IFT 3395: Fondements de l'apprentissage automatique

Rapport de la compétition Kaggle

Participants

Mehdi Lagnaoui (20178135)

Akram Znini (20255127)

Introduction

Cette compétition vise à développer un modèle de classification de texte capable de prédire la catégorie de documents en utilisant des techniques d'apprentissage automatique. En explorant plusieurs modèles et méthodes de prétraitement, nous avons optimisé les hyperparamètres et ajusté les configurations pour identifier la méthode la plus performante. Les résultats montrent des différences significatives en termes de scores F1 entre les modèles et soulignent l'importance des techniques de prétraitement et de réglage des hyperparamètres pour des performances optimales.

Exploration des données

Avant de procéder au prétraitement, nous avons effectué une analyse des données fournies pour comprendre leurs caractéristiques principales :

Structure des données de test:

- Nombre de documents dans l'ensemble d'entraînement: 9422
- Nombre de caractéristiques: 26354

Analyse du déséquilibre des classes:

Distribution des classes:

- Classe 0: 7124 échantillons (75.61%)
- Classe 1: 2298 échantillons (24.39%)

Ratio de déséquilibre: 3.10

Stratégie de prétraitement

Sur la base de notre analyse exploratoire, nous avons choisi de faire 3 étapes de prétraitement:

1. Transformation TF-IDF (Term Frequency-Inverse Document Frequency)

Nous avons choisi d'utiliser TF-IDF pour prendre en compte non seulement de la fréquence des termes mais aussi de leur importance relative dans le corpus et réduire l'impact des mots très fréquents mais peu discriminants.

2. Rééquilibrage des données avec SMOTE (Synthetic Minority Over-sampling Technique)

Pour traiter le déséquilibre des classes, nous avons appliqué l'algorithme SMOTE pour améliorer la robustesse du modèle pour la classe minoritaire 1.

3. Standardisation / Normalisation des caractéristiques:

Pour améliorer les performances dans l'étape 1, on a utilisé la standardisation des caractéristiques pour avoir une moyenne de 0 et un écart-type de 1 en utilisant la formule : $z = (x - \mu) / \sigma$.
Ceci améliore la convergence des algorithmes (plus précisément la régression logistique qu'on a utilisé dans l'étape 1).

Étape 1 (dépasser les points de référence)

Algorithme

Dans cette étape, nous avons choisi d'utiliser la régression logistique pour sa simplicité et son efficacité prouvée pour les tâches de classification textuelle. Nous avons expérimenté plusieurs variantes de l'algorithme pour optimiser ses performances.

Méthodologie

Nous avons effectué nos tests en divisant l'ensemble d'entraînement selon un ratio 80/20 (entraînement/validation). La métrique d'évaluation choisie est le score F1 macro.

Résultats et Discussion

Notre modèle de base, la régression logistique simple, a obtenu un score F1 de 0.5959. Ces performances limitées s'expliquent principalement par des problèmes de convergence lors de la descente de gradient, causés par les différences d'échelle entre les caractéristiques.

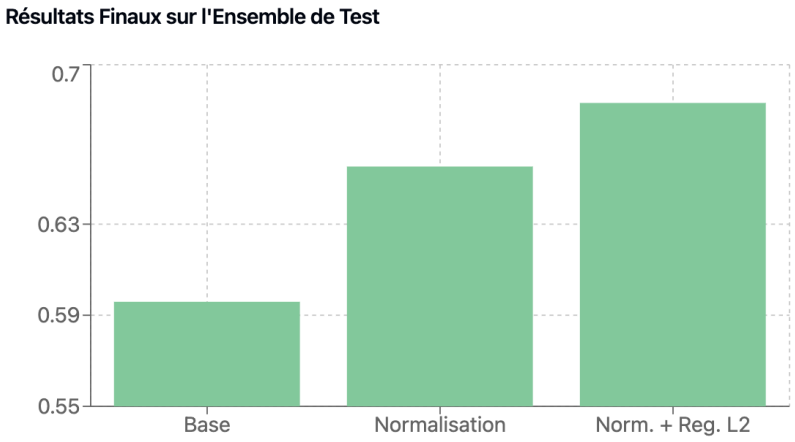
Pour remédier à ce problème, nous avons implémenté la normalisation des caractéristiques, qui a significativement amélioré les performances avec un score F1 de 0.6553 (amélioration de +5.94 points). Cette amélioration s'explique par la mise à l'échelle uniforme des caractéristiques (moyenne 0, variance 1), permettant une descente de gradient plus stable et une convergence plus rapide et précise.

