**REPORT**

**Team OLD\_AI**

|  |  |
| --- | --- |
| Student | Tasks |
| Câmpean Adriana | Documentatie ,Arhitectura |
| Manoila Bogdan Liviu | Dataset ,Implementare,Evaluare |
| Man Lucia | Testare si validare |

**SemEval-2025 Task** Caracterizarea multilingvă și extragerea narativelor din știrile online. - Identificarea principalelor teme sau subiecte discutate în știri. Sumarizarea textului

[**https://github.com/manoila20/Summerize-Classify-Text**](https://github.com/manoila20/Summerize-Classify-Text)

**1.Abstract:**

Această arhitectură propune un proces complet pentru analiza și prelucrarea textului utilizând tehnici de învățare automată și prelucrare a limbajului natural (NLP). Fluxul începe cu încărcarea și pregătirea datelor, unde datele sunt încărcate dintr-un fișier CSV, iar clasele rare sunt grupate pentru a echilibra distribuția. În etapa de preprocesare a textului, sunt eliminate elementele inutile precum semnele de punctuație, emoji-urile și URL-urile, iar textul este normalizat prin conversia acestuia la litere mici. Datele sunt apoi împărțite în seturi de antrenament și testare, iar în etapa de conversie a datelor în format Hugging Face, se creează obiecte Dataset pentru antrenare.

În continuare, se configurează un model de clasificare utilizând modelul pre-antrenat DistilBERT, care este pregătit pentru antrenare folosind un tokenizer specific. Antrenarea se face în cadrul unui trainer, care este configurat cu hiperparametrii adecvați, iar performanța modelului este evaluată pe setul de testare. Rezultatele sunt vizualizate, inclusiv distribuția categoriilor și cele mai frecvente 5 clase.

Pe lângă clasificare, sistemul include și funcționalitatea de sumarizare a textului cu modelul pre-antrenat T5, care generează rezumate concise ale textelor. În final, se prezintă un sistem care permite extracția și procesarea textelor din URL-uri, generând un rezumat al unui articol și clasificându-l utilizând DistilBERT.

**2. Keywords**

* **Prelucrarea limbajului natural (NLP)**
* **Învățare automată**
* **Clasificare text**
* **DistilBERT**
* **T5**
* **Sumarizare automată**
* **Preprocesare text**
* **Tokenizare**
* **Modele pre-antrenate**
* **Împărțirea datelor**
* **Antrenare și evaluare a modelelor**
* **Distribuția categoriilor**
* **Extracție de text din URL**
* **Clasificare documente**
* **Vizualizarea datelor**

**3.Introducere**

In era digitală actuală, volumul de informație disponibil pe internet crește exponențial, iar capacitatea de a procesa și analiza aceste date într-un mod eficient devine esențială. Prelucrarea automată a limbajului natural (NLP) joacă un rol crucial în extragerea de cunoștințe din texte nestructurate, iar în acest context, tehnologiile avansate de învățare automată au demonstrat o eficiență remarcabilă.

Acest proiect propune o arhitectură completă pentru analiza și clasificarea textelor utilizând tehnici de NLP și modele de învățare automată pre-antrenate, cum ar fi DistilBERT pentru clasificare și T5 pentru sumarizare. Obiectivul principal al proiectului este de a crea un sistem care să poată prelucra texte din diverse surse (fișiere CSV, articole de pe web) și să le analizeze atât din perspectiva clasificării (prin alocarea acestora în categorii specifice), cât și a sumarizării automate (pentru a genera rezumate concise ale textelor).

Arhitectura propusă include mai multe etape fundamentale:

1. Încărcarea și pregătirea datelor, asigurându-se că datele sunt accesibile și bine structurate pentru procesare.
2. Preprocesarea textului, care include eliminarea elementelor inutile și normalizarea textului pentru a-l face compatibil cu modelele de NLP.
3. Antrenarea unui model de clasificare pe baza modelului DistilBERT, folosind seturi de date etichetate pentru a învăța să clasifice texte în categorii relevante.
4. Sumarizarea textului cu ajutorul modelului T5, care generează rezumate eficiente ale textelor de intrare.
5. Clasificarea și sumarizarea unui articol de pe un URL, oferind o soluție completă pentru procesarea textelor de pe internet.

Această arhitectură nu doar că îmbunătățește eficiența procesului de analiză a textului, dar contribuie și la soluționarea unor provocări semnificative în domeniul NLP, precum gestionarea volumelor mari de text nestructurat și îmbunătățirea preciziei clasificării și sumarizării textelor. Proiectul poate fi aplicat într-o gamă largă de domenii, de la analiza de sentimente și recunoașterea intențiilor în comunicarea digitală, până la automatizarea proceselor de analiza a documentelor și crearea de rezumate pentru articole, rapoarte și cercetări.

