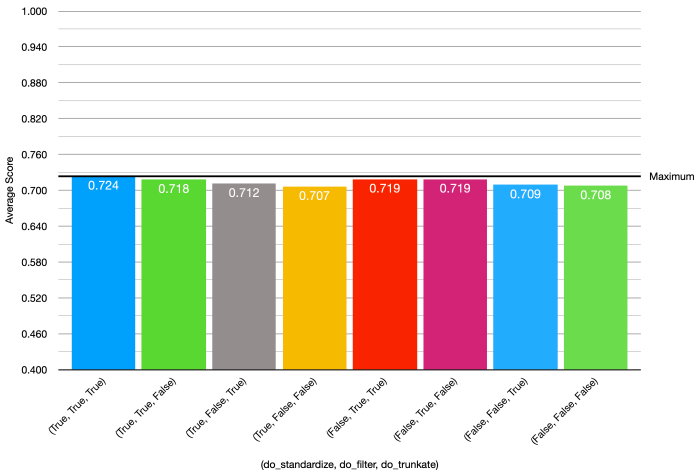

Fig. 2. – Comparaison des performances moyennes des différentes combinaisons de techniques de prétraitement peu importe les modèles d'apprentissage.

Selon ces graphiques, on peut voir que la présence d'une technique de filtrage des données non alpha-numériques et de stopwords est ce qui améliore les performances du prétraitement. La standardisation des données (trimming et minusculisation) semble aussi avoir un impact sur les performances, mais ce n'est pas significatif. Dans otus les cas, le pre-traitement optimal semble utiliser toutes les techniques de pré-traitement implémentées. Cela est logique puisqu'on souhaite distiller les données et standardiser leur format pour ne garder que l'information essentielle à la classification. Le travail des algorithmes d'apprentissage n'en est que facilité.

Prenant cela en compte, on présentera les résultats des différents algorithmes avec l'utilisation de la lemmatisation et du stemming pour la troncature des mots. L'utilisation de CountVectorizer et TF-IDF pour la vectorisation des données sera également comparée.

4. Modèles d'apprentissage

4.1. Naïve Bayes

Le modèle Naïve Bayes est un algorithme d'apprentissage automatique probabiliste basé sur l'application du théorème de Bayes avec une hypothèse d'indépendance forte (et naïve) entre les données. Il estime les probabilités conditionnelles pour chaque classe et calcule ensuite la probabilité a posteriori pour les données d'entrée nouvelles, l'affectant à la classe la plus probable.

4.2. Arbre de décision

Le modèle d'arbre de décision est un algorithme d'apprentissage automatique qui utilise une structure en arbre pour faire des prédictions ou classer des données en fonction de variables d'entrée. Il fonctionne en divisant les données en sous-ensembles plus petits à l'aide d'une série de décisions ou de branches, chacune représentant un test sur une ou plusieurs variables d'entrée. Ce processus se poursuit jusqu'à ce qu'un point d'arrêt optimal soit atteint et ensuite la décision finale est prise en fonction des valeurs attribuées aux feuilles.

Pour décider de l'optimalité d'une scission, on utilise généralement l'indice de Gini ou l'entropie. L'indice de Gini mesure la diversité des échantillons dans un noeud, et plus cet indice est faible, mieux c'est. L'entropie quantifie le degré d'imprévisibilité d'un ensemble de données, et un ensemble avec une entropie faible est considéré comme étant très prévisible. Ainsi, lorsque nous divisons les données en deux groupes distincts, nous cherchons à minimiser l'indice de Gini ou l'entropie pour chaque groupe, ce qui nous aide à choisir la scission optimale.

4.3. Forêt aléatoire

Le modèle de forêt aléatoire représente un ensemble d'arbres de décision appris individuellement à partir des exemples d'entraînement. Ces arbres sont utilisés pour prédire la classe ou la valeur cible d'un exemple de test en prenant le vote majoritaire des prédictions de chaque arbre dans l'ensemble. L'avantage de ce modèle est qu'il résiste bien aux données bruyantes et au surapprentissage, puisque les différents arbres sont moins corrélés entre eux.

4.4. SVM

Les modèles SVM (Support Vector Machine) sont des algorithmes d'apprentissage machine qui fonctionnent en cherchant un hyperplan optimal pour séparer les différentes classes de données. Le but est d'optimiser la marge entre ces deux groupes, ce qui permet un modèle plus robuste aux données aberrantes et une meilleure généralisation sur des données inconnues.

