ml-practic

Claudiu Popoiu, Gulica Silvian

January 2024

1 Introducere

Pentru a rezolva problema filtrarii emailurile de spam, trebuie sa analizam modul ce algoritm din punct de vedere computational este cel mai bun la rezolvarea problemei noastre. În aceasta tema, ne vom concentra pe evaluarea performanței unor algoritmi selectați de învățare automată, aplicându-i pe setul de date Ling-Spam. Scopul nostru este de a determina care dintre acești algoritmi oferă cel mai bun echilibru între acuratețe și eficiență computațională în sarcina de clasificare a email-urilor ca spam sau non-spam, luând în considerare diversele abordări de preprocesare ale datelor (descrise mai jos).

2 Metodologie

2.1 Setul de date

Setul de date Ling-Spam este format dintr-o colecție de email-uri categorisite în mai multe foldere. Fiecare email este etichetat ca spam sau non-spam, cu această informație reflectată în titlul fiecărui fișier. Pentru antrenarea modelului ne vom folosi de primele 9 foldere dintr-unul din cele 4 foldere, iar pentru testare de al 10-lea folder.Pentru usurinta, noi am mutat peste tot conținutul celor 9 foldere într-un folder *train*, iar folderul 10 a fost redenumit în *test*.

2.2 Preprocesarea datelor

O caracteristică distinctă a setului de date Ling-Spam este că include deja diverse metode de preprocesare aplicate pe instanțe, cum ar fi lematizarea și extragerea cuvintelor de oprire (stop words), distribuite în patru foldere diferite. Având în vedere această preprocesare preexistentă, am decis să nu aplicăm tehnici suplimentare de preprocesare. Scopul nostru este de a testa eficiența algoritmului Naive Bayes în funcție de aceste metode de preprocesare deja implementate, presupunând că aceasta poate oferi o perspectivă mai amplă asupra performanței algoritmului în condiții variate de preprocesare (anume, atunci cand nu aplicam preprocesare - folderul *bare* -, atunci cand aplicam lematizare - folderul *lemm* -, atunci cand extragem cuvintele de stop - folderul *stop* -,

si atunci cand aplicam si lematizare si extragerea cuvintelor de stop - folderul *lemm_stop* -.

3 Naive Bayes

Pentru sarcina de clasificare a email-urilor spam, am optat pentru implementarea algoritmului Naive Bayes. Motivele pentru care am făcut acest lucru este ca Naive Bayes un algoritm relativ ușor de implementat, dar cu performanțe foarte bune pe seturi de date textuale. N.B. are o eficiență foarte mare in sarcini de clasificare ale textului.

3.1 Implementarea algoritmului

Am implementat algoritmul Naive Bayes în Python, folosind biblioteci standard precum NumPy pentru gestionarea operațiunilor matematice. Implementarea se concentrează pe calculul probabilităților condiționate și pe utilizarea acestora pentru a clasifica email-urile ca fiind spam sau non-spam. În plus, am aplicat tehnica de smoothing Laplace pentru a gestiona problemele de zero-probabilitate care pot apărea în cazul cuvintelor necunoscute.

Procesul nostru de implementare a inclus etapele următoare:

1. **Citirea și procesarea email-urilor**: Email-urile din setul de date au fost citite folosind biblioteca ‘os‘. Am utilizat o funcție personalizată, ‘tokenize‘, pentru a descompune textul fiecărui email în cuvinte. În funcție de prezența prefixului “spm” în titlul fișierului, am etichetat fiecare email ca spam sau non-spam.

2. **Construirea modelului de probabilități**: Pentru fiecare cuvânt din setul de date, am calculat frecvența apariției acestuia în email-urile spam și non-spam, respectiv. Aceasta a implicat utilizarea unui dicționar Python pentru a stoca frecvențele, împreună cu aplicarea tehnicii de smoothing Laplace pentru a evita problemele legate de probabilități zero.

3. **Clasificarea email-urilor**: Am definit o funcție, ‘classify_email‘, care calculează probabilitatea ca un email să fie spam sau non-spam pe baza frecvențelor cuvintelor. Aceasta a fost realizată prin aplicarea formulei lui Bayes și logaritmarea probabilităților pentru ca produsul probabilităților să nu fie aproximat la 0 (pentru că lucrăm cu probabilități foarte mici)

4. **Evaluarea acurateței**: Acuratețea modelului a fost evaluată prin compararea clasificărilor generate de model cu etichetele reale ale email-urilor din setul de testare. Am calculat acuratețea ca procentajul de email-uri clasificate corect.

