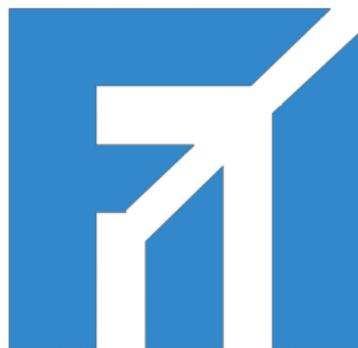


UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Detectarea știrilor false prin modele statistice și  
rețele neuronale: Bayes Naiv versus BERT**

propusă de

**Maria-Valeria Maxim**

**Sesiunea:** iulie, 2025

Coordonator științific

**Asist. Drd. Irimia Cosmin**

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IAȘI

**FACULTATEA DE INFORMATICĂ**

**Detectarea știrilor false prin modele  
statistice și rețele neuronale: Bayes  
Naiv versus BERT**

**Maria-Valeria Maxim**

**Sesiunea: iulie, 2025**

Coordonator științific

**Asist. Drd. Irimia Cosmin**

Anexa II

Avizat,

Îndrumător lucrare de licență,

Asist. Drd. Irimia Cosmin.

Data: .....

Semnătura: .....

## **Declarație privind autenticitatea conținutului lucrării de licență**

Subsemnata **Maxim Maria-Valeria** domiciliată în **România, jud. Botoșani, localitatea Lozna, sat Lozna, strada Salcămilor, nr. 8**, născută la data de **03 iunie 2004**, identificat prin CNP **60403070048**, absolventă a Universității „**Alexandru Ioan Cuza**” din **Iași**, Facultatea de Informatică specializarea **informatică**, promoția 2025, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul: **Detectarea știrilor false prin modele statistice și rețele neuronale: Bayes Naiv versus BERT** elaborată sub îndrumarea domnului **Asist. Drd. Irimia Cosmin**, pe care urmează să o susțin în fața comisiei este autentică, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea autenticității, consumând inclusiv la introducerea conținutului ei într-o bază de date în acest scop. Declar că lucrarea de față are exact același conținut cu lucrarea în format electronic pe care profesorul îndrumător a verificat-o prin intermediul software-ului de detectare a plagiatului.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: .....

Semnătura: .....

Anexa III

## **DECLARAȚIE DE CONSUMĂMÂNT**

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „**Detectarea știrilor false prin modele statistice și rețele neuronale: Bayes Naiv versus BERT**”, codul sursă al programelor și celealte conținuturi (grafice, multimedia, date de test, etc.) care însotesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea ”Alexandru-Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Maria-Valeria Maxim**

Data: .....

Semnătura: .....

# Cuprins

<b>Motivație</b>	<b>2</b>
<b>Introducere</b>	<b>3</b>
<b>1 Prezentarea domeniului</b>	<b>5</b>
1.1 Stirile false . . . . .	5
1.2 Lucrări din domeniu . . . . .	5
1.2.1 Fake News Detection on Social Media: A Data Mining Perspective [2] . . . . .	5
1.2.2 How to fine-tune BERT for text classification? [6] . . . . .	7
1.2.3 An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages[7] . . . . .	8
1.3 Aplicații similare . . . . .	9
1.3.1 PolitiFact . . . . .	9
1.3.2 Snopes . . . . .	9
1.3.3 Full Fact . . . . .	10
1.4 Tehnologii utilizate . . . . .	11
1.4.1 Python . . . . .	11
1.4.2 Frameworkuri și tooluri auxiliare . . . . .	11
<b>2 Noțiuni teoretice</b>	<b>12</b>
2.1 Noțiuni teoretice despre Bayes Naiv . . . . .	13
2.2 Noțiuni teoretice despre BERT . . . . .	15
<b>3 Implementare</b>	<b>18</b>
3.1 Analiza setului de date . . . . .	18
3.2 Bayes Naiv . . . . .	22
3.2.1 Preprocesarea datelor pentru Bayes Naiv . . . . .	23

3.2.2	Transformarea cuvintelor în vectori de caracteristici . . . . .	26
3.2.3	Antrenarea modelului Bayes Naiv . . . . .	27
3.2.4	Verificarea stabilității modelului . . . . .	28
3.3	BERT . . . . .	31
3.3.1	Preprocesarea datelor pentru BERT . . . . .	31
3.3.2	Împărțirea setului de date . . . . .	32
3.3.3	Tokenizarea datelor . . . . .	33
3.3.4	Antrenarea modelului . . . . .	34
3.3.5	Analiza antrenării . . . . .	36
3.4	Aplicație web . . . . .	37
<b>4</b>	<b>Evaluare</b>	<b>40</b>
4.1	Definirea metricilor de evaluare . . . . .	40
4.2	Rezultatele obținute pentru Bayes Naiv . . . . .	42
4.3	Rezultatele obținute pentru BERT . . . . .	44
4.4	Analiza comparativă a modelelor . . . . .	46
<b>Concluzii</b>		<b>49</b>
Implementări de viitor . . . . .	49	
Integrarea mai multor tipuri de știri . . . . .	49	
Integrarea analizei sentimentelor . . . . .	49	
Antrenarea pentru mai multe epoci . . . . .	50	
Integrarea caracterisiticilor non-textuale . . . . .	50	
Actualizarea continuă a modelului . . . . .	50	
<b>Bibliografie</b>		<b>51</b>

# Motivație

În era în care informația devine cea mai puternică unealtă utilizată pentru a ne ușura viața, aceasta devine de asemenea și cea mai puternică armă împotriva noastră. Deși acum avem multe tehnologii care sunt utilizate pentru a combate emailurile spam, pentru a clasifica documentele în diferite categorii, chiar și pentru traducerea automată, aceste *tooluri* pot fi folosite și în scop negativ, spre exemplu pentru a crea știri false și mai convingătoare decât era posibil până în prezent. Anumiți factori psihologici precum **confirmation bias** ( preferăm informațiile care ne confirmă credințele ), **negativity bias** ( știrile negative ne atrag mai mult atenția ) și **backfire effect** ( când ne este demonstrat că ne înșelăm preferăm să ne susținem în continuare ideea inițială decât să recunoaștem că am gresit ) influențează comportamentul utilizatorilor fără ca aceștia să fie conștienți. Aceste mecanisme sunt frecvent utilizate, contribuind la viralizarea știrilor false.

Așadar în contextul în care lumea devine din ce în ce mai susceptibilă la consumul de știri false prin intermediul rețelelor sociale, această lucrare de licență își propune să prezinte o comparație între două abordări diferite asupra detecției de știri false. Vom prezenta aceste metode din punct de vedere al complexității modelelor, al timpului necesar pentru antrenare, dar și al rezultatelor, oferind și o aplicație practică dezvoltată pentru detectarea știrilor false.

# Introducere

Descoperirea știrilor false reprezintă o problemă actuală, dezvoltându-se constant noi metode pentru rezolvarea acesteia.

WEF Global Risks Report este un raport anual publicat de World Economic Forum ( Forumul Economic Mondial ) care identifică și analizează cele mai importante riscuri globale pe termen scurt și lung. Acest raport se bazează pe sondaje realizate în rândul expertilor, liderilor de afaceri, guvernelor și societății civile din întreaga lume. Scopul este de a înțelege mai bine provocările majore care pot afecta economia globală, societatea, mediul și securitatea.

În acest raport dezinformarea se află pe locul 4 în cele mai importante riscuri globale, mai mult de atât este menționat că acesta va fi pe locul întâi în anul 2027. De asemenea, este specificat faptul că proliferarea de conținut fals sau conținut înselător poate duce la complicații în mediul geopolitic. Astfel dezinformarea reprezintă o modalitate accesibilă prin care pot fi afectate intențiile alegătorilor ( setul de date utilizat de lucrarea noastră pune accent pe acest subiect ), poate semăna îndoiești în cadrul publicului cu privire la ceea ce se întâmplă în cazul unei situații de criză ( un exemplu relevant este reprezentat de răspândirea știrilor false în timpul pandemiei de COVID-19[1], acestea au dus la evitarea vaccinării și la utilizarea de medicamente care nu aveau efecte dovedite științific contribuind în final la o rată și mai ridicată a deceselor ). De asemenea, avansarea în domeniul inteligenței artificiale a condus la crearea de conținut fals mult mai convingător, nu doar textual, dar și video, prin imagini sau voce.

Printre primii algoritmi folosiți pentru clasificarea textului se numără **Bayes Naiv**. Bayes Naiv este un clasificator probabilist ce se bazează pe prezumția de independentă a caracteristicilor. Deși pare contraintuitiv ca acest tip de clasificator să ofere rezultate bune în cazul datelor textuale deoarece în general acestea nu sunt independente ci sunt strâns legate contextual, s-a demonstrat că Bayes Naiv oferă rezultate surprinzătoare de bune având în vedere simplitatea arhitecturii modelului.

Pe de altă parte, **transformatorii bidirectionali** au o înțelegere mai bună a relațiilor semantice între cuvinte, datorită capacitatea lor avansată de înțelegere a limbajului natural. Aceștia înțeleg cuvintele dintr-o propoziție ținând cont de contextul complet, atât ce vine înainte cât și ce vine după un cuvânt. **BERT** a oferit rezultate remarcabile în cazul diferitelor task-uri de NLP. Acesta este antrenat pe corpusuri mari de texte, ceea ce îl face să recunoască particularități mai subtile ale limbajului (ironia, stilul lexical), pe când Bayes Naiv se bazează doar pe frecvența cuvintelor.

Vom analiza rezultatele pe care aceste abordări le au în cazul clasificării binare a știrilor false. Vom prezenta ce îmbunătățiri putem aduce în cazul folosirii lui Bayes Naiv prin ajustarea diferiților parametri ai vocabularului, dar și al parametrului care controlează netezirea caracteristicilor. Iar în cazul lui BERT, vom face finetuningul pe datele noastre de antrenament și vom analiza rezultatele pe care acesta le oferă comparativ cu Bayes Naiv.

# Capitolul 1

## Prezentarea domeniului

### 1.1 Știrile false

Înainte de toate, este esențial să înțelegem ce sunt știrile false. Deși nu există o definiție universal acceptată, acest fenomen a fost abordat și interpretat în moduri diferite, în funcție de contextul istoric și social. Pentru prima dată termenul de *stiri false* a fost folosit în 1890 când știrile senzaționaliste din ziare erau destul de comune. În anul 2017 termenul "fake news" ( știri false ) a fost cuvântul anului în British Collins Dictionary, unde a fost definit astfel : "*informații false, adesea senzaționale, diseminate sub pretextul știrilor*".

### 1.2 Lucrări din domeniu

În această secțiune vom discuta despre lucrările științifice care ne-au influențat în timpul cercetării și care ne-au conturat dorința de a aborda această problemă, tehnologiile utilizate pentru a atinge performanțe excelente în domeniul de procesare a limbajului natural și despre alte aplicații care rezolvă problema noastră.

#### 1.2.1 Fake News Detection on Social Media: A Data Mining Perspective [2]

Motivația lucrării noastre a fost conturată de acest articol. Articolul menționat oferă un cadru analitic clar asupra impactului social generat de consumul de știri false difuzate prin intermediul rețelelor sociale.

Platformele de social media reprezintă un mediu accesibil, rapid și gratuit care facilitează distribuirea și consumul de informație. În contrast, sursele traditionale de informare, cunoscute pentru credibilitatea conținutului, cum este cazul agenției Reuters, au început să limiteze utilizatorilor accesul gratuit. Începând cu anul 2024, accesul extins la articolele Reuters presupune deținerea unui abonament, ceea ce îl exclude din categoria surselor de știri complet gratuite.

Reuters nu este un caz izolat, alte publicații, precum The Wall Street Journal sau The New York Times, au adoptat politici similare. Această tendință contribuie la explicarea motivelor pentru care tot mai mulți utilizatori aleg să consume știri din surse informale precum social media, în detrimentul surselor sigure, dar cu acces restricționat. De asemenea, social media permite distribuirea facilă a informațiilor comparativ cu știrile tradiționale și oferă șansa utilizatorilor să discute deschis despre subiectul acestora, creându-se uneori și efectul de *echo chamber*, un spațiu în care utilizatorii sunt expuși doar la opinii asemănătoare cu opiniilor lor, limitând accesul la alte puncte de vedere diferite de cele personale.

În acest articol sunt prezentate strategiile psihologice pe care se bazează răspândirea știrilor false. Este important să înțelegem și această perspectivă deoarece ne ajută să identificăm nu doar modul în care se propagă știrile în rețelele sociale, ci și de ce oamenii aleg să le creadă, să le distribuie și să reacționeze emoțional la acestea.

Factorii psihologici care ne fac să credem că acele informații sunt adevărate sunt următorii:

- **Naive Realism** : Utilizatorii tind să credă că doar percepția lor despre realitate este corectă.
- **Confirmation bias** : Preferăm să citim informații care să ne susțină ideile deja înrădăcinate.

Aceste tendințe psihologice fac descoperirea știrilor false de către consumatori foarte dificilă și conturează necesitatea de a avea o aplicație care să facă această decizie automat. Deși expertiza umană reprezintă *the gold standard*, o aplicație de detectie automată care să fie doar supervizată, poate fi benefică pentru limitarea răspândirii știrilor false.

### 1.2.2 How to fine-tune BERT for text classification? [6]

*How to fine-tune BERT for text classification?* [6] a reprezentat un baseline puternic pentru antrenarea modelului nostru de recunoaștere al știrilor false utilizând BERT. Aceasta a jucat un rol important în alegerea parametrilor pentru antrenarea modelului, dar și pentru a înțelege modul în care este construită arhitectura acestuia.

Preantrenarea modelelor pe un corpus mare textual în prealabil este benefică pentru clasificarea textului și pentru sarcinile de procesare a limbajului natural. Aceasta evită antrenarea unui model de la 0, sarcină care este foarte costisitoare. BERT este bazat pe arhitectura de tip *Bidirectional Multi-layer Transformer* și a fost antrenat pe un corpus mare de text cu sarcina de predicție a următoarei propoziții (*next sentence prediction*) și pentru modelarea limbajului mascat (*masked language modeling*).

BERT conține un encoder cu 12 blocuri Transformer, 12 self-attention heads și un hidden size de 768 de dimensiuni și folosește ca input doar secvențe de lungime maximă de 512 tokeni, de asta este important ca în procesul de tokenizare să păstrăm această dimensiune maximă a textului.

De asemenea, modelul va returna reprezentarea vectorială pentru fiecare *token* din input. Acest vector conține informația despre sensul *tokenului* în context, luând în considerare fraza întreagă. Secvența de 512 tokeni are unul sau mai multe segmente, începutul acesteia este marcat mereu de tokenul [CLS] care conține informația întregii secvențe, iar *tokenul* [SEP] este utilizat pentru separarea secvențelor. Pentru clasificarea textului, peste vectorul ascuns h, asociat *tokenului* [CLS] este adăugat un simplu clasificator **softmax**, rolul acestuia este să mapeze rezultatul rețelei pentru **task-ul** specific. În cazul nostru, clasificatorul softmax mapează rezultatul în 2 probabilități care sunt utilizate pentru predicția clasei corecte real/fals. Pentru fine-tune vom antrena parametrii lui BERT cât și matricea de greutăți W, specifică domeniului prin maximizarea log-probabilității clasei corecte.

Pentru adaptarea lui BERT la sarcina noastră specifică este important să luăm în considerare următoarele:

- Lungimea secvențelor mai mari de 512 tokeni
- Selectarea *layerului* cel mai important pentru clasificarea pe domeniul nostru (BERT are 12 *layer*e și un *pooling layer*)
- Problema *overfittingului* (aceasta este rezolvată prin adaptarea ratei de învățare)

Pentru a combate "Catastrophic forgetting" ( cunoștințele anterioare ale modelului sunt sterse când încercăm să îl specializăm pe sarcina noastră ) este necesar să folosim o rată de învățare mică, precum **2e-5**. Folosind o rată de învățare mai mare nu s-ar ajunge la convergență. În această lucrare este menționat ca ar fi cel mai benefic să antrenăm cel puțin ultimul *layer*. Experimentând, am observat că dacă am face acest lucru, din prima epocă ar exista semne de *overfitting*, aşadar am preferat să înghetăm toate *layerele* și să utilizăm doar *layerul pooler* pentru *finetuning*.