Les noyaux (kernels) sont utilisés dans les SVM pour transformer des données non linéairement séparables en un espace de dimension supérieure où elles peuvent être séparées par un hyperplan linéaire. Généralement, on utilise le noyau polynomial (polynomial) et le noyau RBF (Radial Basis Function). Le noyau linéaire est moins flexible que le noyau RBF, mais il est plus rapide à entraîner et nécessite généralement moins de données. Le noyau RBF est plus complexe et offre une meilleure performance sur des ensembles de données complexes, mais son temps d'entraînement est généralement plus long et nécessite plus de données.

4.5. Multi-Layer Perceptron

Les modèles de Réseaux de Neurones Multicouches (Multi-Layer Perceptrons en anglais) sont un type spécifique de réseau de neurones très basique.

Dans un MLP, il y a généralement au moins trois types de couches interconnectées: une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Leur but est d'apprendre et de modéliser des relations complexes entre les données d'entrée et de sortie. Pour ce faire, ces modèles utilisent une technique de rétropropagation pour ajuster la pondération des connexions entre les neurones dans chaque couche, en minimisant l'écart entre les prédictions du MLP et les valeurs cibles souhaitées.

4.6. Bert & DistilBert

Bert et DistilBert sont deux modèles de traitement automatique de NLP basés sur des transformateurs. BERT, introduit en 2018, est un modèle de deep learning pré-entraîné sur une grande quantité de données qui utilise des techniques de self-supervised learning pour apprendre comment interpréter le contexte et le sens des mots dans une phrase.

DistilBERT, développé en 2019, est une version plus petite de Bert qui tente de réduire la taille du modèle sans compromettre sa performance. DistilBERT est donc moins gourmand en ressources informatiques et a un temps de réponse plus rapide.

Il sera intéressant de comparer les performances de ces deux modèles sur notre corpus de tweets et de voir si la version plus petite de BERT peut rivaliser avec son homologue plus grand.

4.7. Résultats & Comparaison des modèles

Dans cette section, les résultats obtenus avec les différents algorithmes d'apprentissage utilisés seront discutés.