5. **Validarea Cross-Leave-One-Out**: Am implementat și o funcție de validare Cross-Leave-One-Out pentru a evalua robustețea modelului. Aceasta a implicat reantrenarea modelului de fiecare dată când un email era omis

din setul de date de antrenament, oferind astfel o imagine completă a performanței modelului.

4 Compararea algoritmilor si CVLOO

4.1 Implementarea Cross-Validării Leave-One-Out (CVLOO)

Pentru a evalua performanța algoritmului Naive Bayes am implementat strategia de cross-validare Leave-One-Out (CVLOO). Această metodă implică utilizarea fiecărui email din setul de date ca un set de testare unic, în timp ce restul email-urilor sunt folosite pentru antrenarea modelului.

4.2 Compararea cu alți algoritmi

În afară de Naive Bayes, am testat și comparat performanța altor algoritmi populari de clasificare: ID3, k-Nearest Neighbors (kNN) și AdaBoost. Aceste modele au fost implementate folosind biblioteca 'scikit-learn' datorită implementărilor sale eficiente și bine optimizate.

4.3 Vizualizarea și analiza rezultatelor

Pentru a ilustra rezultatele cross-validării și acuratețea fiecărui algoritm pe setul de testare, am creat o serie de grafice folosind biblioteca 'matplotlib'. Aceste grafice vor furniza o reprezentare vizuală a performanței fiecărui model, facilitând compararea directă între ei.

4.4 Interpretarea graficelor

Vom prezenta acuratețea obținută de diferitele algoritmi de clasificare pe patru variante de seturi de date: bare, lemm, lemm_stop și stop. Rezultatele sunt reprezentate vizual în graficele de mai jos, care permit o comparație directă între algoritmi. Fiecare bar reprezintă, de la stânga la dreapta: acuratete ID3, acuratete AdaBoost, acuratete k-NN, acuratete Naive Bayes și acuratete CVLOO Naive Bayes.

4.4.1 Setul de date 'bare'

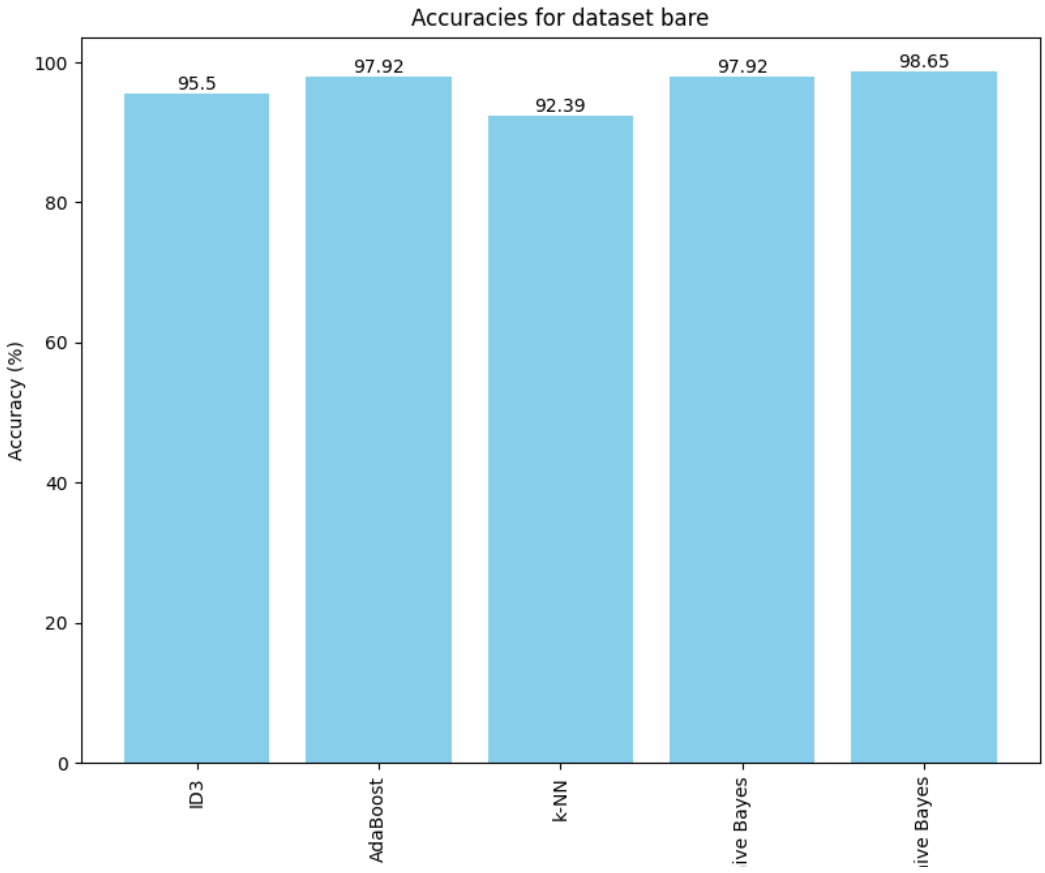

Figure 1: De la stanga la dreapta: ID3, Adaboost,k-NN, Naive Bayes, CVLOO Naive Bayes

Pe setul de date 'bare', care nu a fost supus unui proces de lematizare sau de eliminare a cuvintelor de oprire, observăm că algoritmul Naive Bayes are cea mai bună performanță, urmat îndeaproape de algoritmul Adaboost. Acuratețea de clasificare a acestor modele depășește cea a algoritmilor ID3 și kNN, ceea ce indică o adaptare superioară a Naive Bayes la setul de date ne-preprocesat.

4.4.2 Setul de date ‘lemm‘

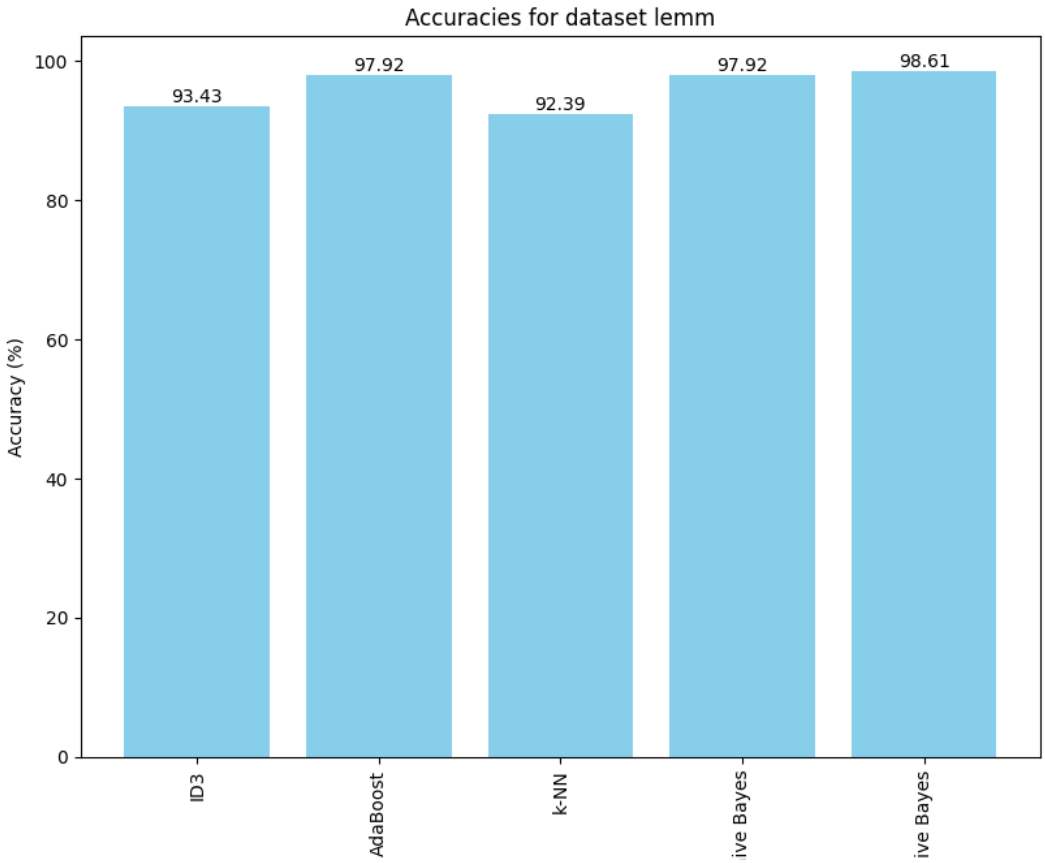

Figure 2: De la stanga la dreapta: ID3, Adaboost,k-NN, Naive Bayes, CVLOO Naive Bayes

Pentru setul de date ‘lemm‘, care a fost preprocesat prin lematizare, observăm din nou că varianta noastră de Naive Bayes și AdaBoost sunt în topul performanței, demonstrând adaptabilitatea algoritmului la datele preprocesate pentru reducerea formelor flexionare ale cuvintelor. În comparație, ID3 și AdaBoost prezintă o acuratețe ușor inferioară.

