Rapport de projet Kaggle - pour IFT3395

Cédric Kamdem
Mandi Vigier
Nom de l'équipe: PTY

12 novembre 2024

Cours : IFT3395 - Apprentissage Automatique
Titre du projet : Classification de Textes

Table des matières

1 Introduction

Ce projet vise à développer un algorithme d'apprentissage automatique pour classifier des documents textuels dans deux catégories prédéfinies, en s'appuyant sur des matrices de comptage de termes. Les défis incluent la gestion des données déséquilibrées, la réduction du bruit et l'optimisation des modèles pour assurer une généralisation efficace.
Notre pipeline comprend le prétraitement des données, la création de caractéristiques enrichies (ex. : scores de sentiment, nombre de mots distincts) et l'évaluation des modèles via le macro F1-score. Parmi les modèles testés, le Naïve Bayes multinomial a obtenu les meilleures performances avec un macro F1-score de 0.7491 après optimisation, surpassant la régression logistique, les SVM et les forêts aléatoires.
Ce rapport détaille notre méthodologie, les choix techniques et les résultats obtenus, tout en proposant des pistes d'amélioration pour des travaux futurs.

2 Conception des fonctionnalités

Pour optimiser le traitement des données textuelles, nous avons appliqué une **lemmatisation**, ce qui a permis de réduire le vocabulaire de 26,354 mots à 23,693 tout en préservant le sens des mots. Cette réduction a diminué la complexité computationnelle tout en maintenant des informations pertinentes pour la classification. Ensuite, nous avons filtré les mots apparaissant dans moins de trois documents, réduisant ainsi le vocabulaire à 8,014 mots, dont seulement 1,322 exclusifs à une classe, ce qui a considérablement réduit le bruit.
Malgré leur fréquence élevée, les **stopwords** ont été conservés. Ces derniers semblent refléter des variations stylistiques importantes, telles qu'un ton formel, informel ou scientifique, qui pourraient être discriminants pour les modèles. Pour valider leur importance, ainsi que celle des autres termes, nous avons utilisé le test **Mann-Whitney U**, adapté aux distributions asymétriques, pour identifier les caractéristiques explicatives les plus significatives.
Par ailleurs, nous avons créé de nouvelles caractéristiques :

- **Nombre de chiffres par texte** : Après avoir constaté une asymétrie dans leur distribution, nous avons appliqué un test Mann-Whitney U (p-value = 0.21), ce qui n'a pas validé cette caractéristique comme pertinente pour la classification.
- **Sentiment** : À l'aide de la bibliothèque `TextBlob`, nous avons attribué des scores de sentiment (polarité et subjectivité) aux textes. Les distributions étant normales, un test t (p-value = 0.0025) a confirmé la significativité de cette caractéristique. Les scores ont été ajustés en ajoutant 1.5, afin de rendre les valeurs positives et compatibles avec certains modèles.
- **Nombre de mots** : Cette caractéristique capture des variations liées à la longueur des textes, pertinentes pour différencier des documents de styles variés (scientifiques, promotionnels, etc.). Un test t (p-value = 0.016) a confirmé son utilité pour la classification.
- **Nombre de mots distincts** : Cette caractéristique, liée à la richesse lexicale, a montré une différence hautement significative entre les classes (p-value = 0.0009), validant ainsi son inclusion.

Ces transformations et sélections ont permis d'enrichir les données tout en réduisant le bruit, optimisant ainsi la qualité des informations utilisées pour la classification.