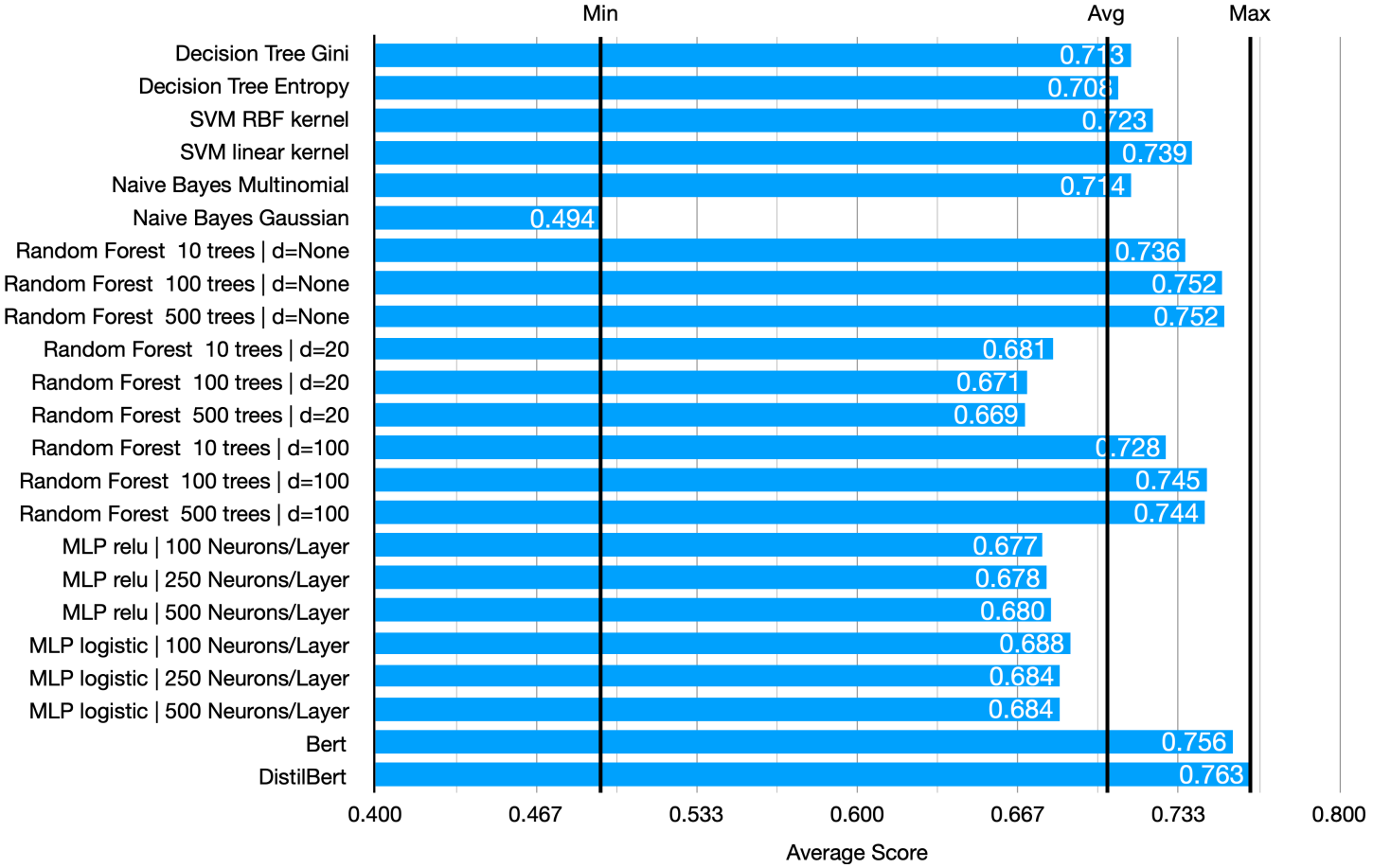

Fig. 3. – Comparaison des performances moyennes des différents algorithmes d'apprentissage. Peu importe les techniques de prétraitement et de vectorisation des données.

4.8. Naives Bayes

Deux variantes du classificateur Naïves Bayes ont été testées ici: Naïves Bayes multinomial et Naïves Bayes gaussien. En moyenne, le Naïves Bayes multinomial a produit un score d'exactitude de 71.4%, tandis que le Naïves Bayes gaussien obtient un score notablement plus bas de 49.4%.

Le Naïves Bayes multinomial est bien adapté aux données textuelles qui sont représentées sous forme de comptage de mots ou de fréquences, ce qui est le cas dans notre problème de classification de texte. En exploitant ces informations sur les fréquences des mots, le Naïves Bayes multinomial a su généraliser et classifier efficacement les données. En revanche, le Naïves Bayes gaussien est conçu pour des données continues et suppose que les caractéristiques suivent une distribution gaussienne. Dans ce contexte de classification de texte oú les données sont représentées de manière discrète, l'application du Naïves Bayes gaussien entraîne une perte de performance significative.

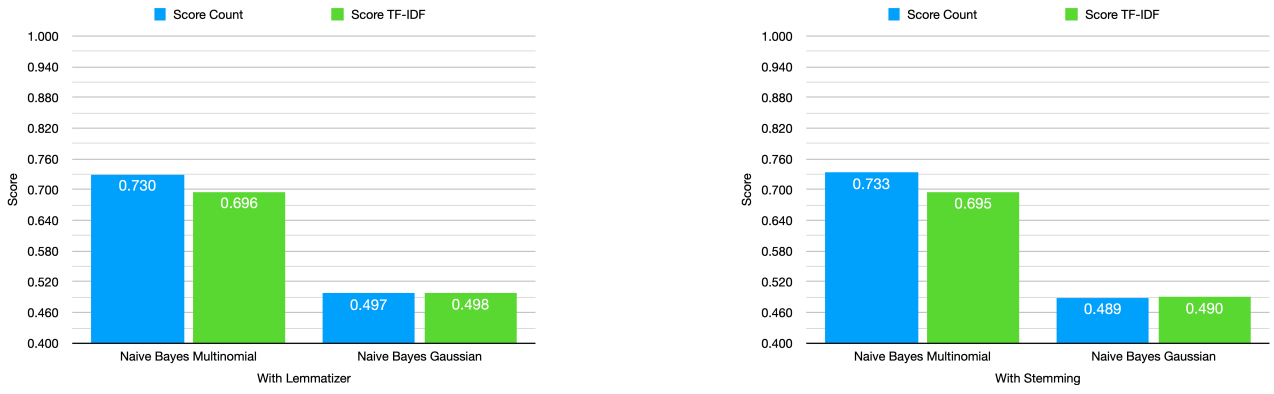

Fig. 4. – Comparaison des performances des Naïves Bayes avec différentes configurations d'architecture.

4.9. Arbre de décision

Nos expérimentations ont révélé des performances comparables entre deux variantes de l'algorithme d'arbre de décision: l'arbre de décision avec le critère de gini et celui avec le critère d'entropie. En moyenne, l'arbre de décision avec le critère de Gini a produit un score d'exactitude de 71.3%, tandis que l'arbre de décision avec le critère d'entropie a obtenu un score légèrement inférieur de 70.8%.

Bien que le critère de Gini ait légèrement surpassé le critère d'entropie en termes de performance, la différence entre les deux n'est pas significative. Ces résultats suggèrent que les deux critères peuvent être efficacement utilisés pour construire des arbres de décision dans le contexte de classification de texte. La similitude des performances entre les deux critères peut être attribuée à la nature du problème de classification de texte et à la distribution des données. Dans certains cas, le critère de Gini peut être plus adapté lorsque les classes sont bien séparées et les données moins sujettes au bruit, tandis que le critère d'entropie peut mieux fonctionner dans des situations où les classes sont plus mélangées et les décisions sont moins évidentes.

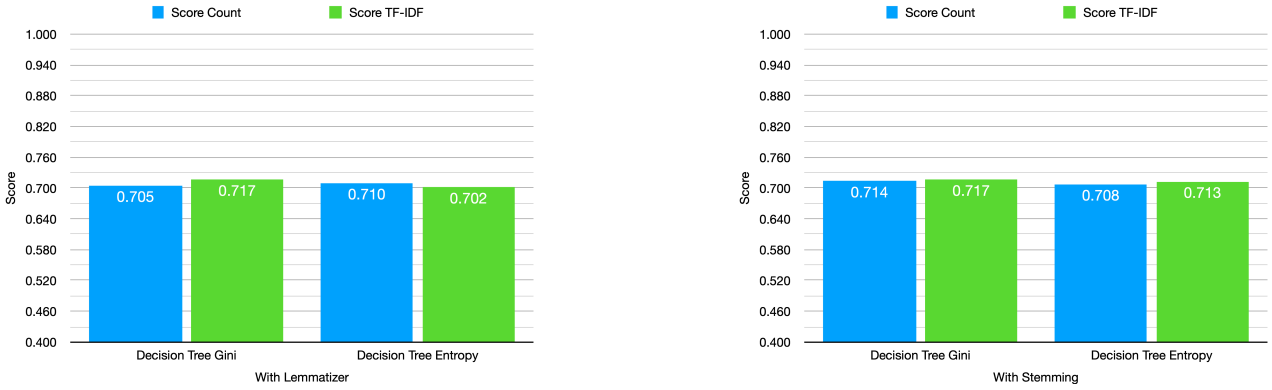

Fig. 5. – Comparaison des performances des arbres de décision avec différentes configurations d'architecture.

4.10. Forêt aléatoire

Dans nos expériences avec la Forêt Aléatoire, nous avons observés des performances variables en fonction des paramètres utilisés. Voici un résumé des observations:

- Augmenter le nombre d'arbres n'a pas toujours un impact significatif sur les performances. Un fois un certain seuil atteint, les performances semblent ne pas s'améliorer significativement.
  - Avec 100 arbres et une profondeur illimitée, un score moyen de 75.2% est obtenu. Cependant, augmenter le nombre d'arbres à 500 n'améliore pas le score et ralentis considérablement l'entrainement.
- Permettre à l'algorithme d'explorer les arbres en profondeur semble être plus bénéfique que de limiter la profondeur des arbres.
  - Peu importe le nombre d'arbres, la profondeur illimitée semble généralement produire de meilleurs résultats. Et ce, même si la limite de profondeur est relativement élevée.

Il est important de noter que le meilleur score obtenu pour des algorithmes de classification classiques de 75.2% a été atteint avec la Forêt Aléatoire pour une utilisation de 100 arbres et en spécifiant la profondeur maximale illimitée. Ce résultat démontre la capacité de la forêt aléatoire à produire des résultats significatifs dans la classification de texte, soulignant son efficacité dans la détection de langage offensif dans les tweets.

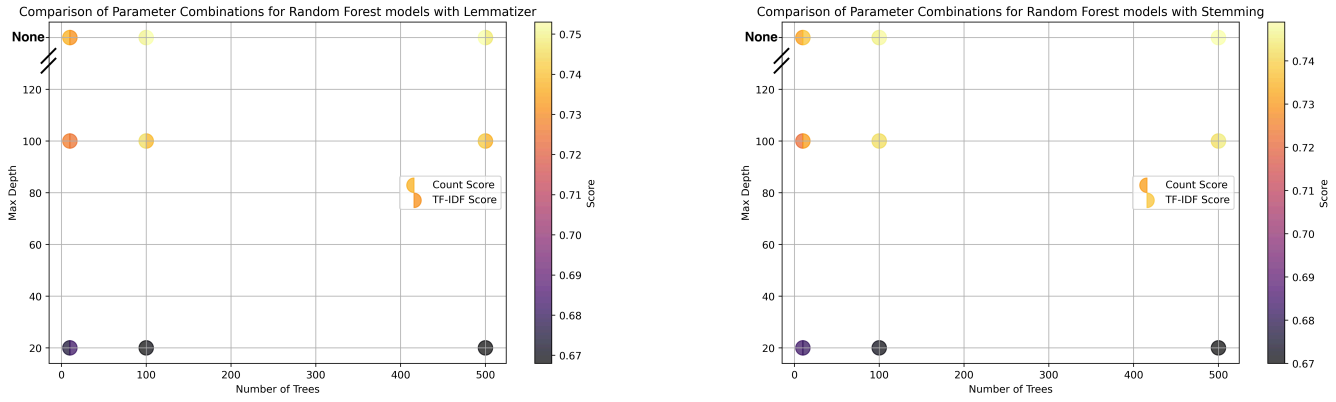

Fig. 6. – Comparaison des performances des forêts aléatoires avec différentes configurations d'architecture.

4.11. SVM

Nous obtenons globalement de bons résultats avec SVM. Nous avons testé ici SVM linéaire et SVM avec noyau RBF. En moyenne, le SVM linéaire a produit un score d'exactitude de 73.9%, tandis que le SVM avec noyau rbf a obtenu un score légèrement inférieur de de 72.3%. Le SVM linéaire fonctionne lorsque les données sont linéairement séparables dans l'espace de caractéristiques, ce qui semble être le cas pour cette tâche de classification de textes.

