



Universitatea
Transilvania
din Brașov

FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

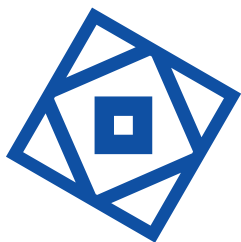
LUCRARE DE LICENȚĂ

Absolvent: Lopătaru Mihnea

Coordonator: Lect. Dr. Plajer Ioana Cristina

Sef. Lucr. Dr. Ing. Danciu Gabriel-Mihail

Brașov
Iulie 2024



Universitatea
Transilvania
din Brașov

FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

LUCRARE DE LICENȚĂ

CAUSE-IT

Absolvent: Lopătaru Mihnea

Coordonatori: Lect. Dr. Plajer Ioana Cristina

Sef. Lucr. Dr. Ing. Danciu Gabriel-Mihail

Brașov
Iulie 2024

Cuprins

Listă de figuri	3
Listă de acronime	4
1 Introducere	5
1.1 Scop	6
1.2 Definiții	6
1.3 Conținutul lucrării	9
2 Metodologii	11
2.1 Starea actuală a cercetării	11
2.2 Metodologia propusă	12
2.2.1 Arhitectura sistemului	13
2.2.2 Curățarea și pregătirea datelor	13
2.2.3 Antrenarea modelului	14
2.2.4 Evaluarea și salvarea rezultatelor	15
2.3 Pașii de cercetare	15
3 Arhitectura Aplicației	16
3.1 Frontend	17
3.1.1 Motivația alegerii platformei web	17
3.1.2 Arhitectura Angular	18
3.1.3 Analiza fluxului de utilizare	19
3.1.3.1 Autentificarea	20
3.1.3.2 Pagina principală	22
3.1.3.3 Încărcarea datelor	23
3.1.3.4 Configurarea modelului	26
3.1.3.5 Vizualizarea rezultatelor	28
3.1.3.6 Profilul utilizatorului	30
3.1.3.7 Pagina de Setări	31
3.2 BridgeAPI	32
3.2.1 Rolul și importanța BridgeAPI	33
3.2.1.1 Separarea funcționalităților	33
3.2.1.2 Performanță și scalabilitate	34
3.2.1.3 Securitate îmbunătățită	34
3.2.1.4 Îmbunătățirea ciclurilor de actualizare	34
3.2.2 Arhitectura BridgeAPI	35
3.2.2.1 Controllerul de specializări	35

3.2.2.2	Controllerul de autentificare	36
3.2.2.3	Controllerul de token-uri	38
3.2.2.4	Controllerul de fișiere	39
3.2.2.5	Controllerul de notificări	41
3.2.2.6	Controllerul de comunicare cu ModelOPS	42
3.3	ModelOPS	45
3.3.1	Descărcarea datelor	45
3.3.2	Curățarea și pregătirea datelor	46
3.3.2.1	Unificarea fișierelor cu denumiri similare	47
3.3.2.2	Curățarea generală a datelor	48
3.3.2.3	Codificarea valorilor de tip text	49
3.3.2.4	Scalarea valorilor numerice	50
3.3.3	Antrenarea modelului și salvarea rezultatelor	51
4	Modelul de învățare automată	53
4.1	Prezentarea algoritmului Random Forest	53
4.1.1	Principiul de funcționare	53
4.1.2	Avantajele și dezavantajele algoritmului	54
4.1.3	Utilizări comune ale algoritmului Random Forest	55
4.2	Scopul și motivația utilizării algoritmului Random Forest	55
4.3	Implementarea modelului	57
4.3.1	Biblioteci utilizate	57
4.3.2	Selectarea caracteristicilor relevante	57
4.3.3	Antrenarea modelului Random Forest	58
4.4	Rezultate și validarea modelului	59
4.4.1	Setul de date artificial	60
4.4.1.1	Generarea datelor	61
4.4.1.2	Rezultatele obținute	64
4.4.1.3	Validarea rezultatelor	66
4.4.2	Setul de date real public	69
4.4.2.1	Descrierea setului de date	69
4.4.2.2	Rezultatele obținute	71
4.4.2.3	Validarea rezultatelor	72
4.4.3	Setul de date real privat	72
5	Concluzii	74
5.1	Dezvoltări viitoare	75
	Rezumat	76
	Abstract	77

Listă de figuri

1	Fluxul aplicației CauselT	10
2	Arhitectura aplicației CauselT	17
3	Interfața de autentificare și înregistrare a platformei CauselT . .	20
4	Diagrama de comunicare între frontend și server pentru autentificare	21
5	Pagina principală a platformei CauselT	22
6	Schimbarea dinamică a iconiței de notificări	23
7	Meniu profil utilizator	23
8	Interfața de încărcare a datelor	24
9	Diagrama de comunicare între frontend și server pentru încărcarea datelor	25
10	Interfața de configurare a modelului (Setări de bază)	26
11	Interfața de configurare a modelului (Setări Avansate)	27
12	Interfața de configurare a modelului (Setări Curățare Date) . . .	27
13	Pagina de vizualizare a rezultatelor	28
14	Lista parametrilor extrași și importanța lor	28
15	Compararea pacienților sănătoși și bolnavi	29
16	Analize statistice ale setului de date	29
17	Informațiile afișate în cadrul paginii de profil	31
18	Pagina de setări a aplicației CauselT	32
19	Diagrama fluxului de comunicare între BridgeAPI și ModelOPS .	44
20	Diagrama fluxului de comunicare pentru descărcarea datelor . .	46
21	Matricea de corelație pentru setul de date artificial	62
22	Colesterol vs Rezultatul bolii	63
23	Indicele de masă corporală (BMI) vs Rezultatul bolii	63
24	Graficul de importanță SHAP pentru setul de date artificial . . .	65
25	Comparația graficului original cu cel generat de Random Forest pentru parametrul Cholesterol	66
26	Comparația graficului original cu cel generat de Random Forest pentru parametrul BMI	66
27	Importanța parametrilor identificată prin PCA	67
28	Importanța parametrilor identificată de Random Forest	68
29	Importanța parametrilor conform adevărului cunoscut	69
30	Graficul de importanță SHAP pentru setul de date real	72

Listă de acronime

.NET Domain (.) NETwork.

API Application Programable Interface.

ASP.NET Core Active Server Pages Network Enabled Technologies.

HTML Hypertext Markup Language.

HTTP HyperText Transfer Protocol.

IIS Internet Information Services.

IT Information Technology.

JSON JavaScript Object Notation.

JWT JSON Web Token.

ML Machine Learning.

REST Representational State Transfer.

SCSS Sassy Cascading Style Sheets.

TMU Taipei Medical University.

URL Uniform Resource Locator.

1 Introducere

În ultimii ani, domeniul medical a înregistrat progrese remarcabile datorită integrării tehnologiilor avansate de IT, în special a celor de învățare automată (machine learning). Aceste tehnologii permit analizarea și interpretarea volumelor mari de date medicale cu o acuratețe și viteză considerabil sporite. Învățarea automată a revoluționat diagnosticarea bolilor, personalizarea tratamentelor și prognoza evoluției stării de sănătate a pacienților.

Astfel de tehnologii se bazează pe algoritmi sofisticati¹ care învață și fac predicții pe baza datelor existente. În contextul medical, acești algoritmi sunt capabili să analizeze seturi complexe de date clinice, imagini medicale și secvențe genetice, extrăgând informații esențiale și identificând tipare subtile. Un exemplu notabil îl constituie utilizarea algoritmilor de machine learning pentru detectarea precoce a cancerului pe baza imaginilor de screening² sau pentru anticiparea riscului de boli cardiovasculare prin analiza datelor clinice.

Algoritmii de ML contribuie semnificativ și la dezvoltarea medicinei personalizate³. Prin analiza datelor genetice și clinice individuale, aceștia pot recomanda tratamente adaptate specificului fiecărui pacient, sporind eficacitatea terapeutică și minimizând riscurile asociate tratamentelor. Astfel, tehnologiile de învățare automată nu doar optimizează tratamentele, ci și îmbunătățesc calitatea vieții pacienților.

Bineînțeles, integrarea acestor algoritmi în practica medicală prezintă și provocări semnificative, cum ar fi asigurarea confidențialității și securității datelor, interpretabilitatea rezultatelor și validarea clinică a acestora. În ciuda acestor provocări, beneficiile substanțiale ale acestor tehnologii sugerează un potențial enorm pentru transformarea serviciilor medicale.

Această lucrare își propune cercetarea aplicării tehnologiilor de machine learning în medicină, concentrându-se pe dezvoltarea unei aplicații care permite încărcarea și analiza seturilor de date ale pacienților suferind de anumite afecțiuni. Aplicația identifică caracteristicile esențiale și importanța acestora în cadrul bolii, oferind totodată analize statistice și comparații detaliate.

¹Algoritmi sofisticati: Seturi complexe de reguli și operații matematice utilizate pentru a prelucra date și a genera predicții.

²Screening: Procedură medicală de examinare pentru detectarea unei boli înainte ca simptomele să apară.

³Medicina personalizată: O abordare medicală care ajustează tratamentele și intervențiile în funcție de caracteristicile individuale ale pacientului.

1.1 Scop

Scopul principal al acestei lucrări este explorarea potențialului algoritmilor avansați de învățare automată (Machine Learning - ML) în identificarea cauzelor specifice ale bolilor, prin dezvoltarea și implementarea aplicației CauselT. Inițial dezvoltată în cadrul companiei Siemens pentru a identifica factorii declanșatori⁴ ai cancerului pancreatic, aplicația a demonstrat un succes semnificativ, motiv pentru care a fost extinsă pentru a acoperi o gamă mai largă de afecțiuni. Această adaptabilitate și scalabilitate⁵ subliniază flexibilitatea și capacitatea aplicației de a servi drept un instrument valoros pentru comunitatea medicală.

Obiectivul CauselT este de a oferi medicilor un instrument avansat care să faciliteze nu doar diagnosticarea rapidă a pacienților, ci și înțelegerea cauzelor și factorilor care contribuie la dezvoltarea bolilor. Prin integrarea tehnicilor sofisticate de ML, aplicația urmărește să furnizeze o analiză detaliată a datelor medicale, permițând identificarea tendințelor, tiparelor și corelațiilor care pot juca un rol crucial în elaborarea strategiilor de prevenție, diagnosticare și tratament.

Mai mult, CauselT aspiră să depășească limitările metodologiilor tradiționale de diagnosticare, oferind o perspectivă mai amplă asupra factorilor de risc și a condițiilor preexistente ale pacienților, îmbunătățind astfel precizia diagnosticelor și eficacitatea tratamentelor propuse. În esență, aplicația reprezintă o punte între datele brute și deciziile clinice informate, contribuind la avansarea înțelegerii și managementului bolilor în era digitală.

În concluzie, prin CauselT, această lucrare își propune să aducă o contribuție semnificativă în domeniul medical, oferind o perspectivă nouă asupra diagnosticării și tratamentului personalizat al bolilor, îmbunătățind calitatea îngrijirilor medicale și eficientizând procesele de luare a deciziilor în practica medicală.

1.2 Definiții

Pentru a clarifica termenii și tehnologiile esențiale utilizate în dezvoltarea aplicației CauselT, această secțiune oferă definiții precise și detaliate ale componentelor cheie.

Angular este un cadru de dezvoltare open-source⁶ bazat pe TypeScript⁷,

⁴Factori declanșatori: Elementele sau condițiile care contribuie la apariția unei boli.

⁵Scalabilitate: Capacitatea unui sistem de a gestiona o cantitate crescută de muncă sau de a fi extins pentru a răspunde cerințelor crescute.

⁶Open-source: software pentru care codul sursă este disponibil publicului, permițând oricui să îl inspecteze, modifice și îmbunătățească.

⁷TypeScript: un limbaj de programare open-source dezvoltat de Microsoft, care este un superset al JavaScript și adaugă tipizare statică opțională.

destinat construirii unor aplicații web single-page⁸. Dezvoltat și susținut de către Google, Angular introduce o serie de caracteristici inovatoare, cum ar fi arhitectura bazată pe componente, data binding bidirecțional, injecția de dependențe și multe altele, facilitând astfel dezvoltarea de aplicații complexe și performante [3]. Acest cadru de dezvoltare promovează scrierea codului într-o manieră curată și structurată, facilitând atât o evoluție rapidă, cât și o mentenanță ulterioară ușoară a aplicațiilor.

Conform documentației oficiale Angular [4], acesta oferă un set de unelte și o platformă bine încheiată pentru a simplifica atât construcția interfeței utilizatorului, cât și interacțiunile cu serverul, punând accent pe performanță și reutilizabilitate. Arhitectura Angular oferă un grad de flexibilitate ridicat, permițând construirea unor aplicații scalabile, ușurând tranziția de la simple site-uri web la aplicații de tip enterprise⁹ [2].

C# este un limbaj de programare obiect-orientat, dezvoltat de Microsoft, ce oferă un echilibru între viteză și productivitate în dezvoltarea de aplicații. Este folosit predominant în cadrul platformei .NET pentru a construi o varietate de soluții software, de la aplicații web până la aplicații mobile și servicii cloud [26].

ASP.NET Core, un framework open-source dezvoltat de Microsoft, reprezintă evoluția tehnologiei ASP.NET. Este proiectat pentru a permite dezvoltarea rapidă și eficientă a aplicațiilor web și API-urilor, fiind optimizat pentru cloud. Având o arhitectură modulară și fiind complet decuplat de IIS, ASP.NET Core permite rularea aplicațiilor pe diverse platforme, inclusiv Linux și macOS, asigurând astfel flexibilitate și extensibilitate [27].

Prin utilizarea **ASP.NET Core**, dezvoltatorii pot crea servere web care beneficiază de funcționalități avansate precum injecția de dependențe, rutarea bazată pe atribute, filtrele de acțiune și un sistem bine pus la punct de middleware. Aceste caracteristici, împreună cu suportul pentru tehnologii front-end diverse, fac din ASP.NET Core o soluție ideală pentru construirea aplicațiilor web moderne, performante și scalabile.

Python este un limbaj de programare de nivel înalt, interpretat, care pune un accent deosebit pe claritatea și simplitatea codului. Este cunoscut pentru sintaxa sa simplă și ușor de înțeles, făcându-l accesibil pentru începători, dar suficient de puternic pentru dezvoltatorii experimentați, datorită bibliotecilor sale extinse și cadrelor de dezvoltare (framework-uri) [31].

FastAPI¹⁰ este un framework modern pentru Python, destinat creării de

⁸Single-page: o aplicație web sau un site web care interacționează cu utilizatorul prin re-încărcarea dinamică a unei singure pagini web, în loc de încărcarea paginilor web separate în browser.

⁹Enterprise: aplicații software concepute pentru a satisface nevoile complexe ale organizațiilor mari, în locul utilizatorului individual sau al grupurilor mai mici.

¹⁰FastAPI: Un framework modern, rapid (de înaltă performanță), pentru construirea de API

API-uri web rapid și fără efort. Având la bază standardele ASGI (Asynchronous Server Gateway Interface), FastAPI permite dezvoltarea de aplicații asincrone care pot gestiona mii de cereri simultan, fiind o alegere excelentă pentru proiecte web de înaltă performanță. Prin utilizarea Python type hints¹¹, FastAPI facilitează dezvoltarea rapidă, oferind în același timp validare automată a datelor și generare de documentație interactivă pentru API-uri, cum ar fi Swagger UI¹² și ReDoc¹³ [34].

MinIO este o soluție open-source pentru stocarea obiectelor, optimizată pentru scalabilitate și performanță în medii cloud-native. Suportând arhitecturi de tip scale-out, MinIO este compatibil cu API-ul Amazon S3, facilitând implementarea flexibilă a sistemelor de stocare pentru date nestructurate la nivel enterprise. Proiectat pentru integrare nativă cu Kubernetes, MinIO permite desfășurarea eficientă și gestionarea resurselor de stocare în medii containerizate, adresând cerințele moderne de stocare masivă a datelor. Prin asigurarea compatibilității extinse și focalizarea pe securitatea datelor, MinIO reprezintă o componentă esențială în construcția infrastructurilor de date complexe, oferind o soluție robustă pentru aplicații care necesită acces rapid și sigur la volume mari de date nestructurate [28].

MongoDB este o bază de date NoSQL¹⁴ orientată pe documente, proiectată pentru a oferi scalabilitate înaltă, performanță și o gestionare flexibilă a datelor. Spre deosebire de modelele tradiționale relaționale, MongoDB suportă scheme de date flexibile, permițând dezvoltatorilor să stocheze și să interogheze date într-un format ce se aseamănă cu JSON. Această abordare facilitează integrarea rapidă a datelor în aplicații prin eliminarea necesității de a transforma obiectele aplicației într-un model de date relaționale.

Articolul "Improving Storage and Read Performance for Free: Flat vs Structured Schemas" de Artur Costa [14], abordează eficiența schemelor plate în comparație cu cele structurate în contextul MongoDB, evidențiind modul în care organizarea datelor poate influența performanța de stocare și citire. Autorul subliniază avantajele schemelor plate în termeni de simplificare a accesului la date și reducere a overhead-ului de stocare, oferind astfel o perspectivă valoroasă asupra optimizării performanței bazei de date în aplicații scalabile.

¹¹Python type hints: Anotatii adăugate în codul Python pentru a specifica tipurile de date ale variabilelor, facilitând astfel verificarea tipurilor de date în timpul dezvoltării.

¹²Swagger UI: O interfață de utilizare grafică care permite utilizatorilor să exploreze și să testeze API-uri web documentate prin specificația Swagger.

¹³ReDoc: O aplicație open source care generează documentație pentru API-uri web bazată pe specificația OpenAPI, oferind o interfață de utilizare îmbunătățită.

¹⁴NoSQL: O clasă de sisteme de management al bazelor de date care diferă de modelul tradițional relațional, oferind o manieră de stocare și recuperare a datelor modelată în moduri diverse față de tabelele utilizate în relațional.

1.3 Conținutul lucrării

Această lucrare oferă o înțelegere detaliată a componentelor și tehnologiilor esențiale utilizate în dezvoltarea aplicației CauselT. Vor fi prezentate aspectele legate de dezvoltarea interfeței utilizatorului, componenta de intermediere și procesarea datelor, precum și rezultatele obținute, demonstrând corectitudinea tehnicilor aplicate.

Frontend

Capitolul Frontend detaliază dezvoltarea interfeței utilizatorului folosind Angular 16. Acest capitol va evidenția modul în care acest cadru de dezvoltare facilitează o experiență de utilizare fluidă și dinamică. Se va discuta arhitectura bazată pe componente și integrarea cu serviciile backend pentru a asigura o interactivitate eficientă între utilizatorul final și aplicație.

BridgeAPI

Capitolul BridgeAPI explorează componenta de intermediere a aplicației. Această componentă servește ca un intermediar între interfața grafică și serverul responsabil pentru procesele de învățare automată. În acest capitol se va discuta despre importanța acestei componente în asigurarea securității utilizatorilor și motivele pentru utilizarea unor soluții specifice de stocare a datelor, cum ar fi cloud-ul sau bazele de date.

ModelOPS

Capitolul ModelOps se concentrează pe serverul de Python, componenta principală responsabilă pentru procesarea, curățarea și analizarea datelor. Acest capitol va aborda modul în care serverul furnizează soluții bazate pe valorile încărcate, subliniind rolul esențial pe care această componentă îl joacă în structura generală a aplicației.

Aceste componente colaborează pentru a forma o platformă destinată optimizării procesului de diagnosticare medicală. Aplicația oferă profesioniștilor din domeniul sănătății un instrument avansat, capabil să furnizeze informații critice care pot fi esențiale pentru diagnosticarea și managementul bolilor. Figura 1 ilustrează secvențial etapele fluxului operațional al aplicației, demonstrând interacțiunea dintre utilizatori și platformă.