**4. State of the Art**

În domeniul procesării limbajului natural (NLP), modelele bazate pe arhitectura **transformer** au revoluționat modul în care sunt abordate sarcinile de clasificare, sumarizare și traducere automată. Modele precum **BERT** și **GPT** au setat noi standarde datorită capacității lor de a înțelege contextul limbajului prin învățare bidirecțională. Totuși, pentru a optimiza performanța și resursele, s-au dezvoltat variante mai ușoare ale acestor modele, precum **DistilBERT**, care păstrează precizia, dar cu un consum redus de resurse. În ceea ce privește sumarizarea automată, modelul **T5** (Text-to-Text Transfer Transformer) a dovedit o performanță excepțională, abordând toate sarcinile NLP într-o manieră uniformă și eficientă, generând rezumate coerente și concise. De asemenea, tehnicile de preprocesare, precum eliminarea semnelor de punctuație și a stopword-urilor, sunt esențiale pentru pregătirea textului înainte de utilizarea acestor modele. Aceste progrese permit aplicații extinse în domenii precum analiza sentimentelor, filtrarea limbajului toxic și automatizarea proceselor de generare și clasificare a textului.

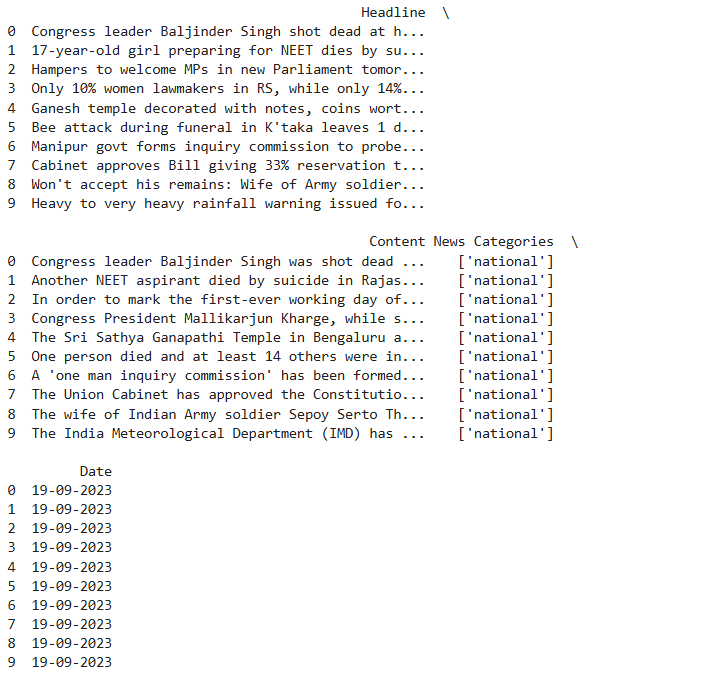
**5. Descrierea Setului de Date și Metode**

**Arhitectura proiectului**

1. **Încărcarea și pregătirea datelor**

**Descrierea datelor:** Setul de date folosit în această lucrare este dataset-ul disponibil pe platforma <https://www.kaggle.com/datasets/adhamelkomy/news-classification-and-analysis-using-nlp>

Se face o verificare a existentei fisierului CSV. In cazul in care fisierul CSV nu exista vom primi un mesaj de atentionare: Fișierul nu a fost găsit: C:\Users….

Se va afisa primele 10 randuri ale fisierului CSV care va arata in felul urmator:

Categoriile rare (care apar de mai puține ori decât pragul stabilit , in cazul nostru 5) sunt reunite într-o singură categorie „Other”. Motivul din spatele acestei acțiuni este de a limita problemele care apar, de obicei, atunci când un model este antrenat cu date rare și neesențiale sau clasificatori și când se tratează clase subreprezentate sau rare.

Astfel ca pentru clasele considerate rare se creează o nouă coloană în DataFrame, numită category\_grouped , iar etichete pentru clasele rare va fi Other.

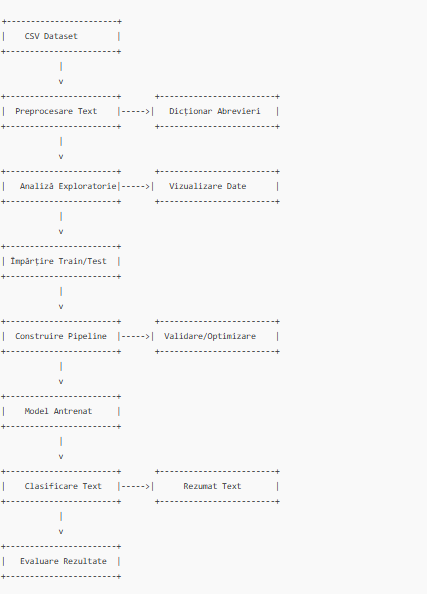
**2. Preprocesarea textului:**

Ca biblioteci pentru procesarea datelor am folosit **pandas, regex, re,** preprocesarea textului **(nltk, string),** și construirea modelului **(sklearn)**,respectiv **BeautifulSoup** pentru extragerea textului din paginile web și pipeline din biblioteca transformers pentru generarea rezumatelor.

Preprocesarea textului este un pas esențial în analiza și prelucrarea limbajului natural (NLP), care presupune curățarea și pregătirea textului pentru a putea fi utilizat într-un model de învățare automată. Scopul este de a reduce complexitatea și zgomotul din datele brute, care ar putea afecta performanța modelului astfel incatmodelul să poată învăța mai eficient și să facă predicții mai precise.

* Elimină semnele de punctuație.
* Elimină emoji-urile folosind biblioteca regex.
* URL-urile din text.
* Elimină cuvintele frecvente și irelevante din limba engleză (ex. "and", "the").
* Înlocuiește abrevierile comune (ex. "LOL" → "laugh out loud").
* textul insensibil la majuscule/minuscule.
* Integrează toate funcțiile de mai sus într-un pipeline pentru procesarea textului.

**METODA 1**



**A.Pregătirea datelor pentru modelul de invatare automata**

Avem următorul set de intrări:

X: setul de antrenament și fiecare element este un articol de știri preprocesat. Acesta conține textul pe care modelul îl va folosi pentru a învăța și a face predicții.

y: Conține categoriile relevante pentru fiecare text din X. Acesta va fi folosit de model ca etichete în timpul învățării.

Modelele de învățare automată din **sklearn** (și multe alte biblioteci) nu pot lucra direct cu etichete textuale. Pentru a transforma etichetele textuale (categorii) din y într-un format numeric pentru a putea efectua calculele matematice necesare, se utilizează clasa **LabelEncoder** din biblioteca **sklearn**, care pot fi utilizate de învățarea automata.

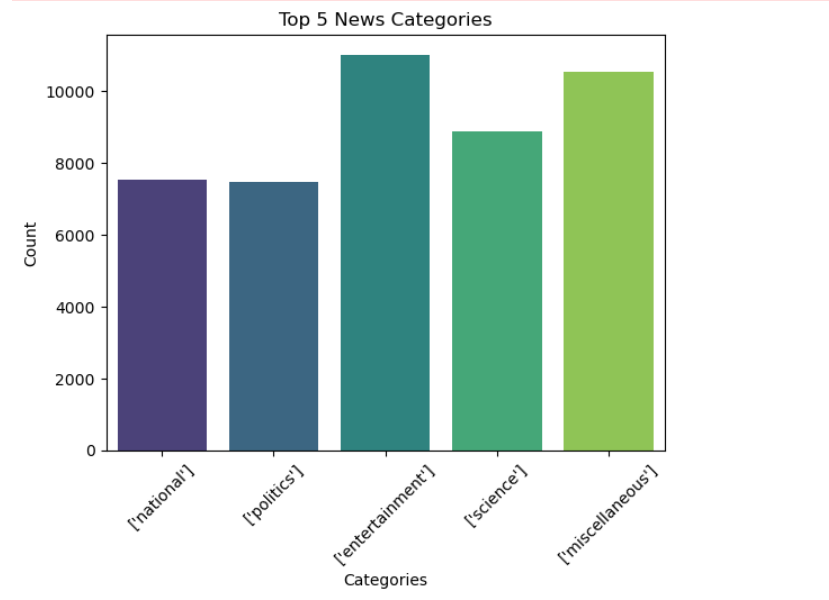
* + Rezultatul este salvat în variabila **y\_encoded**, care conține etichetele numerice.

Avem nevoie de un set de date de antrenament si un set de testare. Astfel ca impartim setul de date cu ajutorul functiei **train\_test\_split** din biblioteca **sklearn** astfel:

* + - 80% sunt alocate pentru antrenament, iar 20% din date sunt alocate setului de testare.
    - random\_state=42: E nevoie de a seta o valoare fixa pentru a avea aceleasi seturi de date de fiecare data cand se ruleaza codul.



Vor fi selectate primele 5 categorii din coloana Categorii de știri în funcție de frecvența lor (de la cea mai frecventă) și va fi afișat un grafic cu bare pentru fiecare categorie selectată care arată numărul de intrări pentru categoriile alese (Primele 5 categorii).

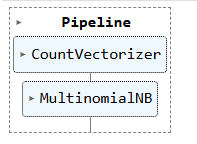
****

**B.****Construirea modelului**

**1.Model principal**

Se construiește și se antrenează modelul de clasificare folosind un **pipeline** (un lanț de pași), aplicand succesiv o serie de transformări sau modele asupra datelor din care sa rezulte vectorizarea textului si antrenarea unui model de clasificare **Naiv Bayes.**

Pipeline-ul conține două etape principale:

* + 1. **CountVectorizer():** in acest proces textul este transformat intr-un vector numeric ,care va fi utilizat de modelul de invatare automata ,folosind metoda “bag of words”. Mai exact fiecare cuvant va fi reprezentat printr-o coloana in vector(contorizeaza aparitia fiecarui cuvant in text)
    2. **MultinomialNB**(): Este un model de clasificare Naiv Bayes, folosit pentru clasificarea textului pe baza probabilităților condiționale.
       - **Multinomial Naive Bayes** este utilizat in algoritmi de clasificare a textului deoarece folosește frecventa de aparitie a cuvintelor din text (vectorizarea) pentru a calcula probabilitatea ca textul să aparțină unei anumite clase.
       - 

Se antrenează modelul pe datele de antrenament.

* + Setul de date de antrenament care conține textul preprocesat.
  + Etichetele corespunzătoare (categorii) pentru datele de antrenament.

În timpul acestui pas, modelul învață să relaționeze textul (caracteristicile numerice extrase din setul de date de antrenament cu etichetele corespunzătoare .

Pe lângă separarea caracteristicilor și etichetelor modelului, modelul este antrenat în așa fel încât să poată învăța clasele de intrare și ieșire și să facă predicții despre exemplele noi, necunoscute.

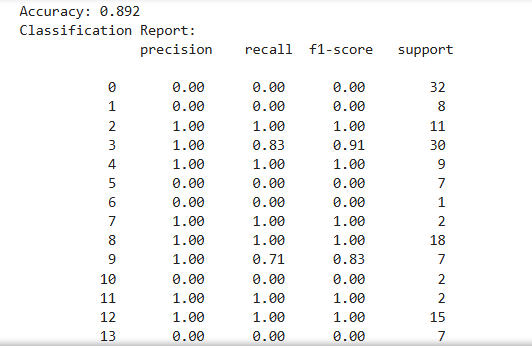
**2.Validare și optimizare**:

Se face evaluarea performanței modelului pe datele de test. Evaluarea se face prin compararea predicțiilor modelului cu etichetele reale din setul de test. La acest pas, modelul folosește textul din setul de test ,aplică aceleași transformări ca și la antrenament (prin pipeline) și apoi face predicții privind categoriile corespunzătoare.

Acuratețea predicțiilor modelului este calculate utilizand functia **accuracy\_score**() din **sklearn.metrics**.Aceasta functie returnează procentul de clasificari facute corect , adică numărul de etichete prezise corect împărțit la numărul total de instanțe si se afișează acuratețea cu o precizie de 3 zecimale.

Sereturnează un raport detaliat al performanței modelului pe fiecare clasă, incluzând următoarele măsuri:

* **Precision:** Proporția predicțiilor corecte pentru o clasă dată.
* **Recall:** Proporția instanțelor dintr-o clasă dată care sunt corect identificate.
* **F1-score:** Media armonică între precision și recall, care oferă o imagine de ansamblu asupra performanței modelului.



Implementam **validarea încrucișată (cross-validation)** pentru a evalua performanța unui model de învățare automată, în acest caz utilizând modelul antrenat anterior. Cross-validatin este o tehnica folosită pentru a estima performanța unui model pe date noi, adica împarte datele în mai multe sub-grupuri și folosește fiecare dintre ele atât pentru antrenament, cât și pentru testare. In cazul nostrum se impart datele in 3 grupuri cu conditia ca distributia etichetelor sa fie similara in fiecare subgroup reducând astfel riscul de overfitting (suprainstruire) și oferind o evaluare mai robustă a performanței modelului care se face pe baza preciziei (accuracy), adică procentul de clasificări corecte din totalul celor făcute. Rezulta o lista cu 3 scoruri de precizie, cate o valoare pentru fiecare subgroup, valoare care reprezintă performanța modelului pe unul dintre subgrupuri.Dupa care calculam media scorurilor de precizie pe toate foldurile, oferid o vedere generală a performanței modelului. Un scor mai mare indică o performanță mai bună a modelului.

Pentru a îmbunătăți performanțele modelului, se utilizează **fine-tuning** prin optimizarea hiperparametrilor folosind **RandomizedSearchCV**. Acesta ajustează parametrii din pipeline-ul modelului, inclusiv:

* Limitează numărul de trăsături extrase din text (5000, 10000 sau fără limită).
* dimensiunea n-gramurilor utilizate, cum ar fi unigrame (1,1) sau bigrame (1,2).
* Controlează regularizarea în modelul MultinomialNB, ajutând la prevenirea overfitting-ului, printr-o distribuție uniformă pe intervalul [0.1, 2.0].

Pentru validarea modelului, se folosește **StratifiedKFold** cu 5 grupuri pentru cross-validation, amestecând datele și asigurând reproducibilitatea rezultatelor. **RandomizedSearchCV** caută combinațiile optime de parametri aleatoriu, realizând 5 iterații și evaluând modelul pe baza preciziei.

La finalse obține modelul final optimizat care a fost găsit în urma procesului de fine-tuning. Modelul este cel care are parametrii optimi găsiți anterior de RandomizedSearchCV. Antrenează modelul optimizat (best\_model) pe datele de antrenament. Acesta utilizează cei mai buni parametri identificați anterior pentru a învăța din datele de antrenament**.**



**C. Evaluare clasificare și rezumare**

**1.Clasificare**

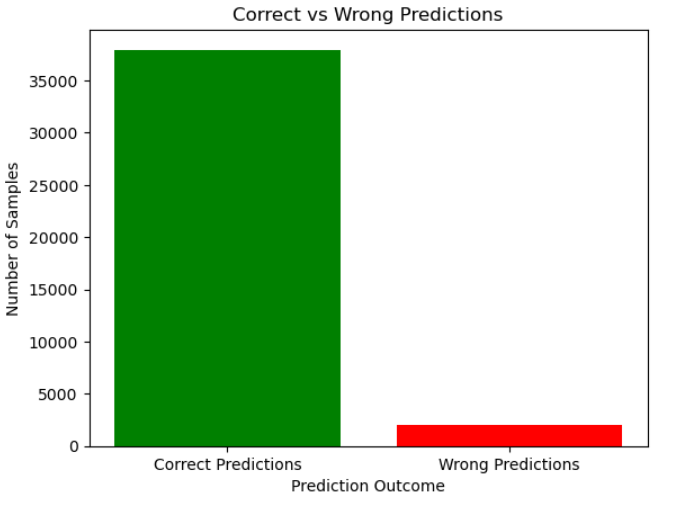
Face predicții pe setul de testare utilizând modelul antrenat anterior (best\_model). Calculul preciziei (accuracy) pentru modelul pe setul de testare. Precizia reprezintă procentajul de predicții corecte**.**

Afișează precizia modelului pe setul de testare în format cu 3 zecimale.

Pentru a oferi o intelegere mai buna a performantei modelului se calculează și afișează numărul de predicții corecte și incorecte realizate de model pe setul de test.







**2.Rezumare**

Pentru a crea un rezumat scurt al unui text mai lung se foloseste un model pre-antrenat din biblioteca transformers de la **Hugging Face**. Datele de intrare este textul continut de o pagina web care se presupune a fi o stire si returneaza un rezumat al acestuia.

**Modelul folosit:** Se folosește facebook/bart-large-cnn, un model pre-antrenat de tip BART optimizat pentru sarcini de rezumare**.**

**Pipeline: pipeline("summarization")** este un API simplificat care gestionează toate detaliile de inferență (tokenizare, procesare etc.).

-Lungimea maximă a rezumatului în token-uri sa fie 150.

**-**Lungimea minimă a rezumatului 50**.**

-Dezactivează eșantionarea aleatorie pentru a obține un rezumat determinist**.**

Rezumatul este generat folosind modelul preantrenat „facebook/bart-large-cnn” de la Huggingface.

**D. Clasificare si rezumat text web**

* Preia conținutul text dintr-o pagină web folosind BeautifulSoup.
* Curăță și preprocesează textul folosind funcțiile definite anterior.
* Clasifică textul curățat folosind modelul antrenat.
* Generează un rezumat al textului folosind model pre-antrenat din biblioteca transformers de la **Hugging Face**.

**E. Rezultat**

[*https://www.medicalnewstoday.com/articles/11-str*](https://www.medicalnewstoday.com/articles/11-str)

*ong-predictors-dementia-20-years-early-diagnosis-diabetes-obesity*

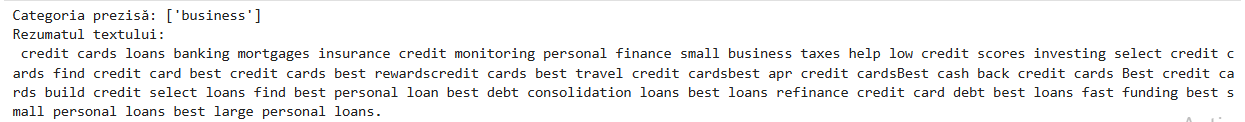
**Exemplu 1:**

*Categoria prezisă: ['Health\_\_\_Fitness']*

*Rezumatul textului:*

*New report nonprofit research organization rand cognitive ability functional limitations physical health strongly predict risk developing dementia years onset. Current uptake cognitive testing among older adults remains low. Research suggests participation would increase tests free easily accessible supported improved treatment options early detection alzheimer ’ disease related dementias especially crucial million americans age existing diseasemodifying treatments.*

**Exemplu 2:**



**METODA 2**

* 1. **Arhitectura proiectului**

**1. ÎNCĂRCAREA ȘI PREGĂTIREA DATELOR**

└──> Încarcă fișier CSV

└──> Verifică existența fișierului și afișează mostre din date

└──> Grupează clasele rare în categoria "Other" pentru echilibrare

**2. PREPROCESAREA TEXTULUI**

├──> Elimină elementele inutile:

│ ├──> Semne de punctuație

│ ├──> Emoji-uri

│ ├──> URL-uri

│ └──> Stopword-uri

└──> Normalizează textul (convertire la litere mici)

**3. ÎMPĂRȚIREA DATELOR**

├──> Împarte în seturi de antrenament și testare:

│ ├──> X\_train: Textele preprocesate

│ └──> X\_test: Textele preprocesate

└──> y\_train, y\_test: Etichetele numerice

**4. CONVERSIA DATELOR ÎN FORMAT HUGGING FACE**

├──> Creează obiecte Dataset pentru seturile de antrenament și testare

**5. CONFIGURAREA MODELULUI DE CLASIFICARE**

├──> Utilizare model pre-antrenat DistilBERT

├──> Definește tokenizer pentru pregătirea textului

└──> Tokenizează datele pentru antrenare

**6. CONFIGURAREA TRAINER-ULUI**

├──> Definește hiperparametrii pentru antrenare:

│ ├──> Numărul de epoci

│ ├──> Dimensiunea batch-ului

│ ├──> Strategia de salvare și evaluare

└──> Definește funcția pentru calcularea acurateței

**7. ANTRENAREA ȘI EVALUAREA MODELULUI**

├──> Antrenează modelul pe setul de antrenament

├──> Evaluează modelul pe setul de testare

└──> Afișează acuratețea

**8. VIZUALIZAREA DISTRIBUȚIEI CATEGORIILOR**

└──> Creează grafice pentru distribuția categoriilor și cele mai frecvente 5 categorii

**9. SUMARIZAREA TEXTULUI CU T5**

└──> Utilizează modelul pre-antrenat T5 pentru a genera rezumate ale textului

**10. CLASIFICAREA ȘI SUMARIZAREA UNUI URL**

├──> Extrage textul unui articol de pe un URL

├──> Sumarizează textul folosind T5

└──> Clasifică rezumatul cu modelul DistilBERT

**B.Pregătirea datelor pentru modelul de invatare automata**

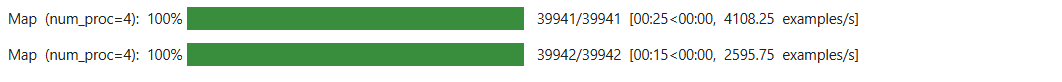
Avem nevoie de un set de date de antrenament si un set de testare. Astfel ca impartim setul de date cu ajutorul functiei **train\_test\_split** din biblioteca **sklearn** astfel:

* + - 80% sunt alocate pentru antrenament, iar 20% din date sunt alocate setului de testare.
    - random\_state=42: E nevoie de a seta o valoare fixa pentru a avea aceleasi seturi de date de fiecare data cand se ruleaza codul.

Diferenta fata de metoda 1 este ca numarul de date de antrenament si testare reprezinta doar 20% din setul de date complet.

**C****. Transformare date**

Se transformă datele text și etichetele lor într-un format compatibil cu Huggingface pentru a fi utilizate mai târziu în antrenarea unui model DistilBERT. Se converteste subsetul de texte de antrenament intr-o lista simpla de siruri , repectiv subsetul etichetele de antrenament intr-o alta lista. Formatarea este necesara pentru a aplica functia de tokenizarea pe loturi si pentru a utiliza API-ul Trainer de la Huggingface pentru antrenarea modelului.



D. **Configurarea si pregatirea modelului de clasificare:**

**Distilbert-base-uncased** – este numele modelului pre-antrenat folosit , fiind o versiune mai rapida si mai compacta a **BERT** ,utilizeaza mai putini parametri si ignora diferenta intre litere mai si mici. Se creaza tokeni (reprezentari numerice) din text folosind modelul **distilbert-base-uncased pre-antrenat** , prin impartirea textului in cuvinte ,sub-cuvinte sau caractere , se adauga tokeni speciali necesari pentru model ( **CLS** – pentru clasificare , respectiv **SEP** pentru delimitare) ,dupa care se normalizeaza textul ( se elimina diferentele intre litere mari si mici).

Se adauga un strat complet conectat de neuroni deasupra modelului **BERT** pentru a returna probabilitati pentru fiecare clasa de clasificatori pe care urmeaza sa se aplice functia Softmax.

Modelul nu poate înțelege texte brute, așa că este nevoie să le convertească într-o formă numerică astfel ca tokenizarea va procesa fiecare text din coloanal ‘Content’ din CSV. Va imparti textul in set de tokeni. E necesar ca toate textele sa aiba aceasi lungime de 128 de tokeni(prin adaugare de tokeni speciali sau prin trunchiere).Apoi fiecare token va fi mapat intru-un identificator numeric care corespunde vocabularului modelului.

Se proceseaza loturi de date simultan , utilizand 4 procese care vor rula in paralel.

* 1. **Configurarea trainer-ului:**

Argumentele necesare pentru antrenarea modelului folosind biblioteca **Huggingface Transformers** ,prin crearea unui obiect de tip **TrainingArguments**, care va fi folosit pentru a configura procesul de antrenare și evaluare:

-numarul de epoci 6 – va trece de 6 ori prin setul de date de antrenamant pentru a invata mai bine

- dimensiunea lotului 16 – la fiecare pas de antrenament ,modelul va procesa 16 texte simultan.

-dimensiunea lotului pentru evaluare model 32

-numarul de pasi de incalzire 200 – in primii 200 de pasi va creste gradual rata de invatare

- termen de regularizare 0.01 – ajuta la prevenirea suprainvatarii prin ajustarea greutatilor mari ale modelului.