4.4.3 Setul de date ‘lemm_stop‘

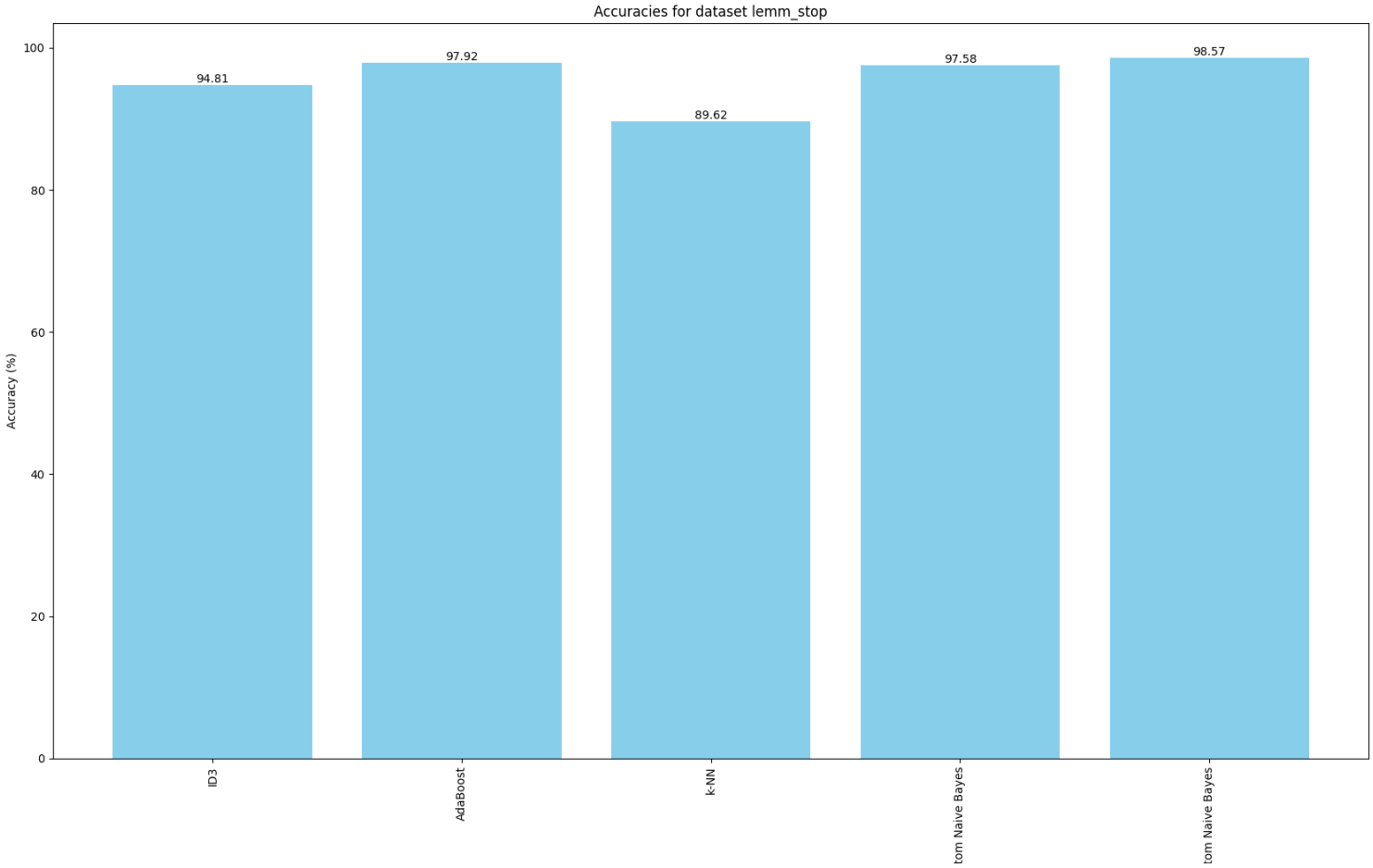

Figure 3: De la stanga la dreapta: ID3, Adaboost,k-NN, Naive Bayes, CVLOO Naive Bayes

Pentru ‘lemm_stop‘, care combină lematizarea cu eliminarea cuvintelor de oprire, rezultatele sunt mixte. Deși Naive Bayes continuă să prezinte rezultate impresionante, AdaBoost înregistrează o acuratețe mai bună. Această variație sugerează că preprocesarea extensivă poate să nu fie întotdeauna benefică pentru modelele de învățare.

4.4.4 Setul de date ‘stop‘

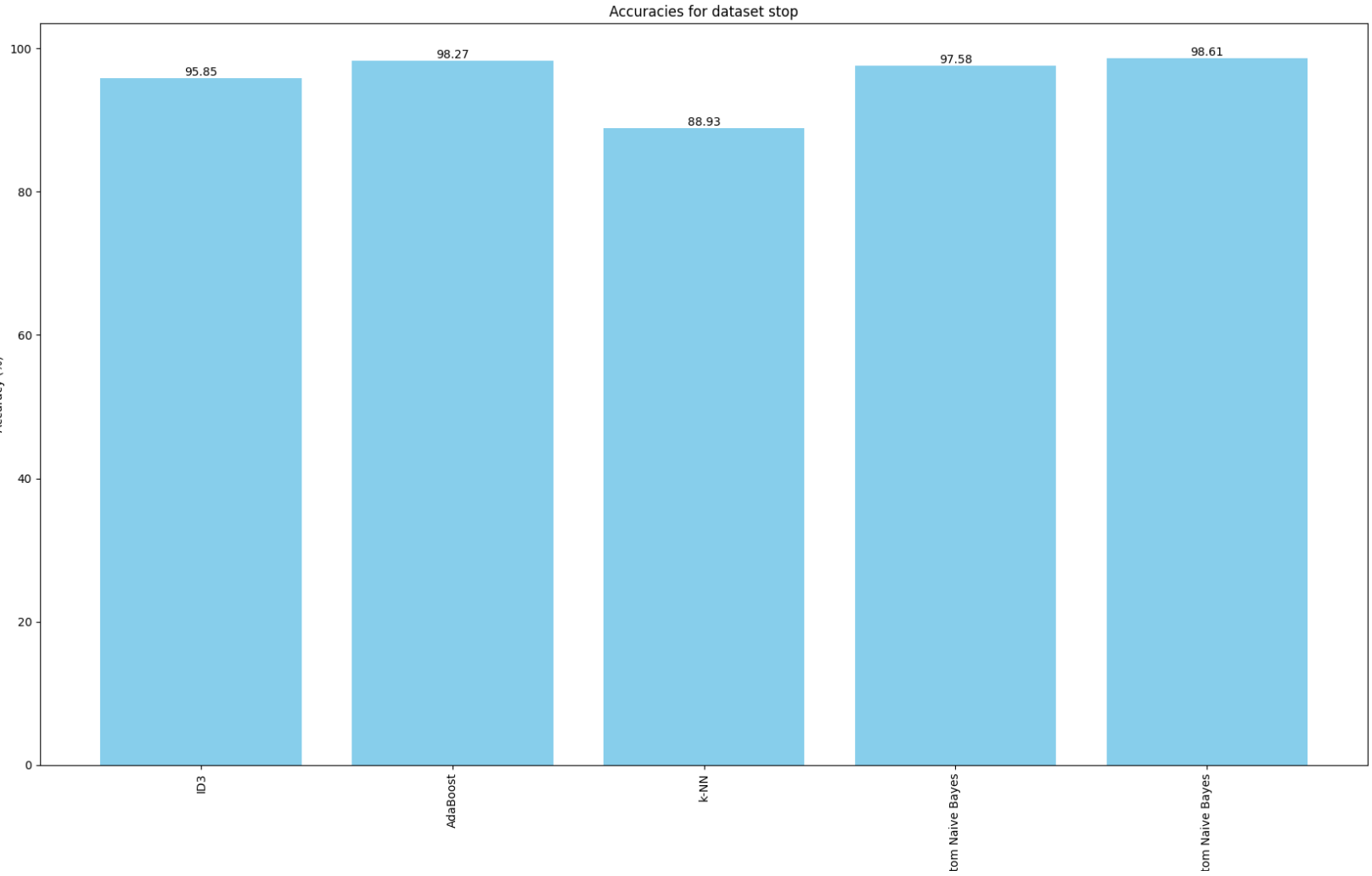

Figure 4: De la stanga la dreapta: ID3, Adaboost,k-NN, Naive Bayes, CVLOO Naive Bayes

În cele din urmă, pentru setul de date ‘stop‘, unde doar cuvintele de oprire au fost eliminate, Naive Bayes își menține performanța solidă. Interesant este că

AdaBoost da dovadă in continuare de o performanță mai bună, ceea ce indică că eliminarea cuvintelor de stop are o influență pozitivă asupra acurateței pentru modelele generate de ID3 și AdaBoost, dar nu și de Naive Bayes (a carui acuratețe a scăzut puțin în comparație cu setul 'bare'.

Putem observa faptul că în toate cazurile de mai sus, acuratețea CVLOO pentru N.B. este > 98.5 ceea ce indică o performanță consistentă.