### 1.2.3 An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages[7]

Această lucrare reprezintă una dintre lucrările fundamentale în domeniul descoperirii filtrării spamului și reprezintă un punct de plecare pentru algoritmul dezvoltat și analizat de noi, deși a fost necesar să adaptăm anumiți parametrii pentru setul nostru de date.

În articol, autorii analizează performanța lui Bayes Naiv comparativ cu utilizarea de **cuvinte cheie** precum "be over 21", "only \$" utilizate ca metode pentru detectarea spamului. În analiza performanței modelului a fost utilizat un *threshold* care măsoară cât de negativă este pierderea clasificării unui email legitim ca fiind spam, pentru a introduce o decizie bazată pe risc.

Printre tehniciile de preprocesare a textului, s-au numărat : **lematizarea** cuvintelor, **utilizarea unei liste de stop-words**, **eliminarea atașamentelor** și a **tagurilor HTML**. Aceștia au analizat modificările performanței aduse de **lematizarea textului** și eliminarea cuvintelor frecvente, ajungând la concluzia că lematizarea a dus la îmbunătățirea performanței, în schimb utilizarea listei de *stop-words* nu avut un efect vizibil probabil din cauza metodei de selecție a atributelor, care nu alegea caracteristicile frecvente.

De asemenea, aceștia au utilizat tehnica de **10-fold cross-validation** ( aceasta presupune împărțirea setului de date în 10 submultimi denumite *folduri*, mai apoi modelul este antrenat pe toate submultimile utilizând o singură submultime pentru evaluarea modelului ). Această metodă permite o analiză mai robustă a performanței, comparativ cu utilizarea unui singur set de date împărțit în setul de antrenare și în setul de testare, deoarece pentru cea din urmă rezultatele pot fi diferite în funcție de modul în care sunt împărțite datele.

La finalul experimentelor conduse de autori, s-a concluzionat că utilizarea lui Ba-

yes Naiv a avut rezultate superioare comparativ cu metoda clasică de filtrare utilizând liste de cuvinte cheie.

## 1.3 Aplicații similare

### 1.3.1 PolitiFact

**PolitiFact**<sup>1</sup> este un website dedicat verificării afirmațiilor oficialilor aleși și ale altor persoane de pe platforma lor. Aceștia utilizează un sistem de *rating* denumit Truth-O-Meter care clasifică afirmațiile în "True," "Mostly True," "Half True," "Mostly False," "False," și "Pants on Fire." Procesul de clasificare este făcut prin căutări în baze online, consultarea altor surse și utilizarea expertilor pentru a stabili veridicitatea informațiilor.

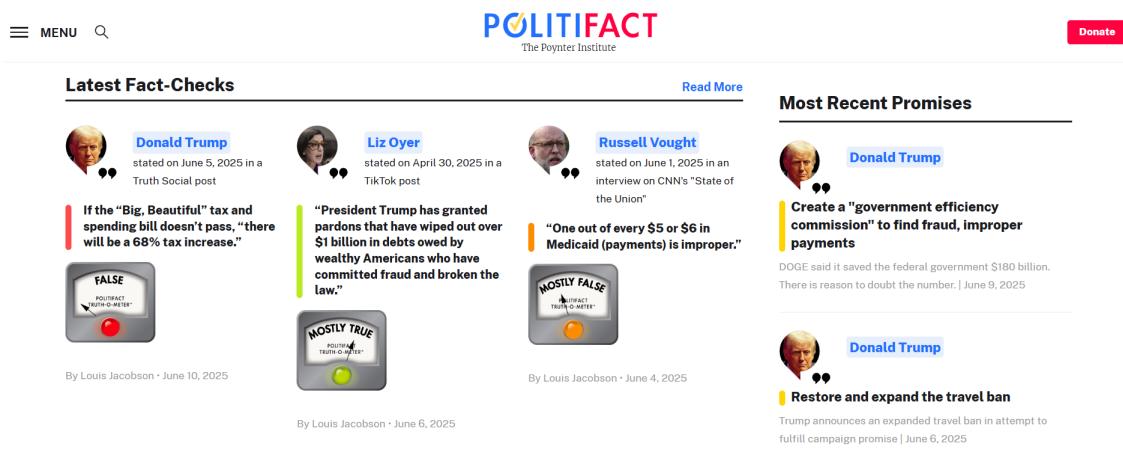


Figura 1.1: PolitiFact

### 1.3.2 Snopes

**Snopes**<sup>2</sup> se bazează pe expertiza umană a jurnaliștilor și a cercetătorilor pentru verificarea știrilor. Aceștia utilizează instrumente de inteligență artificială<sup>3</sup> dar nu în scopul clasificării știrilor ci pentru îmbunătățiri experiența utilizatorilor prin generarea de rezumat.

<sup>1</sup><https://www.politifact.com/>

<sup>2</sup><https://www.snopes.com/>

<sup>3</sup><https://www.calpoly.edu/news/cal-poly-and-snopes-develop-ai-service-fact-check-your-questions>

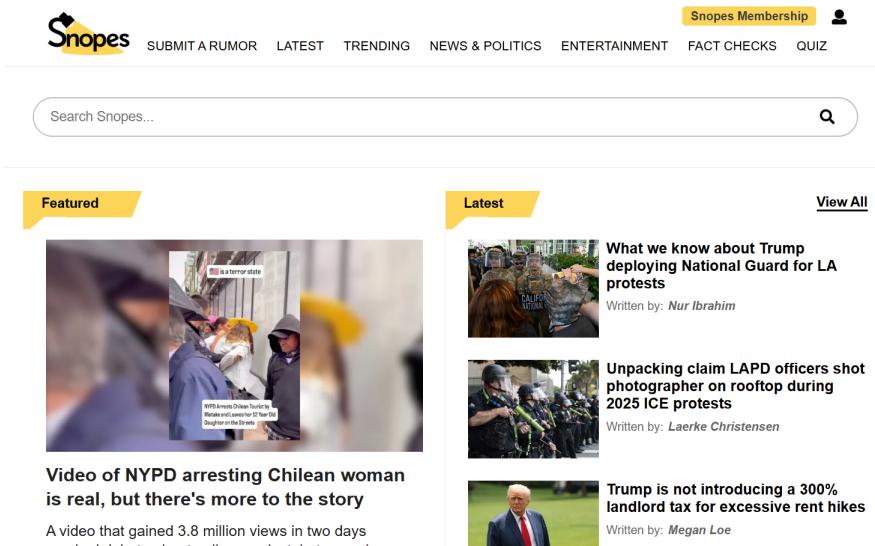


Figura 1.2: Snopes

### 1.3.3 Full Fact

**Full Fact**<sup>4</sup> combină atât expertiza umană cât și sisteme simple de NLP, precum prezicerea potrivirii/lipsei potrivirii propozițiilor, utilizând vectorizarea propozițiilor și analiza entităților. În prezent folosesc un instrument de AI generativ.

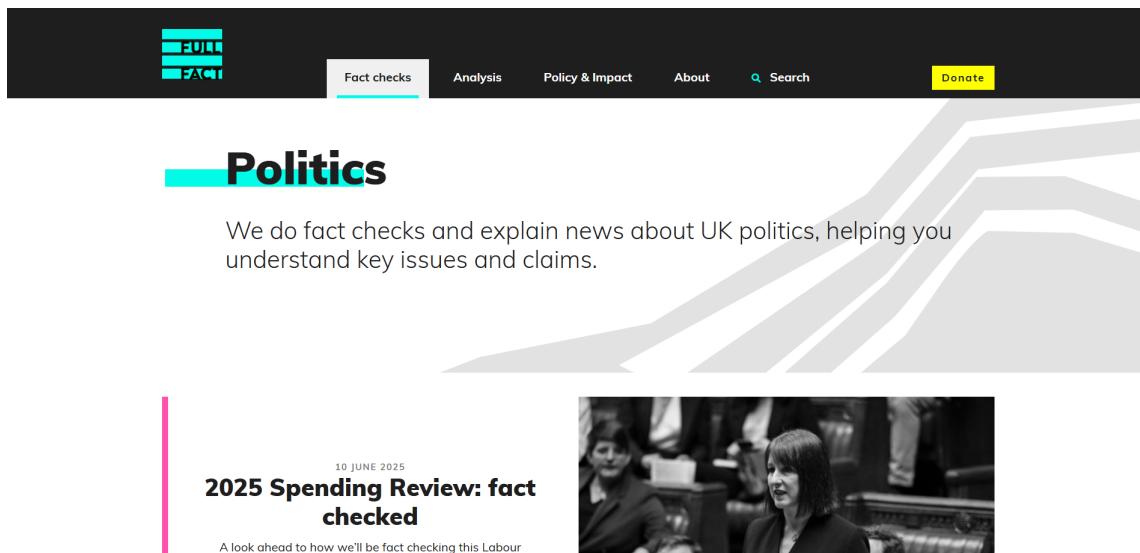


Figura 1.3: Full Fact

Aplicația noastră aduce un plus de valoare datorită automatizării complete a fluxului decizional folosind modele avansate de preprocesare a limbajului natural precum BERT, împreună cu un clasificator bayesian. Snopes și Politifact se bazează în princi-

<sup>4</sup><https://fullfact.org/politics/>

pal pe expertiza umană pentru verificarea știrilor, iar FullFact utilizează doar sisteme simple de NLP, pe când aplicația dezvoltată de noi integrează atât detectarea automată a știrilor false de tip propagandă, cât și rezumarea automată a conținutului. Aceasta reprezintă un instrument eficient mai ales în momentul de față, când știrile false se răspândesc foarte rapid prin intermediul rețelelor sociale.

## 1.4 Tehnologii utilizate

### 1.4.1 Python

Limbajul de programare folosit pentru realizarea proiectului a fost **Python**. Acest limbaj are un suport foarte bun pentru procesarea limbajului natural și învățare automată, dar și pentru integrarea cu backendul. Cu ajutorul acestuia putem integra prompt modelele dezvoltate cu partea de backend.

### 1.4.2 Frameworkuri și tooluri auxiliare

#### Hugging face

**Hugging Face** este o librărie open-source pentru python care are numeroase modele Transformer preantrenate pentru diverse *task-uri*. Am ales această bibliotecă deoarece ne-a permis să implementăm eficient modelul BERT și să îl adaptăm pentru clasificarea știrilor false. Hugging Face oferă suport pentru tokenizarea avansată a datelor și metode simple de finetuning a modelului clasic.

#### Flask

**Flask** este un micro-framework web pentru Python care oferă un mediu de lucru flexibil. Am ales flask pentru partea de backend deoarece ne-a permis să dezvoltăm rapid un server REST API pentru a gestiona cererile către cele două modele și răspunsurile acestora legate de textul știrii.

#### React

**React** este o bibliotecă JavaScript folosită pentru crearea interfețelor interactive și dinamice. Am ales React datorită modului eficient de gestionare al DOM-ului virtual care permite actualizări fluide a paginii web.

# Capitolul 2

## Noțiuni teoretice

În această lucrare ne propunem să abordăm problema clasificării știrilor în două categorii: știri reale și știri false. Procesul de **clasificare** constă în atribuirea unei observații, caracterizată printr-un set de trăsături extrase, unei clase dintr-un ansamblu finit de clase posibile. În cazul de față, fiind vorba despre doar două categorii, ne aflăm în față unei probleme de **clasificare binară**.

Pentru a soluționa această sarcină, vom utiliza metode de **învățare supervizată** (sistemul este antrenat pe baza unui set de date pentru care este cunoscut răspunsul corect sub forma etichetei asociată fiecărei observații). Prin contrast, în cadrul **învățării nesupervizate**, algoritmii operează doar pe baza datelor de intrare, fără a avea acces la etichetele corespunzătoare, urmărind identificarea unor structuri sau tipare latente în date (spre exemplu algoritmii de clustering).

Am optat pentru utilizarea a doi algoritmi complementari, respectiv Bayes Naiv și BERT pentru a sublinia diferențele între modelele statistice și cele bazate pe rețele neuronale în cazul detectării știrilor false. Fiecare dintre cei doi algoritmi prezintă o abordare particulară asupra modului în care datele sunt preprocesate și **tokenizate**, a vitezei de antrenare și a resurselor computaționale necesare pentru acest proces, dar și al rezultatelor pe care le-am obținut utilizându-i. Aceste diferențe precum și scenariile de utilizare în care am înclina spre un model specific vor fi prezentate în capitoalele următoare. În acest capitol ne vom concentra pe noțiunile teoretice necesare pentru a înțelege diferențele dintre cele 2 abordări în clasificarea știrilor false.

## 2.1 Notiuni teoretice despre Bayes Naiv

Bayes Naiv este un **clasificator bayesian** care se bazează pe **independență condițională** dintre trăsături. Desi face această presupunere a fost dovedit că acesta oferă rezultate surprinzător de bune în clasificarea textului.

În acest caz documentele sunt reprezentate sub forma unui **bag of words**, o mulțime neordonată de cuvinte în care poziția lor este ignorată păstrându-se doar frecvența cuvintelor. În figura 2.1 putem observa o știre transpusă în această formă: fiecare cuvânt este extras împreună cu frecvența acestuia.

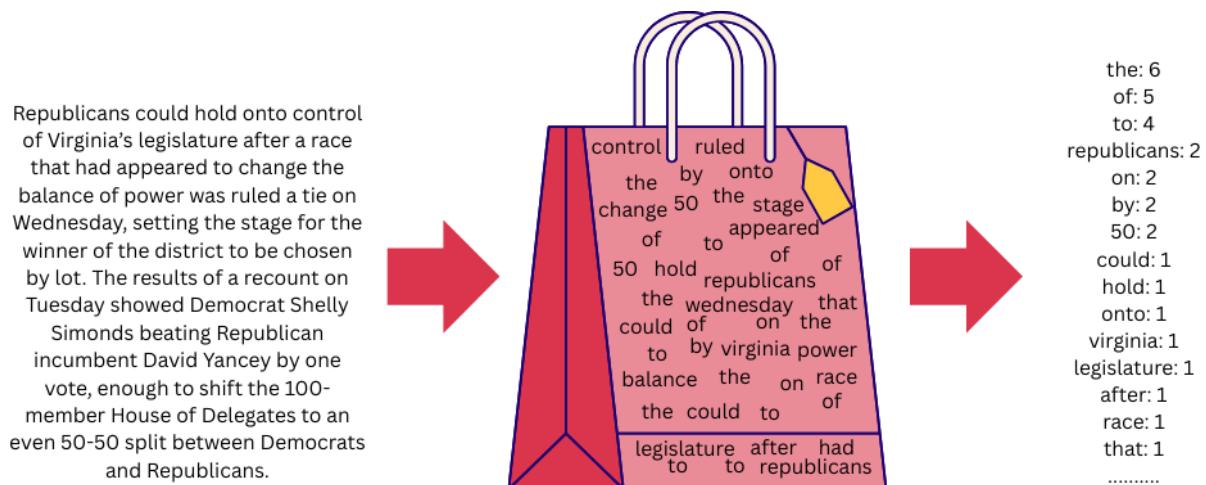


Figura 2.1: Reprezentarea unui document sub forma *bag of words*

Un alt aspect important al algoritmului este că acesta descrie un **clasificator probabilistic** și returnează clasa care are probabilitatea a posteriori maximă fiind dat documentul. Pentru a calcula această probabilitate utilizăm formula 2.1:

$$\hat{c} = \arg \max_{c \in \mathcal{C}} P(c | d) \quad (2.1)$$

$$P(c | d) = \frac{P(d | c) \cdot P(c)}{P(d)} \quad (2.2)$$

Utilizând regula lui Bayes pentru calcularea probabilității condiționate, prezentată în ecuația 2.2, putem substitui  $P(c | d)$  această formulă în ecuația initială. Pentru fiecare clasă numitorul  $P(d)$  va fi același deoarece ne întrebăm care este cea mai probabilă clasă pentru documentul  $d$ , astfel îl putem elmina. Astfel, ajungem la următoarea formulă:

$$\hat{c} = \arg \max_{c \in \mathcal{C}} P(d | c) \cdot P(c) \quad (2.3)$$

Presupunând că documentul  $d$  este reprezentat printr-o mulțime de cuvinte sau trăsături  $x_1, x_2, \dots, x_n$ , și aplicând ipoteza de independentă condiționată a trăsăturilor (de aici provine naivitatea), rezultă:

$$P(d | c) = \prod_{i=1}^n P(x_i | c) \quad (2.4)$$

Astfel, formula finală a clasificatorului devine:

$$\hat{c} = \arg \max_{c \in \mathcal{C}} P(c) \cdot \prod_{i=1}^n P(x_i | c) \quad (2.5)$$

### **Etapele algoritmului:**

1. Calculează probabilitatea *a priori* pentru fiecare clasă:  $P(c)$
2. Calculează probabilitățile condiționate:  $P(x_i | c)$  pentru fiecare trăsătură  $x_i$
3. Pentru un document nou, evaluează funcția de decizie:

$$\hat{c} = \arg \max_{c \in \mathcal{C}} P(c) \cdot \prod_{i=1}^n P(x_i | c)$$

4. Documentului îi atribuie clasa  $\hat{c}$  rezultată

Un caz special ce trebuie să fie luat în considerare este întâlnirea în setul de antrenament a cuvintelor specifice unei singure clase, spre exemplu dacă întâlnim cuvântul "violence" doar în știrile false, aceasta va avea probabilitatea zero, ducând automat la probabilitatea întregii clase pentru acel document să fie zero (probabilitățile pentru fiecare cuvânt fiind înmulțite). Pentru a rezolva această problemă putem utiliza **netezirea Laplace**.

$$\hat{P}(x_i | c) = \frac{\text{count}(x_i, c) + 1}{\sum_{x \in V} (\text{count}(x, c) + 1)} = \frac{\text{count}(x_i, c) + 1}{\sum_{x \in V} (\text{count}(x_i, c) + |V|)} \quad (2.6)$$

În cazul cuvintelor care nu au apărut în setul de date de antrenare dar apar în cel de testare, soluția este să le ignorăm.

Algoritmul complet este prezentat în figura 2.2, acesta fiind preluat din capitolul 4 din cartea "Speech and text recognition" [4].

```

function TRAIN NAIVE BAYES(D,C) returns  $V$ ,  $\log P(c)$ ,  $\log P(w|c)$ 

for each class  $c \in C$            # Calculate  $P(c)$  terms
     $N_{doc}$  = number of documents in D
     $N_c$  = number of documents from D in class c
     $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$ 
     $V \leftarrow$  vocabulary of D
     $bigdoc[c] \leftarrow \text{append}(d)$  for  $d \in D$  with class  $c$ 
        for each word  $w$  in  $V$            # Calculate  $P(w|c)$  terms
             $count(w,c) \leftarrow$  # of occurrences of  $w$  in  $bigdoc[c]$ 
             $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$ 
    return  $logprior$ ,  $loglikelihood$ ,  $V$ 

function TEST NAIVE BAYES( $testdoc$ ,  $logprior$ ,  $loglikelihood$ ,  $C$ ,  $V$ ) returns best  $c$ 

for each class  $c \in C$ 
     $sum[c] \leftarrow logprior[c]$ 
    for each position  $i$  in  $testdoc$ 
         $word \leftarrow testdoc[i]$ 
        if  $word \in V$ 
             $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$ 
    return  $\text{argmax}_c sum[c]$ 

```

Figura 2.2: Algoritmul Naiv Bayes folosind netezirea Laplace [4]

## 2.2 Noțiuni teoretice despre BERT

BERT este un model bazat pe Transformatori dezvoltat de Google. **Transformatorii** sunt **rețele neuronale** care au o structură specifică ce include un mecanism denumit **attention**. BERT utilizează mecanismul de *self-attention* (ia în considerare și tokenul curent), acesta poate fi gândit ca modalitatea prin care integrăm informații din *tokenurile* ce încunjoară cuvântul, ajutându-le să relateze între ele.

Mecanismul de **attenție** este cel care face transformatorii speciali, acesta măsoără importanța cuvintelor dintr-o anumită secvență și combină reprezentările acestora. **Atenția** în acest context vine sub forma unui *attention head*, care utilizează 3 matrici: query (elementul curent care este comparat cu inputurile precedente), key (un element precedent care este comparat cu elementul curent pentru a determina similaritatea).

tea ) și value ( valoarea unui element precedent care este ponderată și însumată pentru a calcula rezultatul elementului curent ) . Ne putem gândi la acestea ca utilizând o întrebare ( query ) , cautăm indicii cuvintelor ( key ) care ne răspund la aceasta pentru a putea extrage informații ( value ) care să ne ajute să construim reprezentarea contextuală a tokenurilor. Un alt element important al arhitecturii lui BERT este reprezentat de *positional encoding* care ne ajută să ținem cont de ordinea cuvintelor. Astfel utilizând mecanismul de *self-attention* și *positional embeddings* vom obține **embedding-uri contextuale** care țin cont de poziția cuvintelor și de relațiile dintre acestea. *Embedding-urile contextuale* pot ajuta la clusterizarea documentelor similare.

BERT ( Bidirectional Encoder Representation from Transformers) reprezintă un avans major în procesarea limbajului natural. Acesta are acces la un context bidirecțional, spre deosebire de modelele anterioare, autoregresive precum GPT-1, care erau antrenate pe predicția următorului cuvânt dintr-o frază. BERT analizează simultan textul care vine din ambele direcții încercând să prezică cuvântul din mijloc. Acesta a fost antrenat pe 2 *task-uri*:

- **Mascarea cuvintelor** → în procesul de antrenare 15% din tokeni au fost manipulați de procedura de mascare, apoi modelul a fost antrenat să ghicească forma corectă a acestor tokeni.
- **Predicția propoziției următoare** → având la dispoziție 2 propoziții modelul trebuie să prezică dacă acestea sunt adiacente.

Putem utiliza cunoștințele anterioare ale modelului și să îl specializăm pentru recunoașterea știrilor false prin **finetuning**. Datorită **finetuning-ului**, complexitatea arhitecturii lui BERT, cu numeroase *layere* Transformer și cei 110 milioane de parametri, nu mai reprezintă un impediment în utilizarea sa practică pe o sarcină specifică. Cunoștințele necesare pentru adaptarea modelului au fost preluate din lucrarea "How to fine-tune BERT for text classification?" [6] și din Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, Daniel Jurafsky and James H. Martin [4].

Pentru clasificarea secvențelor, reprezentăm întreaga intrare care urmează să fie clasificată printr-un singur vector. Pentru BERT este necesar să adăugăm un nou token unic [CLS], este adaugat ca prefix la începutul fiecărei propoziții atât în faza de antrenare cât și în cea de encodare. Vectorul de ieșire pentru intrarea tokenului [CLS] este

utilizat ca intrare pentru o rețea neuronală care are scopul de a face decizia clasificării știrilor.

**Finetuningul** unui clasificator pentru detecția știrilor false implică învățarea un set de weight-uri  $W_c$ , pentru a mapea vectorul de output pentru tokenul [CLS] - $h_{cls}$ - la un set de scoruri pentru clasele real/fals.

Pentru a clasifica o instanță, trece textul știrii prin modelul preantrenat pentru a genera tokenul  $h_{cls}$ , pe care îl vom înmulți cu  $W_c$ , iar vectorul rezultat va fi trecut printr-un strat softmax pentru a mapea rezultatele într-un set de probabilități corespunzătoare celor 2 clase.

Finetuningul valorilor din  $W_c$  necesită un set de date de antrenament etichetate. Antrenamentul se desfășoară în mod obișnuit, cross-entropia între ieșirea softmax și răspunsul corect este folosită pentru a învăța aceste weight-uri. O diagramă completă a procesului de finetuning este prezentată în figura 2.3.

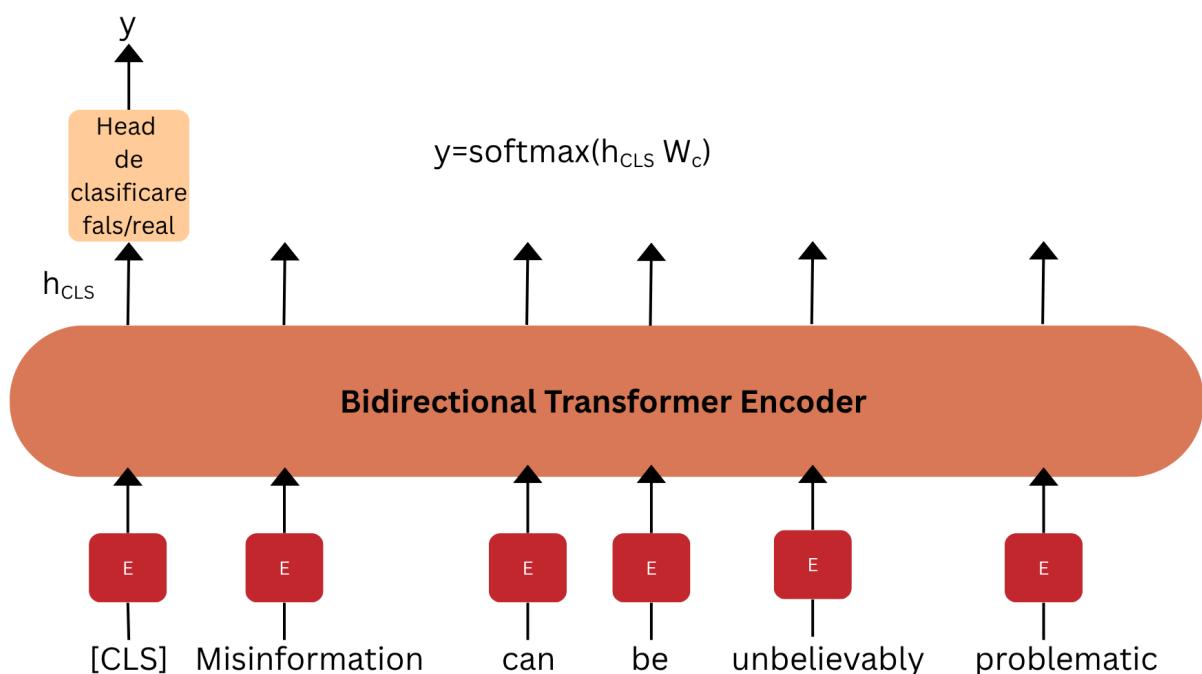


Figura 2.3: Clasificarea secvențelor folosind BERT. Vectorul de ieșire pentru tokenul [CLS] reprezintă intrarea unui clasificator simplu (în cazul nostru o rețea neuronală).

Având în vedere modul de antrenare al modelului BERT și capacitatea acestuia de a genera *embedding-uri contextuale*, spre deosebire de abordarea lui Bayes Naiv care se bazează doar pe frecvența cuvintelor este de așteptat ca acesta să ofere rezultate superioare în clasificarea știrilor.

# Capitolul 3

## Implementare

### 3.1 Analiza setului de date

Setul de date utilizat pentru a conduce experimentele asupra celor doi clasificatori este fake-and-real-news-dataset<sup>1</sup> disponibil public pe platforma Kaggle. Acest set de date conține știri din perioada 2015-2018 în limba engleză și este utilizat în cercetarea legată de detecția automată a știrilor false. Conține articole extrase din surse cunoscute pentru publicarea știrilor fie reale, fie false. Sursa acestor date este ISOT Fake News Dataset<sup>2</sup>, un set de date dezvoltat de University of Victoria's ISOT (Information Security and Object Technology) Research Lab din Canada.

Articolele reale au fost colectate de pe cunoscutul site de știri reuters.com<sup>3</sup>, iar cele false au fost colectate de pe diferite site-uri care sunt cunoscute pentru răspândirea știrilor false, dar despre care nu există informații în documentația datelor.

Pentru a înțelege mai bine structura și caracteristicile *datasetului* am realizat o analiză exploratorie asupra sa. Acest pas este crucial deoarece rezultatele clasificatorilor utilizati sunt puternic influențate de calitatea datelor de antrenare. În această etapă ne vom face o idee generală asupra distribuției știrilor, categoriile cărora aparțin și asupra vocabularului specific.

---

<sup>1</sup><https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset>

<sup>2</sup><http://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/>

<sup>3</sup><https://www.reuters.com/>

## Numarul de stiri reale si false

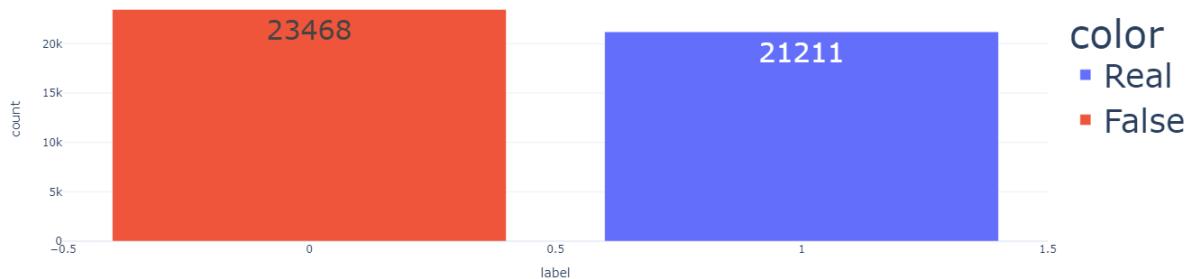


Figura 3.1: Distribuția știrilor în funcție de eticheta real/fals

În diagrama 3.1 putem observa că avem o diferență foarte mică între numărul știrilor reale și false, însă aceasta nu este destul de mare încât să ne afecteze modelul (5%) sau să fie necesară augmentarea datelor.

Tipurile de stiri false

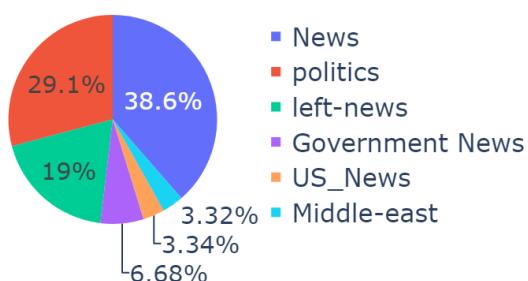


Figura 3.2: Tipurile de știri false

Tipurile de stiri reale

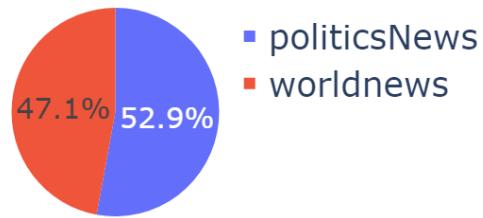


Figura 3.3: Tipurile de știri reale

În diagramele 3.2 și 3.3 sunt prezentate distribuțiile domeniilor din care fac parte atât știrile reale cât și cele false. Putem remarcă că știrile false provin din domenii mai variate comparativ cu știrile reale, oferind modelului un vocabular mai diversificat. Aceasta diversificare ar putea să fie și negativă deoarece, este posibil ca știrile respective să nu respectă un tipar clar pe care să îl poată învăța modelele. În schimb, știrile reale sunt concentrate doar pe două domenii principale și oferă modelului un set mai restrâns de contexte, ceea ce poate influența capacitatea de generalizare a acestuia, dar este posibil ca acestea să fie scrise în aceeași manieră ceea ce ar fi benefic deoarece ar oferi caracteristici comune pe care modelul le-ar putea valorifica. Vom analiza influența acestora în secțiunea de evaluare.

O diagramă de tip **word cloud** este o reprezentare vizuală a frecvenței cuvintelor

într-un text. Acest tip de diagramă ne ajută să analizăm în linii mari textul pentru a observa eventualele particularități comune ale știrilor reale și ale știrilor false. În figurile 3.4 și 3.5 se poate observa distribuția termenilor în contextul celor două clase. Diagramele au fost realizate după aplicarea pașilor de preprocesare descriși în secțiunea următoare.