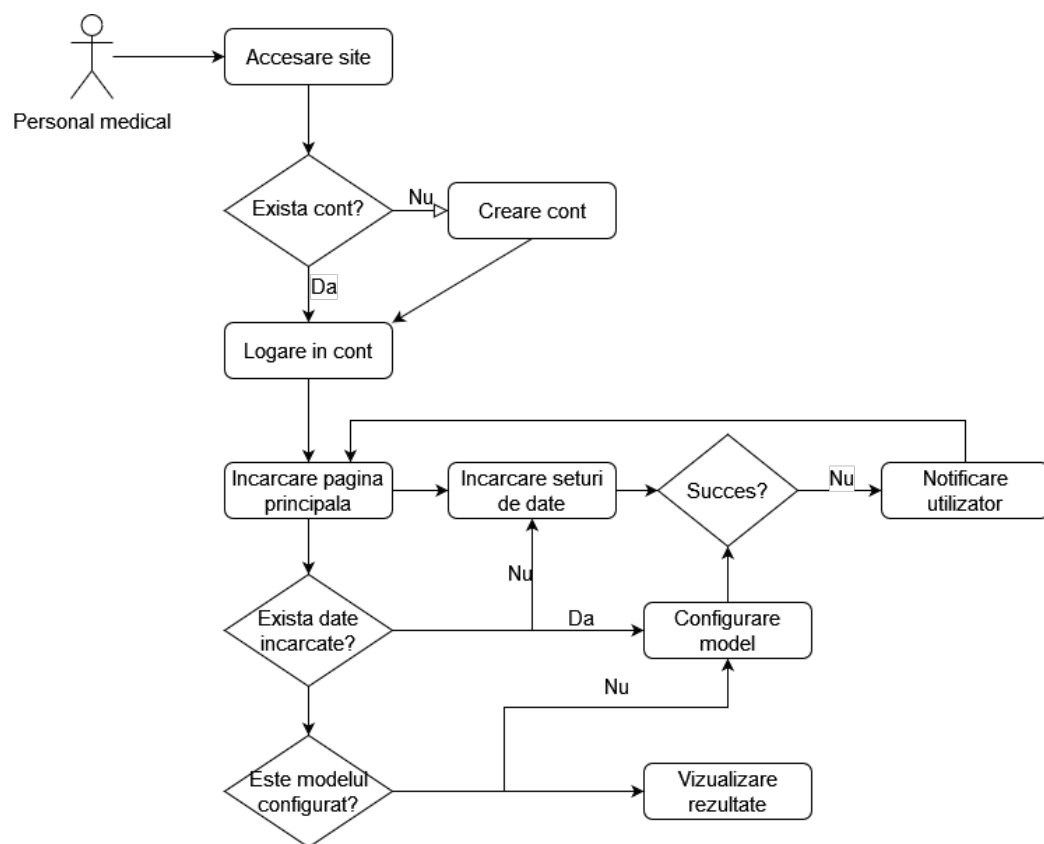


Figura 1: Fluxul aplicației CauselT

2 Metodologii

Capitolul Metodologii descrie abordările și tehnicile utilizate pentru dezvoltarea și implementarea aplicației CauselT. Se va prezenta starea actuală a cercetării în domeniul algoritmilor de învățare automată aplicată în medicină și tehnologiile utilizate în dezvoltarea platformei. În plus, capitolul va detalia metodologia propusă, incluzând arhitectura sistemului, procesele de curățare și pregătire a datelor, antrenarea modelului și evaluarea performanței. Un segment important este dedicat pașilor de cercetare, subliniind colaborarea cu compania Siemens și experți medicali din cadrul proiectului IHelp [25].

2.1 Starea actuală a cercetării

În ultimii ani, utilizarea algoritmilor de învățare automată în scopuri medicale a avut o dezvoltare semnificativă, datorită progreselor în domeniul inteligenței artificiale și a capacității tot mai mari de prelucrare a datelor. Această secțiune prezintă o perspectivă asupra tehnologiilor curente utilizate în dezvoltarea de soluții software în domeniul medical, evidențiind tehnologiile de frontend, backend și baze de date adoptate în proiectul actual.

Algoritmi de machine learning în medicină

Utilizarea algoritmilor de ML în medicină se concentrează pe îmbunătățirea proceselor de diagnosticare, personalizarea tratamentelor și optimizarea managementului resurselor. Studii recente, cum ar fi cele realizate de Badawy et al. [6] și Sidey-Gibbons et al. [33], au demonstrat capacitatea algoritmilor de ML de a analiza seturi mari de date medicale, detectând anomalii și corelații care ar putea rămâne neobservate în analiza umană. Exemple relevante includ algoritmi pentru detectarea precoce a cancerului, predicția evoluției bolilor cronice și dezvoltarea de noi medicamente. Această tendință spre digitalizare și automatizare în medicină subliniază nevoia de soluții software avansate, integrate, capabile să prelucreze și să analizeze date complexe în timp real.

Tehnologii frontend: Angular 16

Angular 16 reprezintă o alegere modernă pentru dezvoltarea interfețelor pentru utilizator, oferind un set bogat de funcționalități care facilitează crearea de aplicații web single-page dinamice. Prin adoptarea Angular 16, proiectul beneficiază de performanță îmbunătățită, o mai bună gestionare a stărilor și un model de dezvoltare bazat pe componente, care sprijină modularitatea și reu-

tilizarea codului. Această tehnologie este în concordanță cu cerințele actuale de agilitate și eficiență în dezvoltarea web.

Serverul de C#: ASP.NET Core 8

ASP.NET Core 8 marchează ultima generație a platformei open-source de la Microsoft pentru construirea de aplicații web și servicii. Alegerea ASP.NET Core 8 pentru serverul de C# reflectă angajamentul proiectului către performanță, securitate și scalabilitate. Oferind suport cross-platform, acest framework permite o integrare eficientă cu alte tehnologii și sisteme, facilitând dezvoltarea de soluții robuste pentru industria medicală.

Serverul de python: Python 3.10

Python 3.10 aduce îmbunătățiri semnificative în ceea ce privește sintaxa și funcționalitățile limbajului, sprijinind dezvoltarea de algoritmi de ML și procesarea avansată a datelor. Prin utilizarea Python, proiectul se aliniază la standardele comunității științifice și de cercetare, profitând de un ecosistem bogat de biblioteci și cadre de lucru specializate în inteligență artificială și învățare automată.

Sisteme de stocare: MinIO și MongoDB

Adoptarea MinIO, o soluție de stocare a obiectelor optimizată pentru cloud și scalabilitate, împreună cu MongoDB, o bază de date NoSQL orientată pe documente, reprezintă o arhitectură de date modernă și flexibilă, adecvată pentru gestionarea volumelor mari de date structurate și nestructurate. Această combinație optimizează performanța și eficiența procesării datelor, facilitând implementarea de soluții scalabile și de încredere în domeniul medical.

Prin integrarea tehnologiilor de ultimă generație precum Angular 16, ASP.NET Core 8, Python 3.10, MinIO și MongoDB, proiectul actual beneficiază de cele mai avansate unelte, ce pot rezolva problemele medicinei moderne. Astfel, este construită o platformă robustă dar și modulară, oferind personalului medical un suport în combaterea bolilor.

2.2 Metodologia propusă

Această secțiune detaliază metodologia utilizată în dezvoltarea și implementarea platformei CauselT, acoperind arhitectura sistemului, procesele de curățare a datelor, antrenarea modelelor de învățare automată și evaluarea performanței acestora.

2.2.1 Arhitectura sistemului

Platforma CauseIT este construită pe o arhitectură cu trei straturi, fiecare având un rol specific în procesul de analiză și procesare a datelor. Această structură modulară asigură scalabilitatea și flexibilitatea necesară pentru a gestiona date complexe și voluminoase din domeniul medical.

- **Frontend:** Stratul de interfață cu utilizatorul este dezvoltat utilizând Angular 16. Acesta oferă o experiență de utilizare fluidă și interactivă, facilitând accesul rapid la funcționalitățile platformei.
- **Server intermediar:** Serverul intermediar este implementat folosind ASP.NET Core și are rolul de a gestiona comunicarea între frontend și backend. Acesta preprocesează și validează cererile utilizatorilor, asigurând securitatea și integritatea datelor transmise.
- **Server ModelOPS:** Backend-ul, dezvoltat în Python 3.10, se ocupă de procesarea avansată a datelor și antrenarea modelelor de învățare automată. Acesta este responsabil pentru execuția algoritmilor și furnizarea rezultatelor analitice.

Această arhitectură pe trei niveluri permite separarea clară a funcționalităților, optimizând atât performanța cât și securitatea sistemului.

2.2.2 Curățarea și pregătirea datelor

Un set de date medical tipic conține atât valori numerice, cât și categorice. De exemplu, un astfel de set poate include informații despre pacienți precum genul (categoric: M/F), vârsta (numeric), obiceiuri (de exemplu, fumatul, consumul de alcool - numeric), simptome (de exemplu, tuse, dificultăți la înghițire - numeric) și diagnostice (de exemplu, prezența sau absența cancerului pulmonar - categoric). Fiecare rând reprezintă un pacient, iar fiecare coloană reprezintă o caracteristică specifică a acestuia.

În cadrul seturilor de date utilizate în această lucrare, există o coloană care indică stadiul bolii, aceasta putând fi fie numerică, fie categorică. Restul coloanelor, fie ele numerice sau categorice, reprezintă rezultatele analizelor medicale ale diferiților parametri. Fiecare set de date conține informații despre mai mulți pacienți, asigurând astfel o bază de date robustă și variată pentru analizele ulterioare.

Procesul de curățare a datelor este esențial pentru a obține un set de date robust și fiabil. Aceasta implică mai mulți pași importanți:

- **Eliminarea liniilor și coloanelor cu un procentaj ridicat de valori lipsă:** Se elimină rândurile și coloanele care conțin multe valori lipsă pentru a asigura integritatea datelor. Se definește un prag (de exemplu, 70%) peste care datele incomplete sunt eliminate.
- **Eliminarea valorilor anormale utilizând metode statistice:** Se utilizează tehnici statistice pentru a identifica și elimina valorile aberante (outliers) care ar putea distorsiona analizele. Valorile considerate anormale sunt eliminate pentru a îmbunătăți calitatea datelor.
- **Standardizarea valorilor text:** Se convertește textul la litere mici și se elimină spațiile inutile pentru a asigura consistența. Aceasta ajută la evitarea duplicatelor și la îmbunătățirea acurateții analizelor.
- **Identificarea și eliminarea coloanelor duplicate:** Se identifică și se elimină coloanele duplicate care pot introduce redundanță în date. Se utilizează funcții de verificare a similarității coloanelor pentru a detecta duplicatele.

2.2.3 Antrenarea modelului

Antrenarea modelului de învățare automată implică următorii pași:

- **Pregătirea setului de date:** După curățare, datele sunt împărțite în seturi de antrenament și de testare. De obicei, 70-80% din date sunt utilizate pentru antrenament și restul pentru testare.
- **Selectarea caracteristicilor:** Caracteristicile relevante sunt selectate pe baza importanței lor statistice și a relevanței pentru problema analizată. Se utilizează metode statistice pentru a evalua semnificația caracteristicilor.
- **Antrenarea modelului:** Modelul este antrenat folosind setul de date de antrenament. Parametrii modelului sunt ajustați pentru a optimiza performanța.
- **Optimizarea procesului:** Antrenarea modelului este un proces intensiv din punct de vedere computațional. Pentru a optimiza resursele, setul de date este împărțit în bucăți (chunk-uri) mai mici, care sunt procesate secvențial. Aceasta evită utilizarea excesivă a memoriei și îmbunătățește viteza de antrenare.

2.2.4 Evaluarea și salvarea rezultatelor

După antrenare, modelul este testat pentru a evalua performanța și a salva rezultatele. Aceasta implică:

- **Testarea modelului:** Modelul este evaluat utilizând setul de date de testare. Metricile de evaluare oferă o imagine de ansamblu asupra performanței modelului.
- **Salvarea rezultatelor:** Rezultatele evaluării și parametrii importanți identificați de model sunt salvați pentru analiză ulterioară. Rezultatele includ:
 - **Parametrii importanți:** Lista parametrilor care influențează boala specifică, împreună cu un procentaj care indică gradul de influență.
 - **Grafice:** Grafice vizuale care ilustrează diferențele valorilor parametrilor pe diverse stadii ale bolii, ajutând specialiștii medicali să identifice tendințe și intervale relevante.
 - **Analize statistice:** Vizualizări statistice ale setului de date, cum ar fi distribuția bolii pe vârste și sex, oferind o perspectivă detaliată asupra structurii datelor.

Această metodologie asigură că platforma CauselT este capabilă să proceseze și să analizeze datele medicale în mod eficient, oferind profesioniștilor din domeniul sănătății un instrument avansat pentru diagnosticarea și managementul bolilor.

2.3 Pașii de cercetare

Cercetarea și dezvoltarea soluției CauselT s-a desfășurat în cadrul companiei Siemens, pe o perioadă de 5 luni. S-a utilizat algoritmul Random Forest pentru a identifica simptomele și importanța lor în cancerul pancreatic, în colaborare cu medici de la diferite spitale din Taipei. În urma confirmării succesului de către medici, s-a efectuat un stagiul de cercetare pentru a generaliza descoperirile făcute astfel încât algoritmul să poată fi aplicat pentru mai multe tipuri de boli.

3 Arhitectura Aplicației

Arhitectura aplicației CauselT este proiectată pe trei straturi fundamentale: frontend, un server intermediar și un server backend dedicat procesării avansate prin modele de învățare automată și inteligență artificială (Figura 2). Această structură stratificată este intenționat concepută pentru a sprijini scalabilitatea și modularitatea, fiecare componentă principală funcționând autonom și permițând astfel expansiunea rapidă a sistemului.

Complexitatea arhitecturală răspunde cerințelor stringente ale domeniului medical, unde acuratețea și fiabilitatea¹⁵ sunt esențiale. Precizia este crucială, deoarece orice inexactitate, chiar și minoră, poate avea consecințe grave asupra sănătății pacienților. Din acest motiv, structura sistemului a fost proiectată pentru a maximiza precizia și a minimiza riscul de erori, contribuind astfel la optimizarea continuă a procedurilor de diagnosticare și tratament.

Serverul intermediar joacă un rol esențial în această configurație, asigurând nu doar securitatea datelor clinice sensibile¹⁶, dar și gestionarea eficientă a fluxurilor de date. În plus față de aceste funcții, serverul intermediar este responsabil pentru stocarea datelor în baze de date și în cloud¹⁷, garantând că informațiile sunt corect organizate și accesibile pentru serverul backend. Organizarea și gestionarea adecvată a datelor sunt cruciale pentru procesarea eficientă și extragerea de informații relevante prin tehnici de inteligență artificială.

Astfel, arhitectura aplicației CauselT nu doar asigură un nivel înalt de securitate și eficiență în prelucrarea datelor, ci și o adaptabilitate și scalabilitate¹⁸ crescută, indispensabile într-un domeniu în continuă evoluție și cu cerințe variabile.

¹⁵Fiabilitate: Capacitatea unui sistem de a funcționa corect pe o perioadă de timp definită, fără defecțiuni.

¹⁶Date clinice sensibile: Informații medicale personale care necesită protecție specială din cauza caracterului lor confidențial.

¹⁷Cloud: Tehnologie care permite stocarea și accesarea datelor și aplicațiilor prin internet, în loc de pe un hard disk local.

¹⁸Scalabilitate: Capacitatea unui sistem de a gestiona o creștere a încărcării prin adăugarea de resurse suplimentare.

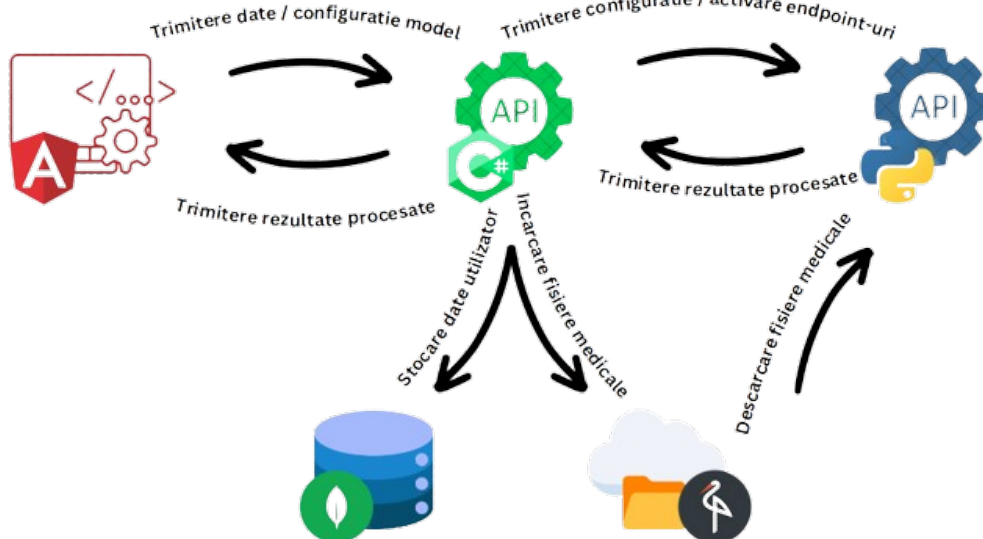


Figura 2: Arhitectura aplicației CauselT

3.1 Frontend

Interfața utilizator a platformei CauselT constituie punctul primar de interacțiune pentru utilizatorii finali, fiind în mod special adaptată pentru a răspunde nevoilor personalului medical. Această componentă este concepută pentru a se adapta la diverse niveluri de competențe tehnice ale utilizatorilor, asigurând astfel accesul simplificat la funcționalitățile necesare desfășurării activităților medicale. Designul Frontend-ului este optimizat pentru a combina intuitivitatea cu eficiența, integrând accesibilitatea cu un design profesionist. Prin aceasta, se garantează că medicii pot naviga ușor în cadrul platformei, accesând eficient funcțiile esențiale, ceea ce este vital pentru susținerea performanței într-un mediu medical solicitant. Conform studiului "User Interface Design for Healthcare Applications", implementarea unui design intuitiv și securizat este esențială pentru îmbunătățirea eficacității și securității aplicațiilor destinate domeniului medical [1].

3.1.1 Motivația alegerii platformei web

Interfața utilizator a platformei CauselT este implementată sub forma unei platforme web, utilizând Angular pentru dezvoltare. Alegerea acestei tehnologii este fundamentată pe avantajele cheie care răspund nevoilor specifice din domeniul medical:

- **Accesibilitate universală:** Platformele web oferă capacitatea de a accesa aplicația de pe orice dispozitiv cu browser și conexiune la internet. Acest lucru este esențial pentru personalul medical care necesită acces rapid și facil la datele pacienților, indiferent de locația lor – fie în spital, fie în deplasare.
- **Consistență și actualizări centralizate:** Actualizările sistemului se implementează direct pe server, devenind instantaneu disponibile pentru toți utilizatorii, fără necesitatea de a instala manual actualizări pe fiecare dispozitiv. Aceasta asigură că toți utilizatorii lucrează cu cea mai recentă versiune a aplicației, esențial pentru conformitatea cu reglementările de securitate și funcționalitate din sănătate.
- **Scalabilitate:** Aplicațiile web se pot scala eficient pentru a deservi un număr mare de utilizatori simultan, ceea ce este crucial în spitale și clinici mari unde accesul la informații trebuie să fie constant și rapid.
- **Costuri reduse de întreținere și dezvoltare:** Folosind o platformă web, costurile asociate cu dezvoltarea și întreținerea software-ului sunt reduse, deoarece codul poate fi utilizat pe multiple platforme, eliminând necesitatea suportului specific pentru fiecare sistem de operare.

Aceste beneficii subliniază adecvarea unei platforme web pentru CauselT, oferind o soluție eficientă și flexibilă pentru gestionarea necesităților complexe ale profesioniștilor medicali.

3.1.2 Arhitectura Angular

Arhitectura Angular a aplicației CauselT este concepută pentru a oferi o soluție scalabilă, modulară și eficientă, crucială pentru gestionarea interacțiunilor complexe în domeniul medical. Aplicația utilizează o structură bazată pe componente, fiecare destinată unei funcționalități specifice. O componentă în Angular este o clasă TypeScript care servește ca bloc constructiv al aplicației, încapsulând logica, datele și prezentarea asociate unei anumite părți a interfeței grafice. Aceasta poate include șabloane HTML pentru structura vizuală, stiluri SCSS pentru aspectul estetic și cod TypeScript pentru gestionarea comportamentului.