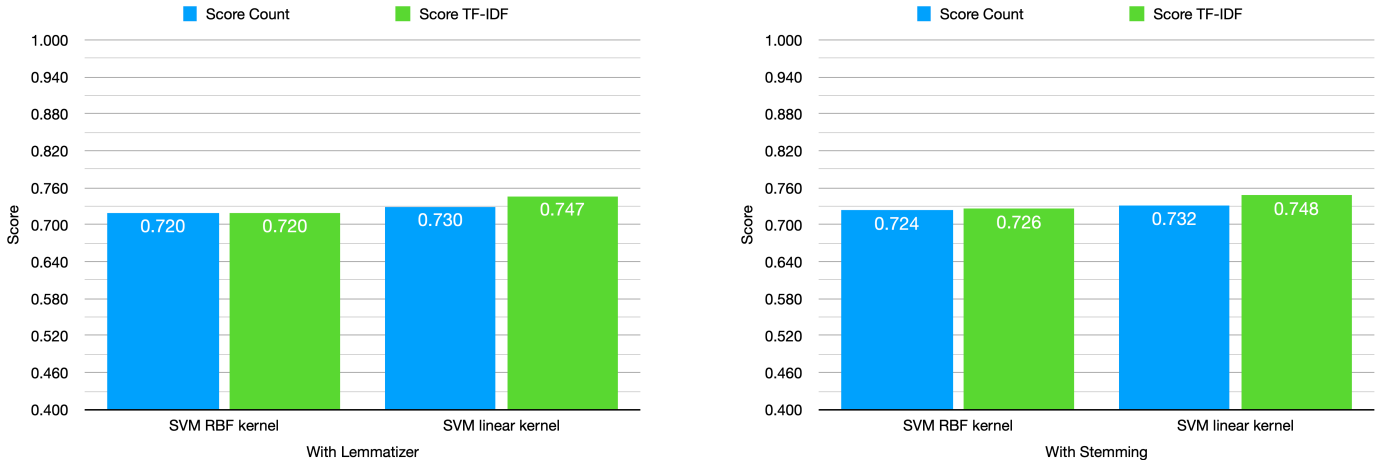

Fig. 7. – Comparaison des performances des SVM avec différentes configurations d'architecture.

4.12. Perceptron

Nos expérimentations avec le Perceptron Multicouche (MLP) ont révélé des résultats intéressants, mettant en lumière l'impact subtil de diverses configurations d'architecture sur les performances du modèle. En moyenne, l'utilisation de la fonction d'activation ReLU avec différentes tailles de couches a produit des scores d'exactitude relativement proches, variant de 67.7% à 68%. De même, l'utilisation de la fonction d'activation logistique avec une taille de couche de 100 neurones a également produit un score comparable de 68.8%. Ces résultats indiquent une certaine stabilité des performances du MLP dans notre contexte, avec des variations relativement modestes malgré les changements dans la taille des couches et le type de fonction d'activation. Cependant, il est intéressant de noter que la fonction d'activation logistique semble légèrement surpasser ReLU dans notre cas spécifique.

L'observation selon laquelle l'augmentation de la taille des couches n'entraîne pas nécessairement une amélioration significative des performances peut être attribuée à plusieurs facteurs. Tout d'abord, l'ajout de neurones supplémentaires peut augmenter la complexité du modèle, mais cela peut également entraîner un surajustement aux données d'entraînement et une généralisation moindre aux données de test. De plus, la nature de la tâche de classification de texte peut limiter les avantages potentiels de l'augmentation de la taille des couches, car les informations pertinentes peuvent être capturées de manière efficace même avec des architectures plus simples. Enfin, les différences de performances entre ReLU et la fonction logistique peuvent être attribuées à leurs propriétés différentes. ReLU est connue pour sa capacité à atténuer le problème de disparition du gradient et à accélérer l'apprentissage, tandis que la fonction logistique peut mieux gérer les valeurs extrêmes et les non-linéarités dans les données. Selon les scores obtenus et considérant le temps élevé nécessaire à l'entraînement comparativement aux autres méthodes d'apprentissage classique, les MLP ne semblent pas en valoir la chandelle. Il serait intéressant de tester les effets sur le score de l'augmentation du nombre de couches cachées.

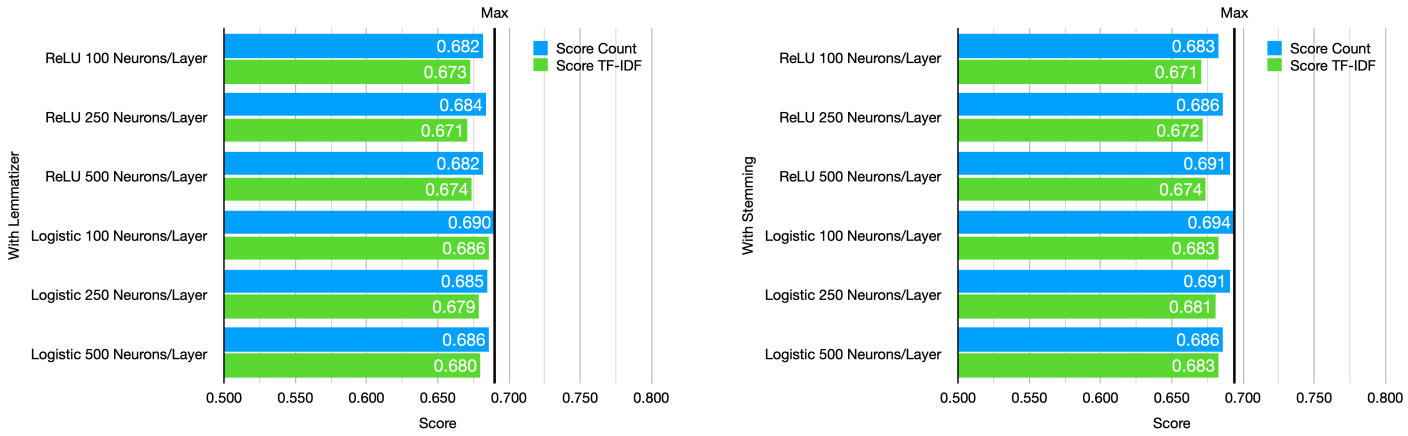

Fig. 8. – Comparaison des performances des MLP avec différentes configurations d'architecture.