Nous avons ensuite exploré la gestion du déséquilibre des classes, obtenant des scores F1 de 0.5819. Cependant, cette approche, même combinée avec la normalisation (score F1 : 0.6058), n'a pas apporté d'amélioration significative par rapport à la normalisation seule.

L'ajout de la régularisation L2 a été testé avec différentes valeurs du paramètre λ. Nous avons obtenu des scores F1 de 0.6339 pour λ=0.1, 0.6413 pour λ=0.01, et 0.6568 pour λ=0.001. La régularisation aide à éviter le surapprentissage en pénalisant les grands poids, tout en améliorant la généralisation du modèle et en stabilisant l'apprentissage.

La configuration optimale sur notre ensemble de validation s'est révélée être une régression logistique avec normalisation des caractéristiques et une légère régularisation L2.

Pour les soumissions finales sur l'ensemble de test, nous avons évalué les trois variantes les plus prometteuses : la régression logistique de base, la régression logistique avec normalisation des caractéristiques, et la régression logistique combinant normalisation et régularisation L2 (λ=0.01). Cette dernière configuration s'est révélée la plus performante sur l'ensemble de test, atteignant un score F1 de 0.68321.



Etape 2 (Compete)

Algorithmes

Dans cette phase, nous avons exploré plusieurs algorithmes de classification pour améliorer nos performances par rapport à la régression logistique de l'étape précédente. Notre sélection s'est portée sur des algorithmes reconnus pour leur efficacité en classification textuelle : MultinomialNB, particulièrement adapté aux données textuelles avec une distribution multinomiale, SVM linéaire (LinearSVC) pour sa capacité à gérer les espaces de grande dimension, et Random Forest qui offre une bonne capacité de généralisation grâce à son ensemble d'arbres de décision. Nous avons également exploré une approche d'ensemble learning combinant Random Forest et MultinomialNB pour tenter de tirer parti des forces de chaque algorithme.

Méthodologie

Pour cette étape, nous avons adopté une approche plus rigoureuse avec :

- Utilisation de la validation croisée (k=5) avec score F1 macro pour une évaluation plus robuste
- Gestion du déséquilibre des classes (0: 75.6%, 1: 24.4%) via :
    - Pondération des classes ('class_weight="balanced"')
    - Test de SMOTE pour le rééquilibrage
- Sélection de caractéristiques avec Random Forest pour réduire la dimensionnalité

- Tests des hyperparamètres différents pour chaque modèle

Résultats

MultinomialNB :

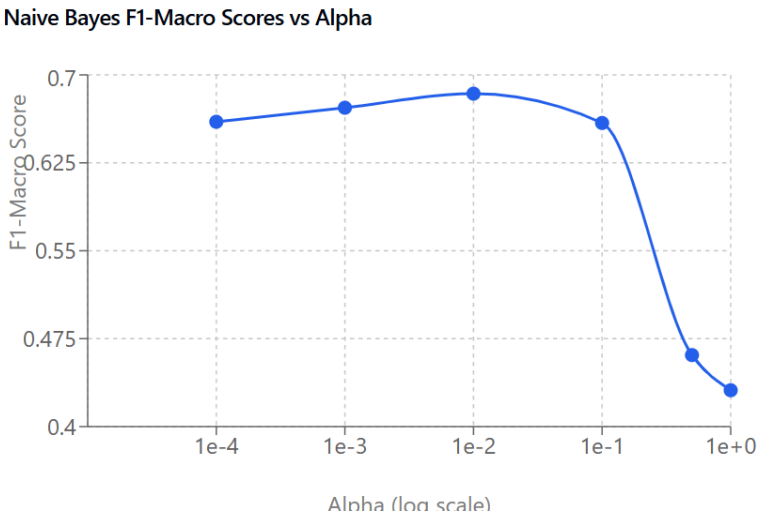
La version de base s'est révélée remarquablement efficace avec :