Componentele permit o extindere și întreținere eficientă, oferind claritate și organizare în arhitectura software și adaptându-se flexibil la cerințele în schimbare ale domeniului medical. Datorită izolării funcționalităților, modificările sau îmbunătățirile pot fi efectuate pe componente individuale fără a afecta restul aplicației.

În plus față de componente, aplicația integrează servicii esențiale pentru comunicarea cu serverele. Serviciile în Angular sunt clase singleton¹⁹ care oferă funcționalități precum recuperarea datelor, logica de afaceri și interacțiunea cu bazele de date, care sunt accesate de multiple componente. Aceasta separare între prezentare și servicii contribuie la un cod mai curat și mai ușor de întreținut.

Securitatea este consolidată prin protecții la nivel de rute, care sunt mecanisme de control ce asigură că utilizatorii pot accesa anumite părți ale aplicației numai dacă îndeplinesc condițiile necesare, cum ar fi autentificarea sau autorizarea.²⁰ Acest lucru este gestionat prin guard-uri de rută, care pot permite sau refuza dinamic accesul la rutele respective.

Interfețele și interceptorii sunt alte instrumente vitale în Angular, utilizate pentru a uniformiza și valida comunicațiile de rețea. Interceptorii sunt clase speciale care pot modifica cererile și răspunsurile HTTP înainte ca acestea să ajungă la server sau înainte de a fi procesate de aplicație.²¹ Aceasta permite manipularea datelor, autentificarea cererilor, gestionarea erorilor și alte activități transversale importante.

Toate aceste elemente structurale formează o interfață grafică intuitivă și accesibilă, concepută pentru a minimiza complexitatea pentru utilizatorii finali — în acest caz, personalul medical. Designul CauseIT facilitează ca personalul medical să execute operațiuni critice eficient, asigurând că toate detaliile tehnice și procedurile de validare sunt gestionate automat. Prin urmare, se creează un mediu de lucru optimizat și securizat, esențial în contextul sensibil al datelor medicale.

3.1.3 Analiza fluxului de utilizare

Navigarea prin platforma web CauseIT începe întotdeauna cu accesarea aplicației dintr-un browser de internet. Indiferent de dispozitivul utilizat — fie că este un desktop, laptop, tabletă sau smartphone — utilizatorii pot deschide aplicația printr-o conexiune la internet și un browser compatibil. Aceasta oferă flexibilitate și accesibilitate universală, esențială pentru personalul medical care trebuie să acceseze rapid și eficient datele pacienților din diverse locații.

¹⁹Un singleton este un obiect de care se permite crearea unei singure instanțe, furnizând un punct de acces global la aceasta.

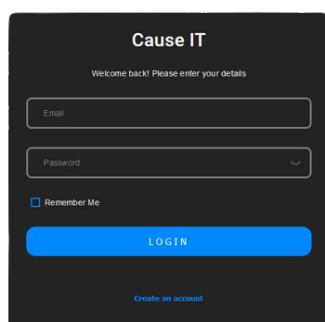
²⁰Autorizarea este procesul prin care un sistem determină dacă utilizatorul are permisiunea de a accesa o resursă sau a efectua o operațiune.

²¹HTTP (Hypertext Transfer Protocol) este protocolul folosit pentru transmiterea datelor pe internet.

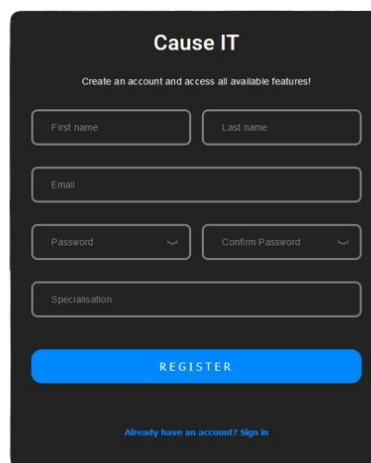
3.1.3.1 Autentificarea

La accesarea platformei, utilizatorul este întâmpinat de pagina de autentificare. Dacă acesta dispune deja de un cont, va trebui să introducă datele de autentificare, adică adresa de email și parola, după cum se arată în Figura 3(a). Activarea opțiunii „Remember Me” permite omiterea reintroducerii datelor de acces în viitoarele utilizări ale platformei, acest mecanism fiind totuși temporar pentru a crește securitatea sistemului.

Pentru utilizatorii noi, există un buton pentru crearea unui cont nou, care deschide fereastra de înregistrare, ilustrată în Figura 3(b). În această fereastră, sunt solicitate detalii cum ar fi nume, prenume, email, parolă și specializarea medicală a utilizatorului, care trebuie selectată dintr-o listă predefinită de opțiuni. Procesele de autentificare și de înregistrare se finalizează prin apăsarea butonului de „Login”, respectiv „Register”.



(a) Pagina de autentificare



(b) Pagina de înregistrare

Figura 3: Interfața de autentificare și înregistrare a platformei CauseIT

La finalizarea procesului de autentificare sau înregistrare, aplicația efectuează verificări riguroase ale datelor furnizate. Este esențial ca toate câmpurile obligatorii să fie completate, formatul textului să fie adecvat și parola să îndeplinească criteriile de complexitate necesare. Dacă aceste condiții sunt îndeplinite, datele sunt trimise prin serviciul de autentificare către serverul intermediar pentru analize suplimentare. În cazul înregistrării, de exemplu, se verifică dacă adresa de email nu a fost anterior utilizată.

Procesul de comunicare între pagina de autentificare și server este detaliat în Figura 4, care ilustrează pașii succesivi de validare și autorizare.

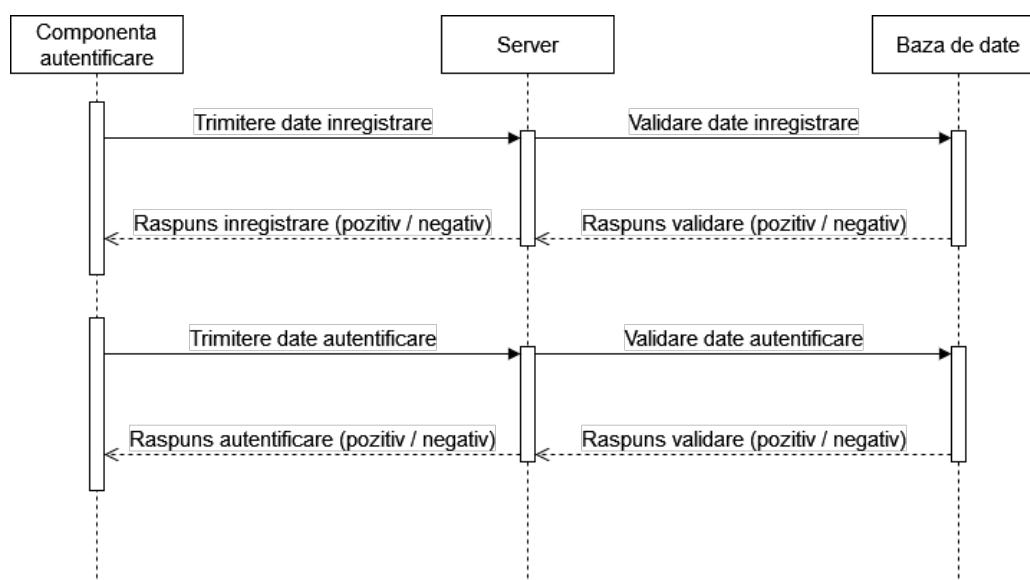


Figura 4: Diagrama de comunicare între frontend și server pentru autentificare

Dacă serverul confirmă validitatea datelor, în contextul înregistrării, informațiile utilizatorului sunt salvate într-o bază de date, iar un răspuns HTTP de confirmare este emis, redirectionând utilizatorul către pagina de autentificare; în cazul autentificării, se emite un răspuns HTTP de confirmare însoțit de două tokenuri: unul de acces și unul de refresh.

Tokenul de acces joacă un rol crucial în facilitarea și securizarea accesului la diversele funcționalități ale platformei. Este destinat unei utilizări pe termen scurt și este stocat în spațiul local al browserului. Rolul său este de a valida accesul rapid și sigur la serviciile aplicației fără verificări repetate ale credențialelor utilizatorului.

Pe de altă parte, tokenul de refresh are o durată de viață mai extinsă și este destinat să rămână invizibil și inaccesibil utilizatorului, fiind stocat în cookie-urile browserului. Funcția sa principală este de a asigura că tokenul de acces rămâne valid. Dacă tokenul de acces expiră, serverul folosește tokenul de refresh pentru a genera un nou token de acces, permițând utilizatorului să continue navigarea fără întreruperi. Cu toate acestea, dacă tokenul de refresh expiră sau este invalidat, accesul utilizatorului la serviciile platformei este suspendat temporar, necesitând reintroducerea credențialelor pentru a asigura continuarea accesului securizat.

Aceste mecanisme de securitate sunt vitale pentru protejarea datelor sen-

sibile și pentru asigurarea unei experiențe de utilizare fără riscuri pe platforma digitală.

3.1.3.2 Pagina principală

După autentificarea reușită, utilizatorul este redirecționat către pagina principală a aplicației, accesul la aceasta fiind condiționat de verificarea stării de autentificare prin intermediul unui guard. Această pagină servește drept interfață centrală de navigare, unde utilizatorul are la dispoziție mai multe opțiuni pentru gestionarea activităților, așa cum este ilustrat în Figura 5. Navigarea poate fi realizată folosind bara laterală sau cardurile interactive care facilitează accesul la funcționalități cheie.

- **Upload Data:** Această secțiune permite utilizatorilor să încarce date medicale, conținând analizele pacienților ce suferă de o anumită boală
- **Configure your AI:** Utilizatorii pot ajusta parametrii algoritmului de inteligență artificială pentru a optimiza acuratețea predicțiilor.
- **Results:** Această pagină oferă acces la rezultatele analizelor AI, incluzând predicții și evaluări statistice.

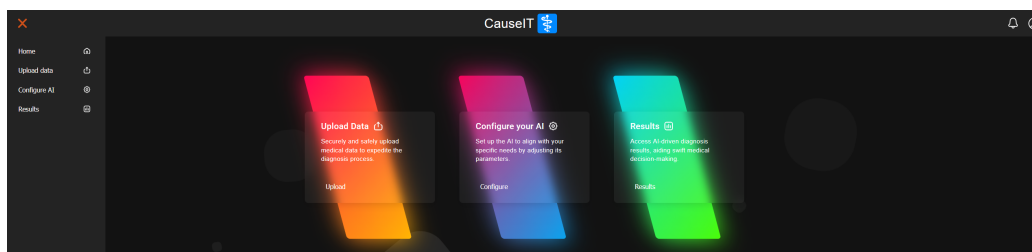
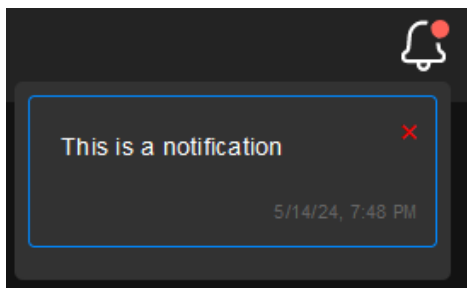


Figura 5: Pagina principală a platformei CauselT

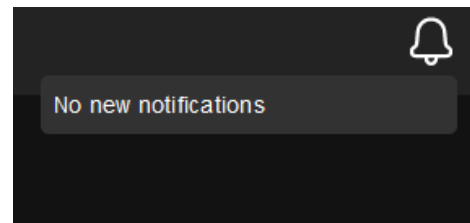
În plus, pe pagina principală sunt prezente două iconițe funcționale:

- **Notifications** (iconița de clopoțel): Aceasta arată notificările care sunt trimise utilizatorului în timp real pentru a informa despre progresul încărcării datelor și al procesului de antrenare. Dacă există notificări necitite, iconița se modifică dinamic pentru a alerta utilizatorul. Notificările active sunt prezentate în Figura 6(a), iar absența notificărilor este ilustrată în Figura 6(b).
- **Account** (iconița de profil): Plasând cursorul peste această iconiță, se deschide un meniu cu trei opțiuni, prezentate în Figura 7:

1. **Profile:** Permite utilizatorului să vizualizeze informațiile specifice profilului său
2. **Settings:** Oferă posibilitatea de a modifica etichetele definite de utilizator, precum și datele încărcate de acesta (ștergerea anumitor fișiere sau eliminarea completă a unei categorii de date)



(a) Exemplu de notificare activă



(b) Ecran fără notificări noi

Figura 6: Schimbarea dinamică a iconiței de notificări

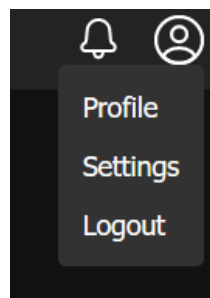


Figura 7: Meniu profil utilizator

Această pagină funcționează ca un dashboard, furnizând un hub central prin care se poate accesa integralitatea funcționalităților disponibile în platformă, facilitând astfel navigarea eficientă.

3.1.3.3 Încărcarea datelor

Interfața de încărcare a datelor poate fi accesată direct din pagina principală sau utilizând meniul de navigație lateral, fiind la rândul ei protejată de un guard. Această pagină conține un formular care trebuie completat de utilizator, așa cum este ilustrat în Figura 8.

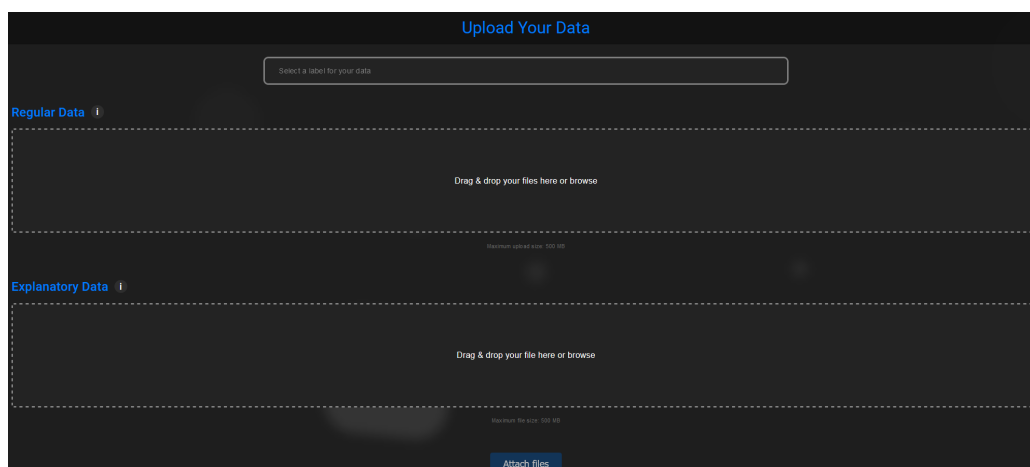


Figura 8: Interfața de încărcare a datelor

Prima componentă a formularului de încărcare solicită utilizatorului să specifice o etichetă pentru date. Utilizatorii pot defini o nouă etichetă introducând denumirea acesteia sau pot selecta din lista de etichete anterior create. Importanța etichetelor constă în organizarea datelor încărcate în categorii distincte, facilitând astfel diferențierea seturilor de date în scopuri analitice și prevenind eventualele confuzii sau erori în timpul proceselor de antrenament al modelului. Utilizarea unei etichete existente adaugă datele noi la cele existente, reantrenând modelul pentru a include și aceste date recent adăugate.

După specificarea etichetei, utilizatorul trebuie să încarce două tipuri de seturi de date:

- **Date Regulate:** Această secțiune permite încărcarea datelor biologice ale pacienților. Utilizatorul poate încărca mai multe fișiere conținând diferiți biomarkeri, respectând limita de spațiu disponibil.
- **Date Explicative:** În această secțiune, utilizatorul are opțiunea de a încărca un fișier ce detaliază numele coloanelor din setul de date încărcat anterior. Deși această secțiune este opțională, încărcarea unui astfel de fișier este recomandată dacă numele coloanelor sunt codificate, facilitând astfel procesul de interpretare a analizelor. Fișierul explicativ permite algoritmului să realizeze corelații directe între codurile și numele coloanelor.

Pentru clarificări suplimentare privind formatul datelor necesar, fiecare secțiune include un simbol informativ. Plasând cursorul mouse-ului peste acest simbol, se deschide un panou explicativ care detaliază cerințele și formatul adecvat pentru încărcarea datelor. Această funcționalitate asigură că utili-

zatorul încarcă datele în mod corespunzător, facilitând astfel procesul analitic efectuat de algoritmi.

După încărcarea datelor necesare, utilizatorul apasă pe butonul **Attach Files** pentru a iniția transmiterea fișierelor către server și stocarea acestora. Procesul este structurat în mai multe etape, ilustrate clar în Diagrama 9.

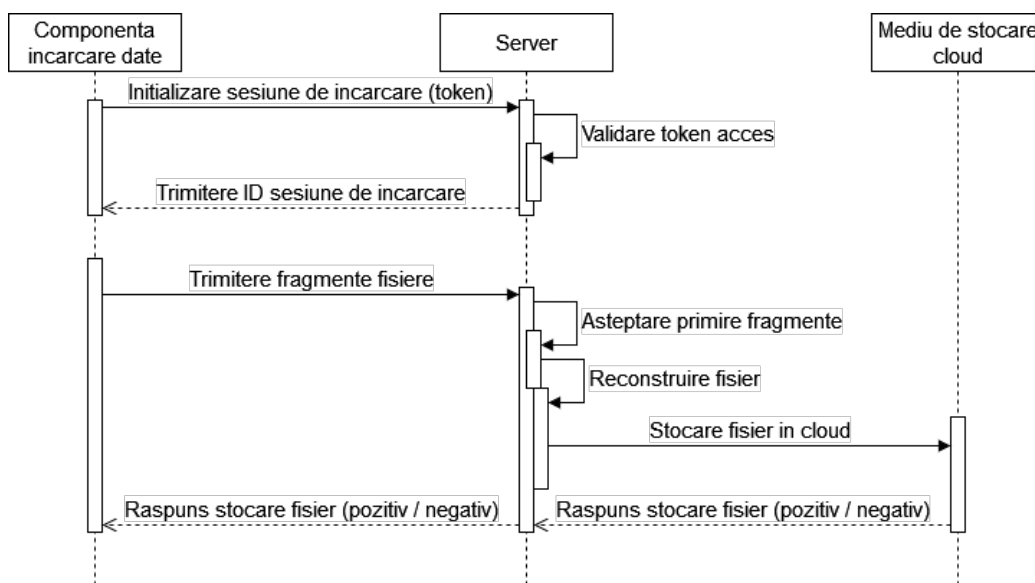


Figura 9: Diagrama de comunicare între frontend și server pentru încărcarea datelor

Inițial, sistemul generează o cerere de inițializare a unei sesiuni de încărcare, trimițând tokenuri de acces către server (Pasul 1). Dacă sesiunea este aprobată, serverul validează tokenurile (Pasul 2) și răspunde cu un ID de sesiune (Pasul 3). Utilizatorul începe apoi transmiterea datelor, segmentate în pachete mici, cunoscute sub denumirea de 'chunk-uri' (Pasul 4).

Serverul așteaptă recepționarea tuturor chunk-urilor (Pasul 5), urmând ca acestea să fie reasamblate pentru a reconstrui fișierul original (Pasul 6). Fișierul este apoi stocat într-un mediu de stocare cloud (Pasul 7). Confirmarea stocării fișierului este transmisă înapoi la interfața utilizatorului, indicând dacă procesul a fost realizat cu succes sau dacă au apărut erori (Pasul 8). Utilizatorul primește un răspuns final privind statusul încărcării, fie pozitiv, fie negativ (Pasul 9).

Este important de menționat că procesul de încărcare a datelor este optimizat să ruleze în paralel cu alte activități ale utilizatorului în browser, astfel încât funcționalitatea acestuia să nu fie întreruptă. Utilizatorul poate continua să navigheze sau să folosească alte funcționalități ale aplicației în timp ce datele sunt încărcate. La finalizarea încărcării, utilizatorul va fi notificat printr-o alertă

în browser, informându-l că procesul de transmitere a datelor a fost completat cu succes sau nu.

Această structură detaliată a procesului de încărcare a datelor asigură eficiența și securitatea în gestionarea informațiilor, prevenind pierderile de date și erorile de transmisie, și îmbunătățește experiența utilizatorului prin gestionarea eficientă a resurselor și comunicarea interactivă.

3.1.3.4 Configurarea modelului

Interfața de configurare a modelului, protejată de un guard, oferă utilizatorilor posibilitatea de a stabili parametrii de predicție și de a modifica anumite caracteristici ale modelului. Platforma include și opțiuni de curățare automată a datelor prin diferite procese de analiză și procesare, pe care utilizatorii le pot activa selectând căsuțele corespunzătoare, reprezentate în Figura 10.

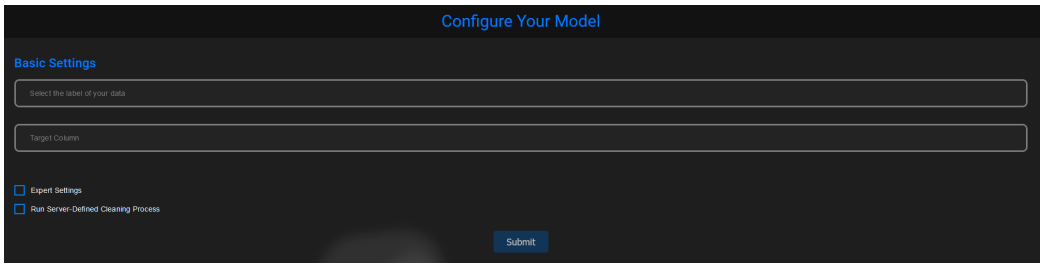
The image shows a web interface titled "Configure Your Model" in a dark-themed box. Under the "Basic Settings" section, there are two input fields: "Select the label of your data" and "Target Column". Below these fields are two checkboxes: "Expert Settings" and "Run Server-Defined Cleaning Process". A "Submit" button is located at the bottom right of the form.

Figura 10: Interfața de configurare a modelului (Setări de bază)

Dacă niciuna dintre opțiuni nu este selectată, utilizatorii trebuie să introducă doar setările de bază. Prima opțiune implică selectarea etichetei datelor; aceste etichete sunt predefinite de utilizator. Alegerea etichetei adecvate este crucială deoarece determină setul de date pe care se va antrena modelul. Ulterior, utilizatorii trebuie să specifice coloana țintă, care indică de obicei parametrul de interes pentru practician.

Activarea opțiunii **Setări Avansate** dezvăluie un formular suplimentar, ilustrat în Figura 11. Acest formular permite utilizatorilor să ajusteze configurația modelului. De exemplu, într-un model de tip Random Forest, utilizatorii pot modifica adâncimea maximă, starea aleatoare sau dimensiunea blocului de date — acesta din urmă referindu-se la numărul de rânduri pe care algoritmul le poate procesa, facilitând gestionarea seturilor de date mai mari. Aceste setări, fiind avansate, necesită o înțelegere profundă a algoritmilor de învățare automată și a tehnologiei informației, deoarece configurațiile necorespunzătoare pot influența semnificativ performanța modelului. Notabil, unele valori prestabilite sunt prezente în acest formular, reprezentând configurația de bază a modelului.

Expert Settings

Algorithm type: Random Forest

Max Depth: 5

Random State: 42

Excluded Columns from Training (comma separated)

Chunk Size: 10000 rows

Figura 11: Interfața de configurare a modelului (Setări Avansate)

Similar, activarea opțiunii **Execută curățarea definită de server** deschide un alt formular, prezentat în Figura 12. Acest formular afișează de asemenea valorile implicite care vor fi utilizate de procesul de curățare dacă utilizatorul nu le modifică, dar optează pentru curățare. Setările pentru codificare și scalare sunt selectabile, necesitând completarea suplimentară a identificatorului pacientului (de obicei un ID) și a unei liste de coloane de exclus din procesul de curățare. Este important de notat că procesul de curățare va altera structura și formatul datelor, astfel recomandându-se precizarea exactă a coloanelor care nu trebuie modificate. Ultimele două slidere indică procentajul maxim permis de valori lipsă pentru coloane și rânduri. Dacă o coloană sau un rând conține valori lipsă care depășesc acest procentaj setat, acea coloană sau acel rând va fi eliminat din setul de date pentru a asigura integritatea și calitatea datelor.

Cleaning Options

Select encoding type: Label Encoding

Select scaling type: Standardize

Patient Identifier Column

Excluded Columns from Cleaning (comma separated)

Row Threshold: 50%

Column Threshold: 30%

Figura 12: Interfața de configurare a modelului (Setări Curățare Date)

Aceste setări conferă utilizatorilor puterea de a personaliza algoritmul pentru setul lor de date, asigurând predicții cât mai precise. Mai mult, oferă un element de modularitate, permițând algoritmului să se adapteze la cerințele utilizatorului.

După finalizarea configurării, utilizatorul poate iniția procesul de antrenare al algoritmului apăsând pe butonul **Submit**. La activarea acestuia, aplicația declanșează automat secvențial endpoint-urile serverului pentru a procesa și analiza datele în mod eficient. Acest demers presupune o încărcătură computațională semnificativă, motiv pentru care utilizatorul este informat că

va primi o notificare odată ce procesarea este completă. Între timp, este posibilă explorarea altor funcționalități ale platformei, cu condiția menținerii sesiunii active. Odată ce notificarea este emisă, utilizatorul va avea acces la rezultatele antrenării.

3.1.3.5 Vizualizarea rezultatelor

Pagina de vizualizare a rezultatelor (Figura 13) este securizată printr-un mecanism de control al accesului, asigurând astfel că doar utilizatorii autorizați pot consulta informațiile afișate. Utilizatorii trebuie să selecteze o etichetă pentru a vizualiza rezultatele asociate acesteia.

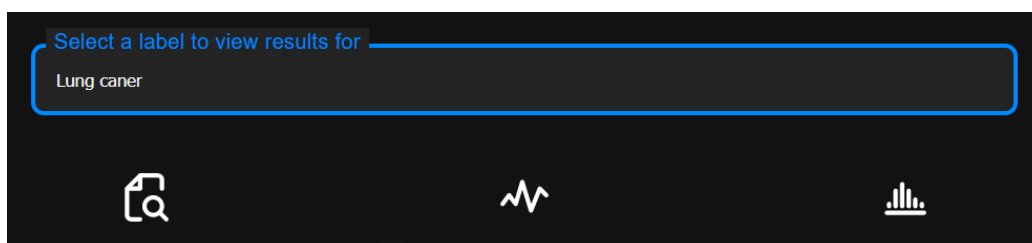


Figura 13: Pagina de vizualizare a rezultatelor

După selectarea etichetei, utilizatorii au acces la trei butoane care afișează conținut specific:

- Opțiunea 1: Afișează lista parametrilor extrași împreună cu procentajul de influență asupra bolii (importanța) și acuratețea obținută de model (vezi Figura 14).

Feature	Importance
ALCOHOL CONSUMING	13.689%
ALLERGY	12.591%
SWALLOWING DIFFICULTY	10.443%
COUGHING	9.257%
AGE	8.053%
PEER_PRESSURE	7.331%
FATIGUE	6.73%
YELLOW_FINGERS	6.298%
WHEEZING	5.051%
ANXIETY	4.973%
SHORTNESS OF BREATH	3.799%
CHEST PAIN	3.052%
SMOKING	2.783%
CHRONIC DISEASE	2.778%
GENDER	2.278%

Model Accuracy: 94.0%

Figura 14: Lista parametrilor extrași și importanța lor

- Opțiunea 2: Afișează grafice care oferă o comparație între pacienții sănătoși și cei bolnavi pentru fiecare parametru extras, condiționată de prezența datelor pentru pacienții sănătoși (vezi Figura 15).

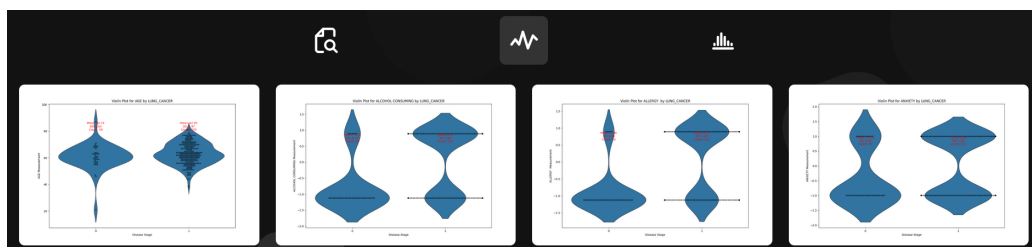


Figura 15: Compararea pacienților sănătoși și bolnavi

- Opțiunea 3: Afișează grafice cu diferite analize statistice ale setului de date încărcat, cum ar fi distribuția bolii pe vârstă și sex, compararea numărului de pacienți sănătoși versus bolnavi, și altele, în funcție de disponibilitatea datelor (vezi Figura 16).



Figura 16: Analize statistice ale setului de date

Pentru obținerea parametrilor, se face o cerere către serverul intermediar, care va returna lista de parametri și importanța lor. Aceste informații sunt apoi afișate în tabelul din Figura 14.

Pentru afișarea graficelor din ambele secțiuni (compararea pacienților și analize statistice), s-a implementat o soluție de virtual scrolling²². Această tehnică a fost necesară deoarece imaginile sunt descărcate dintr-un mediu cloud și primite de la serverul intermediar sub formă de stringuri în format base64.

Base64 este un sistem de codificare care permite transformarea datelor binare în format text, utilizat frecvent pentru transmiterea datelor în medii

²²Virtual scrolling: Tehnică utilizată pentru a îmbunătăți performanța aplicațiilor care afișează o listă mare de elemente prin încărcarea dinamică a acestora pe măsură ce utilizatorul derulează conținutul.

unde textul este preferat, cum ar fi în email-uri și stocarea datelor în formate text. Utilizarea Base64 este justificată de nevoia de a transmite imagini mari în siguranță și eficient. Prin trimiterea imaginilor în acest format, datele pot fi transportate fără riscul de a fi corupte. Un studiu realizat de Chandramouli și Memon (2001) evidențiază importanța și utilitatea codificării base64 în diverse aplicații de transfer de date[12].

Din cauza numărului mare de imagini, este imposibilă trimiterea întregului conținut deodată. Astfel, pentru a optimiza procesul, pe măsură ce utilizatorul navighează prin imagini, alte imagini sunt cerute de la server pentru a completa lista, până când toate au fost încărcate.

În cazul apariției unei erori în timpul procesului, utilizatorul va fi informat și va putea relua procesul.

3.1.3.6 Profilul utilizatorului

Pagina de profil este esențială pentru managementul informațiilor personale ale utilizatorilor aplicației CauselT. Aceasta oferă o vedere de ansamblu asupra datelor esențiale asociate fiecărui utilizator, facilitând astfel identificarea și personalizarea interacțiunilor în cadrul platformei. Pagina este proiectată pentru a fi ușor de navigat, prezentând informațiile relevante într-un format clar și accesibil. Rolul și funcționalitatea paginii de profil

Pe pagina de profil sunt afișate datele esențiale ale utilizatorului, inclusiv un identificador unic, numele, adresa de email și specializarea medicală. Aceasta servește drept punct central pentru gestionarea identității utilizatorului în cadrul aplicației, asigurând că toate interacțiunile și acțiunile sunt corect atribuite. Prin centralizarea acestor informații, pagina de profil facilitează un management eficient și securizat al datelor utilizatorilor.

Figura 17 ilustrează aspectul paginii de profil, incluzând următoarele informații:

- **Identificator unic:** Un șir de caractere alfanumeric unic asociat fiecărui utilizator, asigurând unicitatea și securitatea datelor.
- **Nume:** Numele utilizatorului, folosit pentru personalizarea interacțiunilor.
- **Adresa de email:** Adresa de email a utilizatorului, necesară pentru comunicare și recuperarea contului.
- **Specializarea:** Domeniul medical în care utilizatorul activează, relevant pentru personalizarea experienței și a funcționalităților oferite de aplicație.

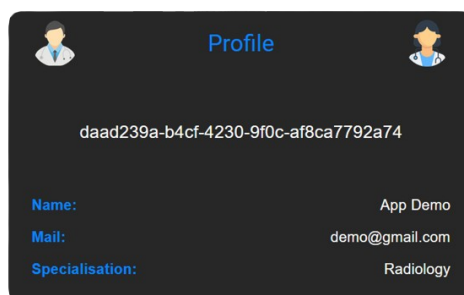


Figura 17: Informațiile afișate în cadrul paginii de profil

Pagina de profil asigură un nivel adecvat de securitate prin controlul accesului la datele personale, garantând confidențialitatea informațiilor utilizatorilor. Datele afișate pe această pagină sunt doar pentru vizualizare, fără a permite modificarea acestora direct prin intermediul interfeței de utilizator. Acest aspect contribuie la protejarea integrității informațiilor și la prevenirea accesului neautorizat.

3.1.3.7 Pagina de Setări

Pagina de setări a aplicației CauselT oferă utilizatorilor un control avansat asupra datelor și notificărilor gestionate în cadrul platformei. Aceasta permite administrarea etichetelor și a fișierelor asociate, precum și vizualizarea și gestionarea notificărilor primite.

În cadrul paginii de setări, utilizatorii pot vizualiza etichetele definite de aceștia și fișierele încărcate în cadrul fiecărei etichete. Această funcționalitate este crucială pentru menținerea unei structuri organizate și pentru asigurarea integrității datelor utilizate de modelul de învățare automată.

Gestionarea etichetelor și a fișierelor

- **Ștergerea unei etichete:** Utilizatorii au posibilitatea de a șterge întreaga etichetă, ceea ce va conduce la eliminarea tuturor fișierelor asociate acelei etichete. Această opțiune este utilă atunci când o etichetă nu mai este necesară sau dacă toate fișierele din cadrul ei sunt considerate irelevante.
- **Ștergerea fișierelor individuale:** În plus, utilizatorii pot șterge fișiere individuale din cadrul unei etichete, oferind astfel un control mai granular asupra datelor încărcate. Această funcționalitate este esențială în cazul

în care un fișier a fost încărcat din greșeală sau dacă datele conținute nu sunt corecte.

Toate aceste operațiuni de ștergere sunt realizate prin trimiterea de cereri către serverul intermediar. Serverul se ocupă de eliminarea fișierelor într-o manieră sigură și eficientă, asigurând că datele sunt gestionate corespunzător și că integritatea sistemului este menținută.

Figura 18 ilustrează pagina de setări, arătând etichetele existente și fișierele asociate, împreună cu opțiunile de ștergere disponibile pentru fiecare etichetă și fișier.

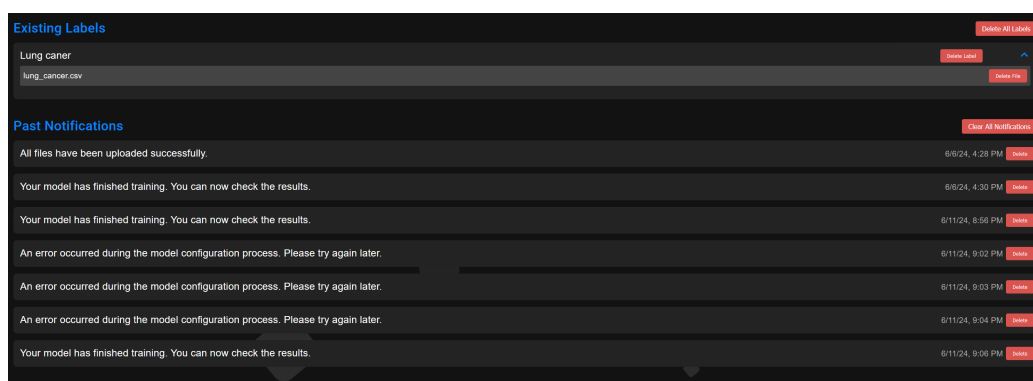


Figura 18: Pagina de setări a aplicației CausalT

Vizualizarea și gestionarea notificărilor

Pagina de setări permite, de asemenea, utilizatorilor să vizualizeze toate notificările primite pe parcursul utilizării platformei. Utilizatorii au următoarele opțiuni pentru gestionarea notificărilor:

- **Ștergerea unei notificări:** Utilizatorii pot șterge notificări individuale, oferind un control detaliat asupra mesajelor care sunt păstrate sau eliminate.
- **Ștergerea tuturor notificărilor:** Pentru o gestionare rapidă și eficientă, utilizatorii au opțiunea de a șterge toate notificările simultan.

Aceste funcționalități asigură utilizatorilor un grad înalt de control și personalizare, permițându-le să mențină un mediu de lucru organizat și să gestioneze eficient datele și notificările.

3.2 BridgeAPI

În contextul evoluției continue a sistemelor informatice complexe, componenta de intermediere a datelor, reprezentată în cadrul platformei Cau-

seIT de BridgeAPI, devine esențială pentru asigurarea integrității, securității și eficienței schimbului de informații între interfețele utilizatorilor și mecanismele de procesare avansată. Acest capitol se concentrează pe descrierea detaliată a arhitecturii și funcționalităților BridgeAPI, explorând rolul său fundamental în facilitarea comunicației securizate și eficiente între componentele aplicative. Prin intermediul acestui server intermediar, CausalT reușește nu numai să optimizeze fluxurile de date, dar și să implementeze protocoale robuste de securitate, esențiale pentru protecția informațiilor sensibile gestionate de platformă.

BridgeAPI servește ca un nod central în arhitectura sistemului, gestionând solicitările de la interfața utilizator și coordonând executarea operațiunilor de procesare a datelor cu ajutorul ModelOPS. Prin urmare, acest capitol va detalia structura internă a BridgeAPI, evidențiind implementarea tehnică și contribuția sa la performanța generală a platformei CausalT. De asemenea, se va discuta despre modul în care BridgeAPI gestionează și securizează stocarea datelor în baza de date și cloud, un aspect crucial în cadrul infrastructurilor informatice moderne.

3.2.1 Rolul și importanța BridgeAPI

BridgeAPI, componenta esențială a arhitecturii sistemului CausalT, funcționează ca un mediator între interfața utilizatorilor (frontend) și serverul de backend care gestionează procesarea datelor (ModelOPS, implementat în Python). Implementarea acestui strat intermediar este importantă pentru optimizarea eficienței, securității și scalabilității platformei.

3.2.1.1 Separarea funcționalităților

Implementarea BridgeAPI facilitează o separare clară a responsabilităților, permițând serverului Python să se concentreze exclusiv pe computații intensive legate de învățare automată, în timp ce BridgeAPI gestionează interacțiunile cu utilizatorii.

- **Optimizarea procesării:** Serverul Python este optimizat pentru sarcini de antrenare și evaluare a modelului de învățare automată, iar BridgeAPI preprocesează și validează cererile utilizatorilor pentru a asigura că sunt în conformitate cu specificațiile necesare.
- **Gestionarea cererilor:** Prin autentificarea și filtrarea cererilor înainte de a le transmite la ModelOPS, BridgeAPI reduce riscul de erori și securizează fluxul de date.

3.2.1.2 Performanță și scalabilitate

Utilizarea C# și .NET în cadrul BridgeAPI oferă avantaje semnificative legate de performanță și scalabilitate, facilitând gestionarea eficientă a cererilor multiple și a comunicării în timp real prin SignalR.

- **SignalR pentru comunicații în timp real:** Tehnologia SignalR permite BridgeAPI să implementeze comunicații bidirecționale în timp real, esențiale pentru funcționalitățile interactive ale platformei.
- **Scalabilitate modulară:** Separarea funcțiilor între BridgeAPI și serverul Python permite o alocare și scalare independentă a resurselor, optimizând capacitatea de răspuns a sistemului.

3.2.1.3 Securitate îmbunătățită

BridgeAPI joacă un rol crucial în securizarea platformei, implementând protocoale stricte de securitate pentru toate cererile care trec prin acest nod.

- **Barieră de securitate:** Funcționând ca un filtru de securitate, validează și autentifică toate cererile înainte de a le transmite la backend.
- **Conformitate:** Asigură conformitatea cu standardele de protecție a datelor, aplicând protocoale de securitate riguroase pentru a proteja informațiile sensibile ale utilizatorilor.

3.2.1.4 Îmbunătățirea ciclurilor de actualizare

Prin decuplarea frontend-ului de backend, BridgeAPI permite actualizări și îmbunătățiri ale interfeței utilizatorului sau ale logicii de procesare a datelor fără a perturba alte componente ale sistemului.

- **Flexibilitate în actualizări:** Permite implementarea de actualizări și îmbunătățiri specifice fără a afecta funcționarea generală a platformei.

Prin aceste caracteristici, BridgeAPI nu doar îmbunătățește performanța și securitatea platformei CauselT, dar oferă și o fundație solidă pentru adaptarea continuă la cerințele evolutive ale domeniului medical. Această arhitectură sustenabilă promovează inovația și asigură timpi de răspuns reduși, vitali pentru sectorul medical.

3.2.2 Arhitectura BridgeAPI

BridgeAPI este construit folosind platforma .NET Framework și adoptă o arhitectură de tip REST API, optimizând astfel interacțiunea între diferite sisteme software printr-o serie de interfețe standardizate. REST, sau „Representational State Transfer”, este un set de constrângeri arhitecturale pentru serviciile web, care utilizează protocoalele HTTP existente pentru a permite comunicația. Aceasta oferă o metodă intuitivă și scalabilă pentru accesul și manipularea datelor reprezentate ca resurse web, fiecare identificată printr-un URL unic.

Principalele metode HTTP utilizate în REST sunt:

- **GET:** Recuperarea informațiilor despre o resursă specificată.
- **POST:** Crearea unei noi resurse.
- **PUT:** Actualizarea completă a unei resurse existente.
- **PATCH:** Actualizarea parțială a unei resurse existente.
- **DELETE:** Ștergerea unei resurse.

Folosirea REST în cadrul BridgeAPI garantează o integrare fluidă și o interacțiune eficientă între diverse aplicații, facilitând o structură clară și modulară a serviciilor web. Controllerele în C# din .NET Framework gestionează aceste solicitări, asigurând că logica de afaceri este aplicată corect și că răspunsurile sunt generate în conformitate cu cerințele clientului. Aceasta structurare ajută la menținerea unei coerențe și eficiențe în procesarea solicitărilor, oferind o fundație solidă pentru dezvoltarea de aplicații web moderne.

3.2.2.1 Controllerul de specializări

Controllerul de **specializări** gestionează setul de specializări medicale disponibile în cadrul aplicației. Aceste specializări sunt predefinite și nu pot fi modificate de utilizatori, ci doar de administratorii care au acces la server. Specializările reprezintă diferite domenii ale medicinei și sunt esențiale în procesul de înregistrare, fiecare medic alegând o specializare corespunzătoare calificărilor sale.

- **Adăugarea specializărilor:** Administratorii pot introduce noi specializări în sistem pentru a reflecta evoluțiile din domeniul medical. Procesul include asignarea unui identificador unic fiecărei specializări pentru a asigura gestionarea eficientă.

- **Accesarea specializărilor:** Specializările pot fi accesate prin diverse metode, fie prin ID-ul unic, fie prin numele specializării, oferind flexibilitate în utilizarea acestor informații în alte procese ale aplicației.
- **Actualizarea specializărilor:** Modificările în oferta de specializări sunt posibile doar prin intervenția administratorului, care poate actualiza detaliile specializărilor pentru a se alinia la noile standarde sau cerințe ale sectorului medical.
- **Ștergerea specializărilor:** Similar, eliminarea specializărilor din sistem este controlată strict de administratori, asigurându-se că eliminările nu afectează integritatea datelor utilizatorilor existenți sau alte procese dependente de acestea.

Controlul strict al acestor specializări asigură că platforma rămâne actualizată și relevantă pentru nevoile profesionale ale utilizatorilor săi, oferind o bază de date precisă și sigură pentru înregistrarea și gestionarea medicilor în funcție de specializarea lor.

3.2.2.2 Controllerul de autentificare

Controllerul de autentificare din cadrul serverului BridgeAPI gestionează cererile legate de autentificarea utilizatorilor, oferind două funcții principale: înregistrarea și autentificarea.

În cadrul procesului de **înregistrare**, controllerul de autentificare gestionează cererile venite de la interfața utilizator prin intermediul unui endpoint dedicat. Acest proces implică următoarele etape principale:

- **Validarea Datelor:** Se efectuează o verificare a integrității și formatului informațiilor transmise, cum ar fi adresa de email și parola. Acest pas include și confirmarea că adresa de email nu este folosită anterior în sistem.
- **Hashing-ul Parolei:** Odată validată informația, parola este supusă unui proces de hashing pentru criptare înainte de a fi stocată în baza de date. Procesul folosește funcția `HashPassword`, care implementează algoritmul PBKDF2 cu HMAC-SHA256. Acest proces este descris detaliat în articolul "Optimization of PBKDF2 using HMAC-SHA2 and HMAC-LSH Families in CPU Environment" de Choi Hojin și Seo Seog [13]. Procesul include generarea unui salt aleatoriu, care, combinat cu parola, asigură unicitatea rezultatului hash pentru parole identice.

- **Stocarea în Baza de Date:** Ulterior procesului de hashing, datele utilizatorului, inclusiv parola criptată, sunt înregistrate în baza de date.

Funcția HashPassword

```
private string HashPassword(string password)
{
    byte[] salt = new byte[128 / 8];
    using (var rng = RandomNumberGenerator.Create())
    {
        rng.GetBytes(salt);
    }

    string hashed = Convert.ToBase64String(KeyDerivation.Pbkdf2(
        password: password,
        salt: salt,
        prf: KeyDerivationPrf.HMACSHA256,
        iterationCount: 10000,
        numBytesRequested: 256 / 8));

    return Convert.ToBase64String(salt) + "||" + hashed;
}
```

Procesul de **autentificare** este crucial în verificarea și validarea credențialelor utilizatorilor, implicând mai multe etape esențiale pentru securizarea accesului în sistem:

- **Verificarea Credențialelor:** Inițial, se compară adresa de email și parola introduse de utilizator cu cele stocate în baza de date. Procesul de verificare a parolei se realizează folosind funcția `VerifyPassword`, care aplică algoritmul PBKDF2 cu HMAC-SHA256 pentru a regenera hash-ul parolei utilizând salt-ul stocat. În acest proces, salt-ul este extras din hash-ul stocat și utilizat pentru a recalcula hash-ul parolei introduse. Compararea acestui nou hash cu hash-ul stocat confirmă autenticitatea parolei.
- **Generarea Token-ului de Acces:** După confirmarea credențialelor, sistemul generează un token JWT (JSON Web Token), care permite utilizatorului accesul la resursele protejate ale aplicației.
- **Emiterea Token-ului de Refresh:** Complementar cu token-ul de acces, se generează și un refresh token, care extinde durata sesiunii de autentificare a utilizatorului, eliminând necesitatea reautentificării frecvente.

Funcția VerifyPassword

```
private bool VerifyPassword(string inputPassword, string
    storedHashWithSalt)
{
    var parts = storedHashWithSalt.Split(new[] { "|" },
        StringSplitOptions.None);
    if (parts.Length != 2)
    {
        throw new InvalidOperationException("Stored hash should
            contain the salt and hash");
    }

    var salt = Convert.FromBase64String(parts[0]);
    var storedHash = parts[1];

    string hashedInput =
        Convert.ToBase64String(KeyDerivation.Pbkdf2(
            password: inputPassword,
            salt: salt,
            prf: KeyDerivationPrf.HMACSHA256,
            iterationCount: 10000,
            numBytesRequested: 256 / 8));

    return storedHash == hashedInput;
}
```

Aceste metode consolidează integritatea și confidențialitatea datelor utilizatorilor, constituind un pilon central al securității sistemului. Implementarea acestor proceduri sporește eficiența procesului de autentificare și oferă un nivel înalt de protecție.

3.2.2.3 Controllerul de token-uri

Controllerul de token-uri joacă un rol fundamental în gestionarea securității sesiunilor utilizatorilor prin emiterea și validarea tokenurilor de autentificare. Aceste tokenuri sunt esențiale pentru menținerea unei stări de autentificare sigure și eficiente în aplicații, conform detaliilor furnizate în [5].

- **Importanța tokenurilor de autentificare:** Tokenurile de autentificare, inclusiv tokenurile de acces și cele de reîmprospătare (refresh tokens), sunt cruciale pentru validarea și menținerea sesiunilor utilizatorilor în aplicații web. Acestea permit utilizatorilor să acceseze resurse protejate fără a reintroduce credențialele, până la expirarea tokenului.

- **Generarea tokenurilor pe server:** Tokenurile sunt generate pe server folosind biblioteci de securitate care asigură criptarea informațiilor. Pentru tokenurile de acces, se utilizează JWT (JSON Web Token), care sunt create și semnate folosind o cheie secretă configurată în server. Tokenurile de refresh sunt generate pentru a permite reînnoirea accesului fără a solicita utilizatorului să se autentifice din nou. Acestea sunt emise cu o durată de viață mai lungă și pot fi înlocuite sau revocate prin cereri specifice la server.
- **Procesul de validare și reînnoire:** La primirea unei cereri cu un token de acces, serverul validează acest token folosind parametri de securitate specifici, verificând semnătura și validitatea acestuia. În cazul în care tokenul de acces a expirat, un token de refresh poate fi folosit pentru a obține un nou token de acces, asigurând astfel continuitatea accesului utilizatorului fără întreruperi.

Această abordare asigură că aplicația menține securitatea datelor și oferă o experiență utilizator fluidă și protejată.

3.2.2.4 Controllerul de fișiere

Controllerul de **fișiere** are rolul de a gestiona încărcările de fișiere realizate de utilizatori, fiecare fișier fiind asociat unei categorii specifice, care reprezintă diferite boli sau condiții medicale. Acest sistem permite organizarea eficientă a datelor și facilitează accesul rapid la informații relevante în funcție de categoria specificată.

- **Inițierea sesiunii de încărcare:** La începutul procesului de încărcare, sistemul inițiază automat o sesiune și calculează numărul total de fișiere pe care utilizatorul intenționează să le încarce, atribuind fiecărei sesiuni un identificator unic. Acest identificator facilitează monitorizarea și gestionarea precisă a procesului de încărcare.
- **Încărcarea fișierelor:** Fișierele sunt încărcate în fragmente (chunk-uri) pentru a facilita transferul eficient și sigur al datelor, chiar și în cazul fișierelor de mari dimensiuni. Sistemul gestionează organizarea acestor fragmente, asigurând reasamblarea corectă a fișierului final în locația destinată stocării pe termen lung. Fragmentarea și reasamblarea permit o gestionare optimizată a resurselor și minimizarea riscului de întrerupere a procesului de încărcare.
- **Gestionarea erorilor și notificările:** În cazul apariției erorilor pe parcursul încărcării, controllerul trimite notificări specifice pentru a oferi feedback

utilizatorului despre starea încărcării. Acest mecanism include notificări de finalizare reușită sau de eroare, contribuind la transparența procesului. Notificările sunt gestionate prin intermediul unui serviciu dedicat, care asigură informarea promptă și clară a utilizatorului.

- **Separarea pe categorii:** Etichetarea fișierelor încărcate cu categorii specifice permite o organizare logică și facilitează accesul rapid la informații. Această caracteristică este crucială în contextul medical, unde eficiența accesului la date poate influența deciziile clinice. Fișierele sunt stocate în directoare specifice etichetelor, asigurând o structură coerentă și ușor de navigat.
- **Listarea etichetelor:** Controllerul oferă posibilitatea utilizatorilor de a obține o listă a etichetelor existente. Aceasta funcție permite utilizatorilor să vizualizeze și să selecteze rapid etichetele relevante pentru încărcarea sau gestionarea fișierelor. Funcționalitatea este esențială pentru o organizare eficientă și pentru identificarea rapidă a categoriilor de interes.
- **Listarea fișierelor:** Utilizatorii pot solicita o listă cu fișierele încărcate, filtrată în funcție de o anumită etichetă. Aceasta funcție facilitează accesul rapid la fișierele necesare și permite o gestionare eficientă a datelor. Prin listarea fișierelor, utilizatorii pot vizualiza și administra conținutul încărcat în cadrul platformei.
- **Ștergerea fișierelor:** Controllerul permite ștergerea fișierelor specificate de utilizatori. Această funcție asigură că datele pot fi gestionate eficient și că informațiile redundante sau învechite pot fi eliminate, contribuind la menținerea unui sistem de stocare curat și organizat. Procedura de ștergere este securizată pentru a preveni eliminarea accidentală a datelor importante.
- **Ștergerea etichetelor:** Utilizatorii au opțiunea de a șterge întregi directoare asociate unei etichete. Aceasta funcție este esențială pentru gestionarea eficientă a categoriilor de date și pentru eliminarea completă a tuturor fișierelor asociate unei etichete care nu mai este relevantă. Ștergerea etichetelor asigură că datele sunt bine organizate și că informațiile depășite nu ocupă spațiu inutil.
- **Curățarea și întreținerea sistemului:** După finalizarea sesiunilor de încărcare, sistemul efectuează operațiuni de curățare automată a directorilor temporari, asigurând gestionarea eficientă a spațiului de stocare și prevenind acumularea de date inutile. Acest proces de curățare este

esențial pentru menținerea performanței și securității platformei, eliminând fișierele temporare și datele redundante.

Prin aceste funcționalități, controllerul de fișiere sprijină gestionarea eficientă a documentelor medicale și a datelor de cercetare, asigurând integritatea și securitatea informațiilor în cadrul platformei.

3.2.2.5 Controllerul de notificări

Controllerul de Notificări este instrumentul principal prin care sistemul comunică automat cu utilizatorii, transmițând notificări generate intern. Aceste notificări sunt esențiale pentru informarea utilizatorilor despre evenimente relevante în timp real, utilizând tehnologia SignalR pentru a asigura livrarea promptă și eficientă a mesajelor.

- **Generarea și transmiterea notificărilor:** Sistemul creează notificări în diverse scenarii, cum ar fi completarea unor procese sau actualizări importante de stare. Odată generate, notificările sunt transmise utilizatorilor prin intermediul NotificationHub, care utilizează SignalR pentru a livra mesaje în timp real. Această funcționalitate asigură că utilizatorii sunt informați imediat ce apar evenimente noi, permițându-le să reacționeze prompt la schimbările din sistem.
- **Recuperarea notificărilor:** Utilizatorii pot solicita vizualizarea notificărilor recente printr-o interfață simplă, care permite accesul la notificările stocate, relevante pentru fiecare utilizator în parte. Această funcționalitate le oferă utilizatorilor o modalitate rapidă de a verifica notificările primite, ajutându-i să se mențină la curent cu evenimentele importante.
- **Actualizarea stării notificărilor:** Funcționalitatea de actualizare permite utilizatorilor să marcheze notificările ca fiind citite, modificând astfel starea acestora în sistem și ajutând la gestionarea eficientă a fluxului de informații. Prin marcarea notificărilor ca citite, utilizatorii pot organiza mai bine notificările primite, concentrându-se pe cele care necesită atenție imediată.
- **Ștergerea notificărilor:** Notificările care nu mai sunt relevante sau care au fost deja gestionate pot fi eliminate, menținând astfel curățenia și relevanța înregistrărilor de notificări. Această funcționalitate permite utilizatorilor să își mențină un inbox de notificări ordonat, eliminând mesajele care nu mai sunt de actualitate.

- **Marcarea notificărilor ca citite:** Această operațiune este crucială pentru menținerea unei interfețe organizate și eficiente, permițând utilizatorilor să se concentreze pe cele mai recente sau importante notificări. Marcarea notificărilor ca citite ajută la distingerea rapidă a mesajelor noi de cele deja vizualizate, îmbunătățind astfel experiența utilizatorului.

Funcția de trimitere a notificărilor

```
public async Task SendNotifications(string user, object
    notification)
{
    await Clients.User(user).SendAsync("ReceiveNotification",
        notification);
}
```

Implementarea acestui controller sprijină o comunicare eficientă între sistem și utilizatori, îmbunătățind experiența acestora prin asigurarea unui răspuns rapid la evenimentele care necesită atenție sau acțiune imediată.

3.2.2.6 Controllerul de comunicare cu ModelOPS

Controllerul de comunicare cu ModelOPS facilitează interacțiunea între interfața utilizatorului și serverul de Python, coordonând fluxurile de lucru analitice pentru prelucrarea datelor. Acest controller nu execută direct operațiunile de antrenare sau pregătire a datelor, ci activează endpointurile corespunzătoare pe serverul de Python, asigurând transmiterea corectă și fără erori a configurațiilor necesare. Comunicarea între BridgeAPI și ModelOPS, esențială pentru aceste operațiuni, este ilustrată în Figura 19.

- **Inițierea sesiunilor:** Sesiunile sunt inițiate prin acest controller, stabilind parametrii și contextul necesar pentru procesele ulterioare. Fiecare sesiune este înregistrată cu un identificator unic, pregătind terenul pentru acțiunile analitice care vor urma. Această inițiere este esențială pentru coordonarea și monitorizarea ulterioară a fluxului de lucru.
- **Descărcarea și pregătirea fișierelor:** Controllerul transmite cereri către serverul de Python pentru a descărca și pregăti fișierele necesare. Aceasta include specificații detaliate pentru curățarea și normalizarea datelor, asigurându-se că formatul și parametrii sunt corect aplicați. Controllerul doar inițiază aceste procese, gestionarea efectivă a datelor fiind realizată de serverul de Python. Acest proces asigură că datele sunt pregătite corespunzător pentru analize ulterioare.

- **Antrenarea modelului:** Similar, controllerul de ModelOPS trimite configurația necesară antrenării modelului către endpointurile de pe serverul de Python. Acesta se asigură că toate setările sunt corecte și completează cererea fără a executa efectiv antrenarea în cadrul controllerului. Această separare permite specializarea serverului de Python în sarcini intensive de calcul, optimizând astfel performanța.
- **Vizualizarea rezultatelor:** Controllerul gestionează descărcarea rezultatelor obținute de către serverul de Python, asigurându-se că acestea sunt prezentate într-o manieră clară și ușor de înțeles. Acest proces include pregătirea datelor pentru afișare și facilitarea accesului la grafice și analize statistice direct în aplicația utilizatorului, promovând o interacțiune eficientă și informativă. Utilizatorii pot vizualiza astfel performanțele modelului antrenat și pot interpreta datele rezultate într-un mod accesibil.

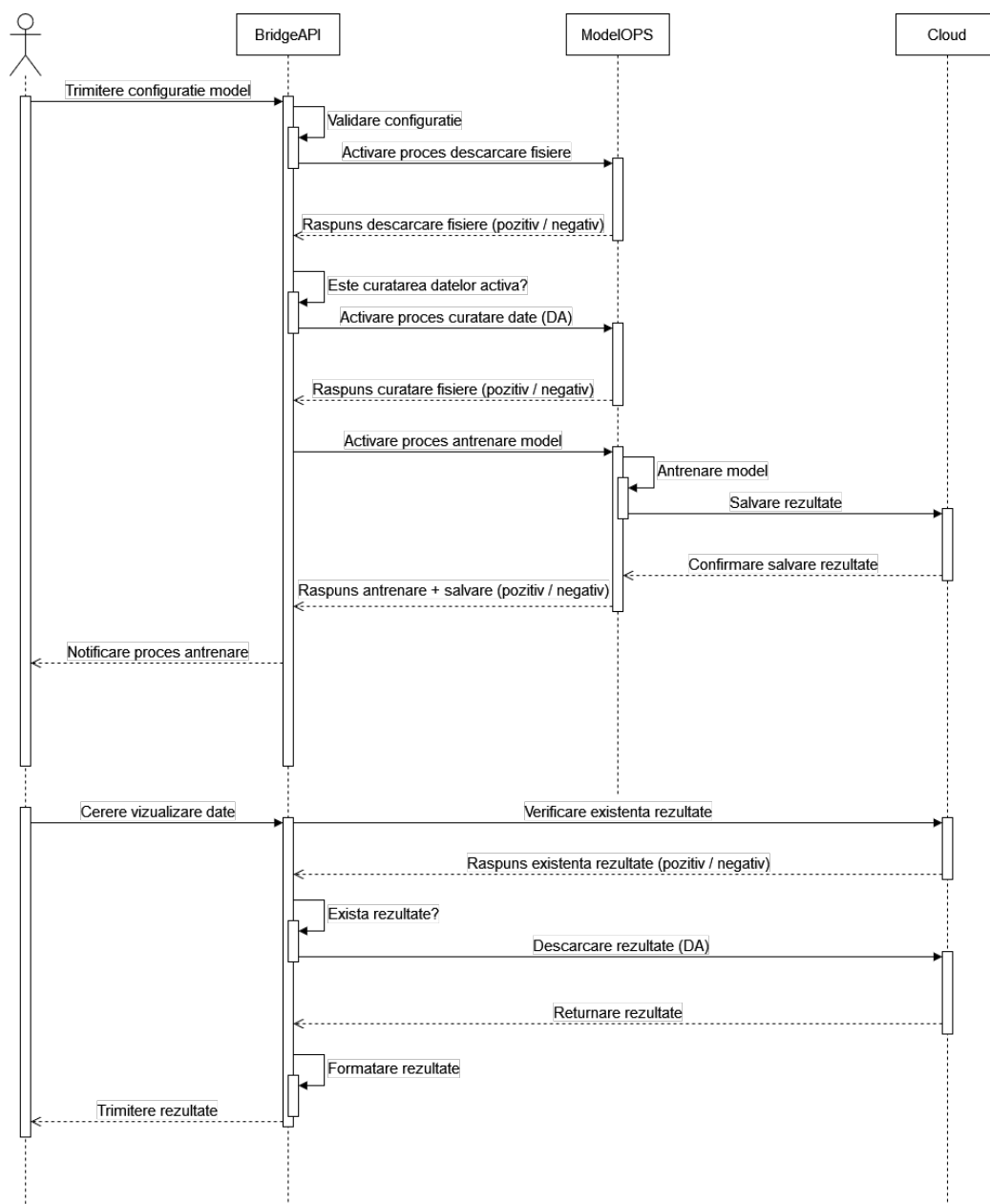


Figura 19: Diagrama fluxului de comunicare între BridgeAPI și ModelOPS

Controllerul de comunicare cu ModelOPS utilizează o serie de metode pentru a implementa funcționalitățile descrise, asigurând o interacțiune eficientă între utilizatori și serverul de Python. Aceste metode includ inițierea sesiunilor, transmiterea cererilor de descărcare și pregătire a fișierelor, configurarea proceselor de antrenare și gestionarea rezultatelor obținute. Prin utilizarea aces-

tor metode, controllerul facilitează un flux de lucru coerent și integrat pentru prelucrarea și analiza datelor medicale complexe.

3.3 ModelOPS

Serverul ModelOPS constituie nucleul analitic al aplicației, gestionând operațiunile de procesare a datelor și antrenarea modelului predictiv. Dezvoltat în Python și implementat ca un server REST API utilizând framework-ul FastAPI, ModelOPS facilitează interacțiunea fluidă și eficientă cu alte componente ale sistemului prin intermediul unor endpoint-uri bine definite.

Această arhitectură permite ModelOPS să execute o serie de procese analitice esențiale, structurate în categorii operaționale, fiecare destinată unui aspect particular al procesării datelor:

- **Descărcarea datelor:** Această categorie se ocupă cu extragerea datelor încărcate de utilizatori, asigurându-se că sunt accesibile pentru etapele ulterioare de procesare.
- **Curățarea și pregătirea datelor:** În această fază, datele sunt curățate de posibile inexactități sau incoerențe și sunt prelucrate pentru a corespunde specificațiilor necesare antrenării modelului. Aceste procese includ normalizarea, codificarea variabilelor categorice și gestionarea valorilor lipsă, pregătind setul de date pentru analize predictive.
- **Antrenarea modelului și salvarea rezultatelor:** Modelul este antrenat folosind seturile de date pregătite, iar rezultatele sunt evaluate și stocate în cloud. Acest proces implică ajustarea parametrilor modelului, validarea acestuia și optimizarea performanței.

Fiecare proces din aceste categorii este conceput pentru a opera secvențial, contribuind la eficiența și modularitatea aplicației. Este esențial ca utilizatorii să adere la un format standardizat al fișierelor pentru a maximiza consistența și acuratețea rezultatelor predictive. Această structură nu doar sporește capacitatea de generalizare a serverului ModelOPS, dar asigură și adaptabilitatea sa în fața diverselor tipuri de date, facilitând astfel o integrare armonioasă într-un mediu de aplicații variat.

3.3.1 Descărcarea datelor

Primul pas în procesul analitic este descărcarea datelor. Acest proces implică extragerea datelor specifice, etichetate de utilizator, dintr-un mediu cloud.

Ulterior, informațiile sunt stocate temporar pe un dispozitiv local. Această metodă garantează acces rapid și neîntrerupt la date, evitându-se astfel solicitările repetate către server, ceea ce ar putea genera întârzieri în analiza datelor și o posibilă suprasolicitare a serverului.

Pentru a extrage fișierele din cloud, se utilizează un client MiniO, configurat în prealabil pe server într-un mod securizat. Procesul începe prin transmiterea către server a identificadorului de utilizator și a etichetei selectate, facilitând astfel localizarea rapidă a datelor necesare. Odată ce datele sunt identificate, acestea sunt descărcate și salvate într-un director temporar. Directorul este eliminat la finalizarea analizei pentru a optimiza utilizarea resurselor de stocare. Prin descărcarea datelor la fiecare sesiune analitică, se asigură utilizarea celor mai recente informații disponibile, prevenindu-se riscul de a lucra cu date învechite sau incomplete.

În cazul unei descărcări reușite, serverul va returna un răspuns afirmativ, în timp ce în situațiile de eșec, va transmite un răspuns negativ, atașând și detalii despre eroarea întâmpinată. Aceste feedback-uri sunt comunicate în timp real utilizatorului prin intermediul serverului, asigurând o informare promptă și eficientă despre statusul operațiunii de descărcare. Detaliile acestui proces sunt ilustrate în Figura 20.

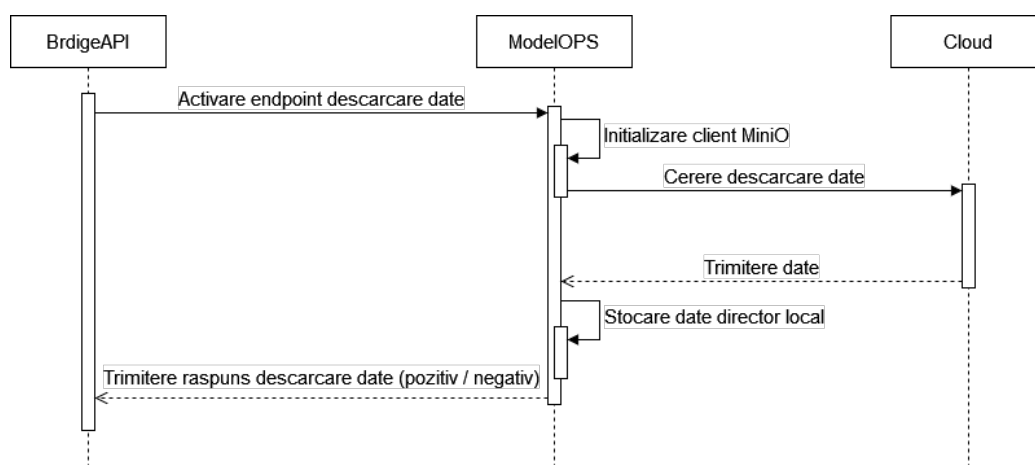


Figura 20: Diagrama fluxului de comunicare pentru descărcarea datelor

3.3.2 Curățarea și pregătirea datelor

Procesul de curățare a datelor este esențial pentru analiza analitică, influențând direct acuratețea rezultatelor. Deși vital, procesul standard oferit de platformă este opțional din cauza naturii sale generice și agresive. Acesta este conceput să funcționeze pe un spectru larg de tipuri de date și aplică

diverse proceduri pentru a asigura compatibilitatea cu modelul de predicție. Totuși, datorită generalizării sale și toleranței scăzute, utilizatorii sunt încurajați să opteze pentru metode de curățare personalizate, care se adaptează mai bine la structura și tipul datelor specifice utilizatorului.

În cadrul procesului de curățare, fișierele stocate în directorul temporar, menționat în secțiunea 3.3.1, sunt accesate și procesate. Datele curățate rezultate vor înlocui datele originale descărcate, dar este important de menționat că aceste modificări nu afectează structura datelor păstrate pe server. Aceasta asigură că datele originale rămân intacte pe server, în timp ce utilizatorii lucrează cu versiuni prelucrate local.

În absența unei soluții personalizate de curățare, utilizatorii pot utiliza metoda standardizată, cu toate riscurile asociate. Procesul de curățare modifică permanent structura setului de date prin aplicarea unei serii de algoritmi de preprocesare, care includ:

- Unificarea fișierelor cu informații similare
- Curățarea generală a datelor
- Codificarea valorilor de tip text
- Scalarea valorilor numerice

Aceste operațiuni asigură compatibilitatea datelor procesate cu modelul de predicție. De asemenea, platforma permite utilizatorilor să ajusteze parametrii de preprocesare pentru a personaliza procesul în funcție de nevoile specifice ale setului de date. Cu toate acestea, abordarea optimă rămâne utilizarea unei metode de curățare dezvoltată specific pentru datele utilizatorului.

3.3.2.1 Unificarea fișierelor cu denumiri similare

Unificarea fișierelor este un proces crucial în gestionarea eficientă a datelor, necesar în situațiile unde dimensiunea maximă a fișierelor este limitată de sistemele de gestionare. Utilizatorii pot împărți seturi mari de date în fișiere mai mici, denumite consistent pentru a indica proveniența comună. Această tactică facilitează reasamblarea datelor într-o structură unitară, simplificând manipularea și analiza lor.

Procesul de unificare este divizat în două etape principale:

- **Identificarea fișierelor cu nume similare:** Această etapă implică gruparea fișierelor dintr-un director temporar, bazată pe un scor de similaritate

între numele fișierelor. Scorul de similaritate este calculat folosind formula 1, care estimează cât de apropiate sunt două șiruri de caractere, returnând un scor între 0 și 100, unde 100 indică identitate completă.

$$\text{Rata de Similaritate} = \left(1 - \frac{\text{Distanța Levenshtein}}{\max(\text{len}(s1), \text{len}(s2))} \right) \times 100 \quad (1)$$

Distanța Levenshtein [21] reflectă numărul minim de modificări (inserții, ștergeri, substituiri) necesare pentru a transforma un șir în celălalt, oferind o măsură clară a similarității.

- **Consolidarea fișierelor:** După gruparea pe baza numelor, fișierele din fiecare grup sunt citite, datele sunt concatenate într-un singur set de date, iar rezultatul este salvat într-un director separat, pentru a nu afecta setul de date original. Aceasta asigură că setul de date rezultat este coeziv și complet.

Aceste procese permit serverului să gestioneze eficient și să reconstruiască seturi de date fragmentate într-un format organizat, facilitând analiza ulterioară și maximizând utilitatea datelor fără a pierde informații esențiale.

3.3.2.2 Curățarea generală a datelor

Procesul de curățare generală a datelor este esențial pentru asigurarea calității acestora în vederea utilizării în algoritmi de învățare automată. Aceasta implică eliminarea elementelor care nu respectă standardele necesare, asigurând astfel că datele sunt curate și structurate corespunzător. Conform articolului [32], curățarea datelor este vitală pentru performanța algoritmilor de învățare automată și, implicit, a inteligenței artificiale. În cadrul acestui proces, se desfășoară următoarele operațiuni:

- **Eliminarea liniilor și coloanelor cu un anumit procentaj de valori lipsă:** Se calculează proporția valorilor lipsă pentru fiecare coloană și linie. Coloanele și liniile care depășesc pragurile de toleranță specificate sunt eliminate pentru a preveni impactul negativ asupra analizei.
- **Eliminarea valorilor anormale (outliers):** Valorile extreme sunt identificate și eliminate folosind scorul Z (z-score), asigurându-se astfel că datele nu sunt distorsionate de anomalii statistice. Valorile care depășesc un anumit prag al scorului Z sunt considerate anormale și sunt excluse. Formula scorului Z [7] este:

$$z = \frac{X - \mu}{\sigma} \quad (2)$$

unde X reprezintă valoarea datelor, μ este media setului de date, iar σ este deviația standard.

- **Standardizarea coloanelor cu valori text:** Valorile text sunt uniformizate prin conversia la litere mici și eliminarea spațiilor de la începutul și sfârșitul fiecărei valori, facilitând astfel comparabilitatea și consistența datelor textuale.
- **Eliminarea coloanelor duplicate:** Coloanele duplicate sunt identificate și eliminate pentru a evita redundanța și pentru a simplifica structura setului de date.

Procesul de curățare generală a datelor este esențial pentru a obține un set de date robust și fiabil, care să sprijine predicțiile și analizele ulterioare. Aplicarea metodică a acestor operațiuni contribuie la eliminarea zgomotului și la îmbunătățirea performanței algoritmilor de învățare automată.

3.3.2.3 Codificarea valorilor de tip text

Codificarea valorilor de tip text este un pas crucial în preprocesarea datelor, esențial pentru a transforma datele categorice în formate numerice pe care algoritmi de învățare automată le pot procesa eficient. Există două metode principale de codificare: codificarea one-hot și codificarea prin etichete. Utilizatorul poate alege între aceste metode în funcție de necesitățile specifice ale analizei.

- **Codificarea one-hot:** Implică transformarea fiecărei categorii unice dintr-o coloană text într-un set de coloane binare. Fiecare coloană rezultată va avea valoarea 1 dacă rândul inițial aparține acelei categorii și 0 în caz contrar. Această metodă este utilă pentru evitarea problemelor de ordonare care pot apărea cu alte metode de codificare. Formula matematică pentru codificarea one-hot este:

$$y_i = \begin{cases} 1 & \text{dacă } x = c_i \\ 0 & \text{altfel} \end{cases} \quad (3)$$

unde x reprezintă valoarea inițială, c_i este categoria specifică, iar y_i este valoarea binară rezultată.

- **Codificarea prin etichete:** Presupune asignarea unei valori numerice fiecărei categorii distincte dintr-o coloană text. Fiecare categorie este mapată la un număr întreg unic. Această metodă este mai simplă și reduce dimensiunea setului de date, dar poate introduce o relație ordinală

falsă între categorii. Formula matematică pentru codificarea prin etichete poate fi descrisă printr-o funcție de mapare:

$$f : C \rightarrow \mathbb{N} \quad (4)$$

unde C este mulțimea de categorii și \mathbb{N} este mulțimea numerelor naturale.

Ambele metode au avantaje și dezavantaje, iar alegerea metodei potrivite depinde de specificul problemei de învățare automată abordate. Codificarea one-hot este de preferat atunci când dimensionalitatea nu este o problemă, adică atunci când valorile unei coloane nu au o varietate foarte mare și, prin urmare, dimensiunea setului de date nu crește semnificativ [11]. Pe de altă parte, codificarea prin etichete este utilă în cazul în care setul de date este deja de dimensiuni considerabile, iar adăugarea unor coloane suplimentare ar putea îngreuna procesul de predicție sau influența negativ rezultatele [10].

Aceste metode de codificare asigură că valorile categorice sunt transformate într-un format adecvat pentru algoritmi de învățare automată, îmbunătățind astfel performanța și acuratețea modelelor predictive. În cazul specific al identificării parametrilor care cauzează o boală, alegerea metodei de codificare adecvate poate avea un impact semnificativ asupra calității analizei și predicțiilor efectuate.

3.3.2.4 Scalarea valorilor numerice

Scalarea valorilor numerice este un pas esențial în preprocesarea datelor pentru algoritmi de învățare automată. Acest proces asigură că toate valorile numerice sunt pe o scară similară, îmbunătățind performanța algoritmilor și acuratețea predicțiilor. Două metode principale de scalare sunt utilizate frecvent: standardizarea și scalarea Min-Max.

■ **Standardizarea:** Presupune transformarea valorilor numerice astfel încât acestea să aibă o medie de 0 și o varianță de 1. Aceasta este utilă pentru algoritmi care presupun date distribuite normal sau care sunt sensibili la variația dintre scale. Formula matematică pentru standardizare [19] este:

$$z = \frac{x - \mu}{\sigma} \quad (5)$$

unde x reprezintă valoarea inițială, μ este media setului de date, iar σ este deviația standard.

- **Scalarea Min-Max:** Transformă valorile numerice astfel încât acestea să fie încadrate într-un interval specific, de obicei între 0 și 1. Această metodă păstrează relațiile proporționale dintre valorile inițiale și este utilă în special pentru algoritmi care nu fac presupuneri despre distribuția datelor. Formula matematică pentru scalarea Min-Max [17] este:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (6)$$

unde x este valoarea inițială, x_{\min} și x_{\max} sunt valorile minimă și, respectiv, maximă din setul de date.

Ambele metode au avantaje și dezavantaje, iar alegerea metodei potrivite depinde de specificul datelor. Standardizarea este de preferat atunci când datele urmează o distribuție normală. Pe de altă parte, scalarea Min-Max este utilă atunci când se dorește păstrarea proporțiilor originale ale datelor.

Aplicarea corectă a tehnicilor de scalare a valorilor numerice contribuie la îmbunătățirea performanței și a acurateței modelelor de învățare automată, asigurând astfel o mai bună interpretare și generalizare a rezultatelor.

3.3.3 Antrenarea modelului și salvarea rezultatelor

Ultima etapă a procesului analitic constă în antrenarea modelului și obținerea rezultatelor, urmată de salvarea acestora. Din perspectiva serverului, când endpointul de antrenare este activat, configurația modelului primită de la serverul intermediar este utilizată pentru inițializarea modelului de predicție. Odată inițializat, modelul este antrenat pe datele descărcate sau pe cele curățate, în funcție de opțiunea selectată de utilizator.

Antrenarea modelului este un proces intensiv din punct de vedere computațional, iar viteza acestuia depinde de volumul și structura datelor. Pentru a optimiza procesul și a evita stocarea unui volum mare de date în memorie, fiecare fișier este împărțit în mai multe fragmente (chunk-uri). Aceste fragmente sunt citite secvențial și trimise către model, asigurând astfel că modelul se antrenează pe toate datele disponibile, evitând în același timp citirea simultană a unui volum mare de date. Această abordare îmbunătățește viteza și reduce resursele computaționale necesare.

După antrenarea modelului, acesta este testat, iar rezultatele și acuratețea sunt salvate. Rezultatele sunt împărțite în mai multe categorii:

- **Parametrii:** Aceasta categorie include o listă de parametri identificați ca influențând boala specifică setului de date, împreună cu un procentaj care indică gradul de influență. Astfel, se poate distinge între parametrii cu impact scăzut și cei cu impact ridicat asupra bolii.

- **Grafice:** Aceasta categorie cuprinde o serie de grafice care ilustrează diferențele valorilor parametrilor extrași de la pacienții din setul de date, pe diferite stagii ale bolii (dacă există). Aceste grafice ajută specialistul medical să identifice tendințele parametrilor pe măsură ce boala evoluează și să determine intervalele de valori ale parametrilor în funcție de condiția bolii.
- **Analyze statistics:** Aceasta categorie include grafice care reprezintă vizual structura setului de date. Aceste grafice pot varia de la distribuția bolii pe vârstă și sex, la distribuția generală a bolii în cadrul setului de date. Este important de menționat că aceste grafice sunt generate pe baza datelor disponibile; absența anumitor valori poate duce la ne-generarea graficului respectiv. De asemenea, generarea acestor grafice ia în considerare denumirile coloanelor, astfel încât o denumire neconvențională a coloanei care reprezintă vârsta pacientului poate duce la excluderea acestuia din analiza statistică.

4 Modelul de învățare automată

În cadrul acestei lucrări, s-a utilizat un model de învățare automată de tip Random Forest pentru a analiza și interpreta datele medicale colectate. Modelul Random Forest este recunoscut pentru capacitatea sa de a gestiona seturi de date complexe și de mari dimensiuni, fiind eficient atât în probleme de clasificare, cât și de regresie. Acest model a fost ales datorită robusteții și acurateții sale, fiind capabil să ofere rezultate relevante și precise în contextul aplicațiilor medicale.

Modelul a fost testat și validat pentru a asigura fiabilitatea rezultatelor. Testarea a inclus evaluarea pe seturi de date diverse pentru a verifica performanța și consistența modelului. Rezultatele obținute au fost validate prin intermediul unor metrici specifice de performanță, asigurând astfel că modelul este adecvat pentru scopul propus.

În continuare, vor fi detaliate structura modelului, procedurile de antrenare și testare, precum și metricile utilizate pentru evaluarea performanței acestuia. Această abordare metodologică a permis obținerea unor rezultate relevante, contribuind la îmbunătățirea procesului de diagnosticare și analiză medicală.

4.1 Prezentarea algoritmului Random Forest

Algoritmul Random Forest este un algoritm de învățare automată utilizat pentru sarcini de clasificare și regresie. Dezvoltat de Leo Breiman în 2001, acest algoritm combină mulți arbori de decizie individuali pentru a forma o "pădure" de arbori, de unde derivă și numele său [9]. Prin agregarea predicțiilor multiplelor modele de arbori de decizie, Random Forest reușește să îmbunătățească performanța generală a modelului, reducând varianța și evitând supraînvățarea (overfitting).

4.1.1 Principiul de funcționare

Arborii de decizie: Fiecare arbore de decizie din cadrul unui model Random Forest este un clasificator de bază, construit prin împărțirea recursivă a datelor de antrenament în subseturi mai mici, bazate pe criterii de impuritate. Aceste criterii includ indicele Gini și entropia, care măsoară nivelul de impuritate sau neomogenitate în date.

Bootstrap Aggregating (Bagging): Random Forest utilizează tehnica de bagging pentru a crea multiple subseturi ale datelor de antrenament. Bagging implică prelevarea aleatorie, cu înlocuire, a subseturilor din setul de date original. Fiecare subset este utilizat pentru a antrena un arbore de decizie indivi-

dual, asigurându-se astfel diversitatea arborilor și reducerea varianței modelului [18].

Votul majoritar și media: În cazul problemelor de clasificare, predicția finală a modelului Random Forest este determinată prin votul majoritar al predicțiilor individuale ale arborilor. Pentru problemele de regresie, predicția finală este media predicțiilor tuturor arborilor. Acest mecanism de agregare asigură o predicție stabilă și robustă.

Matematic, procesul poate fi descris astfel:

1. Se selectează B eșantioane bootstrap din setul de date de antrenament.
2. Pentru fiecare eșantion bootstrap, se construiește un arbore de decizie T_b fără tăiere (pruning).
3. Pentru clasificare se utilizează funcția de decizie [9]:

$$\hat{f}(x) = \text{majoritatea } T_b(x)_{b=1}^B \quad (7)$$

unde $T_b(x)$ este predicția arborelui b pentru observația x .

4. Pentru regresie se utilizează funcția de decizie [18]:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^B T_b(x) \quad (8)$$

4.1.2 Avantajele și dezavantajele algoritmului

Avantaje:

- **Reziliență la overfitting:** Prin combinarea predicțiilor multiple, Random Forest reduce riscul de overfitting prezent în modelele individuale de arbori de decizie. Diversitatea arborilor asigură că modelul final este mai generalizabil și mai rezistent la variațiile din setul de date de antrenament.
- **Capacitatea de a gestiona seturi de date mari și complexe:** Algoritmul poate gestiona cu ușurință un număr mare de caracteristici și observații, fiind eficient în prezența datelor lipsă sau a variabilelor de tip mixt (numerice și categorice).
- **Versatilitate:** Poate fi utilizat atât pentru probleme de clasificare, cât și pentru regresie, adaptându-se ușor la diverse tipuri de probleme și date.

- **Estimarea importanței variabilelor:** Random Forest oferă o măsură a importanței fiecărei variabile în predicție, facilitând interpretabilitatea modelului și identificarea caracteristicilor relevante pentru problema de interes.

Dezavantaje:

- **Complexitate computațională ridicată:** Construirea și predicția cu mulți arbori de decizie necesită resurse computaționale semnificative, ceea ce poate duce la timpi de antrenare și inferență mai lungi, mai ales pentru seturi de date foarte mari.
- **Lipsa interpretabilității:** Deși poate indica importanța variabilelor, structura unui ansamblu de arbori de decizie este dificil de interpretat în comparație cu un singur arbore de decizie. Aceasta poate complica analiza detaliată a relațiilor dintre variabile și predicții.

4.1.3 Utilizări comune ale algoritmului Random Forest

Algoritmul Random Forest este aplicat pe scară largă în diverse domenii datorită versatilității și performanțelor sale:

- **Medicină:** Pentru diagnosticarea bolilor și analiza datelor medicale, Random Forest este utilizat pentru a identifica tiparele și a clasifica afecțiunile medicale pe baza simptomelor și datelor clinice [23].
- **Finanțe:** În domeniul financiar, algoritmul este utilizat pentru detectarea fraudelor și evaluarea riscurilor. Capacitatea sa de a manipula seturi de date complexe îl face ideal pentru aceste aplicații [29].
- **Marketing:** Random Forest este folosit pentru segmentarea pieței și personalizarea ofertelor, ajutând companiile să identifice preferințele clienților și să optimizeze strategiile de marketing [22].
- **Științe sociale:** Algoritmul este aplicat în analiza datelor comportamentale și predicția rezultatelor sociale, oferind insight-uri valoroase în studiile sociologice și de psihologie [15].

4.2 Scopul și motivația utilizării algoritmului Random Forest

În cadrul acestei lucrări, utilizarea unui model de învățare automată a fost esențială pentru analiza și interpretarea datelor medicale complexe. Algoritmul Random Forest a fost selectat datorită capacităților sale superioare de a gestiona date eterogene și de a furniza predicții precise și fiabile.

Principalul scop al utilizării algoritmului Random Forest a fost dezvoltarea unei soluții eficiente pentru diagnosticarea și prognosticarea bolilor, bazată pe datele medicale colectate. Acest model a fost ales pentru a exploata abilitatea sa de a oferi o analiză comprehensivă și de înaltă calitate, adaptată cerințelor unui mediu medical variat și dinamic.

Algoritmul Random Forest a fost considerat adecvat pentru proiectul CauselT datorită următoarelor motive fundamentale:

- **Fiabilitate și reziliență:** Capacitatea Random Forest de a combina multiple modele de arbori de decizie individuali asigură o reziliență sporită și reduce riscul de overfitting. Acest lucru este esențial în domeniul medical, unde datele pot fi zgomotoase și incomplete, iar precizia predicțiilor este crucială pentru luarea deciziilor clinice.
- **Versatilitate în manipularea variabilelor:** Algoritmul poate gestiona eficient variabile de tip mixt (numerice și categorice), fără a necesita transformări complexe. Această versatilitate este deosebit de utilă în contextul datelor medicale, care includ o gamă variată de tipuri de date.
- **Estimarea importanței caracteristicilor:** Random Forest oferă posibilitatea de a evalua importanța fiecărei variabile în procesul de predicție. Această funcționalitate este valoroasă pentru identificarea factorilor critici care influențează diagnosticul și prognosticul medical, facilitând interpretarea și înțelegerea rezultatelor.
- **Performanță și acuratețe:** Algoritmul este recunoscut pentru performanța sa superioară în diverse aplicații, inclusiv în diagnosticarea bolilor și analiza comportamentală. Această reputație de fiabilitate și acuratețe l-a făcut o alegere optimă pentru aplicația dezvoltată.
- **Scalabilitate și eficiență computațională:** Deși Random Forest implică antrenarea mai multor arbori de decizie, algoritmul este scalabil și poate fi optimizat pentru a funcționa eficient pe seturi mari de date. Această caracteristică este importantă pentru aplicații care necesită analiza unor volume mari de date, cum ar fi cele medicale.

În concluzie, algoritmul Random Forest a fost selectat pentru aplicația CauselT datorită capacităților sale avansate de gestionare a datelor complexe și variabile, rezilienței în fața zgomotului și a datelor incomplete, și abilității de a furniza predicții precise și interpretabile. Alegerea acestui model a fost motivată de necesitatea de a dezvolta o soluție analitică eficientă și de încredere pentru diagnosticarea și prognosticul bolilor, aspecte critice în îmbunătățirea proceselor clinice și în luarea deciziilor medicale informate.

4.3 Implementarea modelului

În cadrul acestei lucrări, a fost implementat un algoritm de clasificare folosind Random Forest, utilizând diverse biblioteci din ecosistemul Python. Algoritmul Random Forest este cunoscut pentru capacitatea sa de a gestiona seturi de date mari și complexe, oferind rezultate robuste și interpretabile. Această secțiune descrie în detaliu procesul de implementare, incluzând selectarea caracteristicilor, antrenarea modelului și interpretarea rezultatelor.

4.3.1 Biblioteci utilizate

Implementarea algoritmului Random Forest a fost realizată utilizând următoarele biblioteci esențiale:

- **pandas**: Utilizată pentru manipularea și analiza seturilor de date.
- **numpy**: Folosită pentru operațiuni numerice eficiente.
- **scipy**: Aplicată pentru efectuarea testelor statistice.
- **seaborn** și **matplotlib**: Folosite pentru vizualizarea datelor și interpretarea grafică a rezultatelor.
- **scikit-learn**: Bibliotecă principală pentru construirea și evaluarea modelului Random Forest.
- **shap**: Utilizată pentru interpretarea modelului prin valori Shapley.

4.3.2 Selectarea caracteristicilor relevante

Pentru a determina caracteristicile relevante care vor fi incluse în modelul Random Forest, a fost utilizat testul Kruskal-Wallis. Acesta este un test non-parametric care verifică dacă există diferențe semnificative statistic între distribuțiile fiecărei caracteristici numerice în funcție de categoriile definite de variabila țintă. Utilizarea testului Kruskal-Wallis în acest context are scopul de a reduce dimensionalitatea setului de date, eliminând caracteristicile care nu contribuie semnificativ la clasificare.

Testul Kruskal-Wallis este preferat pentru că poate gestiona date care nu urmează o distribuție normală, fiind astfel mai robust la diverse tipuri de date. Formula statisticii Kruskal-Wallis [20], H , este:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N+1) \quad (9)$$

unde:

- H este valoarea statisticii Kruskal-Wallis,
- N este numărul total de observații,
- k este numărul de grupuri,
- R_i este suma rangurilor pentru al i -lea grup,
- n_i este numărul de observații în al i -lea grup.

Dacă valoarea p asociată testului este mai mică decât pragul de semnificație (de exemplu, 0.05), caracteristica respectivă este considerată semnificativă și este inclusă în model.

Această metodă de selecție a caracteristicilor este esențială pentru reducerea dimensionalității setului de date, ceea ce poate îmbunătăți performanța modelului și reduce timpul de calcul necesar pentru antrenare. Utilizarea testului Kruskal-Wallis în astfel de contexte este bine documentată. De exemplu, un studiu publicat de Xing, Jordan și Karp (2001) în *Proceedings of the 18th International Conference on Machine Learning* menționează că "testul Kruskal-Wallis este adesea utilizat pentru selecția caracteristicilor în problemele de clasificare datorită performanțelor sale și a capacității de a trata date neparametrice" [36].

4.3.3 Antrenarea modelului Random Forest

Procesul de antrenare a modelului Random Forest implică mai multe etape esențiale pentru asigurarea unui model robust și performant. Inițial, setul de date este împărțit în date de antrenament și date de testare într-un raport de 80% - 20%. Această împărțire este crucială pentru evaluarea obiectivă a performanței modelului, asigurând că evaluarea se face pe un set de date care nu a fost utilizat în timpul antrenamentului.

Modelul Random Forest este configurat utilizând biblioteca scikit-learn, având în vedere parametrii esențiali precum:

- **Numărul de arbori (n_estimators):** Numărul de arbori din pădurea aleatorie. Un număr mai mare de arbori poate crește performanța modelului, dar și timpul necesar pentru antrenare.
- **Adâncimea maximă a arborilor (max_depth):** Limitează adâncimea fiecărui arbore pentru a preveni overfitting-ul, adică învățarea excesivă a datelor de antrenament, care poate conduce la performanțe slabe pe datele noi.

- **Ponderile echilibrate ale claselor (`class_weight='balanced'`):** Utilizează ponderi echilibrate pentru a gestiona problemele de clasificare dezechilibrată, unde unele clase pot fi mult mai frecvente decât altele.
- **Stare aleatoare (`random_state`):** Asigură reproducibilitatea rezultatelor, permițând obținerea acelorași rezultate la fiecare execuție a modelului.

Formula pentru calcularea importanței caracteristicilor [9] este esențială în interpretarea modului în care fiecare caracteristică contribuie la deciziile modelului. Importanța unei caracteristici j se calculează folosind:

$$I_j = \sum_{t \in T} \frac{N_t}{N} \Delta I(s_t, t) \quad (10)$$

unde:

- I_j este importanța caracteristicii j ,
- T este setul tuturor arborilor din pădurea aleatorie,
- N_t este numărul de mostre care ajung în nodul t ,
- N este numărul total de mostre,
- $\Delta I(s_t, t)$ este scăderea impurității în nodul t datorată separării pe baza caracteristicii s_t .

După configurare, modelul este antrenat pe datele de antrenament, optimizând criteriul de separare în fiecare arbore pentru a maximiza acuratețea predicțiilor. Performanța modelului este evaluată pe setul de testare folosind acuratețea, definită ca raportul dintre numărul de predicții corecte și numărul total de predicții.

Un aspect important al implementării este salvarea modelului antrenat. Modelul este salvat pentru a permite reutilizarea sa în viitor fără a fi necesară reantrenarea completă, economisind astfel timp și resurse computaționale. Salvarea este realizată folosind funcția `joblib.dump` din biblioteca `joblib`.

Aceste etape asigură dezvoltarea unui model robust, capabil să ofere predicții precise și să fie reutilizat eficient în diferite aplicații.

4.4 Rezultate și validarea modelului

În această secțiune vom prezenta rezultatele obținute în urma testării algoritmului Random Forest pe trei tipuri de seturi de date distincte: seturi de

date artificiale, seturi de date reale publice și seturi de date reale private. Fiecare tip de set de date oferă un context diferit pentru evaluarea performanței algoritmului, permițând o analiză cuprinzătoare a capacităților sale.

1. **Set de date artificial:** Acest set de date a fost generat artificial, fiecare valoare fiind calculată pe baza unei funcții matematice. Scopul acestui set de date este de a simula un set de date real, dar cu avantajul de a cunoaște în prealabil corelațiile dintre parametri și importanța lor. Algoritmul nu are acces la aceste valori, fiind responsabil de identificarea lor în mod independent. Această abordare permite o comparație clară între rezultatele obținute de model și valorile reale, verificând astfel acuratețea și eficiența algoritmului în condiții controlate.
2. **Set de date real public:** Acesta este un set de date real, accesibil publicului, care conține biomarkeri ai pacienților cu risc de cancer pulmonar sau care au dezvoltat deja această boală. În acest caz, importanțele și valorile relevante nu sunt cunoscute anterior. Modelul este testat pe aceste date reale, oferind o evaluare practică a performanței sale într-un context autentic. Această evaluare este crucială pentru a determina aplicabilitatea algoritmului în scenarii reale de diagnostic medical.
3. **Set de date real privat:** Acest set de date a fost furnizat de compania Siemens în cadrul proiectului IHelp, realizat în colaborare cu TMU. Setul de date conține informații despre cancerul pancreatic, fiind mult mai complex, cuprinzând sute de fișiere ale pacienților din diferite spitale. Complexitatea acestui set de date servește ca exemplu al capacității algoritmului de a opera într-un mediu real cu o complexitate ridicată. Deși nu pot fi prezentate explicit valorile din acest set de date din cauza confidențialității, pot fi discutate rezultatele obținute și succesul demonstrat de algoritm. Utilizarea acestor date permite o evaluare aprofundată a algoritmului în condiții de utilizare reală și în medii cu restricții stricte de confidențialitate.

În concluzie, rezultatele obținute pe aceste seturi de date variate subliniază robustețea și versatilitatea algoritmului Random Forest. Performanța acestuia pe seturi de date artificiale confirmă capacitatea sa de a identifica corelații precise, în timp ce testarea pe seturi de date reale demonstrează aplicabilitatea practică și eficiența algoritmului în scenarii autentice și confidențiale.

4.4.1 Setul de date artificial

Pentru evaluarea performanței algoritmului Random Forest într-un mediu controlat, a fost generat un set de date artificial. Acest set de date a fost

creat pentru a simula un scenariu realist de diagnosticare a diabetului de tip 2, permițând evaluarea corelațiilor dintre diferiți parametri clinici și importanța acestora în predicțiile modelului.

4.4.1.1 Generarea datelor

Setul de date generat conține următoarele variabile:

- **Vârstă (Age):** Generată utilizând o distribuție normală cu media de 50 și deviația standard de 12.
- **Indicele de masă corporală (BMI):** Generat utilizând o distribuție normală cu media de 28 și deviația standard de 5.
- **Glucoza (Glucose):** Generată utilizând o distribuție normală cu media de 100 și deviația standard de 15.
- **Tensiunea arterială sistolică (Systolic_BP):** Generată utilizând o distribuție normală cu media de 120 și deviația standard de 15.
- **Tensiunea arterială diastolică (Diastolic_BP):** Generată utilizând o distribuție normală cu media de 80 și deviația standard de 10.
- **Colesterol (Cholesterol):** Generat utilizând o distribuție normală cu media de 200 și deviația standard de 30.
- **Markerul de inflamație (Inflammation_Marker):** Calculat utilizând o distribuție normală absolută cu media de 0.5 și deviația standard de 0.3.
- **Markerul de stres oxidativ (Oxidative_Stress_Marker):** Calculat pe baza markerului de inflamație, cu o componentă aleatoare normală adăugată, având media de 0 și deviația standard de 0.2.
- **Rezultatul bolii (Disease_Outcome):** Clasificat pe baza scorului bolii într-o combinație liniară a variabilelor de mai sus, împărțit în patru categorii: 0 (sănătos), 1 (stadiu incipient), 2 (stadiu mediu) și 3 (stadiu avansat).

O porțiune din valorile generate în cadrul setului de date artificial este prezentată în Tabelul 1.

Age	BMI	Glucose	Systolic_BP	Disease_Outcome
56	21.5	89.7	109.9	0
48	24.9	95.2	117.8	1
58	25.1	105.3	108.1	2
68	26.9	118.4	115.4	3
47	19.5	92.1	91.6	0

Tabela 1: Primele cinci rânduri din setul de date artificial (coloane selectate)

Matricea de corelație (Figura 21) a fost generată pentru a analiza relațiile dintre diferiții parametri ai setului de date. Aceasta joacă un rol crucial în identificarea interdependențelor dintre variabile și în evaluarea relevanței fiecărui parametru pentru predicțiile modelului. Rezultatele acestei analize sunt esențiale pentru a înțelege cum fiecare variabilă contribuie la rezultatul final și pentru a ajusta modelul în consecință.

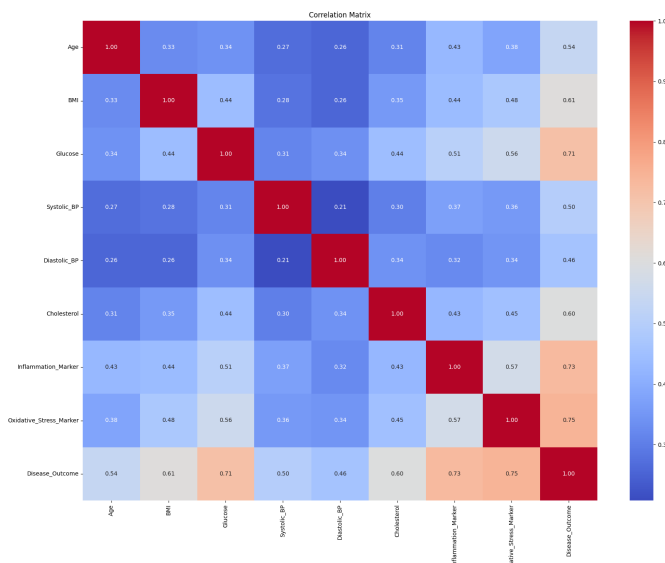


Figura 21: Matricea de corelație pentru setul de date artificial

Vizualizările suplimentare ale relațiilor dintre biomarkeri și rezultatul bolii au fost generate pentru a oferi o perspectivă vizuală asupra modului în care fiecare parametru variază în funcție de gravitatea bolii. În figurile 22 și 23 sunt prezentate exemple de astfel de grafice, arătând diferențele parametrilor colesterol și indice de masă corporală (BMI) pentru fiecare categorie de boală.

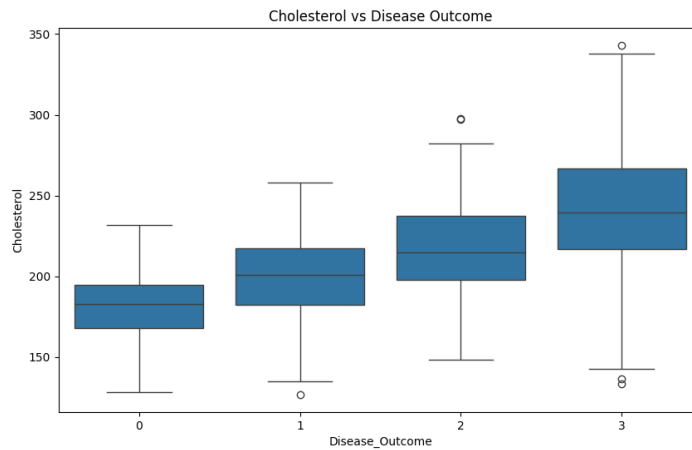


Figura 22: Colesterol vs Rezultatul bolii

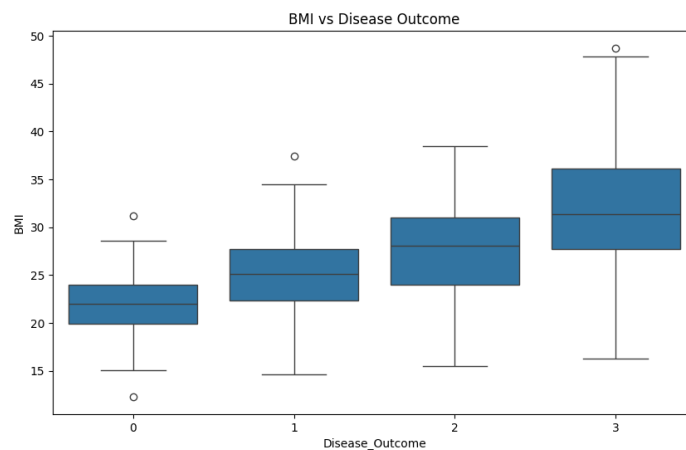


Figura 23: Indicele de masă corporală (BMI) vs Rezultatul bolii

Setul de date artificial oferă un mediu controlat pentru testarea și validarea algoritmului Random Forest, demonstrând capacitatea sa de a identifica corelații și de a face predicții precise bazate pe datele disponibile. Acest set de date permite o evaluare detaliată a performanței modelului în condiții similare cu cele reale, dar controlabile.

4.4.1.2 Rezultatele obținute

În urma rularii algoritmului Random Forest pe setul de date artificial, au fost identificați următorii parametri împreună cu scorurile lor de importanță, prezentați în Tabelul 2

Parametru	Importanță (%)
Glucoză (Glucose)	18.12
Marker de stres oxidativ (Oxidative_Stress_Marker)	16.84
Colesterol (Cholesterol)	16.75
Marker de inflamație (Inflammation_Marker)	11.42
Vârstă (Age)	10.50
Indicele de masă corporală (BMI)	9.84
Tensiunea arterială sistolică (Systolic_BP)	8.43
Tensiunea arterială diastolică (Diastolic_BP)	8.10

Tabela 2: Importanța parametrilor identificați de algoritmul Random Forest

Analiza importanței parametrilor identificați de către Random Forest, a fost realizată folosind SHAP. SHAP (SHapley Additive exPlanations) este o metodă de interpretare a modelelor de învățare automată, bazată pe conceptul de valori Shapley din teoria jocurilor cooperative. SHAP oferă o explicație consistentă și unificată a contribuției fiecărei caracteristici la predicțiile modelului. Prin utilizarea valorilor SHAP, se poate înțelege mai bine importanța relativă a fiecărei caracteristici și modul în care acestea influențează predicțiile.

Conform studiului realizat de Lundberg și Lee (2017), SHAP oferă un cadru robust și coerent pentru interpretarea modelelor complexe de învățare automată [24].

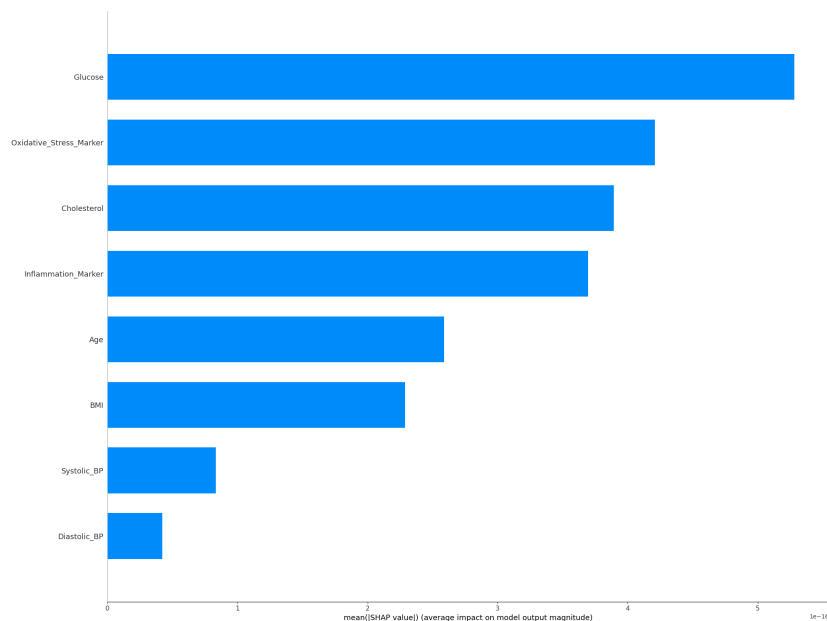
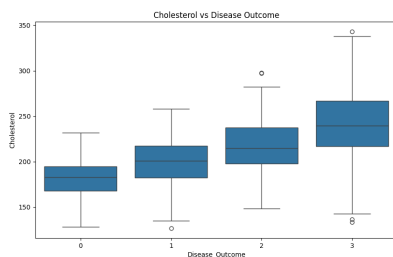


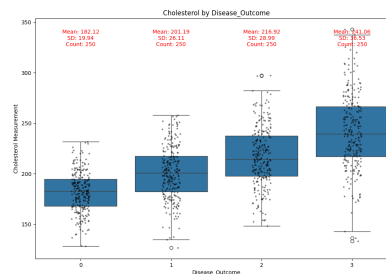
Figura 24: Graficul de importanță SHAP pentru setul de date artificial

Graficul de importanță SHAP prezentat în Figura 24 arată modul în care fiecare caracteristică a influențat predicțiile modelului. Importanța fiecărui parametru este reprezentată prin lungimea barei corespunzătoare. Conform graficului SHAP, glucoza (Glucose) a avut cea mai mare influență asupra predicțiilor modelului, urmat de markerul de stres oxidative (Oxidative_Stress_Marker), colesterol (Cholesterol) și markerul de inflamație (Inflammation_Marker). Aceasta este în concordanță cu scorurile de importanță obținute anterior.

Nu în ultimul rând, în figurile 25 și 26, se poate observa comparația graficului original versus cel generat de Random Forest, care indică distribuția parametrului de-a lungul stagiilor bolii. Din similaritatea lor, se poate concluziona faptul că descoperirile făcute de Random Forest sunt corecte.

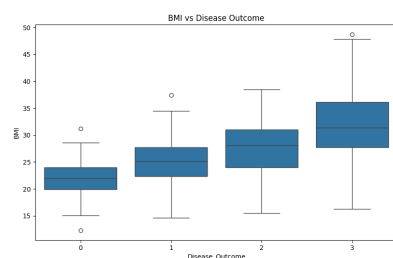


(a) Graficul original

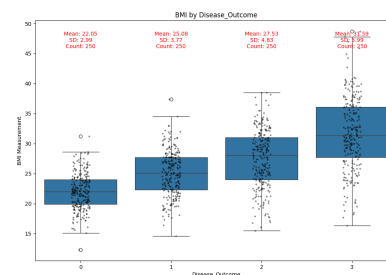


(b) Graficul generat de Random Forest

Figura 25: Comparația graficului original cu cel generat de Random Forest pentru parametrul Cholesterol



(a) Graficul original



(b) Graficul generat de Random Forest

Figura 26: Comparația graficului original cu cel generat de Random Forest pentru parametrul BMI

4.4.1.3 Validarea rezultatelor

Setul de date artificial are rolul de a simula un set de date real, dar într-un mediu controlat unde corelațiile și importanțele parametrilor sunt cunoscute. Astfel, se poate evalua precis comportamentul modelului și se poate calcula o metrică pentru a determina performanța modelului pe seturile de date reale. Acest lucru este esențial, deoarece în seturile de date medicale reale, corelațiile nu pot fi deduse direct din date, acesta fiind scopul algoritmului. În setul de date artificial, testele se desfășoară într-un mediu controlat care permite evaluarea exactă a performanței modelului. Modelul oferă două tipuri de rezultate: identificarea parametrilor relevanți și determinarea importanței acestora, fiind necesare două tipuri de validări.

Prima validare evaluează acuratețea cu care modelul identifică parametrii relevanți. În seturile medicale, există mulți parametri, iar scopul este identificarea celor mai relevanți. Acest test, aplicabil și pe seturile de date reale, constă într-o evaluare a acurateții prin prezicerea stadiului bolii pe baza parametrilor extrași. Setul de date este împărțit în set de antrenament și set de testare, folosind un train-test split clasic, unde 80% din date sunt utilizate pentru antrenament și 20% pentru testare. În cadrul setului de date artificial, modelul a obținut o acuratețe de 85%, conform standardelor pentru modelele de învățare automată [35]. Formula matematică pentru calcularea acurateții [30] este:

$$\text{Acurate\cetea} = \frac{\text{Num\cărul de predic\cții corecte}}{\text{Num\cărul total de predic\cții}} \times 100 \quad (11)$$

Astfel, se poate concluziona c\c modelul identific\c cu succes parametrii relevanți ai bolii.

Cea de-a doua validare confirm\c validitatea modelului \cncr-un context general, fiind aplicabil\c doar pe setul de date artificial, unde corela\ciiile sunt cunoscute \c controlabile. \cncr seturile de date reale, aceste corela\cii nu sunt disponibile direct.

Pentru evaluarea corela\ciiilor dintre variabile, s-a utilizat analiza componentelor principale (PCA). PCA este o metod\c de reducere a dimensionalit\cii care transform\c variabilele corelate \cncr-un set de componente principale necorelate. Aceasta permite identificarea rela\ciiilor dintre variabile \c extragerea caracteristicilor esen\ciale.

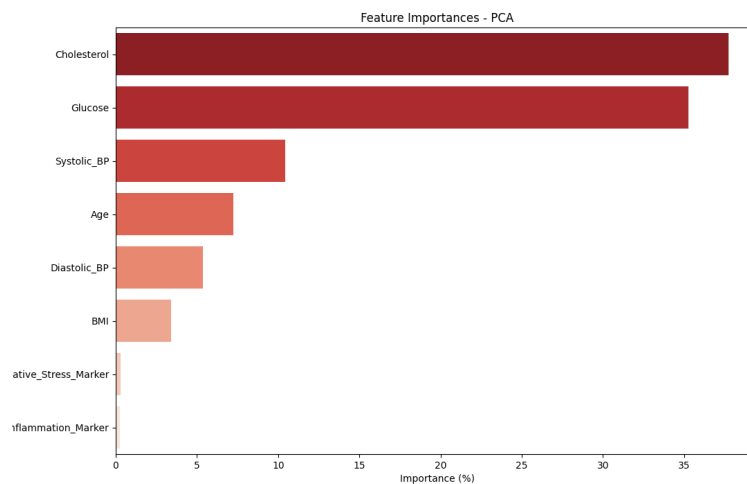


Figura 27: Importanța parametrilor identificată prin PCA

Coeficientul Spearman [16] este utilizat pentru a măsura corelația dintre două seturi de date ordinale. Acesta variază între -1 și 1, unde 1 indică o corelație perfect pozitivă, -1 o corelație perfect negativă și 0 absența corelației. Formula coeficientului Spearman este:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (12)$$

unde d_i este diferența dintre rangurile a două variabile corespunzătoare, iar n este numărul de perechi de valori.

Coeficientul Spearman pentru PCA a fost de 0.45, în timp ce pentru Random Forest a fost de 0.95, indicând o corelație mult mai puternică între parametrii identificați de Random Forest și importanțele reale.

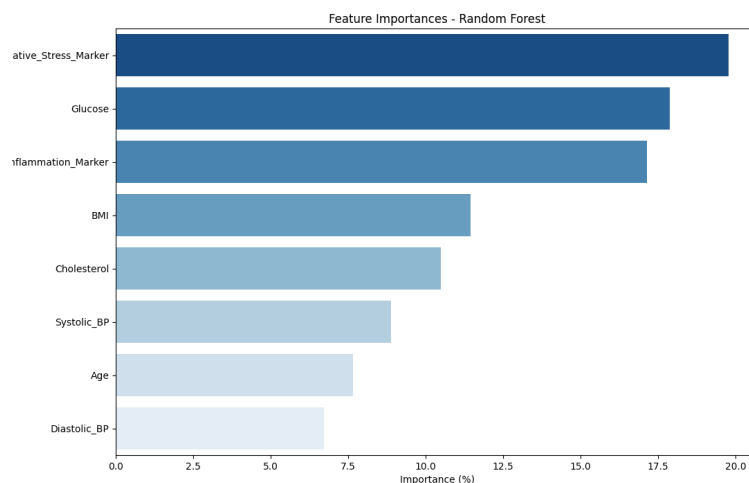


Figura 28: Importanța parametrilor identificată de Random Forest

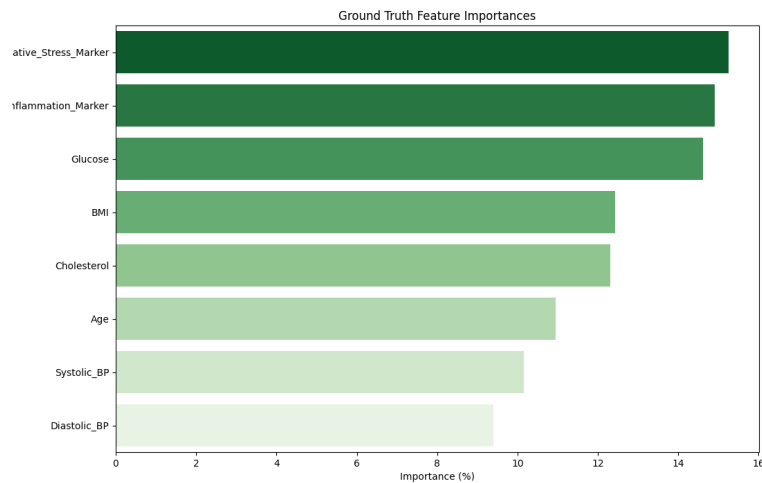


Figura 29: Importanța parametrilor conform adevărului cunoscut

Aceste rezultate arată că Random Forest este mai eficient în identificarea corelațiilor corecte între variabilele setului de date artificial. Performanța superioară a Random Forest se datorează capacității sale de a captura relații non-liniare și interacțiuni complexe între variabile, aspecte pe care PCA, fiind o metodă liniară, nu le poate detecta la fel de bine. Astfel, se poate concluziona că Random Forest este mai potrivit pentru identificarea importanței variabilelor într-un set de date complex și realist, precum cel utilizat în cadrul acestei lucrări.

4.4.2 Setul de date real public

Pentru evaluarea performanței algoritmului Random Forest într-un mediu necontrolat, a fost utilizat un set de date real disponibil public de pe Kaggle, referitor la cancerul pulmonar [8]. Acest set de date permite evaluarea corelațiilor dintre diferiți parametri clinici și importanța acestora în predicțiile modelului, oferind o perspectivă realistă asupra aplicabilității algoritmului.

4.4.2.1 Descrierea setului de date

Setul de date conține următoarele variabile:

- ▣ **Vârstă (Age):** Vârsta pacientului.
- ▣ **Sex (Gender):** Sexul pacientului.

- ▣ **Fumatul (Smoking):** Indică dacă pacientul fumează.
- ▣ **Consum de alcool (Alcohol_consuming):** Indică dacă pacientul consumă alcool.
- ▣ **Alergie (Allergy):** Indică dacă pacientul are alergii.
- ▣ **Tuse (Coughing):** Indică prezența tusei.
- ▣ **Dificultate la înghițire (Swallowing_difficulty):** Indică dacă pacientul are dificultăți la înghițire.
- ▣ **Oboseală (Fatigue):** Indică prezența oboselii.
- ▣ **Respirație șuierătoare (Wheezing):** Indică prezența respirației șuierătoare.
- ▣ **Durere toracică (Chest_pain):** Indică prezența durerii toracice.
- ▣ **Respirație scurtă (Shortness_of_breath):** Indică prezența respirației scurte.
- ▣ **Presiune (Peer_pressure):** Indică presiunea resimțită de la mediul social.
- ▣ **Anxietate (Anxiety):** Indică prezența anxietății.
- ▣ **Degete galbene (Yellow_fingers):** Indică prezența degetelor galbene.
- ▣ **Boală cronică (Chronic_disease):** Indică prezența unei boli cronice.

O porțiune din valorile setului de date real este prezentată în Tabelul 3.

Age	Gender	Smoking	Alcohol_consuming	Coughing	Lung_Cancer
45	M	Yes	No	Yes	Yes
34	F	No	Yes	No	No
67	M	Yes	Yes	Yes	Yes
50	F	No	No	No	No
59	M	Yes	No	Yes	Yes

Tabela 3: Primele cinci rânduri din setul de date real (coloane selectate)

Setul de date real oferă un mediu necontrolat pentru testarea și validarea algoritmului Random Forest, demonstrând capacitatea sa de a identifica corelații și de a face predicții precise bazate pe datele disponibile.

4.4.2.2 Rezultatele obținute

În urma rularii algoritmului Random Forest pe setul de date real, au fost identificați următorii parametri împreună cu scorurile lor de importanță, prezentați în Tabelul 4.

Parametru	Importanță (%)
Dificultate la înghițire (Swallowing_difficulty)	10.24
Tuse (Coughing)	9.81
Consum de alcool (Alcohol_consuming)	8.45
Alergie (Allergy)	7.92
Oboseală (Fatigue)	7.31
Degete galbene (Yellow_fingers)	7.21
Presiune (Peer_pressure)	6.93
Respirație șuierătoare (Wheezing)	6.88
Vârstă (Age)	6.52
Anxietate (Anxiety)	6.48
Fumat (Smoking)	6.29
Boală cronică (Chronic_disease)	6.12
Durere toracică (Chest_pain)	5.95
Respirație scurtă (Shortness_of_breath)	5.78
Sex (Gender)	5.57

Tabela 4: Importanța parametrilor identificați de algoritmul Random Forest pe setul de date real

Graficul de importanță SHAP pentru setul de date real este prezentat în Figura 30. