

Bayes Naive în clasificarea email-urilor spam

Fiodorov Cristian, Galan Andrei-Iulian

January 10, 2024

Abstract

Prin acest raport ne dorim să justificăm alegerea algoritmului Naive Bayes pentru rezolvarea problemei de clasificare a email-urilor spam, atât din perspectiva teoretică, și experimentală. Algoritmul Naive Bayes este bazat pe teorema lui Bayes și totodată pe presupunerea de independență condițională a variabilelor, aceste lucruri fiind considerate potrivite pentru a rezolva problema noastră. Experimentele efectuate pe setul nostru de date confirmă, în comparație cu alți algoritmi studiați, performanța superioară a algoritmului Bayes Naive, susținând astfel alegerea noastră în practică.

1 Introduction

Problema studiată în acest raport este clasificarea email-urilor spam, iar setul de date utilizat pentru evaluare este Ling-Spam¹. Clasificarea precisă a email-urilor este foarte importantă pentru lupta împotriva email-urilor cu un potențial de risc ridicat pentru utilizatorii, care utilizează o astfel de platformă.

Soluționarea acestei probleme, este făcută prin alegerea unui algoritm potrivit, ceea ce duce la obținerea rezultatelor optime. În contextul acestui raport, am analizat și comparat mai mulți algoritmi de învățare automată pentru a selecta cel mai bun în rezolvarea problemei noastre.

Algoritmul Naive Bayes a fost alegerea ideală, datorită principiilor teoremei lui Bayes și ipotezei de independență condițională. Această alegere este susținută și de rezultatele experimentelor noastre, care vor fi detaliate mai jos.

În continuare, vom preciza aspectele teoretice care stau la baza algoritmului Bayes Naive și vom prezenta rezultatele experimentelor noastre pentru a evidenția de ce acesta a fost ales în favoarea altor algoritmi.

2 Argumente Teoretice

Pentru a accentua alegerea noastră pentru algoritmul Bayes Naive, este esențial să explorăm principiile teoretice care stau la baza acestuia.

Algoritmul Naive Bayes se bazează pe teorema lui Bayes, care oferă un cadru matematic solid pentru calculul probabilităților condiționate. În contextul clasificării email-urilor spam, algoritmul ales evaluează probabilitatea că un email să fie spam sau non-spam, dată fiind o serie de proprietăți. Bayes Naive fiind mai mult un algoritm de natură probabilistă, împreună cu principiile teoremei lui Bayes permit actualizarea eficientă a probabilităților în funcție de noi informații.

De asemenea, ipoteza de independență condițională pe care și-o asumă Bayes Naive simplifică modelul nostru, presupunând că datele sunt independente între ele, ceea ce poate reduce complexitatea modelului și timpul de antrenare.

Bayes Naive fiind un algoritm de învățare supervizată, acest lucru implică necesitatea unui set de date de antrenare etichetat pentru a învăța și a face predicții bune. Această caracteristică este esențială în problemele de clasificare, precum cea a email-urilor spam. În cazul nostru, cum avem deja la dispoziție un set de date care să cuprindă email-uri etichetate corespunzător ca spam sau non-spam putem antrena și evalua corect algoritmul.

Pentru a susține alegerea noastră în favoarea algoritmului Naive Bayes, prezentăm acum și aspectele teoretice care au influențat decizia noastră în comparație cu alți algoritmi.

¹Disponibil la <http://www.aueb.gr/users/ion/data/lingspam-public.tar.gz>

2.1 K-Nearest Neighbors (KNN)

KNN este sensibil la scala atributelor și poate întâmpina dificultăți în gestionarea seturilor de date cu dimensiuni mari, cum este și cazul în Ling-Spam. De asemenea, performanța KNN poate scădea în prezența unor caractere cu impact diferit în spațiul de dimensiuni.

2.2 ID3 (Iterative Dichotomiser 3)

ID3 este sensibil la overfitting, în special pe seturi de date mici sau zgomotoase. ID3 ar putea să nu generalizeze bine, deoarece în cazul setului Ling-Spam, pot exista variații semnificative și totodată datele pot fi neomogene.

2.3 Adaboost

Adaboost poate fi sensibil la datele extrem de variabile sau zgomotoase. În setul de date Ling-Spam, unde email-urile pot avea conținut divers și unde există variabilitate este destul de ridicată între exemplele email-uri de spam, Adaboost se poate adapta destul de greu.

3 Argumente Experimentale

Implementarea algoritmului Bayes Naive a condus la rezultate foarte bune în ceea ce privește acuratețea și eficiența în problema clasificării email-urilor spam. Pentru setul de date Ling-Spam, acest algoritm s-a dovedit a fi robust și adaptabil la diferite situații, fiind astfel rezistent la apariția/lipsa zgomotelor, ceea ce este o caracteristică utilă atunci când se lucrează cu date text, totodată având capacitatea de a se comporta bine pe seturi de date, unde variabilitatea era semnificativă.

De asemenea performanța este un element esențial, unde timpul de antrenare și procesare pentru Bayes Naive a fost relativ scăzut așa cum era și de așteptat datorită proprietății de independență condițională, ceea ce îl face o alegere foarte bună pentru seturi de date de dimensiuni mari.

Astfel în urma aplicării algoritmului Bayes Naive, am obținut în urma cross-validării Leave-One-Out eroare de 1,2% pentru întregul set de date, fiind un procent foarte bun, având în vedere faptul ca setul de date a fost unul foarte mare. În ceea ce privește setul de date de testare, acuratețea este una de 0,99% pentru folderul *stop*, 0,99% pentru *lemn-stop*, 0,98% pentru *lemn* și 0,99% pentru folderul *bare*.

Având în vedere complexitatea și diversitatea algoritmilor disponibili, implementarea noastră s-a concentrat în principal pe evaluarea algoritmului Naive Bayes în contextul problemei de clasificare a email-urilor spam. Deoarece nu am efectuat implementări pentru ceilalți algoritmi, vom aborda această secțiune printr-o evaluare generală a potențialelor lor rezultate în cadrul problemei.

3.1 K-Nearest Neighbors (KNN)

KNN, ca și algoritmul Bayes Naive, este o opțiune atractivă pentru problemele de clasificare. Cu toate acestea, în cazul setului de date Ling-Spam, unde dimensiunea și variabilitatea cuvintelor-cheie pot influența semnificativ performanța, KNN poate întâmpina dificultăți în gestionarea eficientă a datelor și în identificarea unor structuri complexe.

Acest lucru se poate observa în urma rezultatelor obținute pe același set de date, astfel că în urma aplicării algoritmului KNN pe datele de testare, am obținut 0,84% pentru folderul *stop*, 0,87% pentru *lemn-stop*, 0,85% pentru *lemn* și 0,87% pentru folderul *bare*, care în comparație cu Bayes Naive sunt mult mai mici, întărind astfel alegerea făcută.

3.2 ID3 (Iterative Dichotomiser 3)

Algoritmul ID3, bazat pe arbori de decizie, poate fi eficient în identificarea relațiilor de decizie. Cu toate acestea, într-un context precum Ling-Spam, unde există multiple cuvinte-cheie și posibile interdependențe, ID3 ar putea suferi de overfitting și dificultăți în generalizare ajungând astfel la clasificări eronate pe datele de testare. La fel ca și la KNN, am obținut rezultate mai mici la testare față de Bayes Naive, respectiv: 0,77% pentru folderul *stop*, 0,82% pentru *lemn-stop*, 0,81% pentru *lemn* și 0,76% pentru folderul *bare*.

3.3 Adaboost

Adaboost, ca și algoritmul Naive Bayes, face parte din categoria algoritmilor de învățare pe bază de ansamblu. În timp ce poate obține performanțe remarcabile, poate fi sensibil la datele zgomotoase și poate necesita ajustări fine ale parametrilor pentru a se adapta corespunzător la specificul setului de date, ceea ce îl face mult mai greu de implementat.

Totuși această complexitate a algoritmului se poate observa, în urma rezultatelor obținute, care nu sunt la fel de bune ca Bayes Naive, însă față de ceilalți doi algoritmi prezentați mai sus, are o acuratețe mai bună: 0,91% pentru folderul *stop*, 0,91% pentru *lemn-stop*, 0,90% pentru *lemn* și 0,90% pentru folderul *bare*.