3 Algorithmes

Pour résoudre notre problème de classification de textes, nous avons sélectionné et testé quatre algorithmes d'apprentissage automatique adaptés à cette tâche. Voici une description des concepts mathématiques derrière chaque algorithme :

1. Régression Logistique : La régression logistique modélise la probabilité qu'une observation appartienne à une classe donnée en utilisant une fonction logistique. Pour une observation x, la probabilité que $y = 1$ est donnée par :

$$P(y = 1|x) = \frac{1}{1 + \exp(-w^T x - b)}$$

Où w est le vecteur des coefficients appris, b est le biais, et x est le vecteur de caractéristiques. L'algorithme maximise la log-vraisemblance des données, ce qui se traduit par une minimisation de la fonction de coût logistique :

$$\text{Coût}(w, b) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

2. Multinomial Naïve Bayes : Cet algorithme est basé sur le théorème de Bayes :

$$P(y|x) \propto P(y) \prod_{j=1}^{d} P(x_j|y)$$

Où $P(y)$ est la probabilité a priori de la classe y, et $P(x_j|y)$ est la probabilité conditionnelle d'observer le terme x_j donné la classe y. Dans le cas du modèle multinomial, $P(x_j|y)$ est calculé en fonction des fréquences des termes dans chaque classe :

$$P(x_j|y) = \frac{n_{j,y} + \alpha}{\sum_k n_{k,y} + \alpha d}$$

Où $n_{j,y}$ est le nombre de fois que le terme j apparaît dans les documents de la classe y, α est un paramètre de lissage, et d est le nombre total de termes.

3. Support Vector Machines (SVM) : Les SVM cherchent à maximiser la marge entre les classes en résolvant l'optimisation suivante :

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{tel que } y_i(w^T x_i + b) \geq 1 \text{ pour tout } i$$

Où w est le vecteur des poids, b est le biais, et x_i sont les vecteurs d'entrée avec leurs étiquettes $y_i \in \{-1, 1\}$. Pour les données non linéairement séparables, un noyau $K(x, x')$ peut être utilisé pour projeter les données dans un espace de dimension supérieure.

4. Random Forest : Une forêt aléatoire est un ensemble d'arbres de décision construits à partir d'échantillons aléatoires des données d'entraînement. Chaque arbre apprend une règle de décision en minimisant une fonction de perte, comme l'entropie ou l'indice de Gini :

$$\text{Entropie} = -\sum_k P_k \log(P_k), \quad \text{Gini} = 1 - \sum_k P_k^2$$

Où P_k est la proportion d'observations appartenant à la classe k dans un nœud. Les prédictions finales sont obtenues par un vote majoritaire parmi tous les arbres.

Ces algorithmes offrent une diversité d'approches, chacune ayant des forces spécifiques pour traiter des données textuelles. Leur performance respective sera analysée dans les sections suivantes.

4 Méthodologie

Dans cette section, nous décrivons les étapes méthodologiques suivies pour entraîner et évaluer les modèles de classification de texte. Cela inclut la répartition des données, la gestion du déséquilibre des classes, et l'exploration des hyperparamètres.

1. Répartition des données : Les données ont été divisées en deux ensembles distincts :

- **Ensemble d'entraînement** : 80% des données ont été utilisées pour entraîner les modèles.
- **Ensemble de validation** : 20% des données ont été réservées pour évaluer la performance des modèles et ajuster les hyperparamètres.

Cette répartition garantit une évaluation fiable tout en minimisant le risque de sur-apprentissage.

2. Gestion du déséquilibre des classes : Notre jeu de données est fortement déséquilibré, ce qui représente un défi pour une tâche de classification binaire. Pour atténuer ce problème, nous avons utilisé la **pondération des classes** dans les modèles. Cette méthode attribue un poids plus élevé à la classe minoritaire lors de l'entraînement, permettant aux modèles de mieux prendre en compte cette classe tout en conservant l'intégrité des données originales.

Nous avons évité le rééchantillonnage des données (sur-échantillonnage de la classe minoritaire ou sous-échantillonnage de la classe majoritaire), car cette technique pourrait introduire des biais ou réduire la diversité des données. En particulier, le sur-échantillonnage risque de créer des doublons artificiels dans la classe minoritaire, tandis que le sous-échantillonnage peut entraîner une perte d'informations dans la classe majoritaire. Ces limitations rendent la pondération des classes plus appropriée pour notre cas.

3. Régularisation et ajustement des hyperparamètres : Pour éviter le sur-apprentissage, nous avons utilisé des stratégies de régularisation adaptées à chaque modèle, ainsi que l'exploration d'hyperparamètres spécifiques :

- **Régression logistique** :
  - Régularisation L_2 pour contrôler la magnitude des coefficients.
  - Paramètre de taux d'apprentissage pour optimiser la vitesse de convergence.
- **SVM** :
  - Ajustement du paramètre C pour équilibrer la marge maximale et les erreurs de classification.
- **Random Forest** :
  - Limitation de la profondeur maximale des arbres pour réduire la complexité.
  - Nombre d'arbres dans la forêt pour améliorer la stabilité des prédictions.
- **Naïve Bayes** :
  - Paramètre de lissage α pour éviter les probabilités nulles.
  - Type de modèle (multinomial ou gaussien) selon les caractéristiques des données.

Pour évaluer les modèles de manière fiable malgré le déséquilibre des classes, nous avons utilisé une **validation croisée stratifiée**. Cette méthode garantit que chaque pli de validation contient la même proportion de classes que l'ensemble original, ce qui est essentiel pour éviter des biais dans l'évaluation des modèles sur des données déséquilibrées.

Pour l'optimisation des hyperparamètres, nous avons combiné la **recherche par grilles aléatoires** et la **recherche bayésienne**. La recherche par grilles aléatoires, qui consiste à échantillonner aléatoirement les combinaisons d'hyperparamètres dans un espace défini, nous permet d'explorer un espace plus vaste avec une moindre consommation de

ressources. La recherche bayésienne, quant à elle, affine les résultats en utilisant des probabilités conditionnelles pour sélectionner les combinaisons d'hyperparamètres les plus prometteuses. Cette combinaison d'approches est particulièrement efficace dans notre cas, car elle maximise la performance tout en limitant le temps de calcul.
Une fois les hyperparamètres optimaux déterminés pour chaque modèle, nous comparerons les résultats approfondis des méthodes, en mettant en évidence leurs forces et leurs faiblesses respectives. Cela permettra de sélectionner le modèle le plus performant pour notre tâche.
Nous avons également décidé de tester différentes valeurs pour les mots en fonction de leur significativité, mesurée à l'aide du test de Mann-Whitney U. Cependant, en raison des contraintes de calcul, le nombre de variables explicatives retenues sera optimisé uniquement pour le **modèle final choisi**, après évaluation des performances de tous les modèles testés. Cela garantit que les caractéristiques sélectionnées seront adaptées aux besoins spécifiques du modèle le plus performant.
4. Évaluation des performances : La métrique principale utilisée pour évaluer les modèles est le **macro F1-score**, qui accorde une importance égale à chaque classe. Cette métrique est adaptée pour des données déséquilibrées. Les performances ont été calculées sur l'ensemble de validation pour comparer les différents modèles.
Cette méthodologie garantit une approche rigoureuse pour maximiser les performances tout en minimisant le risque de sur-ajustement des modèles.

5 Résultats

Dans cette section, nous présentons les résultats obtenus pour les différents modèles testés après optimisation des hyperparamètres. Les performances sont évaluées en termes de macro F1-score, une métrique adaptée pour des données déséquilibrées. Nous analysons également les forces et les faiblesses de chaque méthode.
1. Comparaison des modèles : Les résultats des modèles optimisés sont résumés dans le tableau ci-dessous :

Modèle	**Macro F1-score**
Régression logistique (C=0.1037, pénalité=L_2, solver=liblinear)	0.6946
SVM (C=0.75, noyau=linéaire)	0.6832
Random Forest (arbres=200, profondeur max=15)	0.6754
Naïve Bayes (benchmark, *alpha*=0.01, fit_prior=True)	0.7096

TABLE 1 – Comparaison des performances des modèles avec leurs hyperparamètres optimaux.

2. Analyse des performances :
- La **Naïve Bayes (benchmark)** a obtenu les meilleures performances avec un macro F1-score de 0.7096. Cette méthode est rapide et bien adaptée à des données textuelles comme celles utilisées dans ce projet.
- La **régression logistique**, bien qu'ayant un macro F1-score légèrement inférieur (0.6946), offre une robustesse et une simplicité d'interprétation des résultats.
- Le **SVM** a montré des performances solides (macro F1-score de 0.6832), mais son temps de calcul peut être plus élevé, en particulier pour des jeux de données plus volumineux.

— Le **Random Forest**, avec un macro F1-score de 0.6754, est performant pour capturer des relations non linéaires mais peut être pénalisé par un temps d'entraînement plus long et une complexité accrue.

3. Optimisation spécifique au modèle Naïve Bayes : Étant donné que le modèle Naïve Bayes s'est avéré le plus performant, nous avons poursuivi l'optimisation en utilisant ce modèle pour sélectionner nos top n mots les plus significatifs selon le test de Mann-Whitney U. Après avoir testé différentes valeurs de n, nous avons trouvé que la meilleure configuration était $n = 4400$, ce qui a donné un macro F1-score de 0.7491 avec validation croisée.
Le graphique ci-dessous illustre l'évolution du macro F1-score en fonction de n :

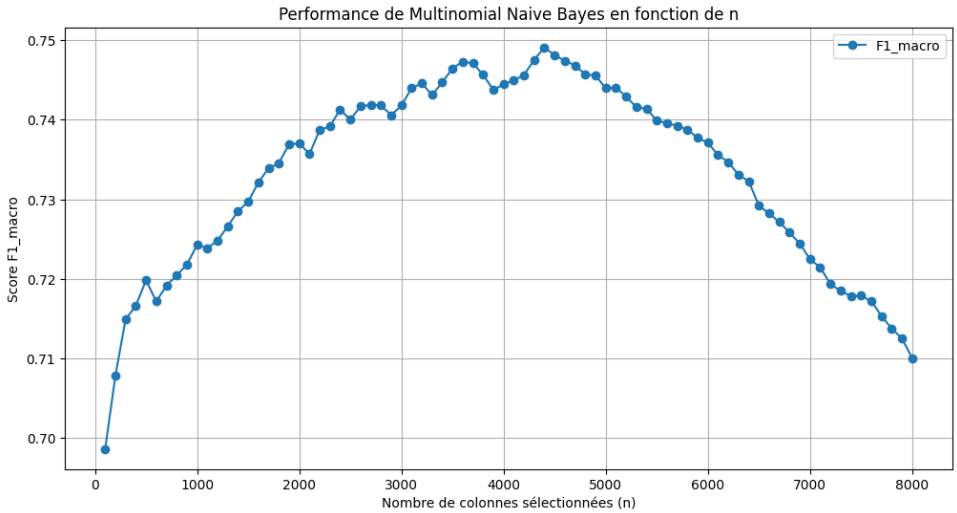

FIGURE 1 – Performance de Multinomial Naïve Bayes en fonction du nombre de mots sélectionnés (n).

4. Analyse approfondie des performances du modèle Naïve Bayes : Pour mieux comprendre le comportement du modèle Naïve Bayes, nous avons analysé ses performances en validation croisée et généré la courbe ROC correspondante. Les graphiques suivants présentent les résultats :

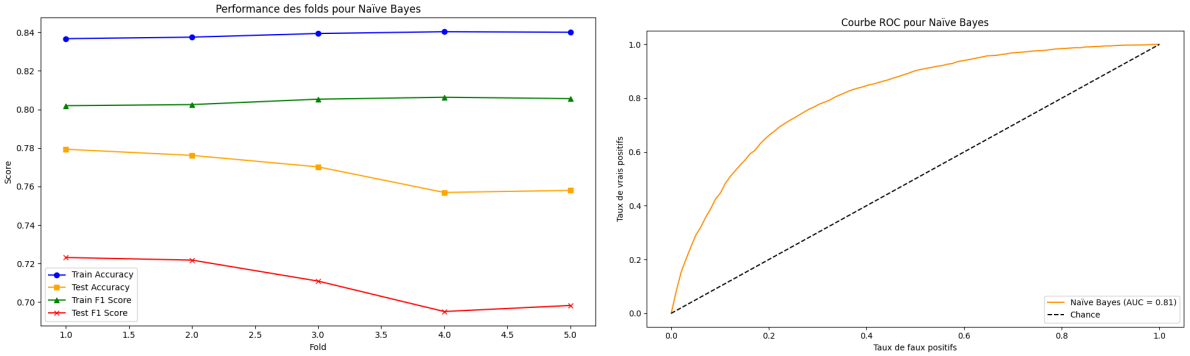

FIGURE 2 – À gauche : Performance des folds pour Naïve Bayes. À droite : Courbe ROC pour Naïve Bayes.

Commentaires techniques :

— **Performance des folds** : Le graphique de gauche montre une légère décroissance des scores de F1 et d'exactitude (accuracy) sur les données de test à mesure que les folds augmentent. Cela indique une bonne généralisation du modèle tout en mettant en évidence des défis pour maintenir des performances constantes sur des sous-ensembles de données variés.
— **Courbe ROC** : Avec une AUC de 0.81, le modèle Naïve Bayes démontre une capacité solide à distinguer les classes positives et négatives. Cela reflète une qualité de classification acceptable pour cette tâche, bien qu'il reste des marges d'amélioration pour réduire davantage les faux positifs et faux négatifs.

5. Conclusion sur les résultats : Les résultats obtenus mettent en évidence que le Naïve Bayes est particulièrement bien adapté pour cette tâche spécifique, suivi de près par la régression logistique. Ces deux modèles offrent un bon compromis entre performance et efficacité. L'optimisation supplémentaire pour le modèle Naïve Bayes en sélectionnant les mots les plus significatifs a permis d'atteindre une performance maximale de 0.7491 en macro F1-score. Le SVM et le Random Forest, bien qu'étant des modèles puissants, pourraient être mieux exploités sur des données présentant des relations plus complexes ou un volume plus important.
Dans les sections suivantes, nous discuterons des implications de ces résultats et des pistes d'amélioration possibles.

6 Discussion

Notre approche met en évidence plusieurs avantages et limites avec des pistes d'amélioration claires.

Avantages

— **Efficacité du Naïve Bayes** : Modèle rapide et performant (macro F1-score de 0.7491), adapté aux matrices creuses.
— **Optimisation ciblée** : Le test de Mann-Whitney U a permis de réduire le bruit tout en conservant les termes significatifs.
— **Validation stratifiée** : Évaluation fiable sur des données déséquilibrées, minimisant les biais.
— **Enrichissement des caractéristiques** : Scores de sentiment et nombre de mots distincts ont amélioré les représentations textuelles.

Limites

— **Sélection des caractéristiques** : Le filtrage par $n = 4400$ repose sur l'hypothèse que les termes restants suffisent pour modéliser la complexité.
— **Dépendance au Naïve Bayes** : Incapable de modéliser des relations complexes entre termes.
— **Optimisation limitée** : Exploration réduite des hyperparamètres des autres modèles (SVM, Random Forest) en raison des contraintes de calcul.
— **Absence d'analyse qualitative** : Aucun examen des cas d'erreurs pour identifier les biais ou lacunes des modèles.

Pistes d'amélioration

- **Représentations avancées** : Intégrer des embeddings comme Word2Vec, GloVe ou BERT pour capturer le contexte sémantique.
- **Modèles complexes** : Tester des modèles neuronaux comme LSTMs ou Transformers.
- **Optimisation avancée** : Appliquer la recherche bayésienne pour des hyperparamètres plus adaptés.
- **Analyse qualitative** : Identifier les cas mal classés pour comprendre les biais présents.
- **Gestion des déséquilibres** : Explorer des méthodes de rééchantillonnage ou des modèles adaptatifs comme XGBoost.

En résumé, notre pipeline offre une base solide et performante, mais des améliorations ciblées pourraient en augmenter la robustesse et l'efficacité pour des applications plus complexes.