4.13. Bert

En moyenne, le modèle BERT a produit un score d'exactitude de 75.6%, tandis que le modèle DistilBert a surpassé ses performances avec un score légèrement supérieur de 76.3%. Ces résultats mettent en évidence l'efficacité des modèles basés sur les transformers pour la classification de texte, en particulier dans le contexte de détection de langage offensif dans les tweets. La différence de performance entre BERT et DistilBERT peut être attribuée aux différences dans la complexité et la taille des modèles. BERT est une architecture de transformer complète, tandis que DistilBERT est une version plus légère et compressée de BERT, conçue pour être plus rapide et plus efficace en termes de mémoire. Malgré sa taille réduite, DistilBERT a conservé une grande partie de la capacité de représentation de BERT, ce qui lui permet d'atteindre des performances comparables voire supérieures dans certaines tâches. En outre, la légèreté de DistilBERT lui permet d'être plus facilement déployé dans des environnement avec des contraintes de ressources, ce qui en fait un choix plus attractif pour les applications pratiques de traitement automatique du langage naturel.

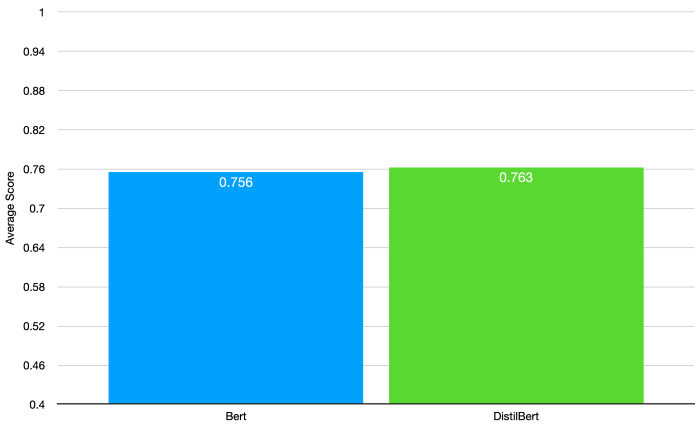

Fig. 9. – Comparaison des performances des modèles BERT et DistilBERT.

4.14. CountVectorizer vs. TF-IDF

À la lumière des résultats, on compare l'usage de la vectorisation par CountVectorizer et TF-IDF pour les algorithmes classiques. Parmis tous les tests effectués, la différence en pourcentage moyenne entre le score obtenu par CountVectorizer et celui obtenu par TF-IDF est de 0.648%. C'est à dire que CountVectorizer a des résultats 0.648% plus élevés que TF-IDF en moyenne. Ce résultat est constant parmi les différents algorithmes d'apprentissage et les multiples tests effectués.

À ne pas confondre avec la différence entre le score moyen produit par ces deux techniques.

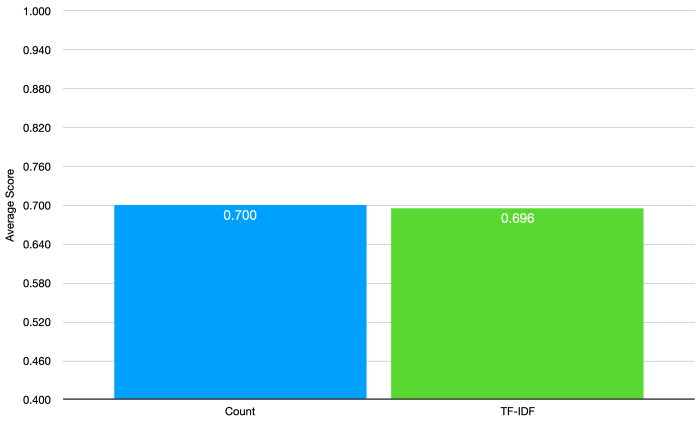

Fig. 10. – Comparaison des performances moyennes des algorithmes classiques utilisant CountVectorizer vs. TF-IDF.