Toate aceste date prezentate mai sus, se pot observa la punctul **5**, respectiv **6** al documentului, unde sunt afișate grafice pentru CVLOO, respectiv acuratețea pe datele de testare.

4 Algoritmul Naive Bayes

Algoritmul lui Bayes Naive este o metodă de clasificare, care se bazează pe teorema lui Bayes și pe presupunerea de independență condițională a caracteristicilor. În continuare vom prezenta pașii și strategiile pe care le-am aplicat în implementare pentru a ajunge la rezultatele obținute:

4.1 Colectarea datelor:

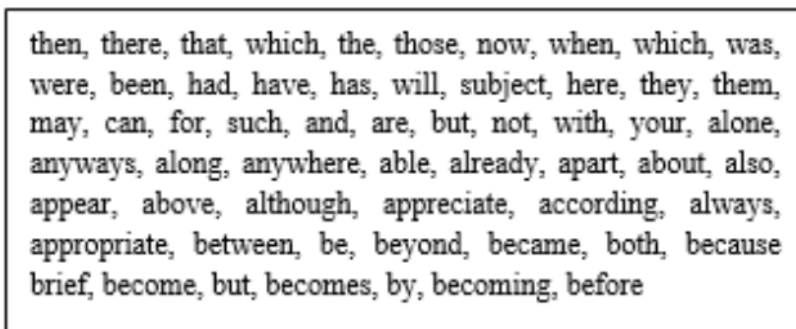
Pentru setul de date am folosit Ling-Spam, unde mesajele erau împărțite pe categorii: *lemn*, *bare*, *stop* și *lemn-stop*. Iar fiecare categorie era formată din 10 foldere, cu fișiere text, unde erau email-uri spam și non-spam.

4.2 Preprocesarea datelor:

În filtrarea spam-ului, preprocesarea informațiilor textuale este foarte critică și importantă. Obiectivul principal al preprocesării datelor text este eliminarea datelor care nu oferă informații utile cu privire la clasa documentului.

Aceste proceduri constau în curățarea și normalizarea datelor, eliminând orice zgomot sau inconsistență. Pașii pe care i-am folosit pentru preprocesarea datelor, pentru detectarea spamului sunt:

1. Cuvintele cu lungimea ≤ 2 sunt eliminate (Exemplele includ: a, is, an, of, to, as, on etc.)
2. Toate caracterele speciale sunt eliminate.
3. Cuvintele stop sunt eliminate.



then, there, that, which, the, those, now, when, which, was, were, been, had, have, has, will, subject, here, they, them, may, can, for, such, and, are, but, not, with, your, alone, anyways, along, anywhere, able, already, apart, about, also, appear, above, although, appreciate, according, always, appropriate, between, be, beyond, became, both, because brief, become, but, becomes, by, becoming, before

Figure 1: Stop words list

4. Algoritmul Stemming Porter este aplicat pentru a aduce cuvântul în forma lor cea mai de bază.

| Words | Stem |
|-----------|---------|
| ponies | poni |
| caress | caress |
| cats | cat |
| feed | fe |
| agreed | agre |
| plastered | plaster |
| motoring | motor |
| sing | sing |
| conflated | conflat |
| troubling | troubl |
| sized | size |
| hopping | hop |
| tanned | tan |
| falling | fall |

Table 1: Exemple de cuvinte și corespondența cuvintelor cheie (stem).

5. Frecvența termenului, care numără numărul de apariții ale termenului într-un document text. Matematic, poate fi reprezentat în felul următor:
 $\text{Term_Frequency_}W_{ij} = tf_{ij}$, unde tf_{ij} este frecvența cuvântului i în documentul j .

4.3 Separarea datelor:

Pentru fiecare categorie, din cele 10 foldere, am ales primele 9 pentru antrenare, iar ultimul a fost păstrat pentru testare.

Setul de antrenament este folosit pentru a învăța modelul, în timp ce setul de testare este folosit pentru a evalua performanța modelului.