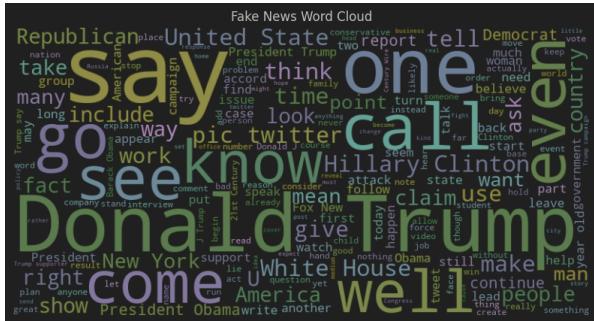


Figura 3.4: Word Cloud știri false

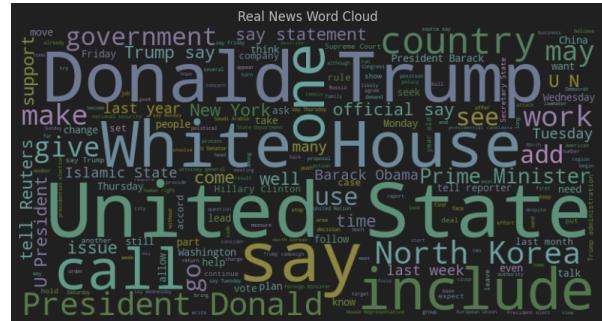


Figura 3.5: Word Cloud știri reale

Tabela 3.1: Caracteristici lexicale extrase din textul știrilor reale

Cuvinte extrase din știrile false	Analiză lexicală
"prime minister", "secretary state", "foreign minister"	Ştirile reale subliniază adesea poziția și responsabilitățile asociate acestiei în loc să se concentreze pe persoanele individuale, aceasta fiind o caracteristică a raportării obiective care își propune să prezinte informații despre evenimente, nu povestiri personale.
"source say", "show", "call", "agree", "official say", "statement", "provide"	Sugerează că știrile reale au dovezi solide care să le susțină povestea, punând accent pe dovezi și pe atribuirea acestora autorităților competente.
"friday", "thursday", "sunday", "monday", "tuesday"	Adverbe temporale ce sugerează o zi exactă, relatările știrilor false se fac pe baza unor evenimente ce s-au întâmplat într-o perioadă clar definită.

Tabela 3.2: Caracteristici lexicale extrase din textul știrilor false

Cuvinte extrase din știrile false	Analiză lexicală
"donald trump", "hilary clinton"	Aceste personalitățile proeminente sunt reprezentate cu o dimensiune foarte mare în diagramă sugerând că o parte a știrilor false se concentrează pe aceștia ( cei doi dominau scena politică în perioada în care a fost construit setul de date) .
"said", "say", "know", "believe", "think", "claim", "fact"	Sugerează că știrile false prezintă în general opinii sau fapte presupuse.
"lie", "love", "attack", "fight", "racist"	Un limbaj emoțional, lipsit de <i>neutralitatea</i> specifică știrilor reale.
"anything", "anyone", "someone", "something"	Pot fi folosite ca și strategie de manipulare pentru a crea un sentiment de incluziune socială mai largă ; evitarea răspunderii atribuind informații <i>cuiva</i> pentru a evita tragedia la răspundere pentru acuratețea afirmațiilor lor.
"day", "today", "year"	Adverbe temporale relative sugerând incertitudinea apariției știrii, a timpului derulării evenimentelor

Analizând tabelele 3.1 și 3.2, putem remarcă anumite relații care ne pot sugera că modelele noastre ar putea să aibă o performanță destul de bună. De asemenea, putem observa că cel mai probabil știrile false din setul nostru de date sunt de tip **propagandă**, fiind caracterizate de un limbaj emotional, exagerarea faptelor pentru a crea un impact mare și lipsa dovezilor clare pentru afirmațiile făcute.

**Bayes Naiv**, utilizând frecvența cuvintelor pentru a face clasificarea poate fi ajutat de faptul că știrile reale au un **vocabulary diferit** comparat cu cel al știrilor false. De asemenea, acesta ar putea să beneficieze de **separarea topicelor**. Știrile false sunt concentrate pe personaje individuale și prezintă opinii asupra acțiunilor acestora, pe când știrile reale se concentrează mai multe pe evenimente și utilizează o narativă neutră. În schimb acest model nu prezintă o înțelegere contextuală a informațiilor și nu poate contura relații complexe între termeni.

**BERT** poate beneficia mai mult de aceste relații și face o distincție mai fină între știrile false și cele reale. Spre exemplu, acesta **poate** învăță că persoanele politice sunt

prezente în ambele tipuri de știri, însă în cadrul știrilor reale este prezentată o narătivă obiectivă despre acestea, pe când în știrile false sunt prezentate narătivuri subiective.

Așadar, caracteristicile observate prin intermediul diagramei ne sugerează relevanța acestora pentru clasificare și separabilitatea lexicală a celor două vocabulare (dacă cele două aveau vocabulare foarte asemănătoare modelele noastre ar fi putut avea dificultăți în procesul de clasificare). Se observă un avantaj clar pe care BERT îl poate avea în acest caz datorită înțelegerii contextului și a relațiilor semantice, însă și Bayes Naiv poate să aibă rezultate bune datorită separabilității termenilor prezenți în fiecare categorie.

## 3.2 Bayes Naiv

Fluxul complet parcurs de date în cazul lui Bayes Naiv este prezentat în diagrama arhitecturală 3.6.

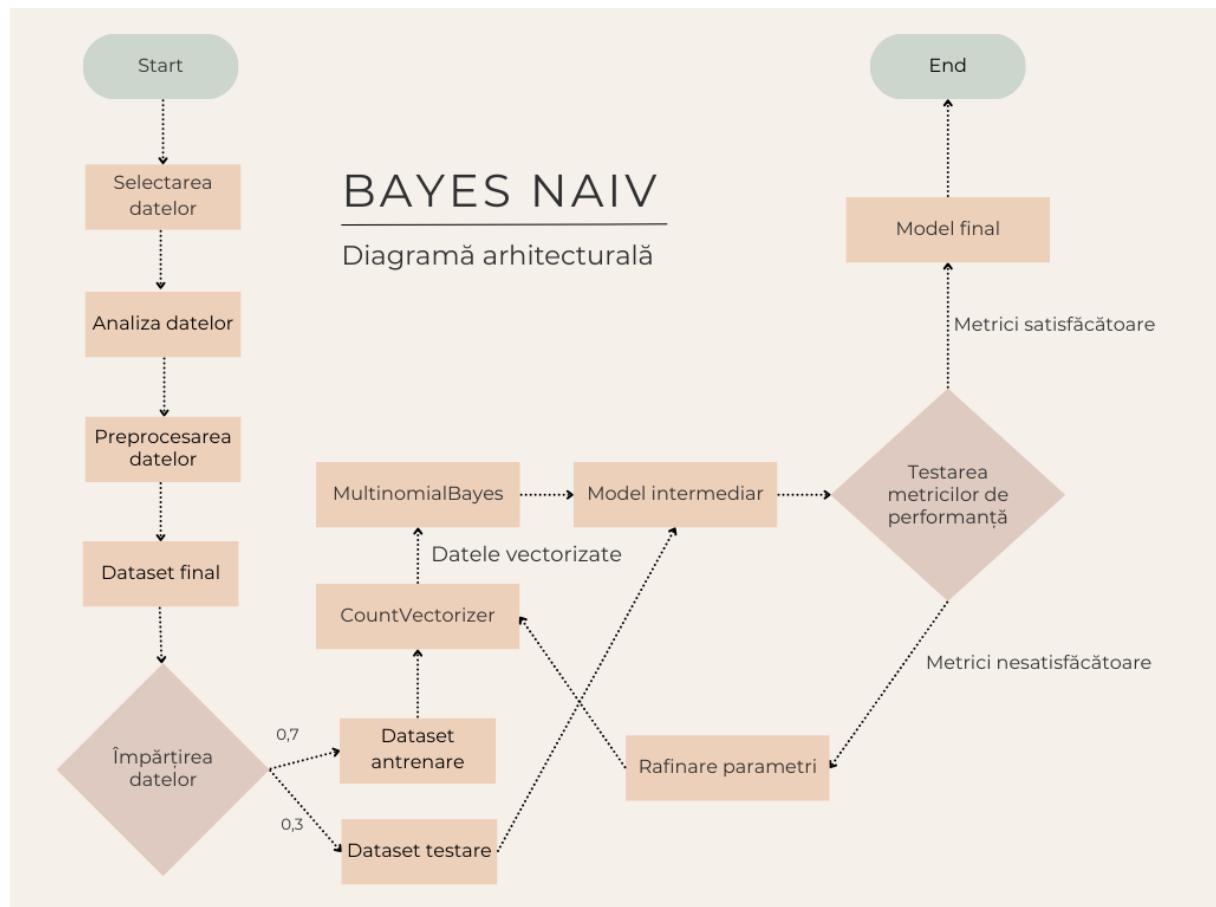


Figura 3.6: Arhitectura Bayes Naiv

### 3.2.1 Preprocesarea datelor pentru Bayes Naiv

Un pas important în construirea unui model de învățare automată este **preprocesarea datelor**. Preprocesarea are un impact semnificativ asupra performanței, iar scopul acestui proces este de a limita zgomotul în date, prezenta acestuia afectând negativ capacitatea modelului de a generaliza. Vom analiza fiecare pas efectuat evidențiind rolul și importanța acestuia.

Întrucât Bayes Naiv se bazează pe analiza frecvențelor cuvintelor, aplicarea unui proces adecvat de preprocesare și curățare a datelor este crucială pentru a asigura performanță optimă a modelului. Pentru a reduce influența zgomotului din date, am realizat o analiză atentă a textului știrilor, în urma căreia am aplicat o serie de etape de preprocesare. După aplicarea etapelor de procesare vocabularul nostru s-a redus cu **24591** de cuvinte (dimensiunea inițială a fost de **121945**, iar cea finală de **97354**).

#### Normalizarea textului utilizând expresii regulate

**Normalizarea** reprezintă procesul de transformare a datelor într-o formă standardizată. **Expresiile regulate** ne vor ajuta să îndeplinim această sarcină. După o analiză atentă a setului de date am observat existența zgomotului mai ales în știrile false. Pentru reducerea acestuia am parcurs următorii pași.

- Pentru știrile false:
  - *eliminarea tagurilor html* → regex #2
  - *eliminarea datelor calendaristice* (datele calendaristice nu ofer informații relevante pentru analiza noastră, mai ales că aceste date se repetă frecvent în știrile false, fiind parte din comentarii) → regex #6
  - *eliminarea linkurilor* → regex #3
  - *eliminarea de usernames* → regex #4
  - *eliminarea semnelor de punctuație* → regex #5
  - *eliminarea referințelor către imagini* ( multe știri false aveau la final referințe către imagini externe ) → regex #7
- Pentru știrile reale, pe lângă eliminarea semnelor de punctuație am adăugat :
  - *eliminarea numelui publicației și al orașului* ( în numeroase articole am observat apariția numelui publicației și al orașului din care a fost făcută) → regex #1

Deși anumite caracteristici ( fapt prezentat în An overview of fake news detection: From a new perspective [8] ), precum : numărul de linkuri, dacă un mesaj conține sau nu semnul întrebării și semnul exclamării, numărul de *hashtag-uri*, numărul de taguri @ dintr-un mesaj s-au dovedit a fi relevante, deoarece vom utiliza n-grame în intervalul (2,3) acestea nu ar fi luate în considerare dacă le-am păstra. Astfel pentru a elimina zgomotul din date și pentru a ne concentra pe trăsăturile pur lingvistice am decis să le eliminăm.

În tabelul de mai jos sunt prezentate expresiile regulate utilizate pentru a curăța textul.

#	Regex	Matches
1	r' ([A-Za-z\s]+)?\/?([A-Z\s]+)?(,?\s[A-za-z]+\.\+\s)?\((Reuters)\)'	WASHINGTON (Reuters), (Reuters), PH-NOM PENH (Reuters), GADSDEN, Ala. (Reuters)
2	r' <[^>]+> &#; &lt; &gt; &nbsp;'	</p><blockquote class="twitter-tweet" data-lang="en"><p lang="en" dir="ltr">
3	r' http[s]?:\/\/\S+'	https://t.co/g8SwgAKtfH
4	r' @\w+'	@MeghanMcCain @shannonrwatts
5	r' [^\w\s]'	?!\$#, () .@ \ etc.
6	r' (\b (January February March April May June July August September October November December)\b \s \d{1,2},\s \d{4})'	December 4, 2017 July 7, 2013 May 8, 2016
7	r' (Featured Photo\sbyp  Image) [\w\ / \s ]+'	Featured image via Lukas Schulze/Getty Images

## Lematizare

Următorul pas al preprocesării, este *lematizarea datelor*. **Lematizarea** este un pas al normalizării datelor și reprezintă *"sarcina de a determina că două cuvinte au aceeași rădăcină, în ciuda diferențelor lor de suprafață"* ( definiție citată din Daniel Jurafsky & James H. Martin ) [4]. Deși lematizarea este mai costisitoare din punct de vedere computațional decât **stemmingul** ( procesul de eliminare a sufixelor ), aceasta oferă cuvinte reale, corect flexionate, și evită problemele de *suprageneralizare* ( stemmingul reduce două cuvinte diferite ca sens la aceeași rădăcină ) sau *subgeneralizare* ( nu reușește să reducă două forme ale aceluiași cuvânt la o rădăcină comună ) asociate metodei de stemming. Astfel, analiza devine mai relevantă, iar cuvintele lematizate contribuie la o mai bună interpretare a factorilor care influențează decizia algoritmului Bayes Naiv. Lematizarea se bazează pe reprezentarea cuvintelor ca **lemma** lor.

Pentru a observa clar diferența dintre lematizare și stemming am efectuat următorul experiment:

Având propoziția: "The children were better at running and studies than their older siblings."

- Rezultat lematizării este: the child be well at run and study than their old sibling.
- Rezultat stemmingului este: the children were better at run and studi than their older siblings.

Pentru a evidenția mai clar diferențele dintre lematizare și stemming, putem observa rezultatele în tabelul 3.3.

Cuvânt original	Lematizare	Stemming
children	child	children
running	run	run
studies	study	studi
better	well	better
older	old	older
siblings	sibling	siblings

Tabela 3.3: Diferențe între lematizare și stemming pentru cuvintele cheie

Se poate observa că, în urma aplicării lematizării, cuvântul *children* este transformat în forma sa canonica de singular, *child*, în timp ce, în cazul aplicării algoritmului de

stemming, acesta rămâne neschimbat. Un comportament similar este observat și pentru cuvântul *siblings*, care este redus la forma *sibling* doar în urma lematizării. În ceea ce privește adverbele, precum *better*, lematizarea reușește să identifice și să transforme cuvântul în forma de bază *well*, în timp ce stemmingul nu face nicio modificare. De asemenea, în cazul substantivului *studies*, algoritmul de stemming produce forma *studi*, care nu corespunde unui cuvânt valid în limba engleză, ceea ce evidențiază natura sa mecanică ce bazată reguli predefinite.

### ***Case folding***

Mai apoi vom face **case folding**, adică vom transforma toate cuvintele în litere mici ( *lowercase* ) . Acest proces are rolul de a micșora dimensiunea vocabularului. Având în vedere că reprezentarea cuvintelor ține cont doar de frecvența acestora ci nu de poziția lor în frază este natural să considerăm cuvintele care apar la începutul propoziției ca fiind aceeași cu cele care se află în centrul frazei. În schimb pot exista cuvinte care au semnificații diferite în funcție de modul în care sunt scrise ( spre exemplu abrevierile ; SALT = Strategic Arms Limitation Talks și cuvântul *salt* care înseamnă sare ) . Pentru a analiza impactul acestei etape asupra performanței modelului, am păstrat atât o versiune a textului original, cu majuscule, cât și una convertită complet în litere mici. Însă în timpul antrenării prin experimente am realizat că versiunea cu litere *lowercase* oferea rezultate la fel de bune ca și cea originală.

### **Eliminarea de stopwords**

Pentru eliminarea cuvintelor **stop words** (cuvinte foarte frecvente precum 'this', 'a' ) am utilizat o listă predefinită din biblioteca nltk.corpus. Această listă cuprinde cuvinte precum: 'because', 'over', 'having', 'than', 'they', 'each'. Am preferat să folosim această abordare în locul întocmirii unei listei de stop words specifică datelor, deoarece conform Daniel Jurafsky & James H. Martin [4] aceasta în general nu îmbunătățește performanța modelului.

## **3.2.2 Transformarea cuvintelor în vectori de caracteristici**

**Vectorizarea** datelor constă în transformarea cuvintelor într-o reprezentare numerică, care să poată fi procesată eficient de algoritmul de învățare automată utilizat. Această etapă presupune maparea textului brut într-un spațiu vectorial, păstrând

informații relevante despre prezența și frecvența termenilor din corpus.

Pentru a implementa acest pas am utilizat **CountVectorizer**. Acest vectorizator reprezintă cuvintele sub forma unei **matrici sparse** ( care conține puține elemente diferite de zero) de tip **termen-document** care conține frecvența fiecarui cuvânt într-un anumit document. Am ales această reprezentare deoarece conținează perfect modul în care funcționează algoritmul Bayes Naiv (acesta folosește abordarea de tip **bag of words** pentru reprezentarea textului explicată în Capitolul 2. În schimb, noi vom utiliza **n-gramme** în loc de cuvinte individuale . **N-grammele** reprezintă grupuri de n cuvinte, spre exemplu 2-gramele ar fi reprezentate de secvențe de cuvinte precum "donald trump", "white house" care pot fi observate și în diagramele Word Cloud(3.5, 3.4). N-gramele ne permit să reducem ambiguitatea textului și să avem un context mai clar (cuvântul good poate să apară atât ca "very good" cât și ca "not good"), însă creează un vocabular mai complex.

În cazul nostru am utilizat n-gramme în intervalul (2,3), deci grupări de 2 și 3 cuvinte. Am utilizat această abordare bazându-ne pe articolul [5], unde a fost demonstrat că acest interval oferă cele mai bune rezultate în cazul detectării spamului. În experimente legate de fenomenul de *overfitting* vom observa cum alegerea intervalului n-gramelor poate influența modelul fie spre *overfitting*, fie spre *underfitting*.

De asemenea pentru a limita zgromotul din date, am utilizat **min\_df=15** și **max\_df=0.7** ; **min\_df=15** semnifică că vor fi ignorati termenii care apar în mai putin de 15 de documente, iar **max\_df=0.7** exclude din vocabular termenii care apar în peste 70% dintre documente, aceștia pot fi considerați ca fiind prea frecvenți sau termeni specifici vocabularului domeniului. Acești parametrii au fost aleși experimental pentru a optimiza performanța modelului și a obține cele mai bune rezultate posibile.

### 3.2.3 Antrenarea modelului Bayes Naiv

După ce am efectuat preprocessarea și tokenizarea datelor, acestea vor fi împărțite în două seturi: **setul de antrenare** ( 70% ) și **setul de testare** ( 30% ) . **Splitarea** a datelor ne permite să utilizăm o parte din date, care nu au fost văzute de model pentru a măsura performanța modelului. **Setul de antrenare** este utilizat pentru construirea modelului, iar **setul de testare** este folosit pentru a evalua performanța modelului, aplicându-l pe date pe care nu le-a văzut anterior. Există numeroase variante ale modelului Bayes Naiv, însă conform lui Dan Jurafsky and James H. Martin [4] s-a de-

monstrat că pentru clasificarea textului nu este atât de important numărul de apariții al unui cuvânt în text pe cât este de importantă prezența acestuia. Așadar, chiar dacă întâlnim un cuvânt de mai multe ori, putem să îi pastrăm frecvența la 1, pentru a face acest lucru este necesar să eliminăm toate duplicatele, iar acest model este denumit **Naive Bayes Multinomial**. Pentru a remedia cazul cuvintelor care sunt prezente doar într-o singură clăsă, am utilizat factorul de netezire  $\alpha=1$ , tehnică denumită *netezirea Laplace*.

Pentru a face procesul de antrenare mai robust am utilizat un **pipeline** în care am încorporat clasificatorul și vectorizatorul. Un pipeline ne permite să aplicăm secvențial vectorizatorul pentru preprocesarea datelor și să folosim un clasificator final. Aceasta ne-a permis să experimentăm cu numeroși parametrii pentru a avea rezultate cât mai bune, dar și să facem mai rapidă inferența modelelor ( folosirea unui model deja antrenat pentru a face predicții ) pentru a prezice veridicitatea știrilor introduse de utilizatori.

### 3.2.4 Verificarea stabilității modelului

La finalul antrenării, este necesar să luăm în considerare și posibilitatea ca modelul nostru să nu se comporte exact cum ne-am așteptă și să facă *overfitting* sau *underfitting*. Deși Bayes Naiv este un model simplu și probabilitatea ca acesta să facă overfitting este destul de scăzută prezența zgromotului în date poate duce la acest fenomen. **Overfittingul** apare când clasificatorul se specializează atât de bine pe datele de antrenare încât tinde să nu generalizeze bine pe date noi. Deși putem observa că avem rezultate bune atât pe antrenare cât și pe testare, fiind o diferență doar de 1% între cele două, vom folosi și o curbă de învățare pentru a evalua modelul. **Curba** de învățare ne arată cum evoluează performanța modelului în funcție de mărimea setului de antrenare. Curvele de antrenare au fost calculate prin **cross-validation**, cu 5 folduri pentru a asigura reducând posibilitatea ca rezultatele modelului să fie influențate de splitarea datelor. În figura 3.7 putem observa curba de antrenare pentru modelul nostru.

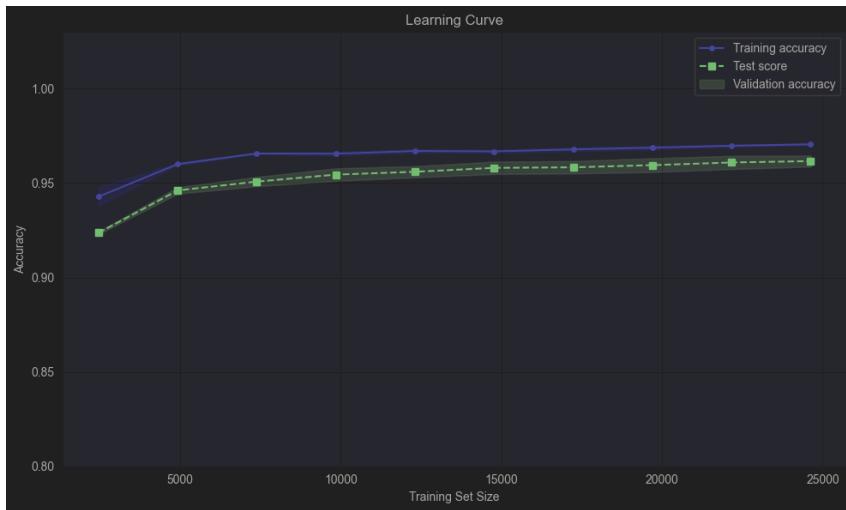


Figura 3.7: Curba de învățare

Acuratețea pe antrenare începe de la un punct destul de ridicat ( aproximativ 92%), iar pe măsură ce mărimea setului de date crește atât acuratețea pentru antrenare cât și cea pentru testare cresc progresiv. Cele două linii se apropie progresiv, ceea ce înseamnă că nu avem semne de overfitting. Putem observa că modelul are rezultate bune chiar și cu seturi mici de date, cu 5000 de exemple de antrenament avem o acuratețe destul de bună, dar cu un set mai mare de date de antrenament acuratețea crește.

Pentru a observa cum s-ar manifesta overfittingul și underfittingul am forțat intenționat modelul să sufere de aceste fenomene fie prin mărirea vocabularului, fie prin limitarea acestuia.

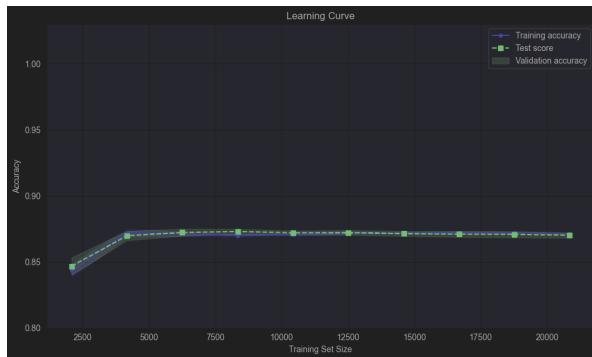


Figura 3.8: Underfitting

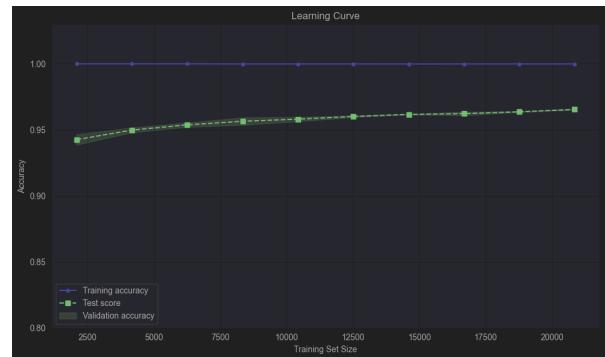


Figura 3.9: Overfitting

In figura 3.8 putem remarca că modelul are *high bias*. **Biasul** este definit ca *incapacitya modelului de a face predictii corecte din cauza faptului că există o anumită diferență*

sau eroare între valoarea prezisă de model și valoarea pe care o au datele. O valoare ridicată a biasului înseamnă că modelul nu se va potrivi îndeaproape cu setul de antrenament, adică va face underfitting. Putem observa că acesta are atât acuratețea la antrenare cât și cea la cross-validation scăzute și egale ( $\sim 86\%$ ) , aşadar acesta nu are performanțe bune nici pe datele de antrenament, nici pe cele de testare. Pentru a forța modelul nostru să facă underfitting am modificat vectorizatorul astfel : am setat numărul maxim de features la 100, am utilizat doar cuvintele unice și ne-am limitat doar la cuvintele care apar în maxim 500 de documente. Astfel vocabularul construit de vectorizator a fost prea mic pentru a reuși să captureze relațiile semantice complexe. Modelul este prea simplu pentru a fi utilizat într-o aplicație de detectie a știrilor false.

În figura 3.9 putem remarcă un model ce suferă de varianță ridicată. **Varianța** reprezintă *capacitatea modelului de a se adapta la un nou subset al datelor de antrenament*. Varianța ridicată înseamnă că modelul este foarte sensibil la modificările datelor de antrenament și are performanțe scăzute pe date noi. Putem observa că acuratețea la testare este mai scăzută decât acuratețea la antrenare, care este maximă, iar între cele două există o diferență de aproximativ 4%. Pentru ca modelul nostru să sufere de overfitting am modificat vectorizatorul prin utilizarea n-gramelor din intervalul (1,3), am luat în considerarea toti termenii ( $\text{min\_df}=1, \text{max\_df}=1$ ), aceste caracteristici au dus la un vocabular mare, însă nereprezentativ pentru date, conținând și numeroase surse de zgomot. În acest caz modelul s-a adaptat prea puternic la datele de antrenament și nu a reușit să generalizeze pe date noi.

### 3.3 BERT

Fluxul complet parcurs de date, în cazul lui BERT este prezentat în diagrama arhitecturală 3.10.

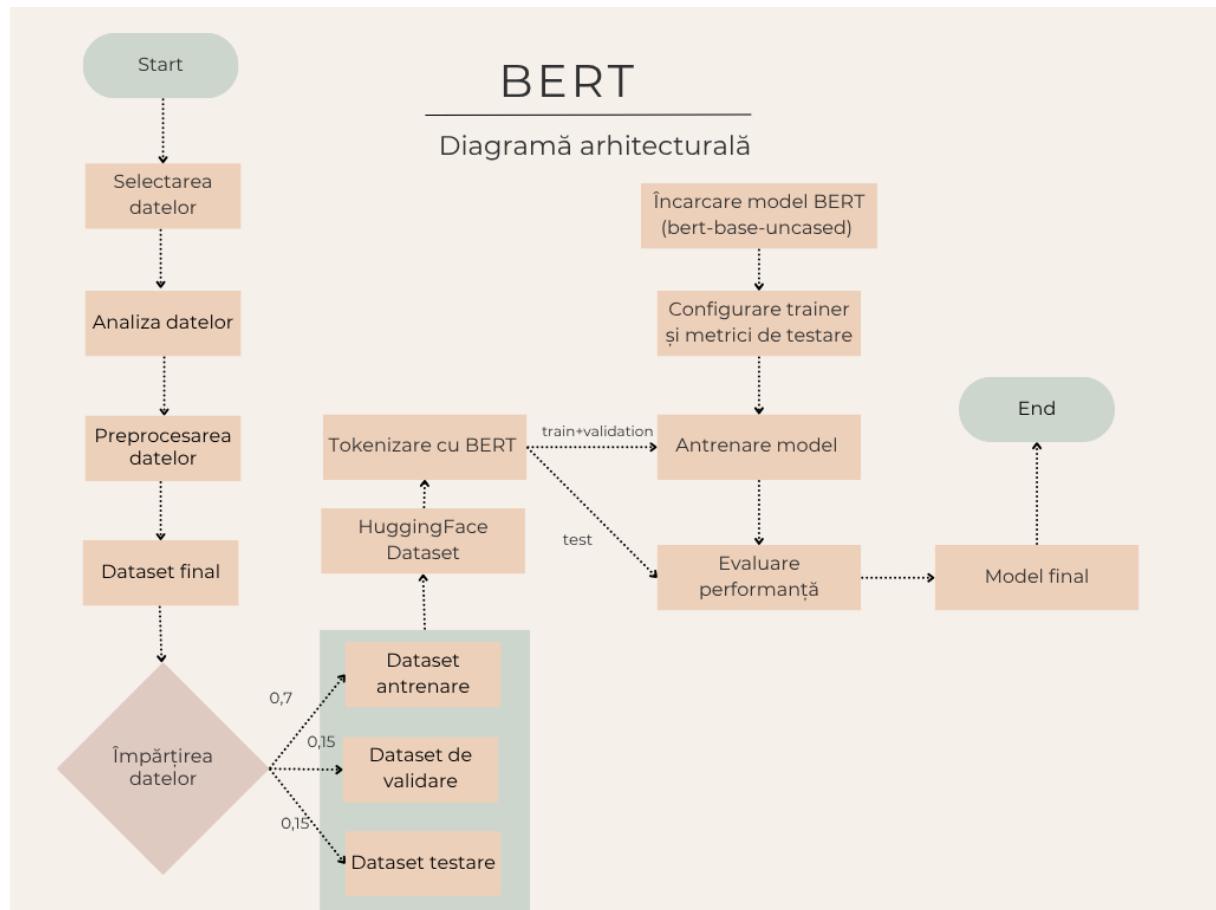


Figura 3.10: Arhitectură BERT

Putem observa că este necesar să parcurgem aproximativ aceeași pași ca și în cazul lui Bayes Naiv, însă sunt diferențe distincte pe care le vom analiza în secțiunile următoare.

#### 3.3.1 Preprocesarea datelor pentru BERT

În cazul lui BERT nu este necesar să facem o preprocesare prea agresivă a textului, deoarece acesta este preantrenat pe un set mare de texte naturale. Modelul astfel a învățat să gestioneze variațiile de limbaj. Așadar am aplicat doar o preprocesare minimă care să nu modifice contextul datelor. Pașii de preprocesare includ:

- eliminarea tagurilor html

- înlocuirea datelor calendaristice cu tokenul special [DATE]
- înlocuirea linkurilor cu tokenul [LINK]
- eliminarea referințelor către imagini
- înlocuirea hashtag-urilor cu tokenul [HASHTAG]
- eliminarea spațiilor suplimentare
- eliminarea numelui publicației și al orașului din știrile reale
- înlocuirea numelor utilizatorilor cu tagul [USERNAME]

Am decis ca unele din informații să fie codificate printr-un token special pentru a nu altera structura semantică a propozițiilor, dar nu le-am păstrat pentru a limita overfittingul în cazul în care unele dintre caracteristici ar fi foarte specifice știrilor false, acestea fiind colectate de pe rețele sociale.

După cum putem vedea, preprocesarea datelor este mult mai simplă și mai rapidă în cazul lui BERT. Nu mai este necesar să facem eliminarea semnelor de punctuație și nici lematizarea datelor, cea din urmă fiind foarte costisitoare computațional.

### 3.3.2 Împărțirea setului de date

Pentru a testa performanța modelului nostru și pentru a remarcă cum evoluează modelul după fiecare epocă de antrenare este necesar să împărțim setul de date în 3 categorii distincte : train , validation și test. Distribuția completă poate fi observată în figura 3.11.

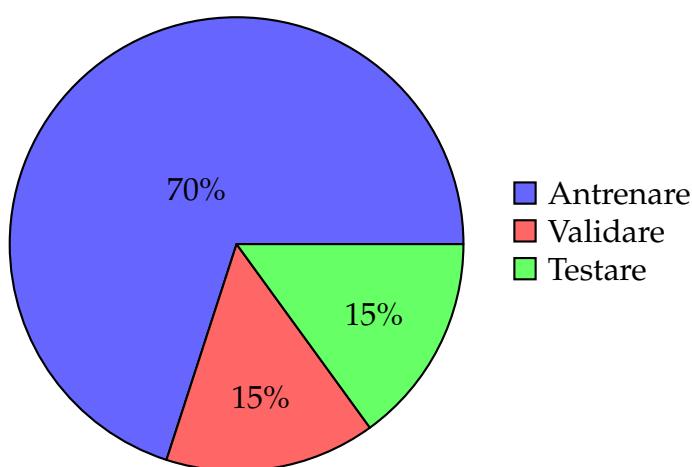


Figura 3.11: Împărțirea setului de date: 70% antrenare, 15% validare, 15% testare

**Setul de antrenare** este utilizat pentru antrenarea modelului. **Setul de validare** este utilizat pentru a monitoriza performanța modelului pe parcursul antrenării, acesta ajută la detecția *overfittingului*, după fiecare epocă pe setul de validare sunt calculate diferite metriki care ne arată cum se descurcă modelul pe date necunoscute pentru a putea remarcă din timp dacă acesta începe să se supraspecializeze astfel încât antrenarea să fie oprită la momentul potrivit. Dacă în timpul antrenamentului pierdere la validare crește și cea la antrenare scade putem utiliza *early stopping* pentru a opri modelul din a face *overfitting*. Iar, **setul de testare** este utilizat pentru a analiza metricile de performanță după antrenarea completă a modelului. Aceste *dataseturi* sunt salvate sub forma unui HuggingFace Dataset deoarece acesta este formatul specific așteptat de Trainer.

### 3.3.3 Tokenizarea datelor

Pentru antrenarea modelului am folosit biblioteca Transformers<sup>4</sup> de la Hugging Face<sup>5</sup>, care oferă un mod flexibil de prelucrare a datelor, antrenare și testare a modelului. Datele brute, pe care le-am preprocesat anterior, trebuie să fie transformate într-un format compatibil cu modelul, mai exact în secvențe de **token ids** ( reprezentate ca input numeric ). Acest proces implică tokenizarea datele, adică transformarea în secvențe de tokeni ( subunități de cuvinte ) specifice lui BERT, urmată de trunchirea textelor mai lungi de 512 tokeni. Această preprocesare s-a aplicat în mod batch pe întreg setul de date pentru a accelera procesul de tokenizare.

În timpul antrenamentului folosim un **data collator** special care adaugă padding pentru a face secvențele să aibă lungimea de 512 tokeni doar unde este necesar în cadrul fiecărui batch, optimizând astfel consumul de memorie și viteza antrenamentului.

Pentru a înțelege mai bine modul în care sunt tokenizate cuvintele, să luăm un exemplu concret : „Misinformation can be unbelievably problematic, especially in 2025.”. Rezultatul tokenizării a fost următorul:

```
['mis', '#in', '#form', '#ation', 'can', 'be', 'un', '#bel',
 , '#ie', '#va', '#bly', 'problematic', ',', 'especially',
 'in', '202', '#5', '.']
```

---

<sup>4</sup><https://huggingface.co/docs/transformers/en/index>

<sup>5</sup><https://huggingface.co/>

Putem remarcă că în acest caz cuvintele sunt împărțite în cuvinte mici denumite subtokensi, această tehnică utilizată de BERT este denumită *Word Piece Tokenization* și are rolul de a face clasificatorul mai robust la variațiile limbajul. De asemenea, subtokensi sunt mai apoi transformați în reprezentări numerice denumite **token IDs** deoarece BERT poate utiliza doar forme numerice ca input.

### 3.3.4 Antrenarea modelului

Pentru task-ul nostru am utilizat modelul preantrenat **bert-base-uncased** căruia îl am făcut **finetuning** pe datele noastre. Primul pas făcut în acest scop a fost să înghețăm **layerele** din modelul de bază și să păstrăm activi doar parametrii din layerul pooler și din stratul de clasificare adăugat ulterior. **Layerul pooler** aplică o transformare asupra vectorului [CLS], care rezumă întreaga propoziție și este folosit în sarcina originală de preantrenare BERT, cum ar fi *next sentence prediction*. În cadrul procesului de *finetuning* dorim să înlocuim acea sarcină generală cu o sarcină specifică, clasificarea binară a stirilor false. Așadar pentru a păstra *embedding-urile contextuale* deja învățate de model și să-l specializez pe taskul nostru am ales să înghețăm toate layerele, în afară de layerul pooler și să utilizăm *embedding-urile contextuale* reprezentate de outputul layerului pooler ca input pentru o rețea neuronală de clasificare.

```
from transformers import TrainingArguments
training_args = TrainingArguments(
    output_dir="bert_training_model_for_fake_news_detection_8",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=5,
    warmup_ratio=0.1,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    logging_steps=400,
    logging_dir=".//logs_8",
    report_to="tensorboard"
)
✓ [12] 61ms
```

Figura 3.12: Parametrii utilizati pentru antrenarea lui BERT

În figura 3.12 putem observa parametrii setați pentru antrenarea modelului. Alegera acestora este în strânsă legătură cu lucrarea "How to Fine-Tune BERT for Text Classification?"[6].

### Rata de învățare

În cadrul unei rețele neuronale rata de învățare reprezintă dimensiunea cu care modelul nostru face pașii spre optimizarea minimului funcției de cost, astfel am ales rata de învățare de  $2e-5$ . Această rată de învățare este destul de mică pentru a evita fenomenul de *catastrophical forgetting*.

### Dimensiunea batch-urilor

Batch-urile nu sunt diferite de cele utilizate în cazul unei rețele neuronale, acestea reprezintă numărul de exemple dintr-un mini-batch care sunt procesate simultan pentru actualizarea parametrilor modelului în cadrul unei iterații atât în propagarea directă (forward propagation), cât și în propagarea inversă (backward propagation). Deși în cazul rețelelor neuronale clasice este utilizat Mini-batch Gradient Descent pentru a minimiza funcția de cost, BERT utilizează un optimizator AdamW. AdamW este o variantă optimizată a algoritmului Adam care combină momentum-ul și adaptarea automată a ratei de învățare. Am ales să folosim dimensiunea unui batch egală cu 8 pentru a avea un echilibru între memoria ocupată și performanța modelului. Experimentând cu batch-uri de 16 am observat că pierderea la antrenare era mai instabilă și nu avea un traseu descendent.

### Numărul de epoci de antrenare

Antrenarea modelului se face pentru un număr de epoci, o epocă reprezintă *trecerea completă prin toate exemplele de antrenament*. Deoarece în timpul antrenării am observat rezultate promițătoare încă din prima epocă am decis că 5 epoci ar fi un număr rezonabil de epoci de antrenare. De asemenea pentru a reduce riscul de overfitting am utilizat *early stopping=2*, acesta oprește antrenarea modelului după 2 epoci în care pierderea pe setul de validare nu scade.

## Pașii de warmup

**Warmup steps** ajută la creșterea graduală a ratei de învățare la începutul antrenamentului pentru stabilizarea actualizării inițiale. Am ales un procent de 10% pentru evita schimbările brusă ale *weight-urilor* și a stabiliza procesul de învățare.

### 3.3.5 Analiza antrenării

În cursul antrenării modelului am analizat atât pierderea ( loss ) cât și acuratețea și scorul AUC pe datele de antrenament și pe cele de validare pe parcursul a 5 epoci, valorile pot fi observate în figura 3.13.

Epoch	Training Loss	Validation Loss	Accuracy	Auc
1	0.093600	0.112812	0.964000	0.993000
2	0.077200	0.083209	0.974000	0.996000
3	0.061000	0.078123	0.976000	0.997000
4	0.047800	0.070301	0.977000	0.998000
5	0.053200	0.067993	0.978000	0.998000

Figura 3.13: Rezultatele din timpul antrenării lui BERT

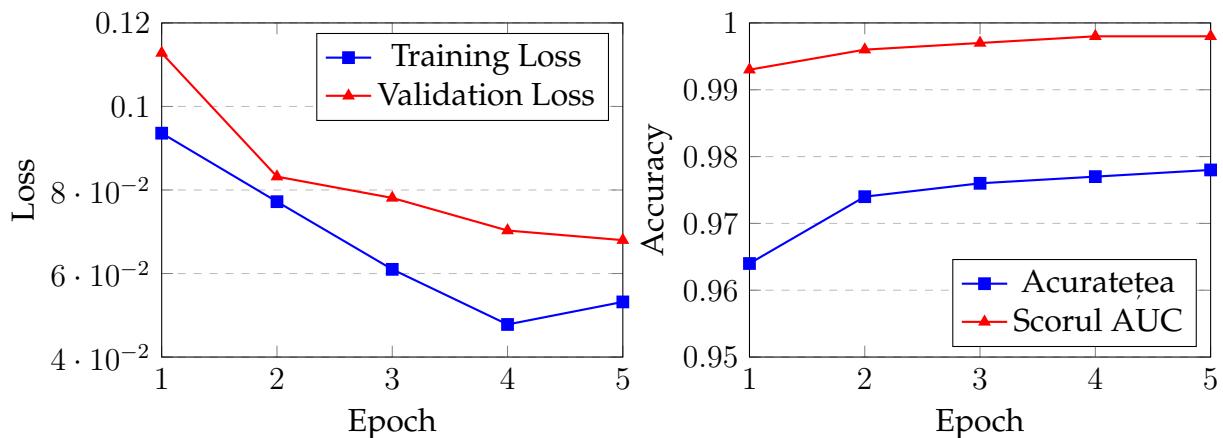


Figura 3.14: Evoluția pierderii (loss) pe seturile de antrenament și validare

Figura 3.15: Evoluția acurateții pe seturile de antrenament și validare

De asemenea în graficele 3.14 și 3.15 putem observa clar modul în care modelul evoluează în funcție de pierdere, acuratețea și scorul AUC. Pierderea pe antrenare scade continuu din epoca 1 până în epoca 4, deși în epoca 5 crește puțin, mici fluctuații

nu trebuie să ne îngrijoreze deoarece pierderea la validare scade în fiecare epocă, ceea ce semnifică că modelul nostru generalizează bine pe date noi și nu memorează datele de antrenament. Această diferență din epoca 5 poate proveni de la regularizarea aplicată modelului BERT. În cazul în care pierderea la antrenare ar scădea și cea la validare ar crește pe parcursul antrenamentului ne-am putea îngrijora legat de overfitting.

Pe parcursul antrenamentului putem remarcă creșterea acurateței și a scorului AUC, arătând faptul că modelul nostru învață eficient. Deși ajungem la rezultate excelente, este posibil ca antrenând modelul un număr mai mare de epoci acesta să aibă performanțe și mai bune.

## 3.4 Aplicație web

Interfața web permite interacțiunea utilizatorului cu modelul nostru de predicție. Aceasta este dezvoltată sub forma unui *single page application* și are 2 componente principale, care vor fi prezentate în această subsecțiune.

### Predicția veridicității știrii

Pentru a prezice dacă o știre este reală sau falsă am decis că vom afișa atât rezultatele oferite de Bayes cât și cele oferite de Bert împreună cu scorul de încredere al fiecărui model în predicția făcută. Modelul Bayes Naiv, oferă o probabilitate asociată fiecărei clase, însă acest scor de încredere nu este foarte exact deoarece acesta suferă de **overconfidence**. De cele mai multe ori, independența condițională a cuvintelor este încălcată, probabilitățile a posteriori finale vor fi mult mai apropiate de 1 sau 0 decât ar trebui să fie; astfel modelul poate fi considerat prea încrezător în predicțiile sale. (afirmații prezentate în Artificial Intelligence: A Modern Approach, Global Edition, 4ed [9] ). În schimb, în capitolul 4 vom observa că predicțiile făcute de acesta sunt destul de precise. În schimb în cazul lui BERT acest rezultat este mai relevant. În figurile 3.16 și 3.17 este prezentată predicția veridicității unei știri.

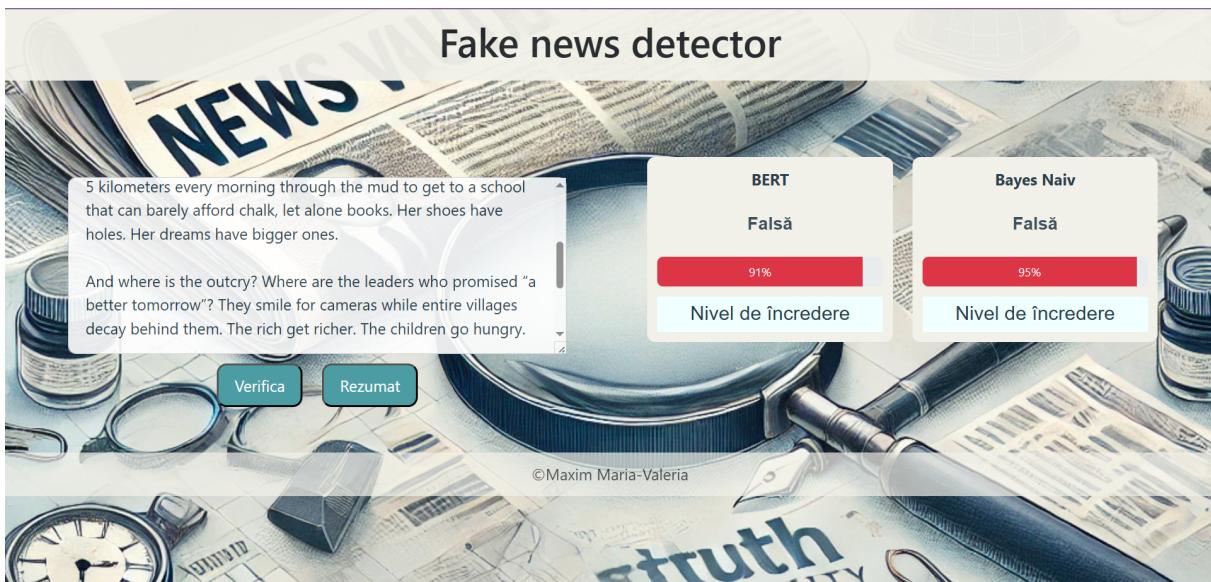


Figura 3.16: Predictia unei stiri false



Figura 3.17: Predictia unei stiri reale

## Rezumarea stiri

Cea de-a doua funcționalitate a aplicației noastre este reprezentată de rezumarea stiri introdusă de utilizator. Rezumatul stiri este făcut utilizând un model de tip Transformer, antrenat special pentru sarcina de *text summarization* ("facebook/bart-large-cnn"). Arhitectura modelului BART ( Bidirectional and Auto-Regressive Transformer ) se bazează pe două componente : un encoder ( similar cu BERT ) și un decoder

autoregresiv ( similar cu GPT ). Encoderul este utilizat pentru înțelegerea contextului bidirectional, iar decoderul este utilizat pentru generarea rezumatului.

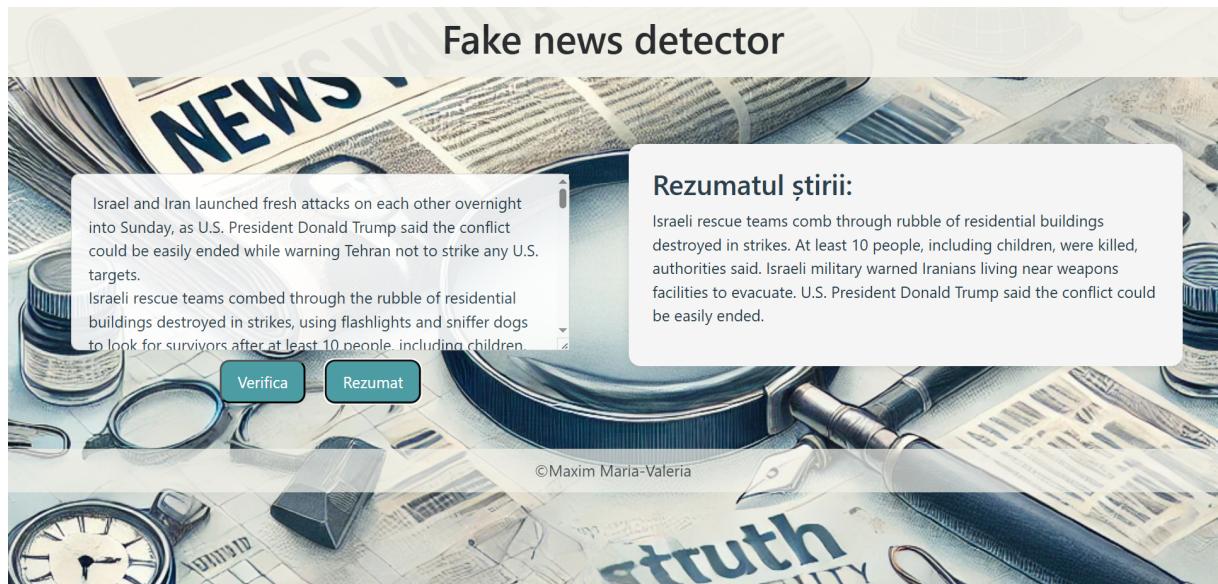


Figura 3.18: Rezumatul unei știri

# Capitolul 4

## Evaluare

**Evaluarea performanței** este un pas important în construirea unui model deoarece prezintă atât o măsură a performanței acestuia, cât și capacitatea sa de generalizare. De asemenea, ne permite și să vizualizăm comparativ performanțele celor două modele pe care le-am utilizat pentru detectarea știrilor false.

### 4.1 Definirea metricilor de evaluare

Pentru a cuantifica performanțele modelelor noastre am utilizat următoarele mătrici:

- **Matricea de confuzie** este o reprezentare utilizată pentru a vizualiza cât de bine funcționează modelul nostru prin compararea rezultatelor obținute de acesta cu cele stabilite inițial în setul de antrenare. Această matrice este construită pe 2 dimensiuni (outputul dat de model și outputul real din setul de date inițial). Matricea de confuzie se bazează pe conceptele de **true positives/negatives** și **false positives/negatives**. În cazul nostru, *true positives* reprezintă știrile care sunt recunoscute ca fiind false și sunt într-adevăr false. Iar *false negatives* reprezintă documentele care sunt într-adevăr știri false, dar modelul nostru le clasifică incorect ca fiind adevărate.
- **Acuratețea** reprezintă procentajul de documente din modelul nostru care sunt clasificate corect. Această metrică nu este indicată în cazul în care clasele nu sunt echilibrate. În cazul în care am avea 98 de știri reale și 2 știri false, dacă am utilizat un clasificator naiv care ar spune că orice știre este reală atunci am avea o acuratețe foarte bună de 98%, însă modelul nostru nu ar reuși să clasifice

nicio știre ca fiind falsă. Acest exemplu ne arată de ce acuratețea nu este cea mai potrivită metrică în cazul claselor dezechilibrate. Deși în cazul nostru clasele nu sunt dezechilibrate (setul de date utilizat conține 23.468 de știri false și 21.211 știri reale), acestea nu sunt nici perfect echilibrate aşadar este necesar să analizăm și alte metriki.

- **Precizia** măsoară procentajul de documente pe care sistemul le-a prezis ca fiind false și chiar sunt false.
- **Recall/True positive rate(TPR)** măsoară procentajul de documente prezente în setul de testare care sunt într-adevăr recunoscute de sistem ca știri false.
- **F1** reprezintă media armonică dintre precizie și recall. Aceasta încorporează cele două metriki într-o singură.
- **ROC AUC (Receiver Operating Characteristic – Area Under Curve)** este o metrică de evaluare care măsoară performanța unui clasificator binar pe baza scorurilor de probabilitate. Aceasta reflectă capacitatea modelului de a diferenția între clasele pozitive și negative indiferent de pragul de decizie ales. Un scor AUC apropiat de 1 indică o separare clară între clase, în timp ce un scor de 0.5 sugerează o clasificare aleatorie.

Pentru o înțelegere mai bună a acestor metriki putem observa în diagrama 4.1 formulele după care se calculează:

		Real		
		1	0	
		1	0	
Prezis	1	true positive	false positive	$\text{precizie} = \text{tp}/(\text{tp}+\text{fp})$
	0	false negative	true negative	
		$\text{recall} = \text{tp}/(\text{tp}+\text{fn})$		$\text{acuratețe} = (\text{tp}+\text{tn})/(\text{tp}+\text{tn}+\text{fp}+\text{fn})$

Figura 4.1: Metriki de performanță

F1 se calculează după următoarea formulă:

$$F1 = \frac{2 * \text{precizie} * \text{recall}}{\text{precizie} + \text{recall}} \quad (4.1)$$

## 4.2 Rezultatele obținute pentru Bayes Naiv

Ultimul pas al dezvoltării modelului nostru este să analizăm rezultatele acestuia.

- Matricea de confuzie

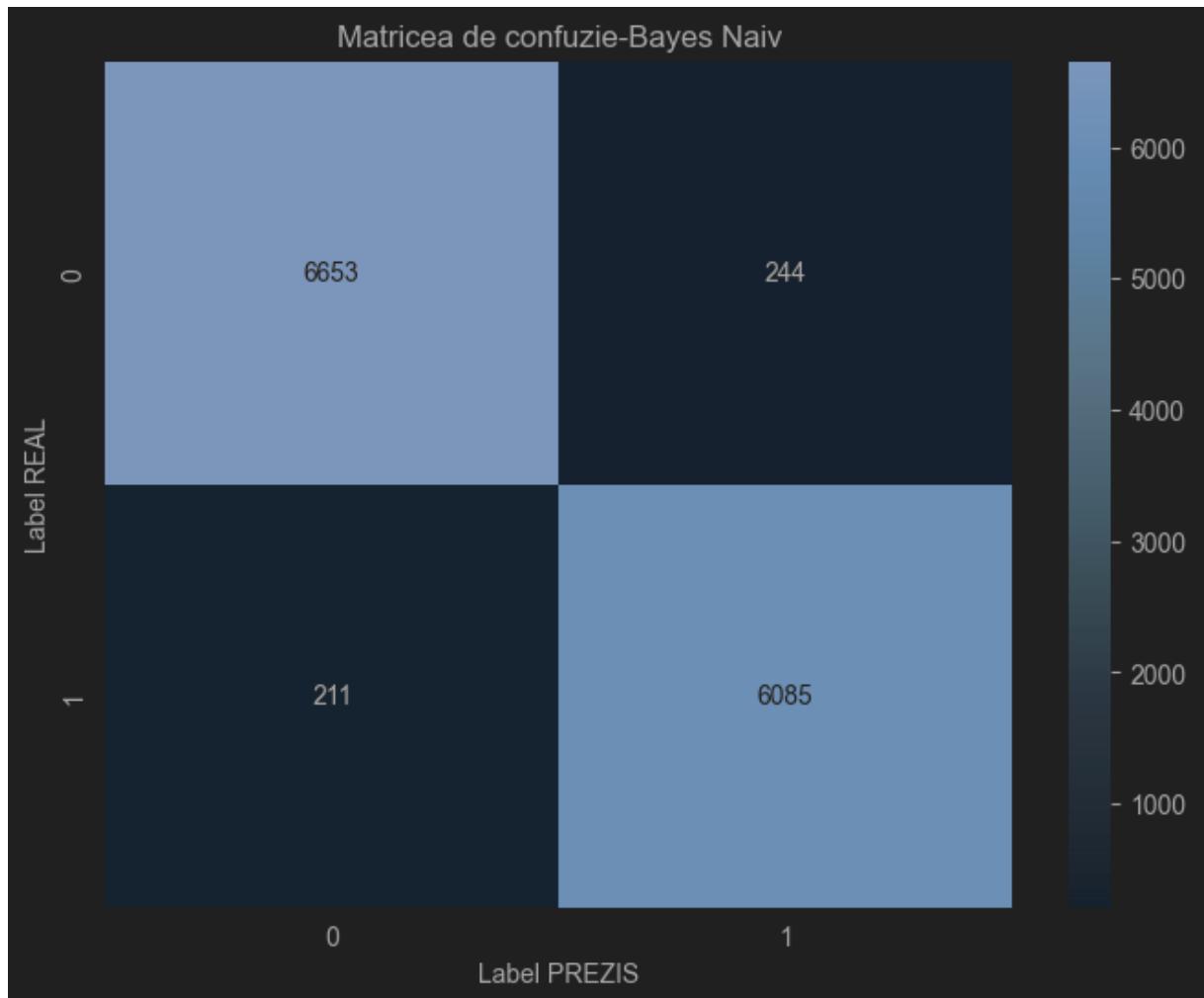


Figura 4.2: Matricea de confuzie pentru Bayes Naiv

Analizând această matrice putem observa că numărul de false positives (o știre reală clasificată ca fiind falsă) este 211, iar numărul de false negatives (o știre falsă clasificată de modelul nostru ca fiind reală) este 244 (deși clasa notată cu 1 este cea a știrilor reale, nu trebuie să facem confuzie între positive ca label și

semnificația acestuia în cazul modelului nostru, în cazul nostru descoperirea unei știri false reprezintă un true positive). Utilizând matricea de confuzie putem să calculăm metrii precum precizia, recall-ul și acuratețea. În acest scop, pentru calcule vom utiliza următoarele valori:

TP(true positive)=6653	FN(false negative)=244
FP(false positive)=211	TN(true negative)=6085

- **Acuratețea** măsoară proporția de clasificări corecte.

$$A = \frac{TP + TN}{TP + TN + FP + FN} = \frac{6653 + 6085}{6653 + 6085 + 211 + 244} = \mathbf{0.9655} \quad (4.2)$$

- **Precizia** va măsura procentul de știri care sunt prezise ca fiind false și sunt într-adevăr false din numărul total de predictii false făcute de model.

$$P = \frac{TP}{TP + FP} = \frac{6653}{6653 + 211} = \mathbf{0.9692} \quad (4.3)$$

- **Recall-ul** măsoară proporția de știri cu adevarat false care au fost clasificate corect de model din numărul total de știri false. Aceasta este denumită și **probabilitatea de detectie a sistemului**.

$$R = \frac{TP}{TP + FN} = \frac{6653}{6653 + 244} = \mathbf{0.9646} \quad (4.4)$$

- **F1-score** ne oferă un mod de a verifica robustețea sistemului.

$$F1 = \frac{2 * P * R}{P + R} = \frac{1.8697}{0.9692 + 0.9646} = \mathbf{0.9668} \quad (4.5)$$

În figura se pot observa aceste rezultate, atât pe setul de testare cât și pe cel de antrenare.

Classification Report (Test):				
	precision	recall	f1-score	support
Fake	0.97	0.96	0.97	6897
Real	0.96	0.97	0.96	6296
accuracy			0.97	13193
macro avg	0.97	0.97	0.97	13193
weighted avg	0.97	0.97	0.97	13193

Classification Report (Train):				
	precision	recall	f1-score	support
Fake	0.97	0.97	0.97	15869
Real	0.97	0.97	0.97	14914
accuracy			0.97	30783
macro avg	0.97	0.97	0.97	30783
weighted avg	0.97	0.97	0.97	30783

Figura 4.3: Raportul metricilor pentru setul de testare

Figura 4.4: Raportul metricilor pentru setul de antrenare

Metricile prezentate anterior sunt calculate în funcție de un singur prag ( pentru modelul nostru pragul implicit este 0.5, clasa care are probabilitatea condiționată mai mare de acest prag este cea prezisă de clasificator ) . Pentru a observa performanța modelului în funcție de toate pragurile posibile am utilizat **ROC curve și PR curve** ( precision recall curve ) .

ROC curve este trăsătă prin calcularea TPR ( recall ) și FPR ( false positive rate ) - formula pentru calcularea FPR este prezentată în ecuația 4.6 - la fiecare prag posibil, apoi reprezentarea grafică a TPR în funcție de FPR. **AUC ( Area Under Curve )** reprezintă probabilitatea ca atunci când îi dăm modelului un exemplu pozitiv și unul negativ, să claseze mai sus exemplul pozitiv decât cel negativ.

$$FPR = \frac{FP}{FP + TN} \quad (4.6)$$

PR curve este creată prin plotarea precizie pe axa y și a recall-ului pe axa x în funcție de toate pragurile.

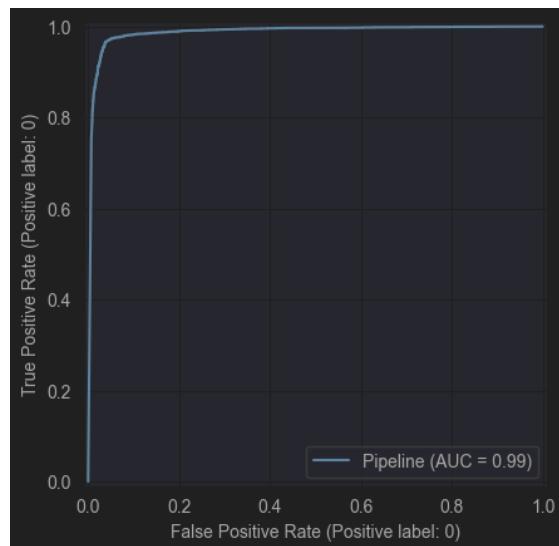


Figura 4.5: Curba ROC-Bayes Naiv

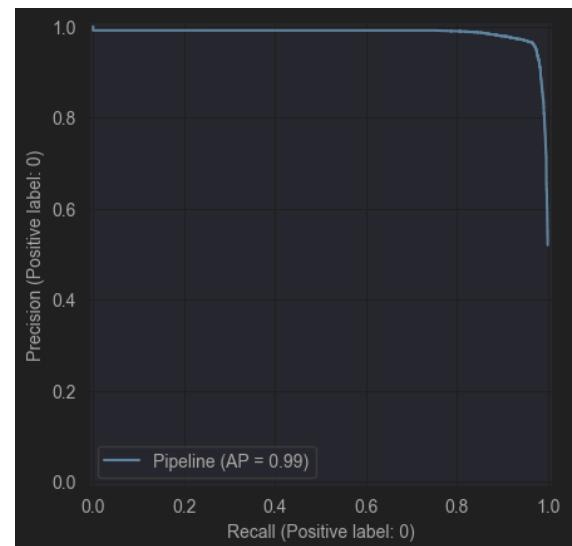


Figura 4.6: Curba PR-Bayes Naiv

### 4.3 Rezultatele obținute pentru BERT

Vom analiza același metriki pe care le-am analizat în cazul lui Bayes și pentru BERT. Pentru a le oferi aceeași set de date vom observa că numărul de exemple testate diferă ( pentru BERT a fost necesar să avem și un set de date de validare ) .

- Matricea de confuzie

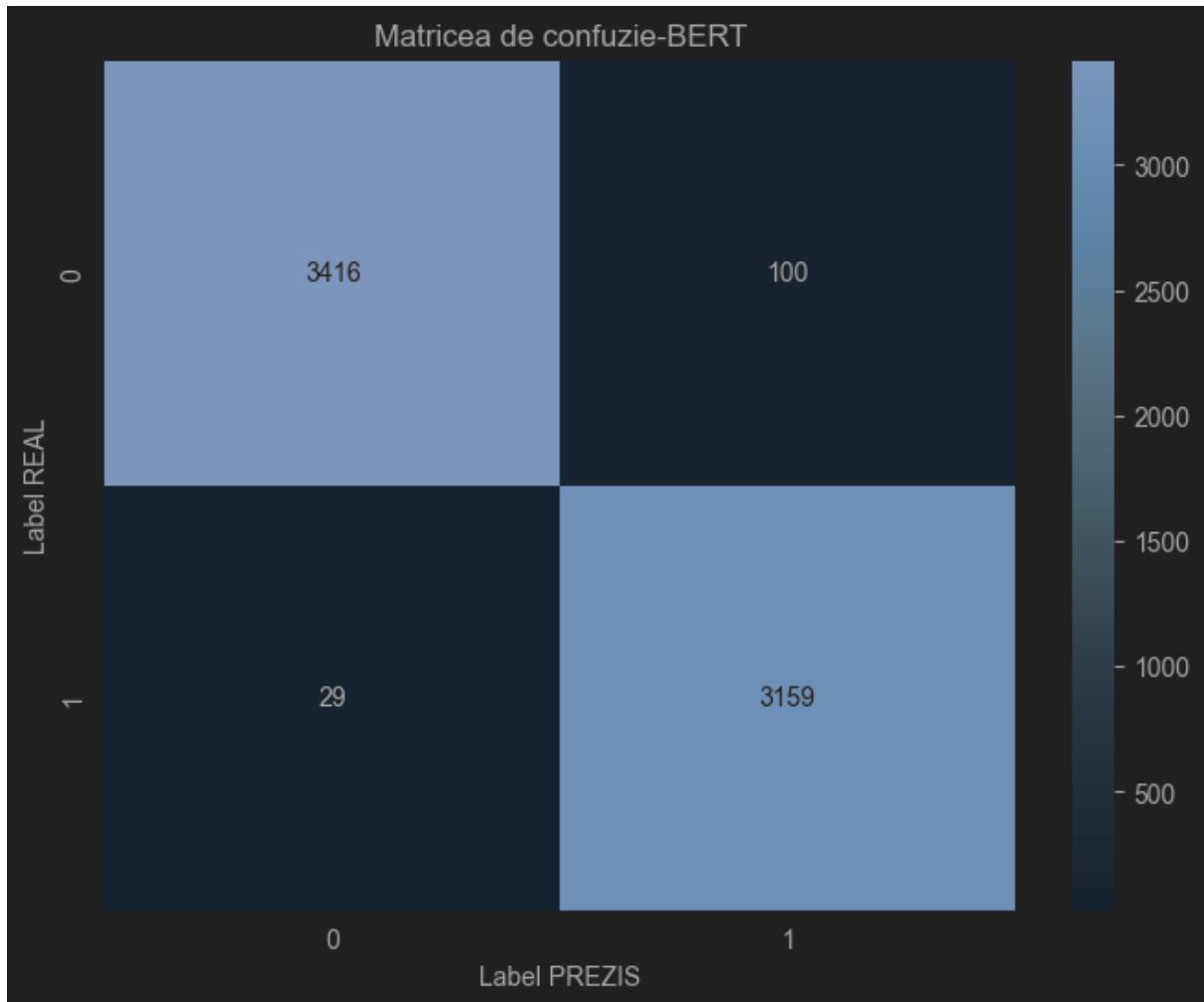


Figura 4.7: Matricea de confuzie-BERT

Pentru a analiza mai ușor metricile vom lua în considerare următoarele:

TP(true positive)=3416	FN(false negative)=100
FP(false positive)=29	TN(true negative)=3159

Pentru a analiza precizia, recall-ul și F1-score vom utiliza formulele definite anterior, aceste metriki având aceeași însemnatate ca și în cazul lui Bayes Naiv.

- Acuratețea

$$A = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3416 + 3159}{3416 + 3159 + 29 + 100} = 0.980 \quad (4.7)$$

- Precizie

$$P = \frac{TP}{TP + FP} = \frac{3416}{3416 + 29} = \mathbf{0.9915} \quad (4.8)$$

- Recall

$$R = \frac{TP}{TP + FN} = \frac{3416}{3416 + 100} = \mathbf{0.9715} \quad (4.9)$$

- F1-score

$$F1 = \frac{2 * P * R}{P + R} = \frac{2 * 0.9915 * 0.9715}{0.9915 + 0.9715} = \mathbf{0.9813} \quad (4.10)$$

- AUC

$$AUC = \mathbf{0.998} \quad (4.11)$$

## 4.4 Analiza comparativă a modelelor

În graficul 4.8 putem observa rezultatele obținute de Bayes Naiv și BERT pentru toate metricile de performanță.

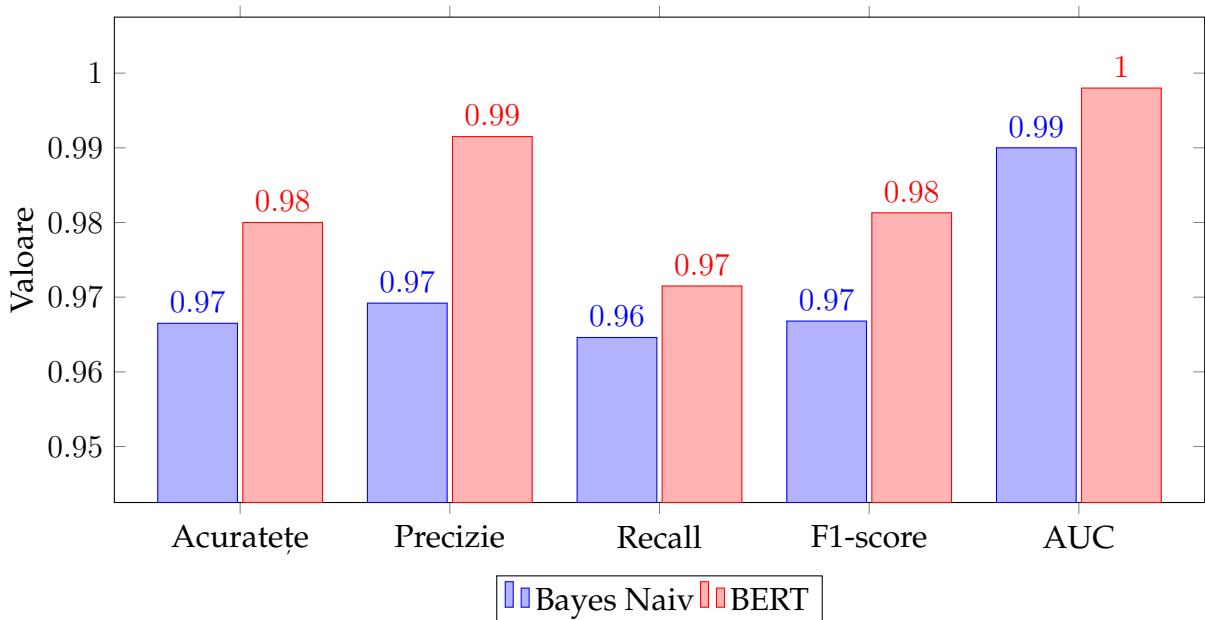


Figura 4.8: Compararea performanțelor între modelele BERT și Bayes Naiv

Pentru Bayes Naiv acuratețea, precizia, recall-ul și F1-score se află toate în jurul valorii de ~97%. Acuratețea ridicată, mai ales în cazul în care clasele noastre sunt destul

de balansate reprezintă o măsură ce ne poate ajuta să ne facem o idee generală despre performanța modelului având în vedere că încorporează TP, FP, TN, FN. Recall-ul ne spune care este probabilitatea de detecție a știrilor false de către clasificator, iar precizia ne asigură ca rezultatele prezise de model ca fiind pozitive sunt exacte. În plus, scorul AUC și aria sub curba precision-recall (PR AUC) fiind egale cu 0.99 evidențiază o bună separabilitate între clase.

În cazul lui **BERT**, acuratețea acestuia este foarte ridicată, respectiv 0.98%, având în vedere că cele două clase sunt destul de echilibrate acest rezultat ne oferă informații despre o performanță excelentă. Deși precizia este puțin mai ridicată decât recall-ul, această diferență nu este semnificativă. Precizia înseamnă că atunci când modelul nostru prezice că o știre este falsă acesta nu se înșeală, recall-ul în schimb ne arată câte din știrile false sunt recunoscute de model. În cazul nostru ne dorim ca ambele metrii să fie ridicate, dar este important ca precizia să fie mai ridicată pentru ca modelul nostru să nu clasifice știri din surse de încredere ca fiind false, deoarece încrederea în sistemul nostru ar scădea dacă o știre dintr-o sursă de încredere ar fi clasificată ca fiind falsă. În general dacă încercăm să îmbunătățim recall-ul, va scădea precizia și vice-versa.

Putem remarcă că BERT obține rezultate mai bune pe toate metricile. După cum putem vedea, precizia în general este mai mare decât recall-ul, ceea ce înseamnă că modelul nostru recunoaște precis cazurile de știri false, însă nu le recunoaște chiar pe toate. Este posibil precizia ridicată să fie rezultatul faptului că știrile reale sunt scrise în manieră jurnalistică și urmează, în general un anumit tipar, comparativ cu cele false care provin din domenii mai variate și nu prezintă un model unitar al textului.

Așadar din punctul de vedere al performanței suntem înclinati să îl alegem pe BERT, deoarece acesta oferă rezultate mai bune. BERT ne dă și un scor de încredere mai exact care este destul de important pentru interacțiunea utilizatorului cu aplicația noastră.

În figura 4.9, putem remarcă durata preprocesării, antrenării și a testării modelelor (pe setul de date de testare). BERT este mai costisitor din punct de vedere computational, chiar și antrenarea acestuia trebuie să fie făcută pe GPU pentru a fi mai rapidă. Așadar în cazul în care avem resurse computaționale limitate și ne dorim un model care să ofere rezultate bune, dar dorim să fie antrenat rapid și să nu fie costisitor din punct de vedere computational, putem tinde să preferăm modelul care folosește Bayes Naiv.

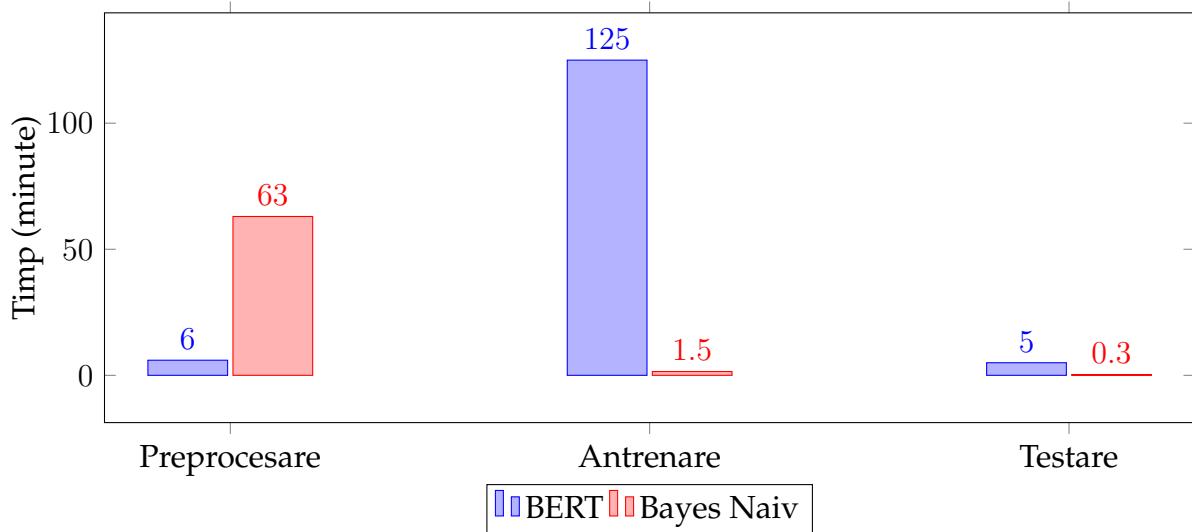


Figura 4.9: Comparație între timpii de preprocesare, antrenare și testare pentru modelele BERT și Bayes Naiv

În concluzie, atunci când construim un model de învățare automată este important să luăm în considerarea multe elemente atât teoretice cât și practice. Este necesar să alegem un set de date reprezentativ pentru problema pe care dorim să o rezolvăm, care să aibă o dimensiune considerabilă pentru a obține rezultate satisfăcătoare. Este necesar să experimentăm cu parametrii și cu modelele pe care le utilizăm pentru a găsi valorile cele mai potrivite în cazul nostru. BERT oferă rezultate superioare, dar acestea vin cu un model mult mai complex, care necesită un cost computațional mai ridicat.

# Concluzii

În concluzie, această lucrare nu propune doar o comparație teoretică a două modele de învățare automată utilizate pentru detectarea știrilor false, ci și o abordare practică, care evidențiază toti pașii esențiali în dezvoltarea unui astfel de sistem. În plus, am integrat modelele dezvoltate și într-o aplicație web împreună cu un sistem de rezumare automată.

Am obținut rezultate impresionante cu ajutorul modelului BERT, datorită valo- rificării contextul bidirecțional, al arhitecturii sale complexe și al preantrenării pe un corpus mare de texte. Însă și Bayes Naiv prezintă o performanță foarte bună având în vedere că este un model simplu ce se bazează doar pe frecvența cuvintelor și face presupunerea *naivă* de independentă condițională a trăsăturilor.

## Implementări de viitor

### Integrarea mai multor tipuri de știri

Lucrarea acesta se concentrează pe detectarea știrilor de tip propagandă, care au un limbaj puternic emotional, acestea utilizează tehnici psihologice pentru a face răspândirea știrilor false mult mai rapidă decât cea a știrilor reale. O implementare interesantă este reprezentată de un sistem capabil să recunoască și alte tipuri de știri false, precum știrile satirice, conpirațiile.

### Integrarea analizei sentimentelor

În lucrarea [10], sunt prezentate multiple metode prin care integrarea analizei sentimentelor în detectarea știrilor false poate să influențeze pozitiv performanța modelor (spre exemplu, analiza sentimentelor pozitive și negative, a faptului că în general o afirmație pozitivă este raportată într-un mod obiectiv, scorul sentimentelor,

prezența emoticoanelor vesele sau triste ). Analiza sentimentelor este un domeniu vast al învățării automate și foarte complex, însă implementarea unui set de trăsături bazate pe acest domeniu poate mări performanța modelelor.

## **Antrenarea pentru mai multe epoci**

Deși BERT a ajuns la o performanță ridicată, antrenarea pentru mai multe epoci poate duce la un rezultat și mai bun. Deoarece nu am observat semne de overfitting, antrenarea poate fi continuată până la stabilizarea pierderii. Acest proces ar fi fost foarte costisitor computațional și am obținut rezultate satisfăcătoare din primele epoci, astfel am decis că antrenarea pentru 5 epoci reprezintă un compromis acceptabil.

## **Integrarea caracteristicilor non-textuale**

Deoarece știrile nu prezintă doar date textuale, ci și imagini, videoclipuri, dezvoltarea unui sistem care să încorporeze și aceste informații este necesară. O direcție viitoare de cercetare ar putea viza construirea unui model multimodal, care combină analiza textuală cu procesarea imaginilor sau a videoclipurilor.

De asemenea, metadatele asociate știrilor, cum ar fi sursa, ora publicării, distribuția pe rețelele sociale sau autorul, pot oferi indicii importante privind veridicitatea informației. Integrarea acestor caracteristici ar putea duce la un sistem mai robust, capabil să identifice modele subtile de dezinformare.

## **Actualizarea continuă a modelului**

Deoarece vocabularul știrilor false și stilul în care sunt scrise acestea se modifică constant este necesară dezvoltarea unui sistem care să se actualizeze continuu cu date noi pentru a menține performanțele modelului în timp.

În acest sens, modelele utilizate în această lucrare, BERT și Bayes Naiv, pot fi adaptate în moduri diferite. Bayes Naiv, fiind un model probabilistic simplu, permite actualizarea incrementală a parametrilor săi prin încorporarea treptată a noilor exemple fără a fi nevoie de reantrenare completă. Acest lucru îl face potrivit pentru scenarii de învățare online. În schimb, BERT necesită o abordare mai complexă, întrucât fine-tuningul său presupune resurse computaționale semnificative, dar acest lucru nu este imposibil.

# Bibliografie

- [1] Ferreira Caceres MM, Sosa JP, Lawrence JA, Sestacovschi C, Tidd-Johnson A, Rasool MHU, Gadamidi VK, Ozair S, Pandav K, Cuevas-Lou C, Parrish M, Rodriguez I, Fernandez JP. *The impact of misinformation on the COVID-19 pandemic*. AIMS Public Health. 2022 Jan 12;9(2):262-277. doi: 10.3934/publichealth.2022018. PMID: 35634019; PMCID: PMC9114791.
- [2] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, “*Fake News Detection on Social Media: A Data Mining Perspective*”, ACM SIGKDD Explorations Newsletter, vol. 19, no. 1, pp. 22–36, 2017.
- [3] Sebastian Raschka,Vahid Mirjalili, *Python machine learning : machine learning and deep learning with python, scikit-learn, and tensorflow 2*
- [4] Daniel Jurafsky and James H. Martin. 2025, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition*. Online manuscript released January 12, 2025. <https://web.stanford.edu/jurafsky/slp3>.
- [5] Nusrat Jahan Euna and Syed Md. Minhaz Hossain, and Md. Musfique Anwar and Iqbal H. Sarker, *Content-based Spam Email Detection Using N-gram Machine Learning Approach*
- [6] Chi Sun, Xipeng Qiu, Yige Xu, Xuanjing Huang, *How to Fine-Tune BERT for Text Classification?*
- [7] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, Constantine D. Spyropoulos, *An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages*
- [8] Bo Hu,Zhendong Mao, Yongdong Zhang, *An overview of fake news detection: From a new perspective*

- [9] Peter Norvig and Stuart J. Russell, *Artificial Intelligence: a modern approach, Fourth Edition*
- [10] Miguel A. Alonso, David Vilares, Carlos Gómez-Rodríguez and Jesús Vilares, *Sentiment Analysis for Fake News Detection*
- [11] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*