4.15. Stemming vs. Lemmatization

À la lumière des résultats, on compare également l'usage du stemming versus de la lemmatization pour la troncation des mots. Parmis tous les tests effectués, la différence en pourcentage moyenne entre le score obtenu par le Lemmatizer et celui obtenu par le Stemming est de -0.263%. C'est à dire que le Stemming a des résultats 0.263% plus élevés que le Lemmatizer en moyenne. Ce résultat est constant parmi les différents algorithmes d'apprentissage et les multiples tests effectués.

À ne pas confondre avec la différence entre le score moyen produit par ces deux techniques.

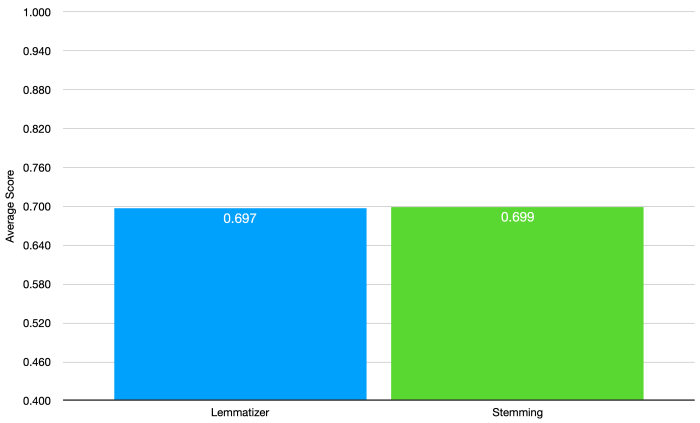

Fig. 11. – Comparaison des performances moyennes des algorithmes classiques utilisant Stemming vs. Lemmatization.

5. Conclusion

Dans le cadre de ce travail pratique, différentes approches pour la classification de texte appliquée à la détection de contenu offensif ont été explorées. À travers les tests, certaines techniques d'apprentissage machine sortent du lot et semblent prometteuses: l'algorithme de forêt aléatoire et le modèle DistilBERT. L'algorithme de forêt aléatoire a émergé comme le meilleur performeur parmi les modèles classiques, avec un score d'exactitude de 75.2%. Cette approche a démontré sa capacité à traiter efficacement des données textuelles complexes et à généraliser avec succès à de nouveaux tweets.

D'autre part, l'utilisation du modèle DistilBERT, une version plus légère et plus rapide de BERT, a également produit des résultats remarquables, avec un score d'exactitude d'en moyenne 76.3%. Ce résultat dépasse surprenament le score moyen de BERT de 75.6%; comme quoi un modèle plus gros et complexe n'est pas toujours avantageux. En combinant des méthodes traditionnelles d'apprentissage automatique avec des approches basées sur l'apprentissage profond, notre étude a démontré l'efficacité de différentes techniques pour la classification de texte. Ces résultats soulignent l'importance de l'exploration de diverses approches et de l'adaptation des méthodes en fonction des besoins spécifiques du problème.

Ceci étant dit, il reste encore des possibilités d'amélioration et des avenues de recherches futures à explorer. Parmi celles-ci, on peut citer l'optimisation des hyper-paramètres des modèles, l'exploration de techniques de prétraitement plus avancés, et l'intégration de données supplémentaires pour enrichir la capacité de généralisation du modèle. Il serait également intéressant de tester des techniques de *data augmentation* afin de balancer le corpus de données qui est présentement composé à 33% de tweets offensants et 66% de tweets non-offensants afin de potentiellement améliorer le score des algorithmes.

6. Fonctionnement du code

1. Ouvrir le jupyternotebook/colab `ift3335_tp2.ipynb`
2. Exécuter les sections suivantes:
   - « Setup »
   - « Training Functions »
3. Ouvrir la section « Running »
   - Exécuter la première cellule de code
   - Dans la deuxième cellule, choisir le modèle/la technique d'apprentissage, le chemin vers le corpus de données et la « batch size » (seulement utilisée pour Bert et DistilBert)
   - Exécuter la deuxième cellule de code