-dupa fiecare 10 pasi de antrenament va fi salvat un log al pregresului

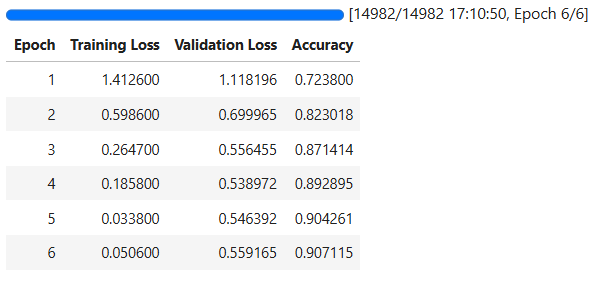
-evaluarea va fi facuta la finalul fiecarei epoci , iar modelul va fi salvat

- doar ultimele 2 modele vor fi salvate. Se face o suprascriere a modelelor vechi.

**F. Antrenarea și evaluarea modelului:**

Se evaluaza performanța modelului pe baza etichetelor prezise și adevărate, calculând **acuratețea** (procentul de predicții corecte). Acesta este un pas esențial pentru a înțelege cât de bine performează modelul pe setul de test. Pentru a face acest lucru se utilizeaza funcția **accuracy\_score** din **sklearn.metrics** pentru a calcula procentajul de exemple pentru care modelul a prezis corect clasa. Cand eticheta prezisa corespunde cu eticheta reala avem o predictie corecta.

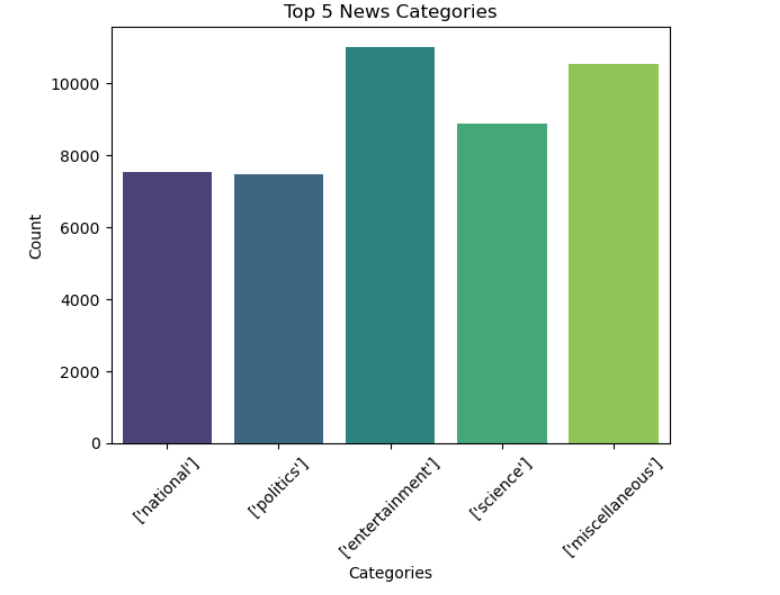
**Trainer** din biblioteca **Huggingface Transformers** este folosit pentru antrenarea si evaluarea modelului. Acesta va itera prin setul de antrenament si va ajusta parametrii modelului in functie de eroarea de predictie.La sfarsitul fiecarei epocii ,Trainer va evalua performanta modelului pe setul de testare.

Se utlizeaza metoda evaluate() a obiectului **Trainer** pentru a evalua modelul pe setul de date de testare. Aceasta metoda returneaza si afiseaza acuratetea cu 3 zecimale pe setul de date. 



**G. Vizualizarea distribuției categoriilor**

Vor fi selectate primele 5 categorii din coloana Categorii de știri în funcție de frecvența lor (de la cea mai frecventă) și va fi afișat un grafic cu bare pentru fiecare categorie selectată care arată numărul de intrări pentru categoriile alese (Primele 5 categorii).



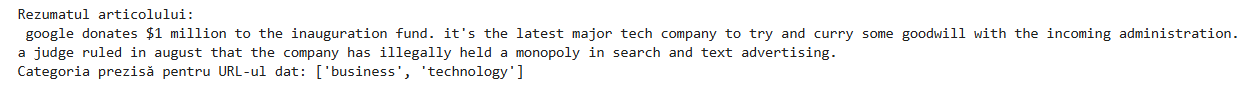
**H.** **Sumarizarea textului cu T5**

Se utilizează modelul pre-antrenat **T5** pentru a genera rezumate ale textelor. Modelul T5 este specializat în a transforma orice sarcină de prelucrare a limbajului natural într-o problemă de tip "text-to-text", iar în acest caz specific, sarcina este sumarizarea unui text. Am ales modelul T5 small pre-antrenat pentru rapiditate si consumul mai mic de resurse. Acesta tokenizeaza textul de intrare (in converteste intr-o secventa de token-uri pe care modelul le intelege) , urmand ca la final dup ace modelul considera ca a gasit un rezumat adevcat sa decodifice rezultatul obtinut de la model intrun text lizibil.

