Rapport du compétition Kaggle

Sidya Galakho, Trinh Ngo
IFT3395

November 14, 2024

Introduction

Dans ce projet, nous participons à une compétition Kaggle axée sur la classification de textes. Le défi consiste à développer un algorithme capable de classer automatiquement de courts documents textuels en deux catégories prédéfinies. Les données fournies incluent des vecteurs de comptage de mots, représentant la fréquence des termes dans chaque document, ainsi qu'une carte de vocabulaire associée. Pour résoudre ce problème, nous avons prétraité les données avec plusieurs techniques et expérimenté différents modèles d'apprentissage automatique. Ces modèles ont été évalués à l'aide du macro F1 score, nous permettant ainsi d'identifier celui offrant les meilleures performances.

Conception des fonctionnalités

Nous avons d'abord supprimé les stop words tels que "you", "then", "them", etc., qui, bien qu'apparaissant fréquemment dans les textes, n'apportent pas d'information significative et n'aident pas à la compréhension du contenu. En les éliminant, nous avons pu nous concentrer sur des termes plus pertinents, susceptibles de contribuer davantage à l'interprétation du document. Ensuite, nous avons appliqué la lemmatisation pour réduire des variantes de mots comme "changing" et "changed" à leur forme de base. Cette technique permet de regrouper les mots appartenant au même champ lexical, en réduisant les variations et en améliorant l'uniformité du texte. Ainsi, elle permet de traiter de manière cohérente les termes similaires, évitant ainsi de traiter différentes formes d'un même mot comme des entités distinctes. Ces deux approches combinées nous ont permis de réduire le nombre de mots pris en compte dans notre analyse, ce qui a diminué la dimensionnalité des données, simplifié le modèle et réduit le bruit.
Enfin, nous avons utilisé la méthode TF-IDF (Term Frequency-Inverse Document Frequency) pour transformer le texte en vecteurs numériques, permettant ainsi de capturer l'importance relative des termes dans le corpus. Cette méthode nous aide à pondérer les mots fréquents mais peu informatifs et à donner plus de poids aux termes rares mais significatifs.

Algorithme

Dans ce projet, nous avons utilisé plusieurs algorithmes de classification pour résoudre la tâche de classification de textes. Le classifieur Bayes naif multinomial a été choisi pour sa simplicité et son efficacité, en particulier lorsqu'il s'agit de traiter des ensembles de données 1ui constent des comptages/fréauences.
Nous avons également utilisé le classifieur Bayes naif complémentaire, une variante qui gère mieux les déséquilibres de classes en ajustant les probabilités des termes rares et fréquents.
En plus de ces modèles basés sur Bayes, nous avons expérimenté avec des modèles plus complexes tels que le classifieur à vecteurs de support (SVM) linéaire, qui est efficace pour les tâches de classification dans des espaces de caractéristiques de grande dimension, et la régression logistique, qui est un modèle linéaire couramment utilisé pour la classification binaire.
Tous ces modèles ont été évalués en fonction de leur performance sur notre ensemble de données, en utilisant des métriques comme le score F1 pour identifier celui qui offrait les meilleurs résultats.

Méthodologie

Dans un premier temps, nous avons divisé les données d'entraînement, en allouant 80% à l'apprentissage des paramètres du modèle et 20% à son évaluation. Par la suite, nous avons réalisé un dernier entraînement en combinant l'ensemble d'entraînement et l'ensemble d'évaluation. Nous avons utilisé une graine de 42 pour garantir la reproductibilité des résultats.
Pour les modèles Bayes naif multinomial et Bayes naif complémentaire, nous avons employé le lissage de Laplace comme hyperparamètre, en testant des valeurs comprises entre 0 et 2 avec un pas de 0,1.
Concernant le SVM linéaire, nous avons opté pour la fonction de perte "hinge loss" avec une régularisation L2, en faisant varier la constante de régularisation dans l'intervalle [1, 10], et un taux d'apprentissage fixé à 0,0001.
Pour l'algorithme du perceptron, nous avons utilisé la même fonction de perte, la même régularisation et le même taux d'apprentissage, en réalisant une validation croisée pour les mêmes valeurs de la constante de régularisation.
Enfin, pour la régression logistique, nous avons appliqué la même approche, cette fois-ci en utilisant l'entropie croisée comme fonction de perte.

Résultat

Le graphique de la figure 1 montre l'évolution du score F1 en fonction de la valeur du paramètre alpha pour le modèle Bayes naïf complémentaire. On observe que lorsque la valeur de alpha est faible, le modèle a tendance à bien s'adapter aux données d'entraînement, avec un score F1 élevé (environ 0,86), mais il montre des signes de surapprentissage car le score de validation est nettement inférieur (environ 0,7). À mesure que l'alpha augmente, le score F1 d'entraînement diminue régulièrement, indiquant que le modèle applique un lissage plus

important, ce qui réduit sa capacité à bien s'ajuster aux données. Le score de validation suit une tendance similaire, se détériorant plus rapidement après un alpha d'environ .5, ce qui montre que le modèle commence à sous-apprendre (underfitting). La meilleure performance semble se situer autour d'un alpha entre 0 et .5, où l'équilibre entre surapprentissage et sous-apprentissage est optimal.
En regardant les figures 2 et 3, nous observons que le complément naïve Bayes est beaucoup plus sensible à la variation de l'hyper paramètre comparé aux deux autres modèles. Toutefois, avec un bon choix de la valeur de l'hyper paramètre le complément naïve Bayes donne un score F1 plus élevé que les deux autres. Ce qui est illustré dans les figures 4, 5 et 6.
Sur Kaggle, complément naïve Bayes donne un score F1 de .72404 alors que les deux autres donnent respectivement .68145 et .67849.

Discussion

L'approche utilisée présente plusieurs avantages, notamment le prétraitement efficace des données textuelles avec l'élimination des stop words et la lemmatisation, ce qui réduit la dimensionnalité et améliore la pertinence des termes. L'utilisation de la méthode TF-IDF permet de mieux pondérer les mots en fonction de leur importance. Les modèles choisis, comme le Bayes Naive Complémentaire, le SVM linéaire et la régression logistique, offrent une diversité d'approches adaptées aux données textuelles. Cependant, les modèles linéaires tels que le SVM et la régression logistique peuvent être limités lorsqu'il s'agit de capturer des relations non linéaires, ce qui pourrait réduire leur performance dans des cas plus complexes. Pour améliorer cette approche, l'intégration de modèles non linéaires, comme les arbres de décision ou les réseaux neuronaux, pourrait offrir des performances plus robustes sur des données plus complexes.

Référence

Naive Bayes Classifier: `https://en.m.wikipedia.org/wiki/Naive_Bayes_classifier`
TF-IDF: `https://fr.m.wikipedia.org/wiki/TF-IDF`

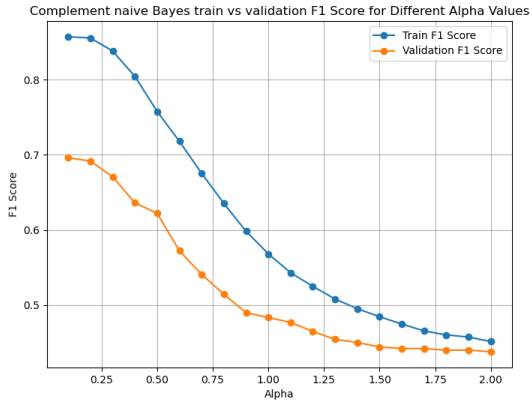

Figure 1: Macro Score F1 vs Alpha pour Bayes naîf multinomial

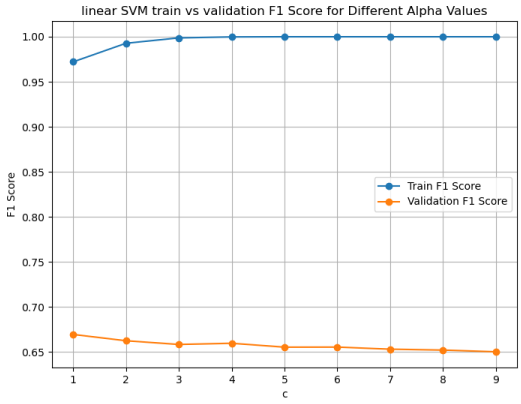

Figure 2: Macro Score F1 vs C pour SVM linéaire

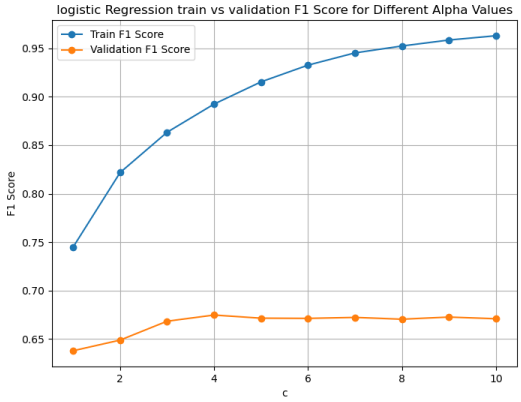

Figure 3: Macro Score F1 vs C pour la régression logistique

	precision	recall	f1-score	support
0	0.88	0.76	0.82	1416
1	0.49	0.70	0.57	469
accuracy			0.74	1885
macro avg	0.69	0.73	0.70	1885
weighted avg	0.79	0.74	0.76	1885

Figure 4: Rapport de classification pour complément naîve Bayes.

	precision	recall	f1-score	support
0	0.82	0.93	0.87	1416
1	0.64	0.38	0.48	469
accuracy			0.79	1885
macro avg	0.73	0.66	0.67	1885
weighted avg	0.78	0.79	0.77	1885

Figure 5: Rapport de classification pour régression logistique.

	precision	recall	f1-score	support
0	0.82	0.89	0.86	1416
1	0.57	0.42	0.48	469
accuracy			0.78	1885
macro avg	0.70	0.66	0.67	1885
weighted avg	0.76	0.78	0.76	1885

Figure 6: Rapport de classification pour SVM linéaire.