Universitatea Politehnica București
Facultatea de Automatică și Calculatoare
Departamentul de Automatică și Informatică Industrială

LUCRARE DE DIPLOMĂ

Identificarea modificărilor stilistice în operele lui Shakespeare și Fletcher prin tehnici de învățare automată supervizată

Coordonator
Ș.l. dr. ing. Băltoiu Andra-Elena

Absolvent
Boriceanu Ioana-Roxana

2023

Cuprins

1 Introducere

Problema identificării modificărilor stilistice într-un text este prezentă în tot mai multe lucrări din domeniul prelucrării limbajului natural datorită numărului mare de aplicații pe care le poate rezolva. În prezent, acest subiect este abordat în cercetări al căror scop este analiza textului pentru a determina diferite aspecte legate de autorul acestora. Un exemplu de astfel de studiu este cel în care se dorește identificarea autorului unui text scris sub anonimat [33]. O altă posibilă aplicație este cea a detectării plagiatului [3]. În această lucrare ne propunem să identificăm modificările stilistice ce apar într-un text, despre care se cunoaște faptul că este rezultatul unei colaborări, pentru a determina care sunt secvențele scrise de fiecare autor.

Pentru a realiza acest lucru a fost nevoie să parcurgem mai multe etape pe care le vom prezenta în continuare. Înainte de a propune o soluție, am analizat abordările recente pentru a identifica atât tehnicile folosite până în prezent, cât și rezultatele obținute în urma utilizării acestora. Soluția propusă de noi constă în rezolvarea unei probleme de învățarea automată supervizată, respectiv clasificare, și în selectarea unor trăsături stilistice relevante prin care să putem defini maniera de scriere a autorului, pentru obținerea unor performanțe cât mai bune.

În Capitolul 2 prezentăm domeniul din care face parte problema abordată. Un scurt istoric al stilometriei, care este o problemă de lingvistică și ale cărei concepte sunt folosite împreună cu tehnici de învățare automată pentru rezolvarea problemei de identificare a autorului, este prezentat în Secțiunea 2.1. Aplicații ale stilometriei sunt prezentate în continuare în Secțiunea 2.2, iar stadiul actual al utilizării metodelor de învățare automată în Secțiunea 2.3.

În Capitolul 3 am descris în detaliu problema abordată și soluția propusă. În introducerea acestui capitol sunt prezentate obiectivele lucrării. Secțiunea 3.1 detaliază problema avută în vedere pentru prezenta lucrare, anume identificarea modificărilor stilistice pentru identificarea autorului în piesele lui William Shakespeare scrise în colaborare cu dramaturgul John Fletcher, prin tehnici de clasificare supervizată. Tot aici sunt prezentate și alte lucrări în care a mai fost abordată această problemă. Secțiunea 3.2 conține descrierea etapelor de dezvoltare pe care le-am urmat pentru a atinge obiectivele lucrării. Detalii despre implementarea soluției propuse se găsesc în Secțiunea 3.3.

Etapa de procesare și reprezentare a textului este descrisă în Capitolul 4. Pentru rezolvarea problemei a fost nevoie mai întâi să pregătim setul de date, proces care a constat în eliminarea informațiilor irelevante din fișierele text și adnotarea scenelor cu numele autorului corespunzător, conform descrierii din Secțiunea 4.1. Apoi a urmat etapa de segmentare a textelor la nivel de scenă. În urma analizei literaturii de specialitate, am extras o serie de trăsături, prezentate în Secțiunea 4.2. Aici am întâmpinat câteva dificultăți datorită dimensiunii mici a bazei de date deoarece, pentru algoritmii de învățare automată selectați, atunci când numărul de semnale este mai mic decât numărul de trăsături, se obțin performanțe foarte scăzute. Problema poartă numele de "curse of dimensionality" și este prezentată tot în această secțiune. Astfel, am ales să segmentăm textul la nivel de frază pentru a crește numărul de semnale în încercarea de a diminua efectele fenomenului menționat anterior asupra performanțelor modelelor. Apoi, pentru a determina care dintre trăsături sunt mai relevante pentru problema analizată am creat 23 de configurații de testare, descrise în Secțiunea 4.3. De asemenea, tot în încercarea de a îmbunătăți rezultatele modelelor și de a rezolva problema apărută datorită dimensiunii mici a bazei de date, menționată anterior, am implementat o serie de metode de procesare a setului de trăsături, precum metode de scalare, metode de selecție a caracteristicilor relevante și o metodă

de reducere a dimensiunii semnalelor. Descrierea modului lor de funcționare se găsește în Secțiunea 4.4.

Pentru etapa de antrenare și testare am utilizat șapte algoritmi de clasificare, descriși pe larg în Capitolul 5. Pentru a analiza performanțele acestora, fiecare model a fost antrenat pe toate cele 23 de configurații de testare și utilizând toate metodele de procesare a setului de trăsături. De asemenea, am analizat influența parametrilor specifici metodelor de învățare automată asupra rezultatelor și impactul pe care îl are aplicarea metodelor de preprocesare a setului de trăsături asupra performanțelor modelelor. Rezultatele obținute în urma experimentelor sunt prezentate în Capitolul 6.

Capitolul 7 prezintă concluziile lucrării și ce ne propunem să realizăm într-o etapă viitoare.

2 Domeniu

"Style is the answer to everything / A fresh way to approach a / Dull or dangerous thing / To do a dull thing with style is preferable / To doing a dangerous thing without it / To do a dangerous thing with style / Is what I call art."- Charles Bukowski ("Style", 1974, [12])

Stilul poate fi caracterizat prin mai multe definiții, una dintre ele fiind ilustrată în poezia "Style" a lui Charles Bukowski, din care am extras strofa de mai sus. În lucrarea sa "A theory of style" [1], James Ackerman definește stilul astfel: "mod de a caracteriza relațiile dintre operele de artă care au fost create în aceeași perioadă și/sau în același loc, sau de aceeași persoană sau grup.". De-a lungul timpului stilul a fost adesea folosit ca determinant principal pentru delimitarea perioadelor istorice în artă prin extragerea unor trăsături din opere și gruparea celor care prezentau elemente comune.

Plecând de la ideea că stilul reprezintă o modalitate de a caracteriza o operă de artă sau orice alt tip de creație, putem merge mai departe și presupune că fiecare persoană are o viziune diferită asupra lucrurilor și o manieră individuală, chiar inconștientă, de a se exprima. De aceea, identificarea și analizarea trăsăturilor stilistice specifice joacă un rol important în cercetările ce urmăresc să înțeleagă sau să interpreteze modul în care un autor și-a exprimat viziunea prin intermediul creațiilor sale.

Stilometria este ramura care se ocupă cu analiza textelor prin utilizarea unor metode statistice și computaționale, în general, cu scopul de a determina diferite informații despre autor.

În continuare, vom prezenta modul în care s-a dezvoltat acest domeniu, care sunt aplicațiile în care se utilizează tehnici specifice și stadiul actual al cercetărilor.

2.1 Scurt istoric

Stilometria apare ca problemă încă din anul 1851, conform [21]. Matematicianul Augustus de Morgan afirmă într-o scrisoare către unul din prietenii săi: "Mă aștept ca un om care scrie despre două subiecte diferite să fie mai de acord cu el însuși decât doi oameni diferiți care scriu despre același subiect. Într-o bună zi, scrierile false vor fi detectate prin acest test." [41]. El sugerează rezolvarea problemelor de atribuire a autorului prin intermediul frecvenței cuvintelor în funcție de lungimea lor.

Ipoteza lui a fost cercetată de Thomas Mendenhall în anii 1880. Acesta a încercat să folosească distribuția lungimilor de cuvinte, încercând să găsească trăsături similare între operele dramaturgilor Bacon, Marlowe și Shakespeare, pentru a determina adevăratul autor al operelor celui din urmă. Deși din studiile lui Mendenhall reiese că pentru a rezolva o problemă de identificare a autorului lungimea cuvintelor nu este un criteriu suficient, lucrările lui au stârnit interesul mai multor cercetători în acest domeniu. Astfel, în anul 1932, lingvistul american George Kingsley Zipf formulează o lege care îi poartă numele, prezentată în [55]. Aceasta are la bază ideea că într-un corpus frecvența unui cuvânt este invers proporțională cu poziția sa într-un clasament al frecvențelor cuvintelor din acel corpus. În anul 1944 statisticianul britanic George Udny Yule prezintă o măsură numită Caracteristica K a lui Yule, care face legătura între media și varianța numărului de apariții ale cuvintelor într-un text. Caracteristica K a lui Yule este prezentată mai amănunțit în [90]. În [22], David Hoover analizează câteva dintre metodele prezentate mai sus și ajunge la concluzia că, folosite separat, niciuna dintre ele nu reprezintă un

criteriu suficient pentru a distinge două texte. Cu toate acestea, ideea din spatele lor, aceea că fiecare autor are un anumit stil specific care poate fi caracterizat de un set de trăsături pe baza cărora se poate diferenția de alți autori, reprezintă fundamentul metodelor de analiză stilistică existente în prezent.

Lucrarea [51] în care Frederick Mosteller și David L. Wallace aplică tehnici de analiză statistică pentru a determina contribuțiile fiecărui autor al Articolelor Federaliste ("The Federalist Papers" de Alexander Hamilton, James Madison, John Jay, publicate în anul 1788) este considerată, așa cum este menționat și în [21], a fi cea care a pus bazele analizei computerizate a stilometriei.

Ulterior problemele de stilometrie au fost abordate prin diverse metode de învățare automată. Un impact important l-au avut rețelele neuronale. Robert Matthews și Tom Merriam au fost printre primii cercetători care au încercat să folosească rețele neuronale, în lucrarea lor [42], pentru a rezolva problema clasică de atribuire a autorului pentru operele lui William Shakespeare. O altă abordare care a apărut în anul 1995 este prezentată în [20]. Holmes și Forsyth folosesc algoritmi genetici pentru a rezolva problema identificării autorilor pentru Articolele Federaliste. Toate acestea și multe altele au făcut posibilă dezvoltarea domeniului până în stadiul în care este astăzi.

2.2 Aplicații

Așa cum am menționat și în Secțiunea 2.1, o problemă clasică a cărei rezolvare implică identificarea elementelor stilistice este aceea de a determina autorul unui text în diferite contexte. Dezvăluirea identității scriitorilor care aleg să publice cărți sub anonimat sau utilizând pseudonime este unul dintre aspectele foarte des abordate prin intermediul stilometriei. În general această problemă se rezolvă prin compararea textului scris sub anonimat cu o mulțime de texte al căror autor este cunoscut. Un exemplu de astfel de "demascare" a autorului este opera literară "Chemarea cucului" ("The Cuckoo's Calling"). Publicată în anul 2013 sub pseudonimul Robert Galbraith, aceasta este una dintre operele literare al cărei autor a fost identificat prin intermediul analizei stilistice. În [33], un grup de cercetători de la Universitatea din Birmingham, conduși de Patrick Juola, au concluzionat că autorul textului este de fapt scriitoarea J.K. Rowling. O altă problemă încă dezbătută este aceea a identității autorului/autoarei ce se află în spatele scrierilor semnate sub pseudonimul Elena Ferrante. O parte din lucrările ce abordează această problemă au fost strânse de Arjuna Tuzzi și Michele Cortelazzo în anul 2017, în cartea [80]. Deși există mai multe ipoteze referitoare la adevărata identitate a Elenei Ferrante, niciuna nu a fost demonstrată până în prezent.

Tot în această categorie se încadrează și aplicațiile care își propun detectarea plagiatului. Există o gamă variată de probleme rezolvate prin intermediul extragerii trăsăturilor stilistice precum: detectarea unor secvențe plagiate dintr-un text scris, un exemplu este prezentat în [3], sau detectarea secvențelor copiate dintr-un cod software [74].

O altă aplicație, tratată în general ca o problemă de clasificare binară, este aceea a verificării autorului unui text. Aceasta are scopul de a determina dacă două texte au fost scrise sau nu de aceeași persoană. În [37], este prezentată o modalitate de rezolvarea a problemei în care se generează o serie de documente "impostor" și se verifică, pe baza extragerii unor trăsături, dacă textul pe care încercăm să îl verificăm este similar sau nu cu unul din textele generate. Există și situații în care se dorește construirea unui profil al autorului textului, nu identificarea lui. Acest lucru se realizează prin analiza documentului cu scopul determinării informațiilor de tip sociologic sau psihologic despre autor ([59], [81], [82]).

Stabilirea ordinii cronologice în care au fost scrise anumite texte reprezintă altă temă de studiu din acest domeniu. O descriere a modului în care a evoluat această subramură se găsește în [71]. Printre lucrările supuse analizei stilistice pentru determinarea succesiunii cronologice se

regăsesc dialogurile lui Platon [8] și operele dramatice ale lui Euripide [38].

Deși stilometria a apărut din nevoia de a rezolva diferite probleme ce implicau prelucrarea textelor, de-a lungul timpului diferite caracteristici ale acestui domeniu au început să fie folosite pentru identificarea unor trăsături stilistice în alte sfere artistice, precum muzica și artele vizuale. Din această categorie fac parte studiile în care datele utilizate sunt de tip audio ([4], [6], [10]) sau de tip vizual ([28], [57]). Un alt exemplu interesant este [79], în care se utilizează metode folosite în general în aplicații ce au ca scop identificarea autorului, pentru a detecta cadre neobișnuite în videoclipuri.

2.3 Stadiul actual

Datorită numărului mare de aplicații pe care își propune să le rezolve și a progresului tehnicilor de învățare automată, stilometria a devenit o zonă importantă de interes pentru specialiștii în prelucrarea limbajului natural. În prezent, unele dintre cele mai importante lucrări din acest domeniu sunt rezultate în urma rezolvării sarcinilor propuse în competițiile organizate de PAN[1]. În continuare vom prezenta câteva lucrări care utilizează diferite tehnici de învățare automată pentru a rezolva problemele ce urmăresc identificarea modificărilor stilistice într-un text. Majoritatea articolelor au fost selectate din setul publicat de PAN, în funcție de trăsăturile selectate pentru rezolvarea problemei, metoda propusă și rezultatele obținute.

2.3.1 Trăsături stilistice

Pentru rezolvarea unei probleme de analiză a textului, este extrem de importantă selectarea trăsăturilor stilistice corespunzătoare. Așa cum am prezentat și în Secțiunea 2.1, au existat încă de la început numeroase încercări de a determina care sunt cele mai eficiente atribute pentru a defini stilul unui autor (vezi de exemplu [45], [90] sau [55]). Conform [11], putem clasifica tipurile de trăsături după cum urmează:

1. Trăsături lexicale
Trăsăturile lexicale ilustrează în general alegerile pe care le face autorul legat de cuvintele pe care le utilizează, frecvența acestora și structura gramaticală a textului. Câteva exemple de astfel de trăsături includ n-gramele de cuvinte, forma sau lungimea cuvintelor, frecvența părților de vorbire, TF-IDF[2].

2. Trăsături sintactice
Trăsăturile sintactice ilustrează maniera în care autorul organizează elementele de limbă într-o propoziție sau frază. Acestea se concentrează asupra modului în care acest aspect influențează semnificația textului. Un exemplu pentru această categorie este felul în care sunt asociate cuvintele pentru a forma înțelesul global al unei fraze.

3. Trăsături specifice aplicației
În majoritatea cazurilor selecția trăsăturilor se face în funcție de problema abordată, pe baza analizei performanțelor obținute.

[1] Plagiarism, Authorship, and Social Software Misuse (PAN) este o serie de competiții anuale, organizate în cadrul Conference and Labs of the Evaluation Forum (CLEF), ale căror task-uri abordează diverse probleme din domeniul stilometriei.

[2] Term Frequency - Inverse Document Frequency este o metodă utilizată pentru a determina cât de relevantă este o structură (în general un cuvânt sau o frază) într-un document în raport cu corpusul (setul de documente) din care face parte documentul [63].

Există lucrări în care s-au utilizat doar un anumit tip de trăsături, de exemplu [44], [49], [36], [62] în care se folosesc trăsături lexicale. În [52], Nath folosește drept trăsătură definitorie cei mai frecvenți 50 de termeni, împreună cu observația că în unele documente există fraze care se repetă. Acesta reușește, propunând o abordare nesupervizată, să obțină cele mai bune rezultate pentru identificarea modificărilor stilistice, în competiția organizată de PAN în 2019. Rezultate foarte bune au obținut și [31] și [86], care folosesc BERT[3] pentru a segmenta textul și a genera reprezentări vectoriale utilizate mai apoi în antrenarea modelului. Anul trecut, în [32], Jiang et al. au utilizat ELECTRA[4] cu același scop.

O metodă mai des întâlnită este folosirea unor combinații ale tipurilor de trăsături. O astfel de abordare a fost utilizată și în [88], care selectează o serie de caracteristici în funcție de rezultatele obținute în timpul antrenării modelelor alese. Prin această tehnică, Zlatkova et al, reușesc să obțină cele mai bune rezultate în competiția organizată de PAN în 2018, pentru determinarea modificărilor stilistice într-un text. Această metodă a mai fost folosită și în [11], [91], [14], [68]. În [72], sunt combinate metodele prezentate în [31], [91] și [88] cu rezultate foarte apropiate de cele ale lucrării câștigătoare [84].

2.3.2 Algoritmi de clasificare

În competițiile organizate de PAN, problema identificării modificărilor stilistice într-un text a luat mai multe forme de la prima ediție și până în prezent. Ideea principală este aceeași în fiecare an: dat fiind un document, să se determine dacă este scris de unul sau mai mulți autori [78]. Există și o excepție în anul 2020, când cerința a fost să se determine dacă există modificări stilistice între două paragrafe consecutive [85].

Pentru rezolvarea acestor aplicații au fost propuse numeroase soluții. Dintre articolele care folosesc rețele neuronale pentru rezolvarea problemei se remarcă [24], care utilizează Parallel Hierarchical Attention Networks (PHAN), obținând rezultate foarte bune în rezolvarea problemei propuse de PAN în anul 2018. În același an, [64] încearcă să rezolve problema prin utilizarea Bidirectional Echo State Network (BESN), obținând însă performanțe mai slabe decât celelalte echipe de cercetători. O altă abordare în care se utilizează rețele neuronale este propusă de Nath în [52], care rezolvă problema prin Siamese Neural Network (SNN) și obține cele mai bune rezultate în competiția organizată de PAN din 2019. În 2021 cele mai bune performanțe sunt obținute de [86], care utilizează un model de clasificare de tip Fully Connected Neural Network (FCNN). În 2022, trei dintre lucrări folosesc astfel de metode, [87], [39] și [32].

Zlatkova et al, în [88], reușesc să obțină cele mai bune rezultate în competiția din 2018. Aceștia încearcă o serie de algoritmi de învățare automată supervizată precum: Support Vector Machine (SVM), Random Forest, AdaBoost[5], MLP[6] și LightGBM[7]. LightGBM fiind cel prin intermediul căruia se obțin performanțe remarcabile.

În mod asemănător, echipa câștigătoare din 2019, încearcă o serie de algoritmi de învățare automată printre care: regresia logistică, arbori de decizie, Random Forest, SVM și Naive Bayes. Obținând cele mai bune performanțe pentru algoritmul Random Forest [31].

Tehnici de învățare automată nesupervizată au fost utilizate în [36], [14], [68]. Rezultate remarcabile s-au obținut în [34] și [52].

[3] Bidirectional Encoder Representations from Transformers (BERT), este un model preantrenat de procesare a limbajului natural, dezvoltat de Google în anul 2018 [17].

[4] Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA), este un model de preprocesare a limbajului natural, dezvoltat de cercetătorii Google AI Language, prezentat în [16].

[5] Adaptive Boosting, este un algoritm de învățare automată supervizată, utilizat în probleme de clasificare [18].

[6] Multi-Layer Perceptron (MLP), este o arhitectură de rețele neuronale cu mai multe straturi, utilizate în general pentru probleme de clasificare și regresie [61].

[7] LightGBM este o bibliotecă open-source pentru învățarea automată, care se bazează pe algoritmul Gradient Boosting Machine (GBM) [35].

3 Problema analizată și soluția propusă

Scopul final al acestei lucrări este dezvoltarea unui sistem capabil să identifice autorii unui text rezultat în urma unei colaborări prin identificarea modificărilor stilistice.

Un prim obiectiv al lucrării este selectarea trăsăturilor adecvate acestui tip de problemă, pentru obținerea celor mai bune performanțe posibile în procesul de detectare a modificărilor stilistice.

Un alt obiectiv ar fi identificarea celor mai potrivite metode de învățare automată supervizată pentru rezolvarea problemei propuse. Într-o primă etapă, vom lua în considerare rezultatele obținute pentru a determina care dintre aceste metode și categorii de trăsături sunt mai potrivite. Pentru a evalua performanța modelelor în experimentele efectuate vom utiliza metrici precum acuratețea, precizia, sensibilitatea și scorul F1.

Acuratețea este măsura folosită în problemele de învățare automată pentru a determina numărul total de predicții corecte (True Positive + True Negative), din totalul de răspunsuri generate (True Positive + True Negative + False Negative + False Positive)

$$Acuratete = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Precizia este o altă măsură utilizată în problemele de învățare automată ce indică numărul de instanțe clasificate corect ca Positive (True Positive) raportat la numărul total de instanțe clasificate drept Positive (True Positive + False Positive)

$$Precizie = \frac{TP}{TP + FP} \tag{3.2}$$

Sensibilitatea (recall) în problemele de învățare automată este o măsură utilizată pentru a ilustra capacitatea modelului de a identifica toate exemplele pozitive dintr-un set de date. Această metrică este definită ca raportul dintre numărul de exemple clasificate corect ca Positive (True Positive) și numărul total de exemple pozitive din setul de date, adică suma dintre numărul de exemple identificate corect ca Positive și numărul de exemple identificate greșit ca Negative (True Positive + False Negative)

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

Scorul F1 este definit ca media armonică dintre precizie și sensibilitate (recall). Un model ce are o precizie ridicată și o sensibilitate scăzută clasifică corect un număr mare de exemple Negative, dar clasifică greșit multe exemple ce sunt Positive. Dacă acesta are însă o precizie scăzută și o sensibilitate mare, înseamnă că identifică greșit mai multe exemple Negative ca fiind Positive, dar detectează un număr mai mare de exemple ce sunt într-adevăr Positive. Scorul F1 este folosit pentru a ilustra relația dintre precizie și sensibilitate. În general, este util atunci când se lucrează cu seturi de date neechilibrate, în care numărul de instanțe din diferite clase este semnificativ diferit

$$F1\_Score = 2 * \frac{precizie * recall}{precizie + recall} \tag{3.4}$$

3.1 Descrierea problemei analizate

Așa cum am menționat și în secțiunile anterioare, una dintre cele mai reprezentative baze de date utilizate în aplicații ale stilometriei este formată din operele lui William Shakespeare. Pentru experimentele realizate am ales operele "Henry VIII" și "The Two Noble Kinsmen", despre care se cunoaște faptul că au fost scrise în colaborare cu dramaturgul John Fletcher. În acest caz particular, plecând de la rezultatele obținute de specialiști de-a lungul timpului, sistemul ar trebui să clasifice corect la final scenele scrise de Fletcher și scenele scrise de Shakespeare.

Autorul operei "The Two Noble Kinsmen", ale cărei prime reprezentații au avut loc în anul 1613, a devenit un subiect dezbătut încă din 1634, când John Waterson publică o primă ediție a piesei cu următoarea inscripție: "Written by the memorable Worthies of their time; Mr. John Fletcher, and Mr. William Shakespeare—Gent" [21]. În cartea sa [5], publicată în anul 1857, Delia Bacon afirmă că autorul adevărat al operelor lui Shakespeare este de fapt reprezentat de un grup de dramaturgi. Cartea ei rămâne una dintre cele mai importante lucrări în care se dezbate acest subiect.

Printre primele lucrări care abordează aceasta problemă se regăsesc: [45] în care Thomas Mendenhall testează ipoteza lui de Morgan, așa cum a fost menționat și în Secțiunea 2.1, urmat de Merriam care aplică metoda lui Morton [50] în mai multe dintre lucrările sale în care analizează textele lui Shakespeare ([47], [48], [46]). O altă lucrare importantă este [42], în care Matthews și Merriam folosesc pentru prima dată rețele neuronale pentru a analiza textele lui Shakespeare. Într-o primă lucrare, aceștia antrenează mai întâi rețeaua pentru a analiza piesa "The Two Noble Kinsmen" și ajung la concluzia că opera a fost într-adevăr rezultatul unei colaborări cu John Fletcher. Într-o a doua lucrare, rețeaua este antrenată pe operele lui Shakespeare și Marlow cu scopul de a determina dacă opera "Edward II" a fost într-adevăr o colaborare între cei doi. În 1994, Ledger și Merriam publică lucrarea [40], în care folosesc ca trăsătură principală frecvența de apariție a literelor. Analizând opera "The Two Noble Kinsmen" cu această metodă, cei doi obțin o posibilă alocare a scenelor în funcție de autor.

O abordare mai recentă a acestei probleme este prezentată în [66], în care un grup de cercetători de la Universitatea din Pennsylvania folosesc o metodă bazată pe rețelele de adiacență a cuvintelor (Word Adjacency Networks). Ei selectează ca trăsătură definitorie a stilului fiecărui autor distanța dintre cuvintele funcționale (stop words). Aplicând această metodă pe operele lui Shakespeare, ajung la concluzia că cele trei părți ale operei "Henry VI" reprezintă excepții din punct de vedere statistic și că cel mai probabil au fost scrise în colaborare cu Christopher Marlowe sau George Peele [30]. Petr Plecháč analizează textele lui Shakespeare utilizând frecvențele structurii ritmice și a celor mai frecvente cuvinte [56]. În urma experimentelor efectuate, acesta ajunge la concluzia că "Henry VIII" este într-adevăr rezultatul unei colaborări între William Shakespeare și John Fletcher.

3.2 Descrierea soluției propuse

Pentru a atinge obiectivele lucrării, prezentate anterior, am parcurs mai multe etape de dezvoltare, ilustrate în Figura 3.1.

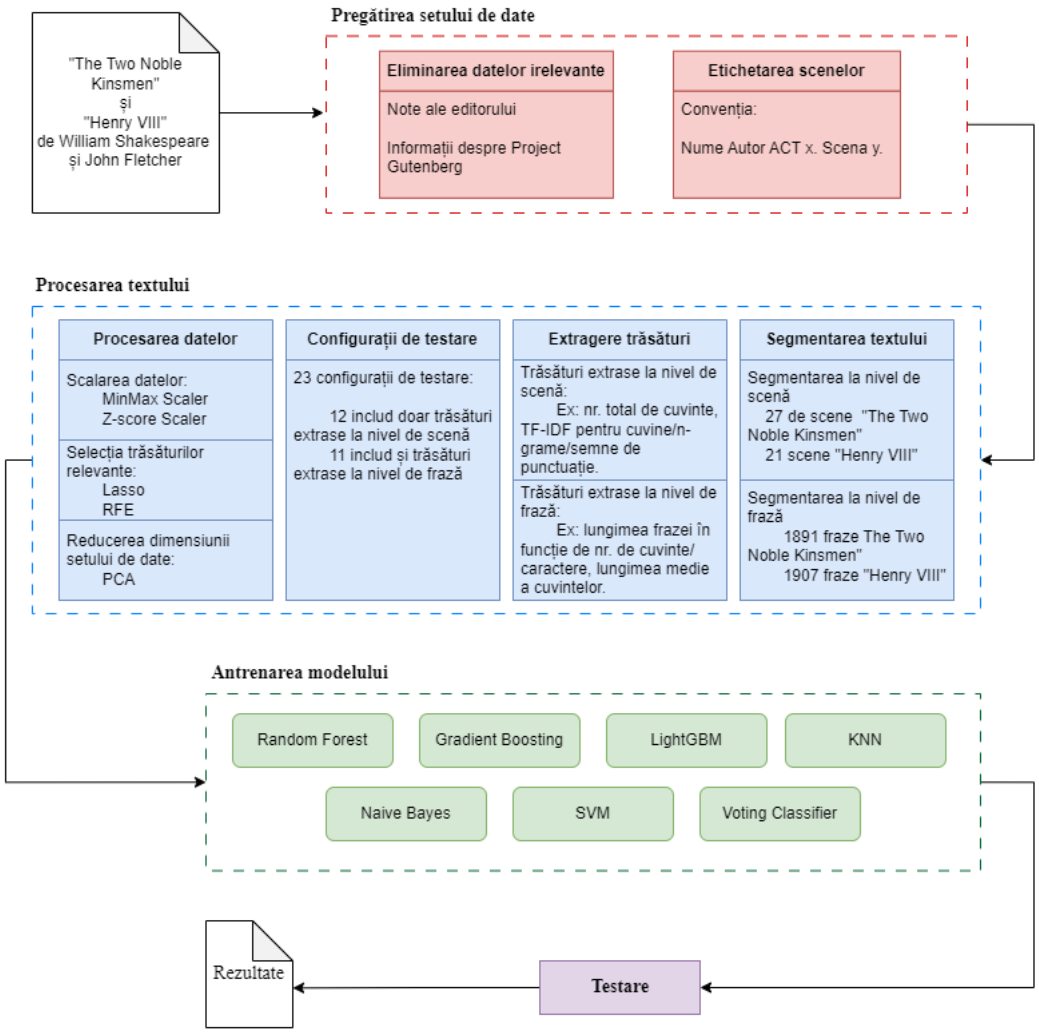

Figura 3.1: Soluția propusă

Selecția celor mai potrivite trăsături

Așa cum am menționat și la începutul capitolului, un prim obiectiv al lucrării este selectarea celor mai potrivite trăsături pentru rezolvarea problemei. Având în vedere acest aspect am urmat o serie de etape pentru procesarea și reprezentarea textelor.

Pregătirea setului de date a constat în eliminarea datelor irelevante din cele două fișiere și etichetarea scenelor cu numele autorului corespunzător, Secțiunea 4.1.

Apoi a urmat partea de procesare a textului. Mai întâi am segmentat textul la nivel de scenă pentru a facilita procesul de extragere a trăsăturilor și am obținut un număr de 48 de semnale din cele două texte: 27 de scene din "The Two Noble Kinsmen", 21 de scene din "Henry VIII". A urmat etapa de extragere a trăsăturilor, descrisă în Secțiunea 4.2. În această fază am selectat mai întâi o serie de caracteristici stilistice la nivel de scenă, dar în urma unor teste preliminare am obținut performanțe slabe pentru algoritmii de clasificare selectați. Problema apare datorită dimensiunii mici a bazei de date și poartă denumirea de "curse of dimensionality". Am decis așadar să analizăm modul în care creșterea numărului de semnale afectează performanțele modelelor, am segmentat textul la nivel de frază și am extras pentru fiecare o serie de trăsături specifice.

Cu aceste două tipuri de caracteristici, la nivel de frază și la nivel de scenă, am creat 23 de configurații de testare, descrise în Secțiunea 4.3. Scopul acestui pas a fost de a analiza pentru ce trăsături/combinații de trăsături se obțin cele mai bune rezultate. În felul acesta, ne putem

da seama care din trăsăturile selectate de noi sunt cele mai potrivite pentru această problemă și care definesc într-adevăr stilul de scris al fiecărui autor.

Tot pentru a verifica relevanța caracteristicilor și pentru a rezolva problema legată de dimensiunea setului de date am implementat o serie de metode de procesare a setului de trăsături, precum metode de scalare a datelor, metode de selecție a trăsăturilor relevante și o metodă de reducere a dimensiunii semnalelor. Toate acestea sunt descrise în Secțiunea 4.4.

Selecția celor mai potrivite metode

Al doilea obiectiv al lucrării este selectarea celor mai potrivite metode de învățare automată pentru rezolvarea problemei. Din acest motiv, pentru etapa de antrenare am utilizat șapte algoritmi de clasificare diferiți, respectiv Gradient Boosting, Random Forest, LightGBM, KNN, Naive Bayes, SVM și un model ce utilizează strategii de votare în cadrul procesului de clasificare. Modul de funcționare al acestor modele este descris pe larg în Capitolul 5.

Toți algoritmii au fost testați pe toate cele 23 de configurații, descrise în Secțiunea 4.3, și pentru toate combinațiile de metode, reprezentate în Tabelul 5.1. Rezultatele obținute în urma acestor experimente sunt prezentate în Capitolul 6.

3.3 Implementare

Codul sursă al proiectului este disponibil pe GitHub. Acesta poate fi vizualizat accesând link-ul: https://github.com/ioana-roxana-b/Proiect_licenta.git.

În directorul Rezultate se găsește un fișier Excel ce conține toate rezultatele obținute în urma experimentelor, pentru toate cele șapte modele de clasificare. Cele două texte modificate, conform descrierii din Secțiunea 4.1, se află în directoarele Train_dataset și Test_dataset.

Proiectul a fost structurat urmând etapele de implementare prezentate în Secțiunea 3.2. Astfel, pentru etapa de pregătire a setului de date, descrisă în Secțiunea 4.1, codul sursă se găsește în:

- **dataset.py** - conține funcția de citire și funcțiile de preprocesare a textului, precum funcția de eliminare a semnelor de punctuație, funcția de transformare a cuvintelor în forma lowercase, funcții de segmentare a textului în cuvinte (tokens);
- **create_vocabs.py** - conține funcții în care sunt formate vocabulare, precum vocabularul ce conține toate cuvintele unice din baza de date, vocabularul cu n-grame, utilizate în etapa de extragere a trăsăturilor.

Implementarea etapei de extragere a trăsăturilor, descrisă în Secțiunea 4.2, se găsește în fișierele:

- **scenes_features.py** - conține funcțiile de extragere a trăsăturilor la nivel de scenă, bazate pe numărul de cuvinte și pe proprietăți ale acestora, precum lungimea medie sau forma comprimată, prezentate în Figura 4.1;
- **sentence_features.py** - conține funcțiile de extragere a trăsăturilor la nivel de frază, prezentate în Figura 4.2;
- **extra_features.py** - conține funcțiile de extragere a trăsăturilor de tip TF-IDF, prezentate în Figura 4.1.

Funcțiile implementate pentru crearea celor 23 de configurații de testare, prezentate în Secțiunea 4.3, se găsesc în:

- **configs.py** - conține funcțiile de formare ale celor 23 de configurații de testare prezentate în Figura 4.3;
- **generate_dataset.py** - conține o funcție care generează o serie de fișiere .csv cu toate cele 23 de configurații de testare, atât pentru setul de antrenare, format din opera "The Two Noble Kinsmen", cât și pentru setul de testare format din opera "Henry VIII". Implementarea acesteia a fost necesară pentru a nu genera seturile de trăsături de fiecare dată când ne dorim să testăm modelele de clasificare.

Funcțiile de preprocesare a setului de trăsături, descrise în Secțiunea 4.4, se află în:

- **additional_functions.py** - pe lângă implementarea funcțiilor de preprocesare a setului de date, în acest fișier se mai găsesc funcțiile de citire pentru fișierele .csv ce conțin seturile de trăsături pentru fiecare configurație, utilizate în timpul procesului de antrenare și testare.

Pentru etapa de antrenare și testare, descrisă în Capitolul 5, am implementat metodele din fișierele:

- **models.py** - conține implementarea celor șapte modele de clasificare;
- **classification.py** - conține funcția utilizată pentru clasificare. Aici se aplică, în funcție de caz, așa cum e prezentat în Tabelul 5.1, funcțiile de procesare a setului de trăsături, apoi se antrenează modelul și se obțin predicțiile pentru setul de testare. La final, se calculează metricile de performanță, acuratețea, precizia, sensibilitatea (recall) și scorul F1, și se salvează rezultatele într-un fișier .csv;
- **run.py** - conține funcții pentru testarea performanțelor metodelor.

În **main.py** se apelează funcțiile de testare a modelelor sau, după caz, alte metode a căror funcționalitate se dorește a fi testată.

4 Procesarea și reprezentarea textului

4.1 Pregătirea setului de date

Întrucât nu am găsit un set de date deja pregătit pentru această aplicație, a fost nevoie să prelucrăm cele două texte selectate pentru analiză. Acest proces a constat în două etape: eliminarea informațiilor irelevante din text și etichetarea scenelor cu numele autorului corespunzător.

4.1.1 Eliminarea datelor irelevante

Ambele opere au fost descărcate de pe Project Gutenberg[1] în format .txt. Pe lângă textul propriu-zis, fișierele conțin un număr mare de note suplimentare, pe care le-am eliminat la începutul etapei de preprocesare a datelor, precum[2]:

1. Informații despre Project Gutenberg: *"Information about Project Gutenberg (one page) We produce about two million dollars for each hour we work. The time it takes us, a rather conservative estimate, is fifty hours to get any etext selected, entered, proofread, edited, copyright searched and analyzed, the copyright letters written, etc. This projected audience is one hundred million readers."*;

2. Note ale editorului: *"A NOTE ON THE TEXT: The text of this Project Gutenberg edition is taken from C. F. Tucker Brooke's 1908 edition of THE SHAKESPEARE APOCRYPHA. Italics have been silently removed in most places, as for proper names, and replaced with ALL CAPS or bracketed text where appropriate."*;

3. Detalii despre operă: *"Presented at the Blackfriers by the Kings Maiesties servants, with great applause: Written by the memorable Worthies of their time; Mr. John Fletcher, Gent., and Mr. William Shakspeare, Gent. Printed at London by Tho. Cotes, for John Waterson: and are to be sold at the signe of the Crowne in Pauls Church-yard. 1634."*.

Prin eliminarea acestor informații ne-am asigurat că setul de date este format doar din conținutul celor două opere selectate, deci doar din date relevante pentru problema noastră.

4.1.2 Etichetarea scenelor

Am decis să aplicăm metode de învățare automată supervizată, respectiv clasificarea binară, și să facem segmentarea textului la nivel de scenă, deci a fost nevoie să atribuim fiecărei secvențe câte o etichetă corespunzătoare autorului care a scris-o. Pentru a face asta am luat ca reper rezultatele obținute în lucrări anterioare. Pentru "Henry VIII", am folosit atribuirea obținută de Jonathan Hope în [23]:

1. Shakespeare: Act I: scenele 1 și 2; Act II: scenele 3 și 4; Act III: scena 2 (parțial); Act V: scena 1;

[1] Project Gutenberg, www.gutenberg.org, este o bibliotecă digitală care oferă acces gratuit la peste 70.000 de cărți în format electronic.

[2] Citatele au fost extrase din fișierul .txt ce conține opera "The Two Noble Kinsmen", ce poate fi descărcat din această sursă: www.gutenberg.org/ebooks/1542.

2. Fletcher: Prolog; Act I: scenele 3 și 4; Act II: scenele 1 și 2; Act III: scenele 1 și 2 (parțial); Act IV: scenele 1 și 2; Act V: scenele 2–5; Epilog.

Pentru opera "The Two Noble Kinsmen" am utilizat atribuirea bazată pe analiza lui Hallet Smith din [67]:

1. Shakespeare: Act I: scenele 1–3; Act II: scena 1; Act III: scena 1; Act V: scena 1 (parțial) și scenele 3-4;

2. Fletcher: Prolog; Act II: scenele 2–6; Act III: scenele 2–6; Act IV: scenele 1 și 3; Act V: scena 1 (parțial), și scena 2; Epilog;

3. Necunoscut: Act I: scenele 4 și 5; Act IV: scena 2.

Scenele al căror autor nu este încă clar stabilit sau la care au contribuit ambii autori nu au fost luate în considerare. De asemenea, deoarece Fletcher pare să fi scris un număr mai mare de scene decât Shakespeare, am mai adăugat câteva scene din "Romeo and Juliet" la finalul ambelor texte, pentru a echilibra setul de date, după cum urmează:

1. The Two Noble Kinsmen: Prolog, Act I: scenele 1, 2, 4, 5;

2. Henry VIII: Act I: scena 3, Act II: scenele 1, 2, 3.

Pentru a putea utiliza aceeași metodă de segmentare a ambelor texte, după etichetarea fiecărei scene cu autorul corespunzător, a fost nevoie să modificăm modul în care se marchează începutul unei noi scene, astfel încât să fie similar pentru ambele opere. Noua convenție adoptată a fost ”Nume Autor ACT x. Scena y.”.

Fiecărei intrări din baza de date îi corespunde o scenă din cele două opere, etichetată corespunzător, conform precizărilor anterioare. Astfel, segmentând textele la nivel de scenă, obținem un număr de 48 de intrări/semnale (27 de scene din "The Two Noble Kinsmen" și 21 de scene din "Henry VIII").

4.2 Extragerea trăsăturilor

Așa cum am menționat și în Capitolul 3, unul din obiectivele principale ale lucrării este selectarea unor trăsături relevante pentru problema abordată.

Am început prin o analiză a literaturii de specialitate, prezentată în Secțiunea 2.3, cu scopul de a identifica trăsăturile pentru care se obțin rezultate bune în problemele de detectare a modificărilor stilistice. În urma analizei lucrărilor, am selectat o serie de trăsături cu potențial pentru problema tratată în această lucrare.

După segmentarea textului la nivel de scenă, conform descrierii din Secțiunea 4.1, am extras trăsăturile prezentate în Figura 4.1.

Trăsături extrase la nivel de scenă	
V1	Nr. cuvinte
V2	Nr. cuvinte funcționale (stop words)
V3	Nr. cuvinte cu formă comprimată (ex: "we'll")
V4	Lungimea medie a cuvintelor
V5	TF-IDF pentru cuvinte funcționale (stop words)
V6	TF-IDF pentru toate cuvintele
V7	TF-IDF pentru cuvinte, fără cele funcționale
V8	TF-IDF pentru n-grame (2-grame)
V9	TF-IDF pentru părți de vorbire
V10	TF-IDF pentru semne de punctuație

Figura 4.1: Trăsături scenă

Trăsăturile extrase la nivel de scenă se pot împărți în două categorii distincte:

1. Vectorii V1-V4 conțin trăsături bazate pe numărul de cuvinte din fiecare scenă și pe proprietăți ale acestora, precum lungimea medie sau forma comprimată;

2. Vectorii V5-V10 conțin trăsături bazate pe metoda Term Frequency - Inverse Document Frequency (TF-IDF).

Term Frequency - Inverse Document Frequency este o metodă des utilizată în problemele de prelucrare a limbajului natural. Aceasta evaluează relevanța unui termen sau a unei structuri dintr-un document, în raport cu întregul corpus de documente. TF - IDF este calculat ca

$$tfidf(t,d,D) = tf(t,d) * idf(t,D) \tag{4.1}$$

Unde $tf(t,d)$ reprezintă frecvența termenului t în documentul d. Aceasta se calculează ca numărul de apariții al termenului t raportat la numărul total de termeni din document. Cu cât cuvântul este mai relevant, cu atât frecvența acestuia este mai mare. TF poate fi calculat cu formula

$$tf(t,d) = \frac{f(t,d)}{\sum f(t',d)} \tag{4.2}$$

unde $f(t,d)$ reprezintă numărul de apariții al termenului t în documentul d, iar numitorul reprezintă numărul total de termeni din documentul d.

IDF măsoară cât de rar apare termenul t în tot corpusul de documente D și se calculează ca inversul frecvenței acelui termen în tot ansamblul de documente

$$idf(t,D) = \log\left(\frac{N}{df(t)}\right) \tag{4.3}$$

unde N este numărul total de documente din corpus, iar $df(t)$ reprezintă numărul de documente în care apare t. Cu cât un termen apare mai rar în corpus, cu atât valoarea IDF este mai mare.

În procesul de extragere a trăsăturilor, am utilizat lista de cuvinte funcționale pentru limba engleză, inclusă în biblioteca NLTK[3]. De asemenea, înainte de a extrage trăsăturile, toate

[3]Natural Language Toolkit (NLTK) este o bibliotecă specifică limbajului Python, utilizată în aplicații pentru prelucrarea limbajului natural [7].

cuvintele din text au fost convertite în formă "lowercase", adică literele majuscule au fost transformate în litere mici. Motivul principal pentru care am făcut acest lucru a fost reducerea dimensiunii vocabularului. Astfel, toate cuvintele identice, dar care erau scrise într-o formă diferită, vor fi tratate la fel în urma conversiei. Această abordare ajută la simplificarea și unificarea datelor și poate contribui la îmbunătățirea performanței modelelor de clasificare.

The Curse of Dimensionality

Reamintim că, segmentând cele două texte la nivel de scenă, obținem un număr de 48 de semnale (27 de scene din "The Two Noble Kinsmen" și 21 de scene din "Henry VIII"). Luând în considerare toate trăsăturile prezentate în Figura 4.1, vom obține o reprezentare de dimensiune 54450 pentru fiecare scenă, reprezentările TF-IDF fiind cele care măresc cel mai mult volumul setului de trăsături.

În literatura de specialitate, situația în care numărul de trăsături este mult mai mare decât numărul de semnale reprezintă un caz tipic pentru problema ce poartă denumirea de "Curse of Dimensionality". Matematicianul Richard E. Bellman este cel care introduce această terminologie în anul 1957. Conceptul se referă la faptul că adăugarea unor dimensiuni suplimentare determină creșterea volumului spațiului Euclidian [75]. În problemele de învățare automată dimensiunea spațiului de reprezentare a setului de date corespunde numărului de trăsături. Pe măsură ce creștem această dimensiune, volumul spațiului de reprezentare se extinde, iar datele se răspândesc din ce în ce mai mult în acest spațiu. Acest fenomen crește complexitatea problemelor și afectează performanțele modelelor. Detalii despre această problemă sunt prezentate și în lucrările [2], [89] și [77].

În problemele de clasificare, acest fenomen mai este întâlnit și sub denumirea de fenomenul Hughes sau paradoxul lui Hughes. În anul 1968, David Hughes publică lucrarea [27] în care arată modul în care creșterea dimensiunii setului de date afectează complexitatea unei probleme de învățare automată. El observă că până într-un anumit punct, performanța unui clasificator crește odată cu creșterea numărului de trăsături. De la un anumit punct însă, adăugarea unor trăsături suplimentare duce la scăderea performanței modelului.

Trăsături extrase la nivel de frază	
V11	Lungimea frazei în funcție de numărul de cuvinte
V12	Lungimea frazei în funcție de numărul de caractere
V13	Lungimea medie a cuvintelor
V14	Nr. cuvinte funcționale (stop words)

Figura 4.2: Trăsături frază

Într-o primă etapă, am încercat să diminuăm efectele fenomenului "curse of dimensionality" asupra performanței modelelor prin creșterea dimensiunii bazei de date. Am segmentat textele la nivel de frază și am extras pentru fiecare câteva trăsături specifice, conform Figurii 4.2.

Astfel, obținem 3798 de semnale (1891 de fraze din "The Two Noble Kinsmen" și 1907 din "Henry VIII"). În acest fel, fiecărei intrări din baza de date îi corespunde câte o frază etichetată conform autorului scenei din care face parte. Vectorul de trăsături asociat fiecărui semnal este alcătuit din caracteristicile stilistice specifice frazei, prezentate în Figura 4.2, și caracteristici ale scenei din care face parte. Numărul total de trăsături obținute prin combinarea celor două tipuri de caracteristici este 54454. În urma unor teste preliminare, am constatat că performanța modelelor de clasificare se îmbunătățește odată cu creșterea numărului de semnale.

4.3 Configurații de testare

Cu trăsăturile prezentate în Secțiunea 4.2 am creat 23 configurații de testare, reprezentate în Figura 4.3, atât pentru a selecta cel mai bun set de caracterisitici stilistice pentru rezolvarea problemei, cât și pentru a putea observa impactul pe care numărul de trăsături selectate îl are asupra performanței modelelor.

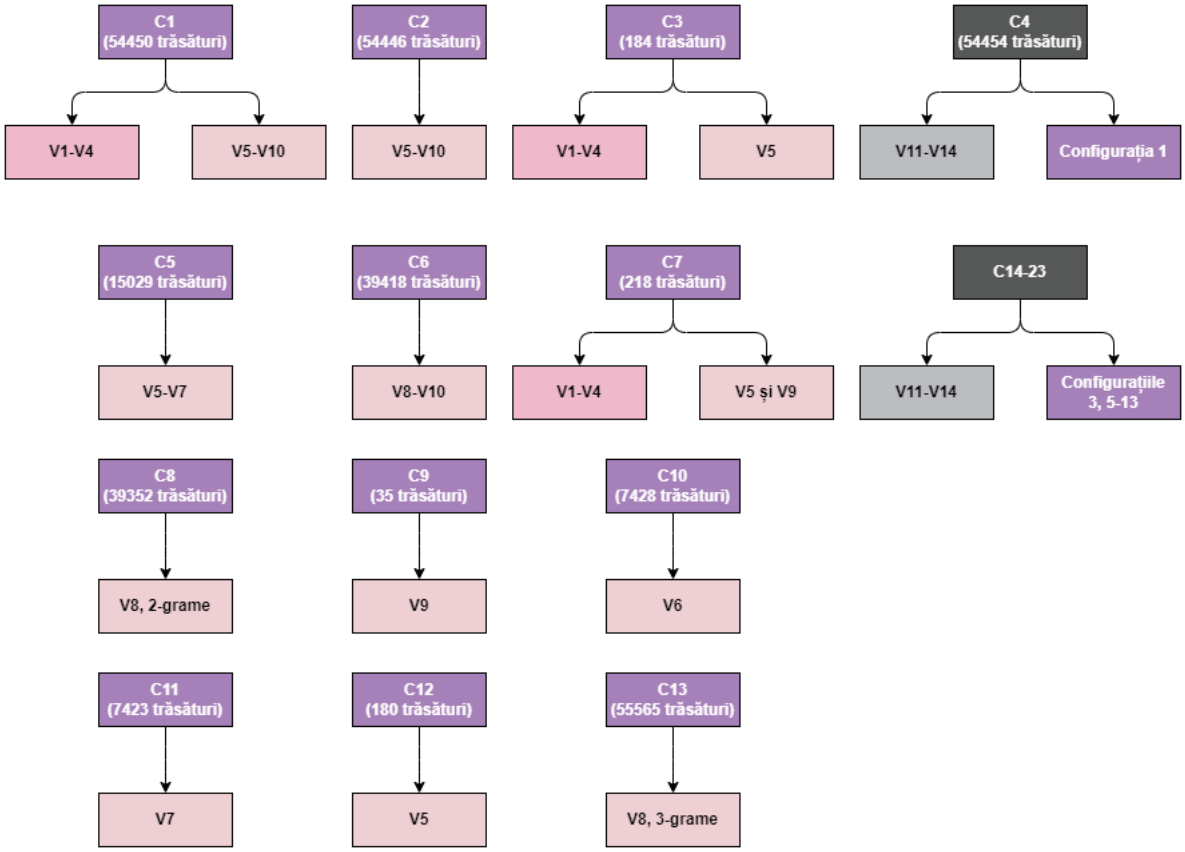

Figura 4.3: Configurații de testare

În experimentele inițiale, am utilizat doar primele patru configurații de testare și doi algoritmi de învățare automată: Random Forest și SVM. Am observat astfel cum ambele modele obțin performanțe foarte bune pentru C4, acuratețe peste 90%, și rezultate încurajatoare pentru celelalte trei, demonstrând că trăsăturile selectate sunt într-adevăr adecvate problemei analizate în această lucrare.

În urma acestor observații, am format și celelalte configurații de testare. Diferența între reprezentarea configurațiilor C1-C3, C5-C13 și reprezentarea configurațiilor C4, respectiv C14-C23, apare datorită modului în care sunt utilizate cele două tipuri de segmentare a textului prezentate în Secțiunea 4.2: segmentarea la nivel de scenă și segmentarea la nivel de frază.

Trăsături la nivel de scenă

Pentru prima categorie (C1-C3, C5-C13) este utilizată segmentarea la nivel de scenă, în care numărul de semnale este 48. Aceste configurații sunt formate doar din combinații de trăsături extrase la nivel de scenă, reprezentate în Figura 4.1. Configurațiile C8-C13 care conțin un singur vector de trăsături au fost create cu scopul de a verifica dacă utilizând doar una din aceste reprezentări se pot obține rezultate la fel de bune ca în cazul în care antrenarea și testarea modelului se face utilizând combinații de caracteristici.

Trăsături la nivel de frază

Pentru a doua categorie (C4, C14-C23) este utilizată segmentarea la nivel de frază în care numărul de semnale este 3798. În C4 sunt combinate toate trăsăturile extrase la nivel de frază, reprezentate în Figura 4.2, cu toate trăsăturile din C1, deci toate trăsăturile extrase la nivel de scenă. Configurațiile C14-C23 sunt create folosind același model și sunt formate din toate trăsăturile extrase la nivel de frază în combinație cu trăsăturile dintr-una din configurațiile C3, C5-C13.

Rezultatele obținute de algoritmii de clasificare pentru aceste configurații de testare vor fi prezentate în capitolele următoare.

4.4 Procesarea setului de trăsături

În urma experimentelor inițiale, în care am antrenat modelele de clasificare fără niciun tip de preprocesare a setului de trăsături, am obținut performanțe destul de scăzute. Analizând literatura de specialitate, am descoperit problema descrisă în Secțiunea 4.2, legată de dimensiunea setului de date, precum și o serie de metode al căror scop este să diminueze efectele fenomenului "curse of dimensionality" asupra performanțelor modelelor.

Astfel, pentru procesarea setului de trăsături am implementat metode de scalare a datelor, metode de selecție a trăsăturilor relevante și o metodă pentru reducerea dimensiunii semnalelor.

4.4.1 Scalarea datelor

Scalarea datelor se referă la procesul prin care toate valorile trăsăturilor sunt modificate astfel încât să se încadreze într-un anumit interval. În anumite probleme de învățare automată, scalarea datelor joacă un rol important în obținerea unor performanțe bune. Această metodă e utilizată pentru situații în care setul de date conține trăsături cu unități de măsură diferite, asigurând eliminarea discrepanțelor dintre valorile acestora. Astfel, prin scalare se reduce influența valorilor extreme ce ar putea afecta performanța modelelor.

Pentru setul nostru de date am implementat două tipuri de scalare diferite: scalarea prin standardizare și scalarea prin normalizare.

Scalare prin standardizare

Scalarea prin standardizare, sau scalare de tip Z-score, transformă setul de trăsături astfel încât să aibă media egală cu zero și deviația standard egală cu unu. Formula utilizată pentru a scala o trăsătură x este

$$x_s = \frac{x - \mu}{\sigma} \tag{4.4}$$

unde x_s este valoarea scalată, x este valoarea pe care dorim să o scalăm, μ este media întregului set de trăsături, iar σ este deviația standard a setului de trăsături. Media se calculează prin adunarea tuturor valorilor din set și împărțirea sumei la numărul total de valori, iar deviația standard se calculează prin determinarea rădăcinii pătrate a varianței setului de date.

Scalarea prin standardizare este utilă în special atunci când datele nu sunt distribuite normal sau conțin valori extreme (outliers). În experimentele noastre am utilizat funcția Standard Scaler disponibilă în biblioteca scikit-learn [54].

Scalare prin normalizare

Scalarea prin normalizare, sau scalare de tip Min-Max, transformă toate valorile setului de trăsături astfel încât să facă parte din același interval numeric, în general [0, 1]. Formula

utilizată pentru scalarea prin normalizare este

$$x_s = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.5}$$

unde x_s este valoarea scalată, x este valoarea pe care dorim să o scalăm, x_{max} este valoarea maximă din setul de trăsături, iar x_{min} este valoarea minimă.

Această metodă este utilizată în general în cazurile în care deviația standard este foarte mică sau distribuția datelor nu este Gaussiană. La fel ca și în cazul scalării prin standardizare, în experimentele noastre am utilizat funcția MinMax disponibilă în biblioteca scikit-learn [54].

4.4.2 Selecția trăsăturilor relevante

Necesitatea implementării metodelor de selecție a trăsăturilor relevante a apărut datorită problemei dimensiunii setului de date, descrisă în Secțiunea 4.2. Metodele de selecție aleg din întregul set de caracterisitici un mic subset în funcție de un anumit criteriu de evaluare a relevanței. Prin reducerea numărului de trăsături, se poate reduce complexitatea problemei și se pot îmbunătăți performanțele algoritmilor de învățare automată. Metodele de selecție a trăsăturilor ajută de asemenea la evitarea fenomenului de supra-adaptare (overfitting), în care modelul învață foarte bine exemplele din setul de antrenare, dar nu este capabil să generalizeze pentru date noi.

Există în literatura de specialitate mai multe tipuri de metode și criterii după care acestea pot fi grupate. Un exemplu ar fi dacă necesită sau nu ca setul de date să fie etichetat, caz în care metodele de selecție a trăsăturilor pot fi supervizate, nesupervizate sau semi-supervizate [13]. Metodele de selecție supervizate pot fi grupate mai departe în: metode de filtrare (filter methods), metode încorporate (embedded methods) și metode de tip wrapper [15].

Metodele de filtrare realizează selecția trăsăturilor prin utilizarea unui criteriu de evaluare a importanței caracteristicilor, independent de setul de date din care fac parte. În general, criteriile utilizate de astfel de tehnici sunt bazate pe metode statistice precum corelația sau informația mutuală. În urma evaluării trăsăturilor se păstrează doar un subset ale cărui valori ale acelor mărimi statistice depășesc un anumit prag prestabilit. Funcționarea metodelor încorporate se bazează, așa cum sugerează și numele lor, pe integrarea procesului de selecție a trăsăturilor în procesul de antrenare al modelului. Metodele de tip wrapper utilizează un algoritm de învățare automată, diferit de cel utilizat pentru crearea modelului, pentru a evalua setul de trăsături. Aceste metode efectuează o căutare euristică în spațiul setului de date, bazându-se pe performanțele modelului utilizat pentru evaluare, pentru a selecta cel mai potrivit subset de trăsături [73].

În lucrarea noastră am utilizat două metode de selecție a trăsăturilor, Least Absolute Shrinkage and Selection Operator (LASSO) și Recursive Feature Elimination (RFE), pentru a evalua impactul lor asupra performanței modelelor de clasificare.

LASSO

Least Absolute Shrinkage and Selection Operator (LASSO) este o metodă de selecție a trăsăturilor de tip embedded. Acest algoritm a fost introdus în anul 1996 de către Robert Tibshirani, în lucrarea [76]. Scopul final al metodei LASSO este de a reduce la 0 ponderile trăsăturilor mai puțin relevante, astfel încât, la final, să rămână doar caracteristicile importante pentru model. Astfel, se obțin modele mai simple și se evită, de asemenea, fenomenul de supra-adaptare (overfitting). Din punct de vedere matematic, obiectivul metodei este de a minimiza următoarea funcție obiectiv

$$\frac{1}{2n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \alpha\|\boldsymbol{w}\|_1 \tag{4.6}$$

în care $\boldsymbol{y}$ este vectorul de etichete al setului de date, $\boldsymbol{X}$ este matricea de trăsături, parametrul α este cel care stabilește gradul de penalizare, iar $\boldsymbol{w}$ este vectorul de ponderi asociat trăsăturilor din matricea $\boldsymbol{X}$.

Un aspect important de menționat aici este că, în lucrarea [76], LASSO este propusă ca o metodă de regresie, în care $\boldsymbol{y}$ este variabila dependentă, iar $\boldsymbol{X}$ variabila independentă, semnalul de intrare. În acest caz, funcția are două componente: una care minimizează eroarea (reziduurile), astfel încât reprezentarea semnalului să fie cât mai aproape de valoarea sa reală, iar a doua componentă reprezintă un termen de penalizare, care are scopul de a reduce ponderile neimportante la 0. În cadrul implementării soluției propuse am utilizat modelul liniar LASSO, existent în biblioteca scikit-learn [54], pe care l-am adaptat pentru problema de clasificare, făcându-l pe $\boldsymbol{y}$ vectorul de etichete, care semnalează autorul fiecărei scene.

RFE

Recursive Feature Elimination (RFE) este o metodă de selecție a trăsăturilor de tip wrapper. Modul de funcționare al RFE are la bază antrenarea unui model de învățare automată și eliminarea în mod iterativ a trăsăturilor considerate neimportante. Selecția se face în urma evaluării performanței modelului pentru fiecare trăsătură sau grup de trăsături, astfel încât, la final, să rămână doar cele pentru care se obțin cele mai bune performanțe. Acest proces se repetă până în punctul în care se ajunge la un număr dorit de trăsături sau până când este îndeplinit un criteriu de oprire specificat.

În general, se poate folosi aproape orice algoritm de învățare automată pentru procesul de antrenare și eliminare a trăsăturilor. Printre cele mai populare metode se numără: regresia logistică, arborii de decizie și SVM. Aceste metode sunt alese în funcție de problema abordată și de modelul principal de învățare automată utilizat.

În implementarea soluției propuse am utilizat metoda Recursive Feature Elimination with Cross-Validation (RFECV), din biblioteca scikit-learn [54]. Diferența dintre RFE și RFECV constă în faptul că în etapa de antrenare și eliminare a variabilelor, în cea de-a doua metodă, se mai adaugă un pas, cel de validare încrucișată (cross-validation). În felul acesta, se evită fenomenul de supra-adaptare (overfitting) și se obține o estimare mai robustă a performanței modelului de învățare automată.

4.4.3 Reducerea dimensiunii semnalelor

Pe lângă metodele de selecție a trăsăturilor, mai există o categorie de algoritmi al căror scop este reducerea numărului de caracteristici. În literatura de specialitate aceste două abordări se găsesc sub denumirile: feature selection și feature extraction. Așa cum am prezentat anterior, metodele de selecție aleg din întregul set de date un mic subset de trăsături ce sunt considerate relevante pentru problemă, în urma unui criteriu de evaluare. Metodele de extragere a trăsăturilor pentru reducerea dimensiunii utilizează transformări matematice pentru a crea un nou set de date din cel inițial, care să conțină caracteristici cât mai relevante pentru problemă, fără a pierde informația esențială. O descriere mai amănunțită a acestor tehnici se găsește în lucrarea [26].

În funcție de principiul de funcționare, metodele de reducere a dimensiunii prin transformarea setului de date se împart în: metode liniare și metode neliniare. Diferența între cele două categorii constă în modul în care acestea modelează relațiile dintre trăsături în momentul construirii noului set de date. Tehnici precum Principal Component Analysis (PCA) și Linear Discriminant Analysis (LDA) fac parte din categoria metodelor liniare. Diferențe între cele două metode au fost studiate în [60], ele folosesc, așa cum sugerează și numele, combinații liniare ale trăsăturilor inițiale pentru formarea noului set de caracteristici. Metodele neliniare, precum Multi-Dimensional Scaling (MDS) sau t-distributed Stochastic Neighbor Embedding (t-SNE),

iau în considerare legături mai complexe, neliniare, între trăsături [58].

PCA

Principal Component Analysis (PCA) este o tehnică populară de reducere a dimensiunii semnalelor. Aceasta poate transforma un set de trăsături multidimensional într-un set mai mic prin crearea unor "componente principale", fără a pierde informațiile relevante din datele inițiale. Noile componente sunt create din combinații liniare ale trăsăturilor inițiale și sunt ordonate în funcție de gradul de variație al datelor pe care îl conțin.

Există mai multe tehnici utilizate de către PCA pentru a reduce dimensiunea setului de date. Printre cele mai populare sunt descompunerea valorilor proprii, detalii despre această metodă se găsesc în [60], și descompunerea valorilor singulare (SVD). În această lucrare am utilizat funcția PCA disponibilă în biblioteca scikit-learn [54] al cărei principiu de funcționare se bazează pe descompunerea valorilor singulare. Principiile matematice din spatele acestei metode sunt descrise în [70] și [29].

5 Antrenarea modelului, testare

În procesul de antrenare am utilizat toate metodele de preprocesare a setului de trăsături, prezentate în Secțiunea 4.4. Am analizat, de asemenea, cum influențează utilizarea lor performanțele modelelor și pentru care combinație de metode obținem cele mai bune rezultate. Modul în care acestea au fost testate este prezentat în Tabelul 5.1.

Pe primul rând al tabelului sunt prezentate numele metodelor. Valorile de pe următoarele rânduri indică dacă metodele sunt utilizate pentru un anumit Caz X (valoarea lui X corespunzând valorilor din prima coloană). Valoarea True corespunde cazului în care metoda este utilizată, iar valoarea False corespunde cazului în care metoda nu este utilizată. Se poate observa de exemplu, că atunci când am folosit metoda LASSO, pentru a selecta cele mai relevante trăsături, a fost nevoie să utilizăm și o metodă de scalare a datelor. De asemenea, important de menționat aici e faptul că RFE are valoarea False pentru toate cazurile, deoarece nu o utilizăm simultan cu metoda LASSO. O variantă alternativă a tabelului ar fi cea în care RFE ar lua valorile metodei LASSO, iar aceasta din urmă ar avea doar valoarea False în toate situațiile.

Caz	StratifiedKFold	PCA	Z-Score	MinMax	LASSO	LASSO treshold	RFE
1	False	False	False	False	False	False	False
2	False	False	False	True	False	False	False
3	False	False	False	True	False	True	False
4	False	False	False	True	True	False	False
5	False	False	True	False	False	False	False
6	False	False	True	False	False	True	False
7	False	False	True	False	True	False	False
8	False	True	False	False	False	False	False
9	False	True	False	True	False	False	False
10	False	True	False	True	True	False	False
11	False	True	True	False	False	False	False
12	False	True	True	False	True	False	False
13	True	False	False	False	False	False	False
14	True	False	False	True	False	False	False
15	True	False	False	True	False	True	False
16	True	False	False	True	True	False	False
17	True	False	True	False	False	False	False
18	True	False	True	False	False	True	False
19	True	False	True	False	True	False	False
20	True	True	False	False	False	False	False
21	True	True	False	True	False	False	False
22	True	True	False	True	True	False	False
23	True	True	True	False	False	False	False
24	True	True	True	False	True	False	False

Tabela 5.1: Combinații ale metodelor de preprocesare a datelor

Metoda prezentă pe a doua coloană este Stratified K-Fold cross-validator, disponibilă în biblioteca scikit-learn [54]. Validarea încrucișată (cross-validation) este o metodă des folosită în problemele de învățare automată. Scopul acestei tehnici este de a evalua performanța modelului pentru diferite subseturi de antrenare și testare. Acest lucru se realizează prin divizarea setului de date inițial în mai multe grupuri, dintre care unul este utilizat pentru etapa de testare și restul sunt folosite în etapa de antrenare. Acest proces se repetă până când toate grupurile au fost utilizate, pe rând, în etapa de testare. Prin validarea încrucișată se evită fenomenul de supra-adaptare (overfitting) și se obține o estimare mai bună a performanței modelului [69].

Stratified K-fold împarte setul inițial în k grupuri egale, cu avantajul că fiecare grup conține aproximativ același număr de instanțe din fiecare clasă. În cazul problemei noastre, prin utilizarea acestei metode, ne-am asigurat că în momentul în care se execută procesul de validare încrucișată, numărul de instanțe ce reprezintă scene/fraze scrise de Shakespeare este similar cu numărul de scene/fraze scrise de Fletcher. În cadrul experimentelor efectuate, parametrul *n_folds* a fost setat la valoarea 2, însemnând că setul de date este împărți de fiecare dată în două grupuri. Pentru situațiile în care am utilizat Stratified K-fold, setul de date a fost format din ambele opere, "The Two Noble Kinsmen" și "Henry VIII". Pentru situațiile în care nu am utilizat Stratified K-fold, setul de antrenare a fost format doar din opera "The Two Noble Kinsmen", iar setul de testare din opera "Henry VIII".

O altă observație aici ar fi legată de modul în care am aplicat metoda PCA în procesul de antrenare. Pentru toate configurațiile de testare, în afară de C4, C9 și C19, în situațiile în care am aplicat metoda PCA, antrenarea modelului a fost făcută pe setul de date format din ambele opere, iar în procesul de testare am utilizat doar opera "Henry VIII". Acest lucru a fost necesar datorită dimensiunii mici a bazei de date și a problemei descrise în Secțiunea 4.2.

Pentru rezolvarea problemei de clasificare, am utilizat șapte algoritmi de învățare automată supervizată, descriși în secțiunile următoare. Toate modelele au fost testate pe toate cele 23 de configurații de testare, prezentate în Figura 4.3, și pentru toate cele 24 de cazuri, prezentate în Tabelul 5.1.

5.1 Random Forest

Random Forest este un algoritm de învățare automată supervizată, utilizat pentru probleme de clasificare și regresie. Așa cum sugerează și numele, principiul de funcționare al algoritmului se bazează pe crearea unui ansamblu de arbori de decizie care lucrează în paralel. Fiecare arbore ia în considerare un subset aleatoriu de trăsături și face o predicție individuală, iar la final este aleasă clasa selectată de cei mai mulți arbori. Algoritmul este prezentat în detaliu în [9].

Unul din avantajele oferite de algoritmul Random Forest este faptul că, în general, nu este foarte afectat de zgomotul prezent în setul de date și nici de posibilele valori lipsă. Prin combinarea rezultatelor arborilor de decizie, reduce varianța datelor și scade posibilitatea apariției fenomenului de supra-adaptare (overfitting).

Pentru rezolvarea problemei de identificare a modificărilor stilistice în texte, algoritmul a mai fost utilizat în lucrări precum [88], [62], [31] și [72] având de fiecare dată performanțe bune. În cadrul experimentelor efectuate în această lucrare am utilizat modelul de clasificare disponibil în biblioteca scikit-learn [54].

5.2 Gradient Boosting

Gradient Boosting este o metodă de învățarea automată supervizată care, la fel ca Random forest, este în general utilizată pentru probleme de clasificare și regresie. Jerome Friedman este cel care introduce acest algoritm în lucrarea [19]. Principiul de funcționare al metodei constă

în creare unui ansamblu de modele slabe ("weak learners"), în general arbori de decizie sau arbori de regresie, cu scopul de a forma un model cât mai robust. În comparație cu modul în care sunt construiți arborii de decizie în cazul metodei Random Forest, în acest caz modelele din ansamblu nu mai sunt construite independent, ci în mod iterativ. Se începe cu antrenarea primului model și pe baza rezultatelor obținute de acesta (valori are erorii, număr de predicții greșite) se construiește următorul. Ideea de bază fiind ca fiecare model nou adăugat să "învețe" din greșelile celui precedent cu scopul de a obține performanțe mai bune. La final, predicția dată de Gradient Boosting se calculează prin combinarea ponderată a tuturor predicțiilor date de modelele din ansamblu.

În probleme de identificare a modificărilor stilistice în texte s-au mai utilizat abordări similare în lucrări precum [88] și [72]. În cadrul acestei lucrări, am utilizat modelul disponibil în biblioteca scikit-learn [54].

5.3 LightGBM

Light Gradient Boosting Machine (LightGBM) este un framework utilizat în probleme de învățare automată, dezvoltat de Microsoft și apărut pentru prima dată ca bibliotecă open-source în anul 2017 [35]. LGBM are la bază aceleași principii de funcționare ca Gradient Boosting, prezentate în Secțiunea 5.2, dar vine cu o serie de îmbunătățiri.

LightGBM folosește o metodă numită Gradient One-Side Sampling (GOSS) pentru a face selecția subsetului de trăsături utilizat în construirea fiecărui arbore din ansamblu. Această metodă analizează setul de date la fiecare pas al algoritmului și selectează subseturile de trăsături pe baza valorilor gradientului. Astfel, GOSS consideră că modelele anterior construite au fost deja antrenate pe subseturile de caracteristici ce au o valoare mică a gradientului și, prin urmare, o valoare mică a erorii. În mod similar, consideră că subseturile de trăsături cu valoare mare a gradientului, ce au asociată o valoare mare a erorii, sunt cele care au cel mai ridicat conținut informațional și le selectează cu o probabilitate mai mare în procesul de construire a arborilor. În felul acesta procesul de selecție al subseturilor de trăsături devine mai eficient și se obține un model mai rapid și mai precis.

O a doua îmbunătățire pe care o aduce LightGBM este utilizarea metodei Exclusive Feature Bundling (EFB) utilizată pentru reducerea dimensiunii semnalelor. Aceasta este o tehnică de tip greedy ce combină trăsături mutual exclusive într-o singură caracteristică fără a pierde conținut informațional. Aceste metode fac algoritmul LightGBM mai rapid și mai eficient în rezolvarea problemelor de învățare automată [83]. În probleme de identificare a modificărilor stilistice este utilizat în lucrările [88], [72].

5.4 KNN

K-Nearest Neighbors (KNN) este un algoritm de învățare automată supervizată utilizat în rezolvarea problemelor de regresie și clasificare. Este considerată una dintre cele mai simple metode de învățare automată datorită faptului că nu face multe presupuneri legate de setul de date și nici nu are nevoie de mulți parametri. Algoritmul clasifică noile elemente pe baza unui criteriu de similitudine al acestora cu cei mai apropiați k vecini din setul de antrenare. Pentru fiecare instanță nouă care trebuie clasificată se calculează distanța dintre noul punct și toate punctele din setul de date. Pe baza rezultatelor obținute, se selectează cei mai apropiați k vecini și se determină clasa majoritară din rândul acestora. Eticheta corespunzătoare acestei clase îi este atribuită noii instanțe. Există mai multe metode folosite pentru calculul distanței între puncte. Cele mai populare sunt distanța Euclidiană, distanța Manhattan și distanța Minkowski.

În probleme de identificare a modificărilor stilistice KNN a mai fost folosită în lucrări precum [72] și [25]. În această lucrare am utilizat modelul KNN din biblioteca scikit-learn [54].

Acest model utilizează distanța Minkowski pentru a determina gradul de apropiere dintre atribute

$$D(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{5.1}$$

în care x_i și y_i sunt elementele vectorilor de trăsături asociați problemei, iar p reprezintă ordinul distanței Minkowski.

5.5 Naive Bayes

Naive Bayes este un algoritm de învățare automată supervizată. Așa cum se poate deduce și din nume, metoda are la bază teorema lui Bayes

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{5.2}$$

Rezultatul ecuației indică probabilitatea condiționată a evenimentului A, dat fiind un eveniment B observat. Într-o problemă de clasificare, evenimentul A poate fi înlocuit cu o clasă C, iar evenimentul B cu un vector de trăsături $\boldsymbol{X}$, astfel formula devine

$$P(C|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|C) \cdot P(C)}{P(x_1, x_2, ..., x_n)} \tag{5.3}$$

Ceea ce face acest model "naiv" este presupunerea că toate trăsăturile sunt condițional independente. Plecând de la această presupunere, în procesul de clasificare, modelul alege clasa cu valoarea maximă a probabilității, utilizând relația

$$\hat{y} = \arg\max_{C_i} P(C_i|X_1, X_2, ..., X_n) \tag{5.4}$$

cunoscută sub denumirea de estimare Maximum A Posteriori (MAP).

Algoritmul Naive Bayes este des folosit în probleme de prelucrare a limbajului natural. Pentru identificarea modificărilor stilistice într-un text, modelul a fost utilizat în [49], [43] și [31]. În aplicația noastră am utilizat modelul Gaussian Naive Bayes, disponibil în biblioteca scikit-learn [54]. Acest model face presupunerea că distribuția trăsăturilor este Gaussiană. Astfel, formula de calcul a probabilității condiționate devine

$$P(x_i|C) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{5.5}$$

unde μ reprezintă media, iar σ reprezintă deviația standard a trăsăturilor din clasa C.

5.6 SVM

Support Vector Machine (SVM) este un algoritm de învățare automată utilizat în probleme de clasificare și regresie. Obiectivul algoritmului este de a găsi un hiperplan într-un spațiu N-dimensional (N - numărul de trăsături) care să poată separa optim datele. În mod ideal, hiperplanul găsit de model este cel mai bun dacă se află la cea mai mare distanță posibilă de cele mai apropiate puncte din setul de antrenare, numite și vectori suport. În general, cu cât

această distanță este mai mare, cu atât este mai mică eroarea de generalizare a clasificatorului. Mai multe detalii legate de acest algoritm se găsesc în [65].

În probleme de identificarea a modificărilor stilistice, algoritmul a mai fost folosit în [49], [53], [88], [31] și [72]. În această lucrare am folosit funcția GridSearchCV împreună cu modelul Support Vector Classifier (SVC), disponibile în biblioteca scikit-learn [54].

Funcția GridSearchCV este o metodă de optimizare utilizată pentru a testa diferite combinații de parametri cu scopul de a determina cea mai bună configurație pentru un anumit estimator. În cazul nostru, parametri testați au fost tipul de kernel și diferite valori pentru parametrul de regularizare C. În urma acestui proces, am ales valoarea 1 pentru parametrul C.

Funcțiile kernel sunt folosite pentru procesul de transformare și separare a datelor. Acestea sunt aplicate tuturor elementelor din setul de date cu scopul de a transforma spațiul original, într-un spațiu multidimensional în care datele devin liniar separabile. Pentru a analiza performanța modelului, am utilizat diferite tipuri de kernel în procesul de antrenare și testare, după cum urmează:

1. Kernel liniar: este cel mai simplu tip de kernel utilizat de modelul SVM. Acesta utilizează produsul scalar dintre vectorii de intrare ca metodă de clasificare

$$K_{\text{linear}}(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle \tag{5.6}$$

   unde $\boldsymbol{x}$ și $\boldsymbol{y}$ sunt vectorii de intrare, iar $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ reprezintă produsul scalar al celor doi vectori.

2. Kernel polinomial (poly) utilizează, așa cum îi sugerează și numele, o funcție polinomială pentru a transforma setul de date într-un spațiu N-dimensional

$$K_{\text{poly}}(\boldsymbol{x}, \boldsymbol{y}) = (1 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle)^d \tag{5.7}$$

   unde d reprezintă gradul polinomului, $\boldsymbol{x}$ și $\boldsymbol{y}$ sunt vectorii de intrare, iar $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ reprezintă produsul scalar al celor doi vectori. Gradul polinomului poate fi setat prin intermediul parametrului *degree*. În cazul experimentelor efectuate parametrul *degree* a fost setat la valoarea 2.

3. Kernel radial (Radial Basis Function, rbf) folosește o funcție radială, bazată pe măsura distanței euclidiene dintre vectorii de intrare, pentru a transforma setul de date într-un spațiu N-dimensional

$$K_{\text{rbf}}(\boldsymbol{x}, \boldsymbol{y}) = e^{(-\gamma \cdot \|\boldsymbol{x} - \boldsymbol{y}\|^2)} \tag{5.8}$$

   unde $\boldsymbol{x}$ și $\boldsymbol{y}$ sunt vectorii de intrare, iar γ este un parametru ce determină gradul de influență al fiecărui exemplu din setul de antrenare asupra vecinilor săi.

4. Kernel sigmoid, aplică funcția sigmoid asupra produsului scalar dintre vectorii de intrare

$$K_{\text{sigmoid}}(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\gamma \cdot \langle \boldsymbol{x}, \boldsymbol{y} \rangle + c) \tag{5.9}$$

   unde $\boldsymbol{x}$ și $\boldsymbol{y}$ sunt vectorii de intrare, γ reprezintă panta funcției sigmoid, iar c este un parametru adițional ce poate fi utilizat pentru a ajusta diferite caracteristici ale acestei funcții.

Rezultatele obținute pentru fiecare tip de kernel, sunt prezentate în Capitolul 6.

5.7 Voting Classifier

Principiul de funcționare al acestui clasificator este cumva asemănător cu cel al algoritmilor Random Forest, descris în Secțiunea 5.1, și Gradient Boosting, descris în Secțiunea 5.2. Doar

că, în cazul unui clasificator de tip voting, ansamblul poate fi format dintr-o serie de modele de învățare automată diferite, selectate în funcție de problema abordată. În cazul acestei lucrări, ansamblul este format din toți cei șase algoritmi de clasificare descriși mai sus.

În funcție de tipul de votare utilizat, există două categorii de metode: hard voting și soft voting. În cazul metodelor de tip hard voting, clasificatorul selectează clasa votată de majoritatea modelelor individuale. Principiul de funcționare al metodelor de tip soft voting se bazează pe distribuțiile de probabilitate asociate predicțiilor fiecărui model individual. Clasificatorul combină aceste distribuții pentru a obține valorile medii ale probabilităților pentru fiecare clasă și selectează clasa cu cea mai mare probabilitate rezultată. Pentru anumite probleme este posibil ca un model de tip voting să obțină performanțe mai bune decât ar obține un model individual datorită faptului că într-un ansamblu deficiențele unui algoritm pot fi compensate de performanțele altui algoritm.

În cadrul experimentelor noastre am analizat performanța clasificatorului pentru ambele tipuri de votare, obținând rezultate asemănătoare.

6 Rezultatele obținute

În cadrul acestui capitol vom prezenta rezultatele obținute în urma efectuării experimentelor. Vom începe prin a analiza modul în care aplicarea metodelor de procesare a setului de trăsături afectează performanțele modelelor. De asemenea, vom ilustra modul de alegere a celor mai potriviți parametri pentru etapa de antrenare și testare și vom arăta care au fost cele mai bune rezultate obținute de fiecare algoritm pentru fiecare configurație de testare din Figura 4.3. Din aceste rezultate vom putea deduce care sunt cele mai relevante trăsături pentru problema abordată și vom analiza cât de potriviți sunt algoritmii de clasificare selectați.

Vom încheia prin prezentarea celor mai bune performanțe obținute pentru fiecare model.

6.1 Influența metodelor de procesare asupra performanței modelelor de clasificare

Pentru a ilustra modul în care aplicarea metodelor de procesare a setului de date afectează performanța modelelor, am efectuat o serie de experimente utilizând algoritmul Random Forest, pentru primele 15 configurații de testare, prezentate în Figura 4.3. Am utilizat acuratețea ca metrică principală pentru a analiza rezultatele clasificatorului. În cadrul fiecărui scenariu de testare, subsetul de antrenare a fost format din opera "The Two Noble Kinsmen", iar subsetul de testare din opera "Henry VIII". Rezultatele obținute în urma experimentelor vor fi prezentate în continuare pentru fiecare metodă.

Tot în această secțiune vom prezenta și modul în care am ales valorile parametrilor specifici fiecărei metode de preprocesare a setului de date.

Influența scalării datelor asupra performanței modelelor

În cadrul procesului de evaluare, am utilizat ambele metode de scalare pentru a determina cum afectează acestea performanța modelului. În urma efectuării experimentelor, am constatat că aplicarea acestor metode individual nu afectează în niciun fel rezultatele clasificatorului.

Cu toate acestea, tehnicile de scalare a datelor joacă un rol important în obținerea celor mai bune performanțe pentru majoritatea modelelor când sunt utilizate împreună cu metodele de selecție a trăsăturilor.

Influența metodei LASSO asupra performanței modelelor

Un parametru esențial care influențează rezultatele acestei metode este α, care reprezintă, așa cum am menționat și în prezentarea Formulei 4.6, gradul de penalizare al ponderilor trăsăturilor neimportante. Pentru a determina cea mai potrivită valoare pentru acest parametru am efectuat o serie de teste în care am analizat performanțele modelelor pentru diferite valori numerice ale lui α, precum [0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.5]. În urma rezultatelor obținute am ales valoarea 0.01.

Așa cum am menționat în subsecțiunea anterioară, deși metodele de scalare utilizate individual nu afectează performanța modelelor, în combinație cu alte metode acestea joacă un rol important în procesul de clasificare. Pentru metodele de selecție a trăsăturilor, precum LASSO, scalarea

datelor este esențială. De aceea, am analizat în ce mod influențează cele două tipuri de scalare rezultatele unui model, în cazul nostru Random Forest, când sunt folosite împreună cu LASSO.

Rezultatele obținute sunt prezentate în Figura 6.1. Pe axa y este reprezentată acuratețea modelului, iar pe axa x sunt reprezentate configurațiile pentru care se obțin aceste rezultate. Se poate observa că, în majoritatea cazurilor, am obținut rezultate mai bune pentru scalare de tip Z-score decât pentru scalare de tip Min-Max.

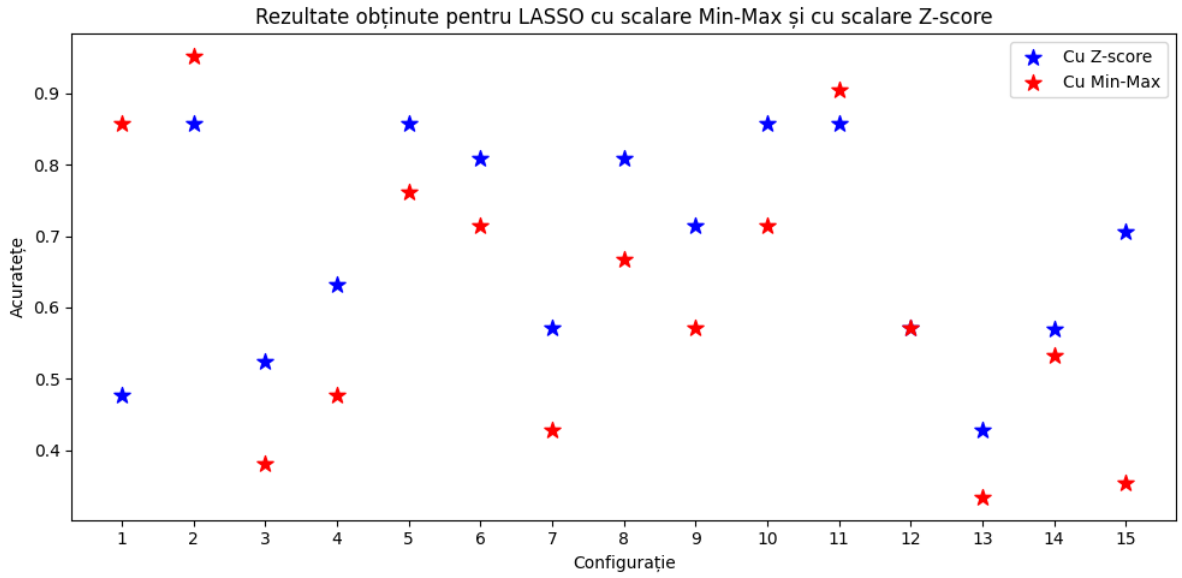

Figura 6.1: Influența tipului de scalare asupra metodei LASSO

Un alt aspect important de menționat aici este faptul că am încercat două abordări pentru selecția trăsăturilor. În prima abordare, la finalul reducerii ponderilor trăsăturilor irelevante, am ales din setul de date toate caracteristicile ale căror ponderi sunt nenule. Într-o a doua abordare, am selectat doar trăsăturile ale căror ponderi au o valoare mai mare decât un anumit prag prestabilit, în cazul experimentelor noastre acesta a fost setat la 0.01. Un motiv pentru care am ales această valoare îl reprezintă faptul că trăsăturile cu valori ale ponderilor foarte mici pot fi considerate neimportante. Un alt motiv îl reprezintă erorile numerice ce pot apărea în timpul procesului de regularizare.

În Figura 6.2 se poate observa diferența între acuratețea modelului Random Forest pentru cazul în care am utilizat LASSO fără o valoare de prag, adică am selectat toate trăsăturile cu ponderi nenule, și pentru cazul în care am aplicat LASSO cu o valoare de prag. Metoda de scalare a datelor utilizată în cadrul acestor experimente este scalarea de tip Z-score.

Pe axa ordonatelor sunt reprezentate valorile acurateții, iar pe axa absciselor configurațiile pentru care s-au obținut acestea. În urma efectuării tuturor experimentelor, am observat că abordarea în care selectăm trăsăturile cu ponderi nenule obține rezultate mai bune pentru majoritatea modelelor și a configurațiilor de testare, fapt ce poate fi observat și în graficul de mai jos.

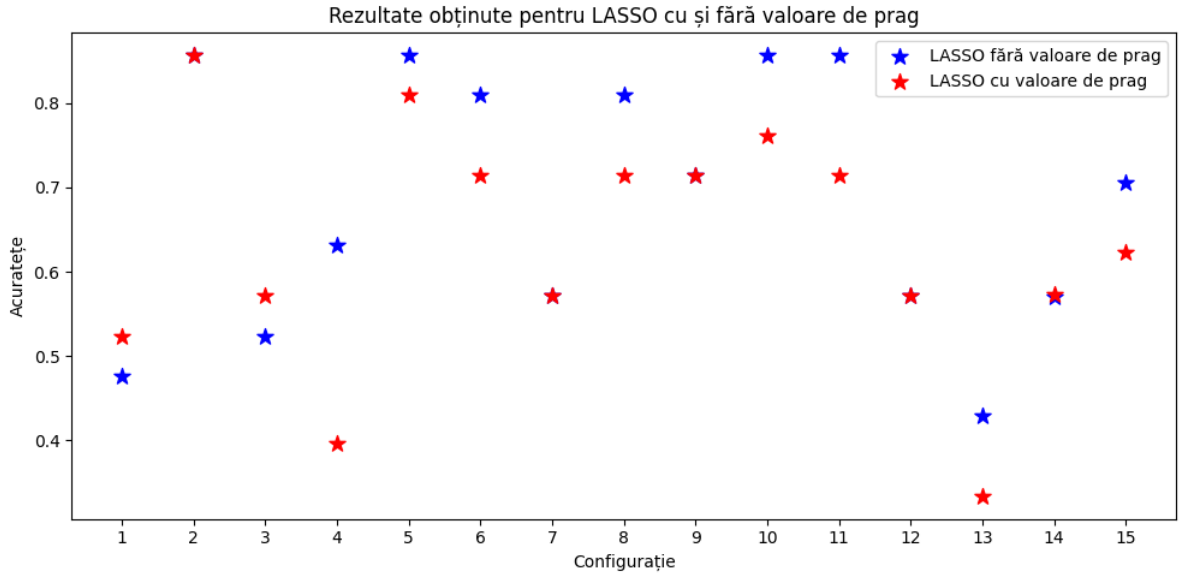

Figura 6.2: Diferența dintre LASSO cu valoare de prag și LASSO fără valoare de prag

Am analizat, de asemenea, modul în care utilizarea metodei LASSO influențează performanțele modelelor. Așa cum am menționat și în introducerea secțiunii, vom ilustra impactul acestei metode doar pentru algoritmul Random Forest și pentru primele 15 configurații de testare.

Rezultatele obținute în urma efectuării testelor sunt prezentate în graficul din Figura 6.3. Pe axa y sunt prezentate valorile acurateții, iar pe axa x configurațiile de testare pentru care s-au obținut aceste valori. Se poate observa cum, cu câteva excepții, 6/15, utilizarea metodei LASSO determină o creștere semnificativă a acurateții modelului de clasificare.

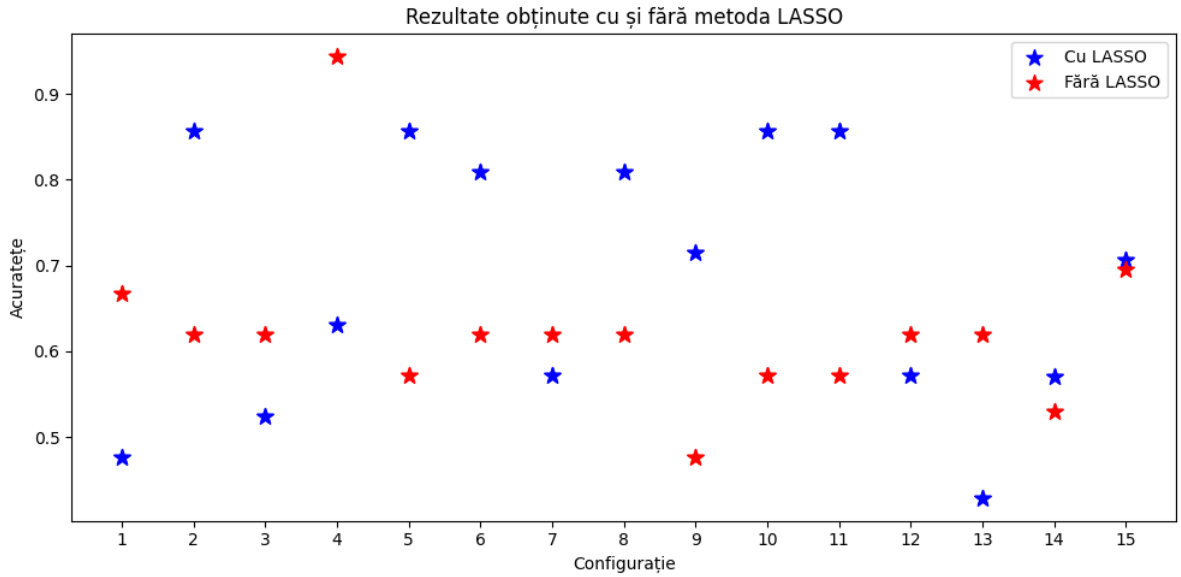

Figura 6.3: Influența utilizării metodei LASSO asupra acurateții

Influența metodei RFE asupra performanței modelelor

Un factor important ce influențează performanța acestei metode este alegerea potrivită a algoritmului de învățare automată utilizat pentru procesul de selecție al trăsăturilor relevante. În cadrul experimentelor noastre, am ales să utilizăm modelul Decision Tree Classifier, disponibil în bi-

blioteca scikit-learn [54]. Pentru parametrul *cv*, care specifică strategia de validare încrucișată utilizată, am folosit valoarea default 5, semnificând că la fiecare pas de validare se împarte setul de date în cinci grupuri, dintre care patru sunt folosite pentru etapa de antrenare, iar unul pentru etapa de testare. Criteriul de evaluare a performanței modelului poate fi setat prin intermediul parametrului *scoring*. În cazul problemei noastre am ales să folosim acuratețea.

Un mic dezavantaj al acestei metode este faptul că e mult mai costisitoare din punct de vedere computațional decât LASSO. Din acest motiv, nu am evaluat-o pentru toate metodele și pentru toate configurațiile de testare. Important de menționat aici este că sistemul de calcul pe care am efectuat experimentele este echipat cu un procesor Intel Core i7-1165G și 16GB de memorie RAM DDR4. Inițial, pentru cea mai simplă configurație, C3, timpul de execuție al metodei era aproximativ 24h. Am reușit să mai îmbunătățim puțin performanța modelului prin setarea parametrului *n_jobs* la valoarea -1, valoarea default fiind None. Acest parametru indică numărul de nuclee care lucrează în paralel în timpul antrenării modelului, iar -1 este echivalent cu faptul că se vor folosi toate nucleele disponibile. Astfel, în momentul de față, pentru C3, timpul de execuție al algoritmului a scăzut la aproximativ 2-3 minute, pentru configurații mai complexe precum C1, C2 sau C5-C11 durează în jur de 30-40 minute, iar pe configurațiile ce utilizează segmentarea la nivel de frază, C4, C14-C23, algoritmul încă are nevoie de câteva ore pentru antrenare. Timpii pot varia în funcție de configurația hardware a sistemului utilizat în cadrul experimentelor.

Pentru a putea analiza totuși diferența dintre rezultatele obținute utilizând metoda RFE și rezultatele obținute utilizând metoda LASSO, am antrenat algoritmul Random Forest pe configurațiile de testare C1-C3, C5-C10, fiind mai puțin complexe. Pentru scalarea datelor am utilizat metoda Z-score. În Figura 6.4 se pot observa rezultatele obținute. Pe axa y sunt prezentate valorile acurateții, iar pe axa x configurațiile de testare pentru care s-au obținut aceste rezultate.

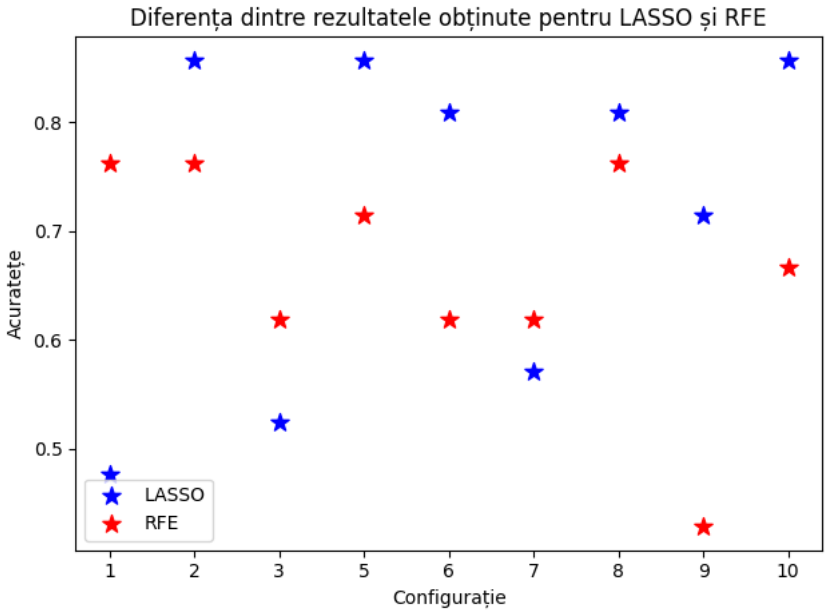

Figura 6.4: Comparație între metoda LASSO și metoda RFE

Observăm că RFE obține rezultate mai bune doar pentru 3/9 configurații și, exceptând C1, diferența între valorile acurateții obținute pentru cele două metode în aceste cazuri nu este foarte mare. Din acest motiv și datorită timpului de execuție mai scurt, am ales să utilizăm metoda LASSO în cadrul procesului de antrenare și testare al algoritmilor de clasificare.

Influența metodei PCA asupra performanței modelelor

Singurul parametru pe care l-am utilizat la această metodă este *n_components* care indică numărul de componente principale ce vor fi selectate la finalul procesului. În urma unei analize a performanțelor modelelor pentru diferite valori ale acestui argument, [5, 10, 25, 50, 100, 500], am observat că numărul de componente selectate nu influențează semnificativ rezultatele. În cadrul experimentelor noastre, parametrul a fost setat la 10.

La fel ca și în cazul celorlalte metode, am analizat influența pe care o are aplicarea metodei PCA asupra performanțelor modelelor. Condițiile de evaluare sunt aceleași, am utilizat algoritmul Random Forest și primele 15 configurații de testare.

Rezultatele obținute în urma experimentelor sunt prezentate în graficul din Figura 6.5. Pe axa y sunt reprezentate valorile acurateții obținute, iar pe axa x configurațiile de testare. Se poate observa că, exceptând configurațiile C1, C4 și C9, utilizarea metodei PCA determină creșterea valorii acurateții modelului.

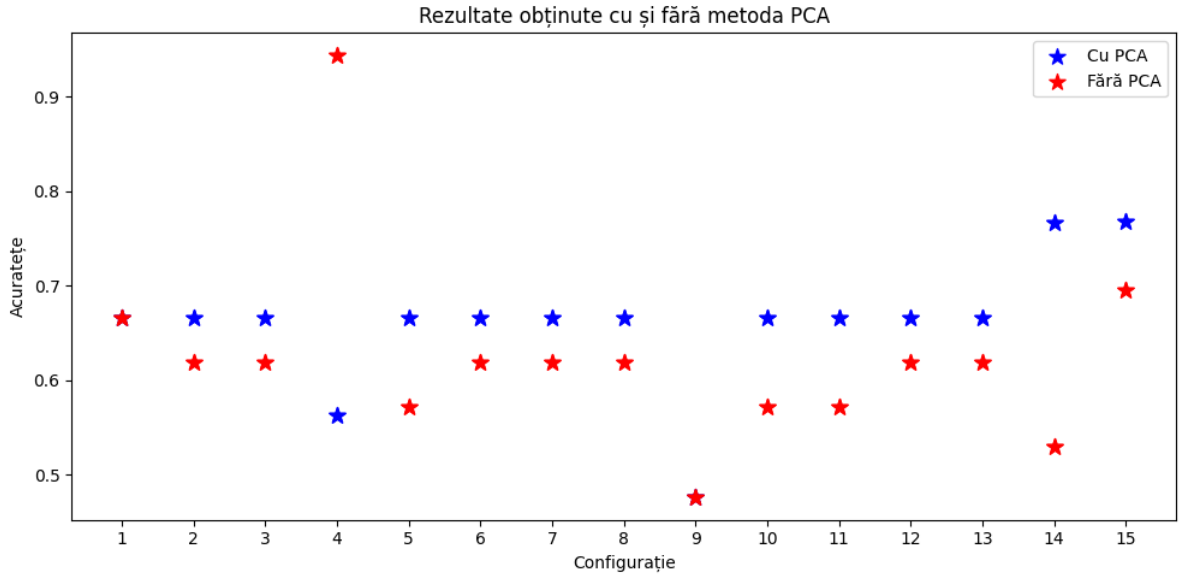

Figura 6.5: Influența utilizării metodei PCA asupra acurateții

6.2 Alegerea parametrilor potriviți pentru modelele de clasificare

Pentru obținerea celor mai bune rezultate în urma procesului de clasificare, un pas important a fost alegerea parametrilor potriviți pentru algoritmii de învățare automată care necesită acest lucru. Astfel, am evaluat performanța modelelor pentru diferite valori ale acestor parametrii, iar rezultatele obținute sunt prezentate în subsecțiunile următoare.

Random Forest

Pentru alegerea numărului optim de arbori de decizie care trebuie generați în timpul procesului de antrenare, am efectuat o serie de experimente pe configurația pentru care am obținut cele mai bune rezultate: C2 împreună cu metoda LASSO de selecție a trăsăturilor și metoda Min-Max de scalare a datelor, întrucât, pentru acest caz, am obținut rezultate mai bune cu acest tip de scalare.

Rezultatele obținute sunt ilustrate în Figura 6.6. Pe axa y sunt reprezentate valorile acurateții, iar pe axa x sunt reprezentate valorile numărului de arbori generați pentru fiecare iterație. Se poate observa că cele mai bune performanțe au fost atinse pentru valori cuprinse între 1000 și 2000, iar creșterea numărului de arbori peste valoarea 2000 determină scăderea acurateții. În cadrul experimentelor efectuate pentru procesul de clasificare am utilizat valoarea 1000 pentru parametrul *n_estimators*, ce corespunde numărului de arbori generați, și valoarea 500 pentru parametrul *random_state*.

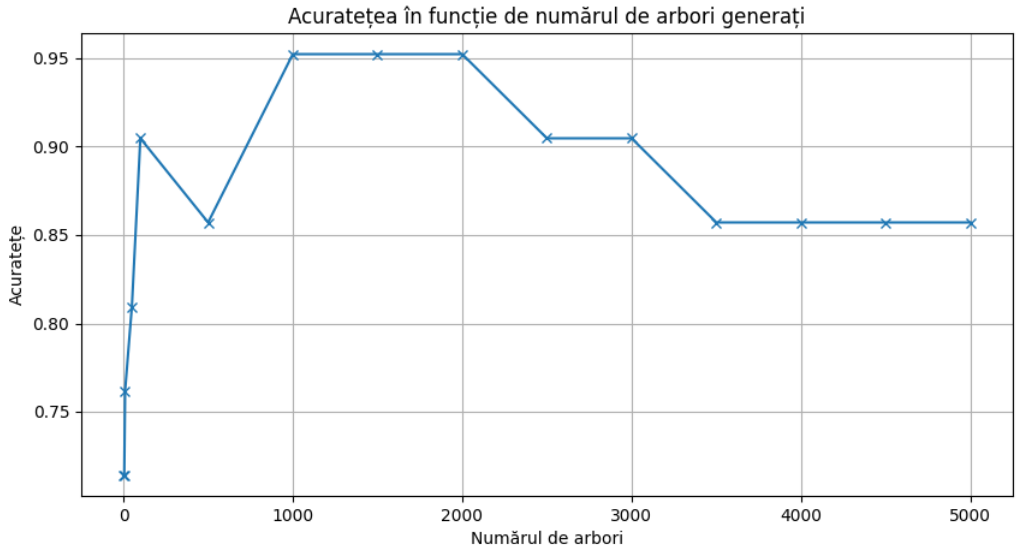

Figura 6.6: Influența numărului de arbori generați asupra acurateții

Gradient Boosting

Am procedat la fel ca în cazul metodei Random Forest pentru alegerea numărului de arbori generați. Am efectuat o serie de experimente utilizând C4, pentru care am obținut cele mai bune rezultate cu acest algoritm, fără aplicarea metodelor descrise în Secțiunea 4.4.

În Figura 6.7 sunt prezentate rezultatele obținute. Acuratețea este reprezentată pe axa y, iar numărul de arbori generați pe axa x. Se poate observa că cele mai bune rezultate se obțin pentru valori mai mari decât 500 și că, din acest punct, un număr mai mare de arbori nu afectează valoarea acurateții. În cadrul experimentelor noastre am ales valoarea 1000 pentru parametrul *n_estimators* și valoarea 500 pentru parametrul *random_state*.

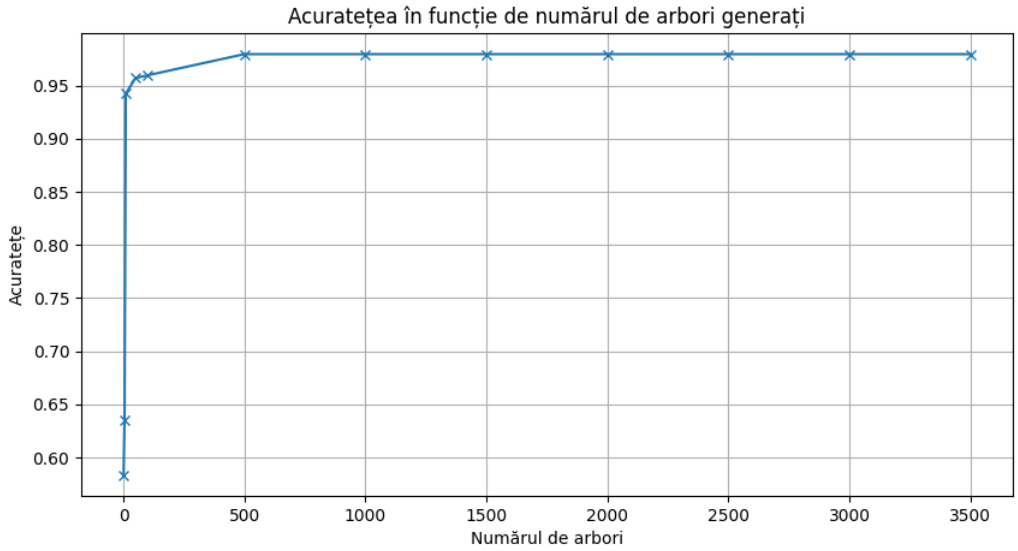

Figura 6.7: Influența numărului de arbori generați asupra acurateții

LightGBM

Pentru a alege valoarea optimă a parametrului *n_estimators*, am efectuat câteva experimente pentru C21, împreună cu metoda LASSO de selecție a trăsăturilor și a metodei de scalare a datelor Z-score. Am obținut diferite valori ale acurateții prezentate în Figura 6.8. Pe axa x este reprezentat numărul de arbori generați, iar pe axa y sunt reprezentate valorile rezultatelor obținute. Așa cum se poate observa și în figură, pentru un număr mai mare de 100 de estimatori valoarea acurateții rămâne constantă. În cadrul experimentelor efectuate am utilizat valoarea 500 atât pentru *n_estimators*, cât și pentru *random_state*.

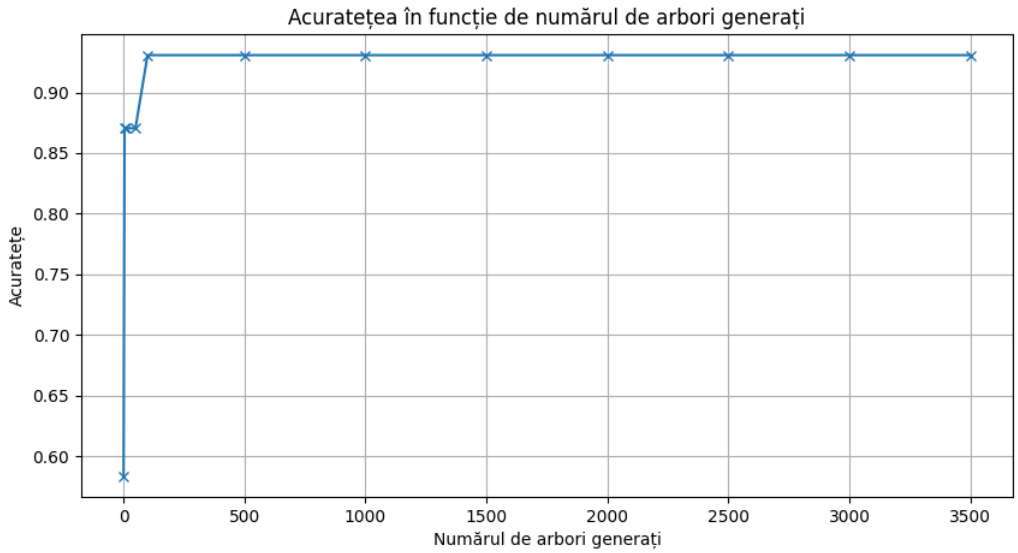

Figura 6.8: Influența numărului de arbori generați asupra acurateții

KNN

Un pas important în implementarea acestei metode este cel de selecție a parametrului *n_neighbors*, ce reprezintă numărul de vecini selectați. Am analizat performanțele obținute de model pentru C11, împreună cu metoda PCA de reducere a dimensiunii vectorilor de trăsături și metoda Z-score de scalare a datelor, și am obținut rezultatele ilustrate în Figura 6.9. Pe axa x este reprezentat numărul de vecini selectați, iar pe axa y valoarea acurateții obținute pentru fiecare caz. În cadrul experimentelor am setat parametrul *n_neighbors* la 5, fiind, așa cum se poate observa și în grafic, valoarea pentru care se obțin cele mai bune rezultate.

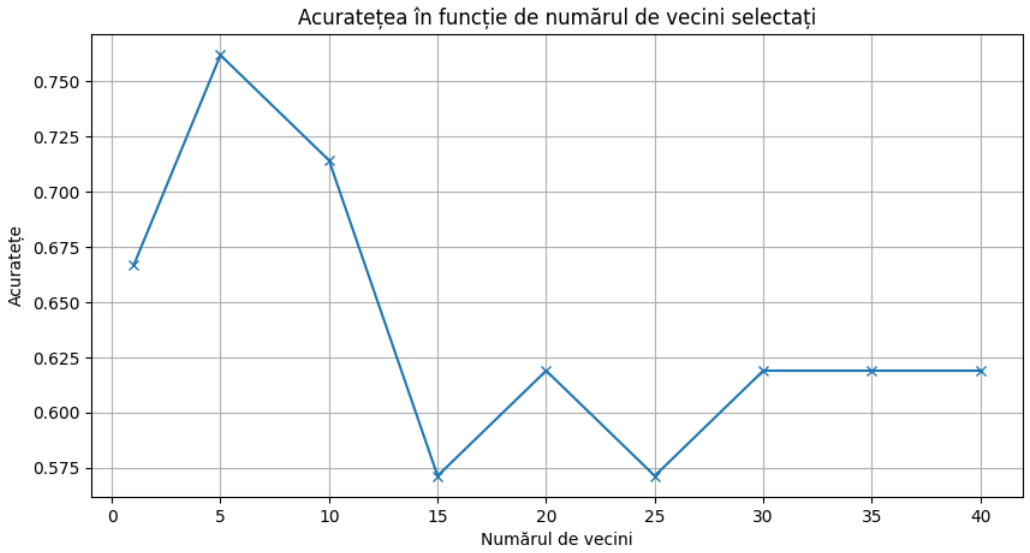

Figura 6.9: Influența numărului de vecini selectați asupra acurateții

SVM

Pentru a ilustra influența tipului de kernel selectat asupra performanței modelului am efectuat o serie de experimente pe C4, împreună cu metoda StratifiedKFold cross-validator, metoda PCA de reducere a vectorilor de trăsături, metoda LASSO pentru selecția trăsăturilor relevante și metoda Min-Max pentru scalarea datelor, fiind configurația pentru care am obținut cele mai bune rezultate. În Figura 6.10 sunt prezentate: pe axa x tipurile de kernel utilizate, iar pe axa y valorile acurateții obținute pentru fiecare situație. Se poate observa că cele mai bune performanțe au fost atinse pentru cazul în care am folosit kernel liniar, urmat, la o diferență relativ mică, de kernelul sigmoid.

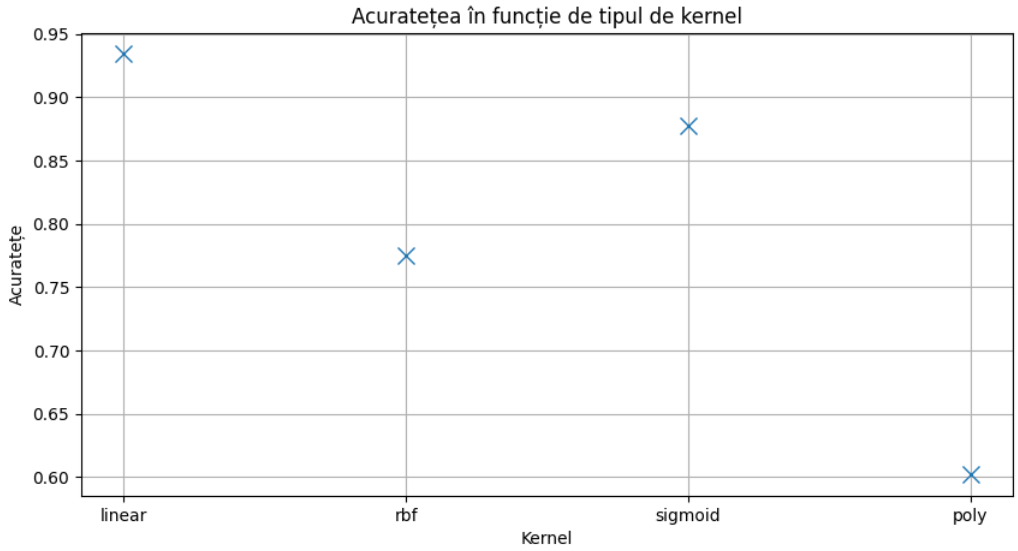

Figura 6.10: Influența tipului de kernel asupra acurateții

Un aspect important de menționat aici este faptul că deși kernelul liniar obține performanțe mai bune decât celelalte tipuri de kernel, pentru majoritatea configurațiilor pe care a fost testat, pentru problema analizată în această lucrare acesta s-a dovedit a fi foarte costisitor din punct de vedere computațional. Pentru configurațiile C15-C23 nu am analizat performanțele modelului cu acest tip de kernel deoarece timpul de execuție depășește 24h. Deoarece s-a dovedit a fi impractic, am decis să facem un compromis și am renunțat la cel mai bun rezultat în procesul de antrenare și testare. Pentru a putea analiza performanța algoritmului pe toate cele 23 de configurații, prezentate în Figura 4.3, și toate combinațiile de metode din Tabelul 5.1, am ales să utilizăm kernelul sigmoid.

6.3 Rezultate pentru fiecare configurație de testare

Tabelul 6.1 ilustrează cele mai bune valori ale acurateții obținute pentru fiecare configurație de testare din Figura 4.3.

Pe primul rând sunt reprezentați cei șapte algoritmi de clasificare selectați pentru rezolvarea problemei, respectiv Gradient Boosting (GB), Random Forest (RF), LightGBM (LGBM), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes (NB), Voting Classifier cu strategie de votare soft (Soft Voting) și Voting Classifier cu strategie de votare hard (Hard Voting). Pe prima coloană sunt prezentate configurațiile de testare.

Config	GB	RF	LGBM	SVM	KNN	NB	Soft Voting	Hard Voting
C1	0.85	0.85	0.66	0.76	0.85	0.83	**0.87**	0.85
C2	0.91	**0.95**	0.71	0.76	0.85	0.85	0.85	0.85
C3	0.70	0.70	0.71	0.71	0.71	0.70	0.75	**0.76**
C4	**0.97**	0.94	0.84	0.87	0.85	0.80	0.92	0.94
C5	0.80	0.80	0.71	0.76	0.85	0.79	0.79	**0.87**
C6	0.71	0.80	0.76	0.80	**0.85**	0.80	**0.85**	0.80
C7	0.66	0.66	0.61	0.76	0.79	0.66	**0.80**	0.76
C8	0.80	0.80	0.76	0.80	0.80	0.80	**0.85**	0.80
C9	0.70	0.66	0.61	0.71	0.75	0.62	**0.79**	**0.79**
C10	0.80	0.85	0.71	0.75	0.80	0.79	**0.87**	0.75
C11	0.90	**0.95**	0.71	0.76	0.90	0.79	0.90	0.85
C12	0.70	0.70	0.76	**0.76**	0.71	0.66	0.70	**0.76**
C13	0.66	0.66	0.71	0.71	0.76	**0.80**	0.71	0.71
C14	0.76	0.76	**0.80**	0.73	0.77	0.69	0.77	**0.80**
C15	0.77	0.77	0.84	0.80	0.77	0.79	**0.87**	0.78
C16	**0.88**	**0.88**	**0.88**	0.84	0.81	0.84	**0.88**	**0.88**
C17	0.76	**0.84**	0.79	0.80	0.77	0.74	**0.84**	0.76
C18	0.79	0.85	0.79	0.85	0.81	**0.89**	**0.89**	0.79
C19	0.62	0.65	0.62	**0.80**	0.71	0.71	0.70	0.70
C20	0.77	**0.81**	0.77	**0.81**	0.77	0.76	0.80	0.77
C21	0.81	0.81	**0.93**	0.78	0.77	0.81	0.87	0.81
C22	0.76	0.76	**0.80**	0.76	0.77	0.74	0.77	**0.80**
C23	0.76	0.76	0.76	0.73	**0.77**	**0.77**	**0.77**	0.76

Tabela 6.1: Cele mai bune valori ale acurateții obținute pentru fiecare configurație

Rezultatele din coloana corespunzătoare metodei SVM sunt obținute pentru cazul în care am utilizat kernel de tip sigmoid.

Cele mai potrivite metode de clasificare

Se poate observa că cea mai mare valoare a acurateții, 97%, am obținut-o cu algoritmul Gradient Boosting pentru C4. Cea mai robustă metodă este Soft Voting cu care obținem cele mai bune rezultate pentru 11/23 configurații, urmată de Hard Voting care atinge cele mai bune performanțe pentru 7/23 configurații.

Dacă nu luăm în considerare metodele de votare, observăm că Random Forest este cel pentru care obținem rezultate foarte bune pentru cele mai multe configurații, 5/23, urmat de LGBM cu 4/23 configurații. Metodele SVM și Naive Bayes obțin cele mai bune valori ale acurateții pentru 3/23 configurații, SVM pentru C12, C19 și C20, iar Naive Bayes pentru C13, C18 și C23. Algoritmul KNN reușește să obțină cele mai mari valori ale acurateții doar pentru C6 și C23, iar Gradient Boosting doar pentru C4 și C16.

Cele mai potrivite trăsături

Analiza rezultatelor obținute pentru cele 23 de configurații poate fi făcută în funcție de mai multe criterii.

Influența dimensiunii setului de date asupra performanțelor

În Secțiunea 4.2 am argumentat că o problemă cu care ne confruntăm este dimensiunea redusă a setului de date (număr mic de semnale) în raport cu numărul mare de trăsături. Din acest motiv am propus configurațiile de testare C4 și C14-C23, în care am îmbogățit setul de date segmentând textul la nivel de frază. Mai exact, am adăugat configurațiilor C1 respectiv C3, C5-C13, unde segmentarea era făcută la nivel de scenă, trăsături extrase la nivel de frază.

Primul criteriu de analiză este modul în care dimensiunea setului de trăsături afectează performanțele algoritmilor. Pentru a face asta vom compara rezultatele obținute pentru C1-C3, C5-C13, ce conțin doar trăsături extrase la nivel de scenă, cu configurațiile ce conțin trăsături extrase la nivel de frază, C4, C14-C23.

Pentru C1 vs. C4, se poate observa că, exceptând metodele KNN și Naive Bayes, pentru C4 obținem rezultate mai bune, deci creșterea numărului de semnale determină o îmbunătățire a performanțelor modelelor. O creștere a valorilor acurateții poate fi observată și pentru C14 în comparație cu C3, excepție făcând din nou algoritmul Naive Bayes. Comentariul legat de modul în care creșterea numărului de semnale implică o creștere a performanțelor, în medie cu 10%, pentru majoritatea modelelor este valabil și pentru C6 vs. C16, C7 vs. C17, C8 vs. C18, C12 vs. C22, C13 vs. C23.

Interesant de observat aici este cum pentru C5 și C15 se obțin rezultate similare, dar pentru metode diferite. C5 obține cele mai bune rezultate pentru algoritmii: Gradient Boosting, Random Forest, KNN și Hard Voting, iar C15 pentru celelalte metode, exceptând Naive Bayes pentru care obțin același rezultat.

Pentru C9 și C19 se obțin rezultate similare pentru toate metodele. Mai observăm cum pentru perechea C10 și C20, creșterea numărului de semnale a dus la îmbunătățirea performanțelor doar pentru 3 metode LightGBM, SVM și Hard Voting. Aceeași observație e valabilă și pentru C11 și C21. În acest caz, C21 obține rezultate mai bune doar pentru metodele LightGBM, SVM și Naive Bayes.

În concluzie, în cele mai multe dintre situații, creșterea numărului de semnale are un efect pozitiv asupra performanțelor.

Variabilitatea rezultatelor pentru fiecare configurație

Un alt aspect interesant de analizat aici este măsura în care performanțele algoritmilor diferă pentru fiecare configurație în parte.

Observăm că pentru configurații precum C3, C8, C12, C16 și C23 se obțin performanțe similare pentru toate metodele. Cu alte cuvinte, alegerea algoritmului nu are un impact semnificativ asupra rezultatelor atunci când utilizăm trăsăturile corespunzătoare acestor configurații. Reamintim că vectorii asociați acestor scenarii de testare se referă la metrici de tipul numărului de cuvinte, forma și lungimea acestora, la cuvintele funcționale și n-grame.

Pentru restul configurațiilor rezultatele diferă de la o metodă la alta, de exemplu pentru C19 obținem 80% acuratețe cu SVM și doar 62% acuratețe cu Gradient Boosting și LightGBM.

Influența tipului de trăsături asupra rezultatelor

Putem analiza, de asemenea, performanțele obținute în funcție de vectorii de trăsături pe care îi conțin, reprezentați în Figura 4.1. Luăm, spre exemplu, configurațiile în care segmentarea este la nivel de scenă, anume C1-C3, C5-C13.

Analizăm întâi configurațiile ce conțin un singur vector de trăsături, anume C8-C13, cu scopul de a verifica dacă există anumite trăsături pentru care se obțin rezultate bune atunci când sunt utilizate individual, nu doar în combinații. În acest mod putem deduce și care ar putea fi cele mai relevante caracterisitici stilistice pentru problema analizată. Observăm că cele mai bune rezultate se obțin cu utilizarea TF-IDF pentru cuvinte, fără cele funcționale (C11), urmată de TF-IDF pentru toate cuvintele (C10) și TF-IDF pentru 2-grame (C8). De asemenea, rezultatele

sugerează că prezența cuvintelor funcționale scade performanțele algoritmilor.

Evaluăm apoi configurațiile formate din mai mulți vectori de trăsături, anume C1-C3, C5-C7, pentru a analiza în ce mod sunt afectate performanțele modelelor dacă utilizăm combinații de caracteristici stilistice. Pentru această categorie, observăm că obținem rezultate bune cu configurațiile C1, C2, C5 și C6. Reamintim că C1 este formată din toate trăsăturile extrase la nivel de scenă, reprezentate în Figura 4.1, C2 conține toate reprezentările TF-IDF, C5 conține toate reprezentările TF-IDF pentru cuvinte, cu și fără cele funcționale, iar C6 este format din reprezentarea TF-IDF pentru 2-grame, reprezentarea TF-IDF pentru părți de vorbire și reprezentarea TF-IDF pentru semne de punctuație. O observație importantă aici este că toate aceste configurații includ trăsăturile pentru care am obținut cele mai bune rezultate și pentru prima categorie analizată, TF-IDF pentru cuvinte, fără cele funcționale (C11), TF-IDF pentru toate cuvintele (C10) și TF-IDF pentru 2-grame (C8).

Pentru a analiza relevanța trăsăturilor am verificat care sunt cele rămase în subsetul de caracteristici în urma aplicării metodei LASSO, adică cele ale căror ponderi au rămas nenule în urma regularizării. Din rezultatele obținute reiese că cele mai importante trăsături fac parte din categoriile: TF-IDF pentru toate cuvintele, TF-IDF pentru cuvinte, fără cele funcționale și TF-IDF pentru 2-grame. Din această observație și rezultatele prezentate anterior, putem concluziona că acestea sunt cele mai relevante seturi de trăsături pentru problema analizată, deci cele ce reprezintă cel mai bine marca stilistică a autorului.

6.4 Rezultate finale

Tabelul 6.2 sumarizează cele mai bune rezultate obținute pentru fiecare dintre cele 7 modele de clasificare.

Pe prima coloană a tabelului sunt prezentați algoritmii de învățare automată selectați, respectiv Gradient Boosting (GB), Random Forest (RF), Voting Classifier cu strategie de votare hard (Hard Voting), LightGBM (LGBM), Voting Classifier cu strategie de votare soft (Soft Voting), K-Nearest Neighbors (KNN), Naive Bayes (NB) și Support Vector Machine (SVM).

Pe a doua coloană sunt reprezentate configurațiile de testare, prezentate în Figura 4.3, iar pe a treia coloană metodele de procesare a setului de trăsături, prezentate în Secțiunea 4.4 și în Tabelul 5.1, pentru care s-au obținut aceste rezultate.

Măsurile de performanță, acuratețea, precizia, sensibilitatea (recall) și scorul F1, sunt prezentate pe primul rând.

Algoritm	Config	Metode	Acuratețe	Precizie	Recall	Scorul F1
GB	C4	-	0.97	0.98	0.97	0.97
RF	C2	LASSO, Min-Max	0.95	0.96	0.93	0.94
Hard Voting	C4	-	0.94	0.95	0.93	0.94
LGBM	C21	LASSO, Z-score	0.93	0.94	0.91	0.92
Soft Voting	C4	Min-Max	0.92	0.93	0.92	0.92
KNN	C11	PCA, Z-score	0.90	0.93	0.87	0.89
NB	C18	LASSO, Z-score	0.89	0.89	0.90	0.89
SVM	C4	StratKFold, PCA, LASSO, Min-Max	0.87	0.87	0.87	0.87

Tabela 6.2: Cele mai bune rezultate

Așa cum se poate observa și în tabel, am obținut cele mai bune rezultate utilizând algoritmul Gradient Boosting, pentru C4. Motivul pentru care celula corespunzătoare metodelor este goală este faptul că aplicarea acestora nu influențează rezultatele modelului pentru această configurație.

Pentru Random Forest am obținut rezultate similare primului model pentru C2 asupra cărei au fost aplicate metoda LASSO de selecție a trăsăturilor și metoda Min-Max de scalare a datelor.

Clasificatorul de votare a obținut performanțe similare pentru cele două strategii: hard și soft, pentru configurația de testare C4. Se poate observa însă că, în timp ce pentru Hard Voting aplicarea metodelor de procesare a setului de date nu afectează rezultatele modelului, pentru Soft Voting se obțin performanțe mai bune în momentul în care se efectuează scalarea datelor.

LightGBM atinge cele mai bune performanțe pentru C21, asupra căreia se aplică metoda LASSO de selecție a trăsăturilor și metoda Z-score de scalare a datelor.

Metodele KNN și Naive Bayes obțin rezultate similare pentru configurațiile C11, respectiv C18. Se poate observa că pentru KNN aplicarea metodelor PCA și Z-score a dus la valori mari ale metricilor de performanță, în timp ce în cazul modelului Naive Bayes aplicarea metodei LASSO de selecție a trăsăturilor relevante este cea care joacă un rol important în obținerea unor rezultate bune.

Rezultatele din tabel pentru modelul SVM au fost obținute pentru cazul în care am utilizat funcția kernel sigmoid. Se poate observa că cele mai bune performanțe au fost atinse pentru C4 asupra căreia au fost aplicate metodele PCA, LASSO și Min-Max. De asemenea, utilizarea metodei Stratified K-Fold cross-validator a jucat un rol important în obținerea acestor rezultate. Pentru aceeași configurație, utilizând kernel liniar, am obținut valori de 93% pentru toate cele patru metrici de performanță. Așa cum am menționat și în Subsecțiunea 6.2, datorită timpului mare de execuție nu am evaluat rezultatele modelului pentru toate configurațiile de testare utilizând kernel liniar, am ales să renunțăm la cel mai bun rezultat și am folosit funcția kernel sigmoid în cadrul experimentelor.

7 Concluzii

Lucrarea și-a propus testarea unor metode de învățare automată pentru identificarea autorului unui text. Am folosit ca bază de date texte aparținând lui William Shakespeare, despre care se cunoaște faptul că au fost scrise împreună cu John Fletcher, și am utilizat concluziile din două lucrări de referință în domeniu cu privire la autorul fiecărei scene pentru a adnota textele.

Pentru a determina care sunt cele mai potrivite metode pentru rezolvarea problemei, am aplicat șapte algoritmi de clasificare supervizată, respectiv Gradient Boosting, Random Forest, Voting Classifier, LightGBM, KNN, Naive Bayes și SVM, cu rezultate încurajatoare.

Am avut în vedere, de asemenea, selectarea unor trăsături relevante pentru a caracteriza stilistic un text. Pentru aceasta, am creat 23 de configurații de testare, 12 conțin trăsături extrase la nivel de scenă și 11 conțin trăsături extrase la nivel de frază. Am analizat performanțele modelelor pentru toate cele 23 de configurații. Pe baza rezultatelor obținute am observat cum, pentru majoritatea cazurilor, creșterea numărului de semnale implică, în medie, o creștere a performanțelor modelelor cu aproximativ 10%.

Am dedus, pe baza performanțelor obținute de modele pentru toate configurațiile de testare și a trăsăturilor selectate de metoda LASSO, că cele mai importante trăsături sunt: TF-IDF pentru toate cuvintele, TF-IDF pentru cuvinte, fără cele funcționale și TF-IDF pentru 2-grame. La polul opus se află reprezentările TF-IDF pentru cuvinte funcționale, TF-IDF pentru semne de punctuație și TF-IDF pentru părți de vorbire care nu reprezintă categorii stilistice definitorii stilului de scris al celor doi dramaturgi. Am observat, de asemenea, cum utilizarea individuală a acestor trăsături nu reprezintă întotdeauna un criteriu suficient pentru a defini maniera de scriere a unui autor, fapt ce reiese și din lucrările prezentate în Capitolul 2.

În continuarea studiului actual intenționăm să analizăm performanțele modelelor de clasificare selectate pentru cazul în care seturile de antrenare și testare sunt formate din scene alese în mod aleatoriu din cele două piese, pentru a ne asigura că algoritmii nu se supra-specializează pe piesa de antrenare, în cazul nostru "The Two Noble Kinsmen". Ne dorim, de asemenea, să abordăm problema și prin tehnici de învățare automată nesupervizată și să testăm și alte tipuri de trăsături, precum distanța dintre cuvintele funcționale și frecvențele structurii ritmice. În plus, ne propunem să evaluăm modul în care integrarea unor modele preantrenate, precum Bidirectional Encoder Representations from Transformers (BERT) sau Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA), în procesul de extragere a trăsăturilor afectează performanța modelelor de clasificare selectate. Totodată, intenționăm să analizăm performanțele algoritmilor aleși pentru noi configurații de testare, formate doar din combinații ale celor mai relevante categorii de trăsături, și să dezvoltăm aplicația astfel încât să poată identifica modificări stilistice și în alte tipuri de text.

Bibliografie

[1] James S Ackerman, „A theory of style", în *The Journal of Aesthetics and Art Criticism* 20.3 (1962), pp. 227–237.

[2] Naomi Altman și Martin Krzywinski, „The curse (s) of dimensionality", în *Nat Methods* 15.6 (2018), pp. 399–400.

[3] Salha M Alzahrani, Naomie Salim și Ajith Abraham, „Understanding plagiarism linguistic patterns, textual features, and detection methods", în *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.2 (2011), pp. 133–149.

[4] Eric Backer și Peter van Kranenburg, „On musical stylometry—a pattern recognition approach", în *Pattern Recognition Letters* 26.3 (2005), pp. 299–309.

[5] Delia Bacon, *The Philosophy of the Plays of Shakspere Unfolded*, BoD–Books on Demand, 2019.

[6] Abdellghani Bellaachia și Edward Jimenez, „Exploring performance-based music attributes for stylometric analysis", în *International Journal of Computer and Information Engineering* 3.7 (2009), pp. 1795–1797.

[7] Steven Bird, Ewan Klein și Edward Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*, " O'Reilly Media, Inc.", 2009.

[8] Leonard Brandwood, „Stylometry and chronology", în (1992).

[9] Leo Breiman, „Random forests", în *Machine learning* 45 (2001), pp. 5–32.

[10] Andrew Brinkman, Daniel Shanahan și Craig Sapp, „Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of josquin", în *Proc. of the Biennial Int. Conf. on Music Perception and Cognition*, 2016, pp. 91–97.

[11] Marcelo Luiz Brocardo, Issa Traore și Isaac Woungang, „Authorship verification of e-mail and tweet messages applied for continuous authentication", în *Journal of Computer and System Sciences* 81.8 (2015), pp. 1429–1440.

[12] Charles Bukowski, *Burning in Water, Drowning in Flame*, Ecco, 1974.

[13] Jie Cai et al., „Feature selection in machine learning: A new perspective", în *Neurocomputing* 300 (2018), pp. 70–79.

[14] Daniel Castro-Castro, Carlos Alberto Rodriguez-Lozada și Rafael Muñoz, „Mixed Style Feature Representation and B-maximal Clustering for Style Change Detection.", în *CLEF (Working Notes)*, 2020.

[15] Girish Chandrashekar și Ferat Sahin, „A survey on feature selection methods", în *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[16] Kevin Clark et al., „Electra: Pre-training text encoders as discriminators rather than generators", în *arXiv preprint arXiv:2003.10555* (2020).

[17] Jacob Devlin et al., „Bert: Pre-training of deep bidirectional transformers for language understanding", în *arXiv preprint arXiv:1810.04805* (2018).

[18] Yoav Freund și Robert E Schapire, „A decision-theoretic generalization of on-line learning and an application to boosting”, în *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[19] Jerome H Friedman, „Greedy function approximation: a gradient boosting machine”, în *Annals of statistics* (2001), pp. 1189–1232.

[20] David I Holmes și Richard S Forsyth, „The Federalist revisited: New directions in authorship attribution”, în *Literary and Linguistic computing* 10.2 (1995), pp. 111–127.

[21] David I Holmes și Judit Kardos, „Who was the author? An introduction to stylometry”, în *Chance* 16.2 (2003), pp. 5–8.

[22] David L Hoover, „Another perspective on vocabulary richness”, în *Computers and the Humanities* 37 (2003), pp. 151–178.

[23] Jonathan Hope, *The authorship of Shakespeare's plays: a socio-linguistic study*, Cambridge University Press, 1994.

[24] Marjan Hosseinia și Arjun Mukherjee, „A Parallel Hierarchical Attention Network for Style Change Detection”, în (2018).

[25] F Howedi et al., „Authorship Attribution of Short Historical Arabic Texts using Stylometric Features and a KNN Classifier with Limited Training Data”, în *Journal of Computer Science* 16.10 (2020), p. 2020.

[26] Xuan Huang, Lei Wu și Yinsong Ye, „A review on dimensionality reduction techniques”, în *International Journal of Pattern Recognition and Artificial Intelligence* 33.10 (2019), p. 1950017.

[27] Gordon Hughes, „On the mean accuracy of statistical pattern recognizers”, în *IEEE transactions on information theory* 14.1 (1968), pp. 55–63.

[28] James M Hughes et al., „Empirical mode decomposition analysis for visual stylometry”, în *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2147–2157.

[29] Jonathan Hui, *Machine Learning — Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)*, Accesat: 2023, 2019, URL: https://medium.com/@jonathan-hui/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491.

[30] Michael Irving, *Computer Analysis reveals Shakespeare's collaborators*, Accesat: 2023, 2016, URL: https://newatlas.com/algorithm-shakespeare-coauthor-marlowe/46130/.

[31] Aarish Iyer și Soroush Vosoughi, „Style Change Detection Using BERT.”, în *CLEF (Working Notes)* 93 (2020), p. 106.

[32] Xinyin Jiang et al., „Style Change Detection: Method Based On Pre-trained Model And Similarity Recognition”, în *training* 1400.7000 (2022), p. 7000.

[33] Patrick Juola, „How a computer program helped show JK Rowling write a Cuckoo's Calling”, în *Scientific American, August 20th* (2013).

[34] Daniel Karas, Martyna Spiewak și Piotr Sobecki, „OPI-JSA at CLEF 2017: Author Clustering and Style Breach Detection.”, în *CLEF (Working Notes)*, 2017.

[35] Guolin Ke et al., „Lightgbm: A highly efficient gradient boosting decision tree”, în *Advances in neural information processing systems* 30 (2017).

[36] Jamal Ahmad Khan, „Style Breach Detection: An Unsupervised Detection Model.”, în *CLEF (Working Notes)*, 2017.

[37] Moshe Koppel și Yaron Winter, „Determining if two documents are written by the same author", în *Journal of the Association for Information Science and Technology* 65.1 (2014), pp. 178–187.

[38] Nancy M Laan, „Stylometry and method. The case of Euripides", în *Literary and Linguistic Computing* 10.4 (1995), pp. 271–278.

[39] Qidi Lao et al., „Style Change Detection Based On Bert And Conv1d", în CLEF, 2022.

[40] Gerard Ledger și Thomas Merriam, „Shakespeare, fletcher, and the two noble kinsmen", în *Literary and Linguistic Computing* 9.3 (1994), pp. 235–248.

[41] R. D. LORD, „Studies in the history of probability and statistics. VIII. De Morgan and the Statistical study of literary style", în *Biometrika* 45.1-2 (Iun. 1958), pp. 282–282.

[42] Robert AJ Matthews și Thomas VN Merriam, „Neural computation in stylometry I: An application to the works of Shakespeare and Fletcher", în *Literary and Linguistic computing* 8.4 (1993), pp. 203–209.

[43] Cristhian Mayor et al., „A single author style representation for the author verification task.", în *CLEF (Working Notes)*, 2014, pp. 1079–1083.

[44] Seifeddine Mechti et al., „Author profiling using style-based features", în *Notebook Papers of CLEF2* (2013).

[45] Thomas Corwin Mendenhall, „The characteristic curves of composition", în *Science* 214s (1887), pp. 237–246.

[46] Thomas Merriam, „The Authorship of Sir Thomas More", în *ALLC Bulletin. Association for Library and Linguistic Computing Bangor* 10.1 (1982), pp. 1–7.

[47] Thomas Merriam, „What Shakespeare wrote in Henry VIII (part I)", în *The Bard* 1 (1979), pp. 81–94.

[48] Thomas Merriam, „What Shakespeare wrote in Henry VIII (part II)", în *The Bard* 2 (1980), 111–n8.

[49] Erwan Moreau și Carl Vogel, „Style-based Distance Features for Author Profiling-Notebook for PAN at CLEF 2013", în *PAN Lab at CLEF 2013*, 2013, Online–proceedings.

[50] Andrew Queen Morton, *Literary detection: How to prove authorship and fraud in literature and documents*, Scribner, 1978.

[51] Frederick Mosteller și David L Wallace, „Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers", în *Journal of the American Statistical Association* 58.302 (1963), pp. 275–309.

[52] Sukanya Nath, „Style change detection by threshold based and window merge clustering methods.", în *CLEF (Working Notes)*, 2019.

[53] Rodrigo Ribeiro Oliveira și Rosalvo Ferreira Oliveira Neto, „Using Character n-grams and Style Features for Gender and Language Variety Classification.", în *CLEF (Working Notes)*, 2017.

[54] Fabian Pedregosa et al., „Scikit-learn: Machine learning in Python", în *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[55] Steven T Piantadosi, „Zipf's word frequency law in natural language: A critical review and future directions", în *Psychonomic bulletin & review* 21 (2014), pp. 1112–1130.

[56] Petr Plecháč, „Relative contributions of Shakespeare and Fletcher in Henry VIII: An analysis based on most frequent words and most frequent rhythmic patterns", în *Digital Scholarship in the Humanities* 36.2 (2021), pp. 430–438.

[57] Hanchao Qi, Armeen Taeb și Shannon M Hughes, „Visual stylometry using background selection and wavelet-HMT-based Fisher information distances for attribution and dating of impressionist paintings", în *Signal Processing* 93.3 (2013), pp. 541–553.

[58] Judy T Raj, *A beginner's guide to dimensionality reduction in Machine Learning*, Accesat: 2023, 2019, URL: https://towardsdatascience.com/dimensionality-reduction-for-machine-learning-80a46c2ebb7e.

[59] Francisco Rangel et al., „Overview of the 3rd Author Profiling Task at PAN 2015", în *CLEF*, sn, 2015, p. 2015.

[60] G Thippa Reddy et al., „Analysis of dimensionality reduction techniques on big data", în *Ieee Access* 8 (2020), pp. 54776–54788.

[61] David E Rumelhart, Geoffrey E Hinton și Ronald J Williams, „Learning representations by back-propagating errors", în *nature* 323.6088 (1986), pp. 533–536.

[62] Kamil Safin și Aleksandr Ogaltsov, „Detecting a change of style using text statistics", în *Working Notes of CLEF* (2018).

[63] Gerard Salton, Anita Wong și Chung-Shu Yang, „A vector space model for automatic indexing", în *Communications of the ACM* 18.11 (1975), pp. 613–620.

[64] Nils Schaetti, „Bidirectional Echo State Network-based Reservoir Computing for Cross-domain Authorship Attribution", în (2018).

[65] Bernhard Schölkopf, Christopher JC Burges, Alexander J Smola et al., *Advances in kernel methods: support vector learning*, MIT press, 1999.

[66] Santiago Segarra et al., „Attributing the Authorship of the" Henry VI" Plays by Word Adjacency", în *Shakespeare Quarterly* 67.2 (2016), pp. 232–256.

[67] William Shakespeare, „Riverside Shakespeare", în (1974), ed. de G. Blakemore Evans.

[68] Shams Al-Shamasi și Mohamed Menai, „Ensemble-Based Clustering for Writing Style Change Detection in Multi-Authored Textual Documents", în CLEF, 2022.

[69] Mahesh Sonawane, *Why do ML models need to be cross validated?*, Accesat: 2023, 2023, URL: https://towardsdev.com/why-do-ml-models-need-to-be-cross-validated-76e0144b40d6.

[70] Carlos Oscar Sánchez Sorzano, Javier Vargas și A Pascual Montano, „A survey of dimensionality reduction techniques", în *arXiv preprint arXiv:1403.2877* (2014).

[71] Constantina Stamou, „Stylochronometry: Stylistic Development, Sequence of Composition, and Relative Dating", în *Literary and Linguistic Computing* 23.2 (Oct. 2007), pp. 181–199.

[72] Eivind Strøm, „Multi-label Style Change Detection by Solving a Binary Classification Problem.", în *CLEF (Working Notes)*, 2021, pp. 2146–2157.

[73] Jiliang Tang, Salem Alelyani și Huan Liu, „Feature selection for classification: A review", în *Data classification: Algorithms and applications* (2014), p. 37.

[74] Matthew F Tennyson, „A replicated comparative study of source code authorship attribution", în *2013 3rd International Workshop on Replication in Empirical Software Engineering Research*, IEEE, 2013, pp. 76–83.

[75] *The Curse of Dimensionality*, Accesat: 2023, 2022, URL: https://www.baeldung.com/cs/curse-of-dimensionality.

[76] Robert Tibshirani, „Regression shrinkage and selection via the lasso", în *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[77] Gerard V Trunk, „A problem of dimensionality: A simple example”, în *IEEE Transactions on pattern analysis and machine intelligence* 3 (1979), pp. 306–307.

[78] Michael Tschuggnall et al., „Overview of the author identification task at PAN-2017: style breach detection and author clustering”, în *Working Notes Papers of the CLEF 2017 Evaluation Labs/Cappellato, Linda [edit.]; et al.* 2017, pp. 1–22.

[79] Radu Tudor Ionescu et al., „Unmasking the abnormal events in video”, în *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2895–2903.

[80] Arjuna Tuzzi și Michele A Cortelazzo, *Drawing Elena Ferrante's Profile: Workshop Proceedings, Padova, 7 September 2017*, Padova UP, 2018.

[81] Ben Verhoeven și Walter Daelemans, „CLiPS Stylometry Investigation (CSI) corpus: A Dutch corpus for the detection of age, gender, personality, sentiment and deception in text”, în *LREC 2014-NINTH INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION*, 2014, pp. 3081–3085.

[82] Cynthia Whissell, „Traditional and emotional stylometric analysis of the songs of Beatles Paul McCartney and John Lennon”, în *Computers and the Humanities* 30 (1996), pp. 257–265.

[83] Soner Yıldırım, *What Makes LightGBM Light*, Accesat: 2023, 2020, URL: https://towardsdatascience.com/what-makes-lightgbm-light-8eb4dc258046.

[84] Eva Zangerle et al., „Overview of the style change detection task at pan”, în (2021).

[85] Eva Zangerle et al., „Overview of the Style Change Detection Task at PAN 2020.”, în *CLEF (Working Notes)* 93 (2020).

[86] Zhijie Zhang et al., „Style change detection based on writing style similarity”, în (2021).

[87] J Zi, L Zhou și Z Liu, „Style Change Detection Based On Bi-LSTM And Bert”, în CLEF, 2022.

[88] Dimitrina Zlatkova et al., „An ensemble-rich multi-aspect approach for robust style change detection”, în *CLEF 2018 Evaluation Labs and Workshop–Working Notes Papers, CEUR-WS. org*, 2018.

[89] Amin Zollanvari, Alex Pappachen James și Reza Sameni, „A theoretical analysis of the peaking phenomenon in classification”, în *Journal of Classification* 37 (2020), pp. 421–434.

[90] S Zubrzycki, „Concerning Yule's characteristic of style”, în *Applicationes Mathematicae* 4.4 (1959), pp. 328–331.

[91] Chaoyuan Zuo, Yu Zhao și Ritwik Banerjee, „Style Change Detection with Feed-forward Neural Networks.”, în *CLEF (Working Notes)* 93 (2019).