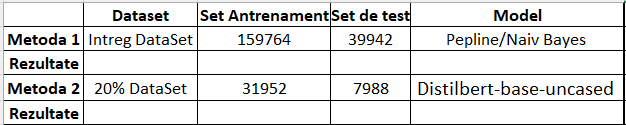
**I.Clasificarea și sumarizarea unui URL**

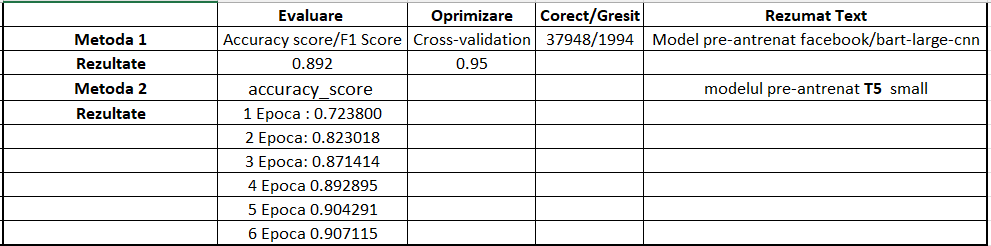
* Extrage textul unui articol de pe un URL.
* Sumarizează textul folosind modelul T5.
* Clasifică rezumatul utilizând modelul DistilBERT.

**J.Rezultate**



1. **Comparatie intre metode**

****

****

**7.Concluzii**

Arhitectura procesului textual descrisă în acest proiect oferă o soluție completă pentru prelucrarea și analiza automată a textelor, integrând etape esențiale precum încărcarea și pregătirea datelor, preprocesarea textului, antrenarea și evaluarea unui model de clasificare, și generarea de rezumate. Această arhitectură asigură o abordare eficientă și scalabilă pentru a procesa texte din diverse surse, prin utilizarea unor modele pre-antrenate de ultimă generație, cum ar fi DistilBERT pentru clasificare și T5 pentru sumarizare.

Procesul de fine-tuning al modelului, prin optimizarea hiperparametrilor cu ajutorul tehnicii RandomizedSearchCV, a permis ajustarea fină a parametrilor esențiali pentru îmbunătățirea performanței, reducerea riscului de overfitting și creșterea preciziei clasificarilor. De asemenea, StratifiedKFold a asigurat o validare corectă și echilibrată, minimizând eventualele erori de evaluare.

Arhitectura implementată include și un modul de vizualizare a distribuției categoriilor, facilitând interpretarea rezultatelor, și un mecanism de sumarizare a textului folosind modelul T5, care permite generarea automată a rezumatelor pentru articole și documente. Acest flux de lucru integrat contribuie semnificativ la automatizarea proceselor de analiză a textelor și poate fi aplicat într-o gamă largă de domenii, inclusiv analiza sentimentelor, detectarea discursului instigator la ură, și procesarea automată a documentelor.

În ansamblu, arhitectura propusă adresează provocările majore ale prelucrării textului nestructurat și furnizează un cadru robust pentru implementarea și evaluarea soluțiilor NLP, având potențialul de a fi extinsă și aplicată în moduri variate și inovative.

**8.Bibliografie**

**1.Raffel, C., Shinn, C., Roberts, A., et al.** (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. arXiv. <https://arxiv.org/abs/1910.10683>

2. **Devlin, J., Chang, M. W., Lee, K., & Toutanova, K.** (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv. <https://arxiv.org/abs/1810.04805>

3. **Hugging Face** (2021). *Transformers: State-of-the-art Natural Language Processing*. <https://huggingface.co/transformers/>

4. **Lample, G., & Charton, C.** (2019). *Deep Learning for Text Classification: A Survey*. arXiv. <https://arxiv.org/abs/1905.02748>

5. **Sanh, V., Debut, L., Chaumond, J., & Wolf, T.** (2019). *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter*. arXiv. <https://arxiv.org/abs/1910.01108>

6. <https://www.kaggle.com/code/adhamelkomy/news-classification-and-analysis>

8. **Jake Vasilakes, Zhixue Zhao, Ivan Vykopal, Michal Gregor,Martin Hyben**, **and Carolina Scarton1** ExU: AI Models for Examining Multilingual Disinformation Narratives and Understanding their Spread

9. **Hsin-Hsi Chen, Chuan-Jie Lin** A Multilingual News Summarizer

10. **Andreea Iana, Goran Glavaš, Heiko** Paulheim MIND Your Language: A Multilingual Dataset for Cross-lingual News Recommendation

11. **David A. Hull ,Gregory Grefenstette** Querying Across Languages: A Dictionary-Based Approach to Multilingual Information Retrieval

12. **Bianca Cavo Figueres ,Tommaso Caselli,Marcel Broersma(2021)** DHQ: Digital Humanities Quarterly Finding Narratives in News Flows: The Temporal Dimension of News Stories