4.4 Calcularea probabilităților prior:

S-au calculat probabilitățile prior pentru fiecare clasă în funcție de frecvența cu care apar în setul de antrenament.

Pentru acest punct s-a folosit formula:

$$P(C_i) = \frac{\text{Numărul de exemple din clasa } C_i}{\text{Numărul total de exemple din setul de antrenament}}$$

4.5 Calcularea probabilităților condiționate:

Formula pentru calcularea probabilităților în contextul algoritmului Bayes Naive este specifică pentru fiecare atribut/caracteristică în parte. Probabilitatea condiționată indică probabilitatea unei anumite valori a unei caracteristici X condiționată de clasa C_i . Formula generală de calculare a acestei probabilități este:

$$P(X = x|C_i) = \frac{\text{Numărul de exemple din clasa } C_i \text{ care au caracteristica } X \text{ cu valoarea } x}{\text{Numărul total de exemple din clasa } C_i}$$

4.6 Calcularea probabilităților condiționate cu corectarea Laplace:

După aplicarea algoritmului Bayes Naive, este adesea util să se facă o corectare a probabilităților condiționate folosind teorema lui Laplace. Această corectare are loc pentru a evita problemele ce pot apărea atunci când întâlnim situații în care unele combinații de valori pentru caracteristici nu au fost observate în setul de antrenare.

Formula corectată pentru probabilitatea condiționată, utilizând teorema Laplace, este exprimată astfel:

$$P(X = x|C_i) = \frac{\text{Numărul de exemple din clasa } C_i \text{ care au caracteristica } X \text{ cu valoarea } x+1}{\text{Numărul total de exemple din clasa } C_i + \text{Numărul posibil de valori pentru caracteristica } X}$$

Această corectare adaugă un termen de 1 atât în numărător, cât și în numitor, pentru a asigura că probabilitățile estimate nu devin zero atunci când întâlnim valori noi sau rare în setul de date. Prin aplicarea teoremei Laplace, algoritmul devine mai robust în fața datelor limitate și poate oferi estimări mai realiste ale probabilităților condiționate.

4.7 Calcularea probabilităților maximum a posteriori:

Probabilitatea maximum a posteriori reprezintă probabilitatea de a aparține unei anumite clase C_i dată o observație a caracteristicilor X_1, X_2, \dots, X_n . Formula generală este dată de:

$$P(C_i|X_1, X_2, \dots, X_n) = \frac{P(X_1, X_2, \dots, X_n|C_i) \cdot P(C_i)}{P(X_1, X_2, \dots, X_n)}$$

Folosind presupunerea de independență condiționată, putem rescrie numărătorul astfel:

$$P(X_1, X_2, \dots, X_n|C_i) = P(X_1|C_i) \cdot P(X_2|C_i) \cdot \dots \cdot P(X_n|C_i)$$

Înlocuind această formulă în prima vom obține ca probabilitatea maximum a posteriori va deveni:

$$P(C_i|X_1, X_2, \dots, X_n) = \frac{P(X_1|C_i) \cdot P(X_2|C_i) \cdot \dots \cdot P(X_n|C_i)}{P(X_1, X_2, \dots, X_n)}$$

4.8 Clasificare:

La final, vom avea doua probabilități, deoarece avem doua clase: spam și non-spam și cea care are probabilitatea cea mai mare, cu ajutorul votului majoritar, va fi rezultatul final.

4.9 Evaluare performanță:

La final, am implementat strategia de cross-validare Leave-One-Out pentru fiecare fisier, pentru evalua performanta algoritmului, ilustrat intr-un grafic de tip Pie Chart și totodata acuratețea obținută printr-un grafic cu bare, prezentate la punctele 5, respectiv 6.

5 Cross-validare Leave-One-Out

În procesul de evaluare a performanței modelelor de învățare automată, o componentă extrem de importantă o deține cross-validarea Leave-One-Out. Această strategie constă prin faptul că fiecare observație din setul de date este utilizată ca set de testare exact o dată, iar restul datelor sunt utilizate pentru antrenare.