- Score F1 local (validation croisée) : 0.710
- Score F1 sur Kaggle (X_test) : 0.73538

Matrice de confusion sur l'ensemble d'entraînement :
[[5698 1426]

 [768 1530]]
Cette matrice montre un bon équilibre dans les prédictions, sans surapprentissage apparent.

Nous avons expérimenté différentes valeurs d'alpha (0.5, 0.1, 1.0), mais leur impact est resté limité: (scores avec validation croisée)



L'application de SMOTE n'a pas amélioré les performances (score sur X_test : 0.71934), alors le déséquilibre des classes n'est peut-être pas aussi problématique qu'on le pensait et peut-être que les transformations peuvent introduire du bruit ou perdre de l'information utile.

LinearSVC :

- Score F1 local (validation croisée) : 0.651
- Score F1 sur Kaggle (X_test) : 0.65133

Matrice de confusion initiale :
[[7124 0]

[2 2296]]

Après optimisation (C=0.1, class_weight='balanced') :

[[7094 30]

[0 2298]]

Ces matrices révèlent un surapprentissage sévère, avec une classification presque parfaite sur l'ensemble d'entraînement mais des performances limitées en test.

RandomForest:

- La version initiale montrait un surapprentissage extrême avec une classification presque parfaite sur l'ensemble d'entraînement
    - Score F1 local (X_test) : 0.530
    - Matrice de confusion:

      [[7122 2]

      [0 2298]]

- L'optimisation des hyperparamètres:
    - profondeur limité: 10
    - augmentation du nombre d'arbre: 200
    - Augmentation du nombre des échantillons minimaux par split: 10
    - Augmentation du nombre des échantillons minimaux par feuille: 5

Cela a permis d'atteindre un F1 local de 0.710 et sur kaggle (X_test) de 0.71699, avec un meilleur équilibre entre les classes.

Matrice de confusion:

[[5699 1425]

[375 1923]]

Cette amélioration montre comment la limitation de la complexité du modèle a permis de réduire significativement le surapprentissage.

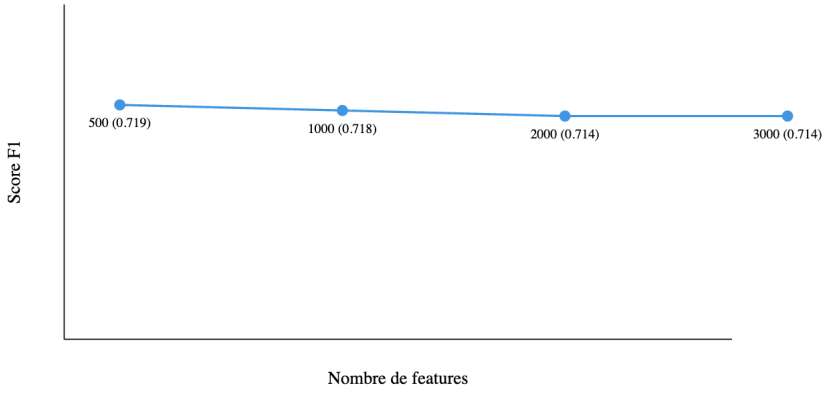
Sélection de caractéristiques avec MultinomialNB :

- Score local avec 500 features : 0.719
- Score Kaggle (X_test) : 0.72407

Matrice de confusion :
[[5629 1495]

[615 1683]]

Tests systématiques des features : (avec validation croisée)



Cette réduction (de 26354 à 500 features) a légèrement amélioré les performances, tout en réduisant significativement la complexité du modèle.

Ensemble Learning (Random Forest + MultinomialNB) :

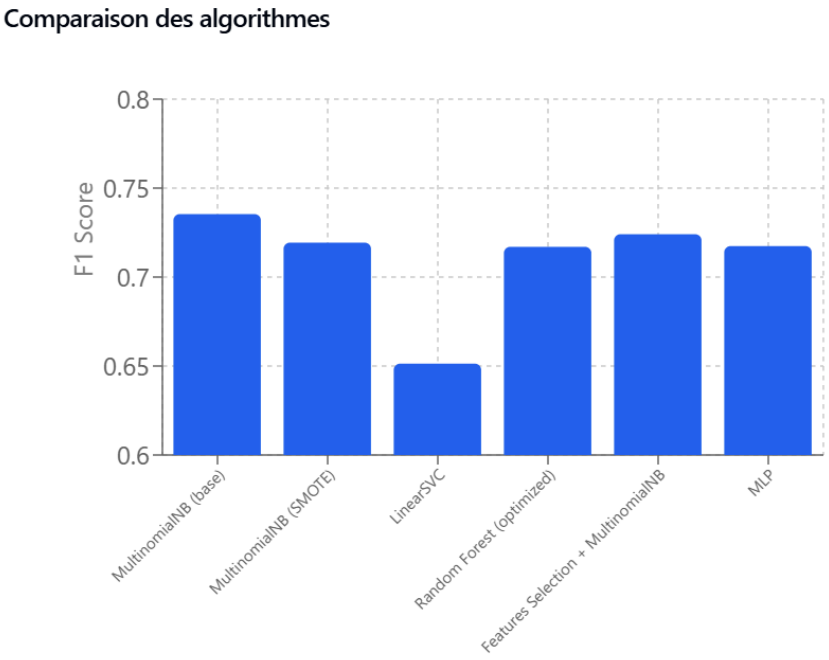
- Score F1 local : 0.710

Matrice de confusion :
[[6042 1082]

[363 1935]]

Cette approche est très complexe mais elle n'a pas apporté d'amélioration significative.

Discussion



L'analyse de nos résultats met en évidence plusieurs points intéressants. Tout d'abord, le MultinomialNB s'est révélé être le modèle le plus performant (F1=0.73538 sur l'ensemble de test), montrant qu'un algorithme simple mais bien adapté aux données textuelles peut surpasser des approches plus complexes.

La gestion du déséquilibre des classes n'a pas eu l'impact attendu. L'utilisation de SMOTE n'a pas amélioré les performances (F1=0.71934), suggérant que le déséquilibre n'était pas aussi problématique que prévu pour notre cas. Les tentatives de rééquilibrage semblent même avoir introduit du bruit dans les données.

Le problème principal rencontré a été le surapprentissage, particulièrement visible avec SVM et la version initiale de Random Forest. Leurs matrices de confusion presque parfaites sur l'ensemble d'entraînement contrastaient fortement avec leurs performances en test. Nous avons pu améliorer les performances du Random Forest en limitant sa complexité (profondeur maximale, nombre minimum d'échantillons), montrant l'importance d'un bon réglage des hyperparamètres.

La réduction du nombre de caractéristiques de 26354 à 500 s'est révélée efficace, maintenant de bonnes performances tout en simplifiant significativement les modèles. Cette approche a permis d'obtenir un bon compromis entre performance et complexité.

Pour améliorer ces résultats, il serait intéressant d'explorer d'autres techniques de traitement du texte, de tester des modèles plus récents comme XGBoost, et d'analyser plus en détail l'importance relative des différentes caractéristiques. Un travail plus approfondi sur la sélection des features pourrait également être bénéfique.