Am calculat rata de eroare pentru fiecare iterație a LOO și am generat un grafic care ilustrează statisticile rezultatelor obținute.

Eroarea CVLOO obținută este de 1.2%. Aceasta indică faptul că, în medie, modelul nostru a făcut predicții incorecte în aproximativ 1.2% din cazuri în cadrul întregului set de date utilizat. Această valoare redusă sugerează o performanță solidă, foarte bună a modelului în procesul de învățare și generalizare pe datele disponibile.

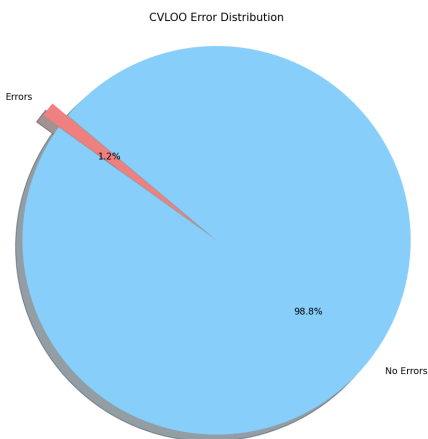


Figure 2: CVLOO pe întregul set de date

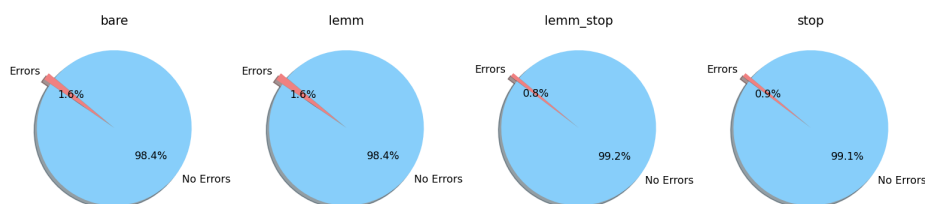


Figure 3: CVLOO împărțit pe cele 4 categorii

6 Acuratețea la testare

În cadrul acestui raport, performanța algoritmului nostru a fost evaluat și pe un set de date de testare folosind termenul de acuratețe. Acuratețea reprezintă un indicator esențial al calității modelului, măsurând raportul dintre predicțiile corecte făcute și totalul cazurilor.

Scopul acestei evaluări este de a demonstra eficacitatea algoritmului în realizarea predicțiilor pe datele nevăzute/ de testare. Astfel că, în urma rezultatelor, am obținut o acuratețe semnificativ mai bună decât dacă s-ar aplica o strategie trivială, cum ar fi ghicirea aleatoare(datul cu banul) sau alegerea când mereu rezultatul este aceeași clasă(spam sau non-spam).

Pentru a ilustra această performanță, am generat un grafic care evidențiază acuratețea obținută în comparație cu strategiile triviale menționate anterior, dar totodată am generat grafice aplicând și alți algoritmi de clasificare, pentru a întări ideea că, Bayes Naive se pliază cel mai bine pe acest tip de problemă.

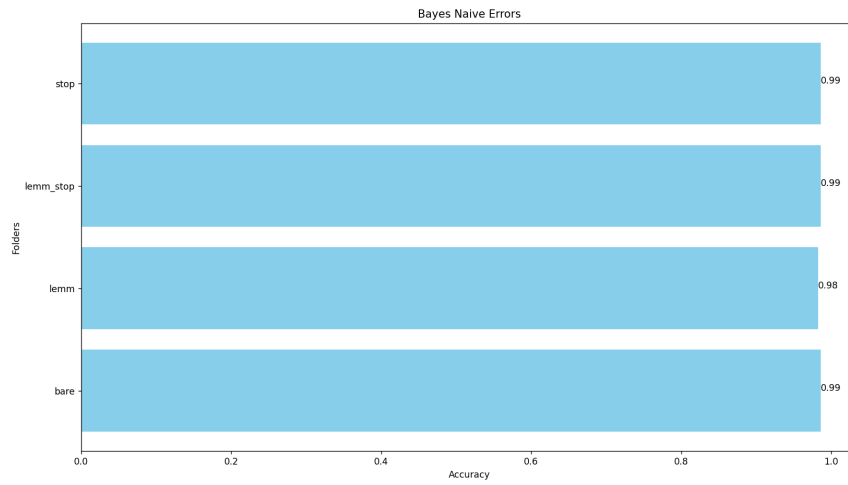


Figure 4: Acuratețea la testare pentru Bayes Naive

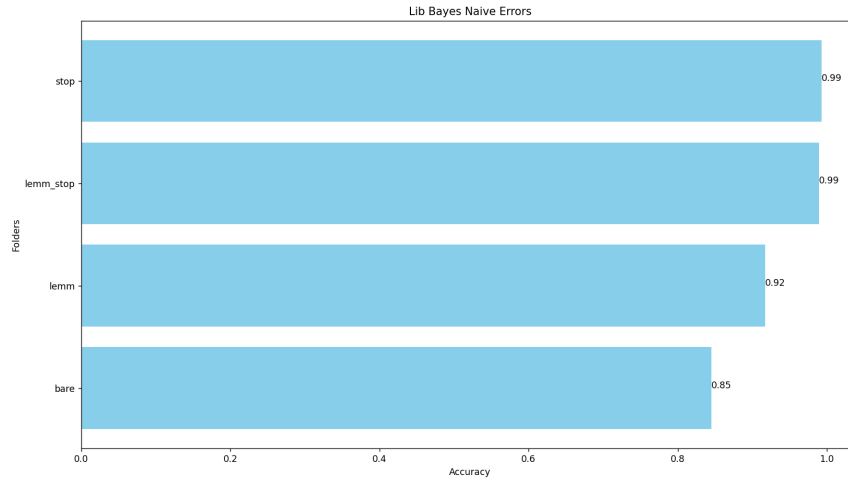


Figure 5: Acuratețea la testare pentru Bayes Naive(din librăria din Python)

Se poate observa faptul ca rezultatele celor două sunt foarte diferite, fiind totuși același algoritm în spate, însă acest lucru se datorează preprocesării datelor(pașii aplicați fiind prezentați la punctul **4.2**), astfel că acest lucru demonstrează încă o data, cât de important este acest pas pentru a antrena un model, în mod corect și pentru a obține rezultatele dorite.

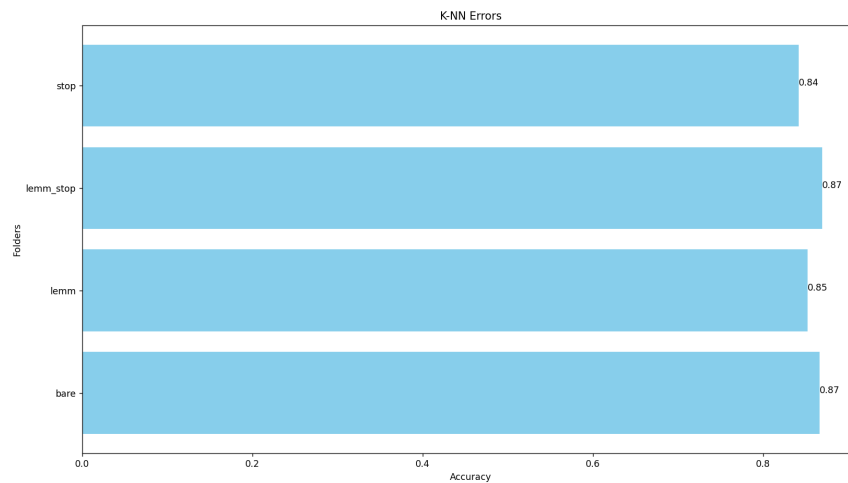


Figure 6: Acuratețea la testare pentru K-NN

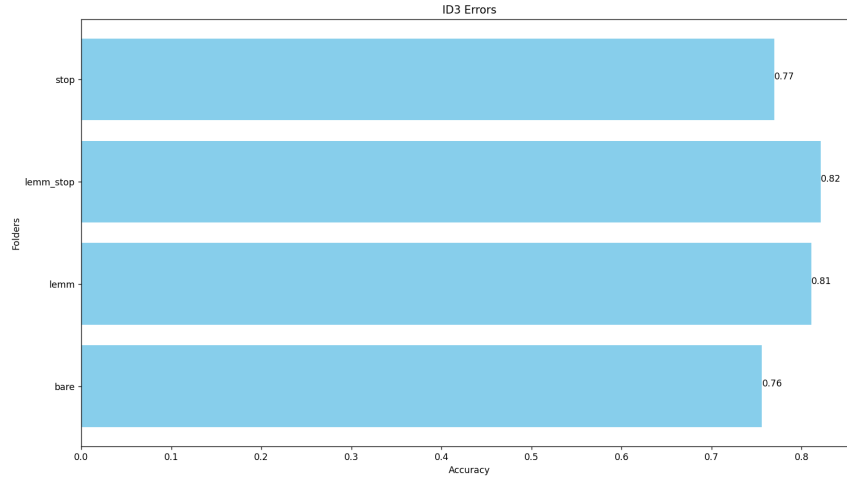


Figure 7: Acuratețea la testare pentru ID3

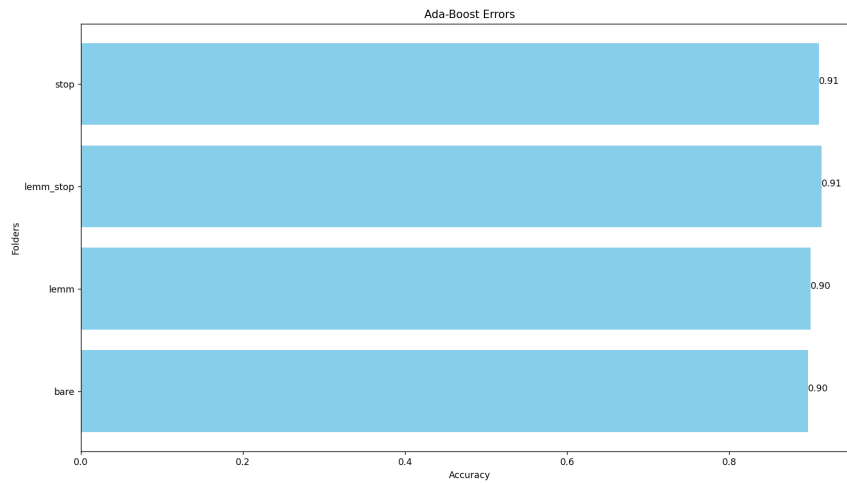


Figure 8: Acuratețea la testare pentru Ada-Boost

7 Concluzie

În urma analizei profunde a setului de date Ling-Spam, este evident că algoritmul de clasificare Bayes Naive s-a dovedit a fi o alegere remarcabilă și superioară în rezolvarea problemei de identificare a email-urilor spam. Acesta a depășit cu succes alte algoritme de clasificare, consolidându-și reputația ca fiind o soluție eficientă și robustă în contextul specific al acestei probleme.

Principala sa putere derivă din abordarea sa "naivă", care presupune independența condiționată între variabilele caracteristice, fapt ce se potrivește bine cu natura datelor din setul Ling-Spam. Modelul Bayes Naive a reușit să identifice corect pattern-urile asociate email-urilor spam, iar performanța sa a fost susținută de o acuratețe notabilă în predicțiile făcute.

Comparativ cu alte algoritme de clasificare, Bayes Naive a evidențiat o capacitate de generalizare superioară și o rezistență la overfitting, ceea ce îl face mai potrivit pentru gestionarea diversității și complexității specifice setului de date Ling-Spam.

References

Setul de date Ling-Spam: http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz

Teorie-Bayes Naive: <https://profs.info.uaic.ro/ciortuz/SLIDES/ml6.pdf>

Aplicații ale algoritmului Bayes Naive: <https://profs.info.uaic.ro/ciortuz/ML.ex-book/SLIDES/ML.ex-book.SLIDES.Bayes.pdf>

Preprocesarea datelor: <https://www.ijste.org/articles/IJSTEV11I11008.pdf>

CVLOO: <https://www.statology.org/leave-one-out-cross-validation/>

Alte informații legate de Bayes Naive: <https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>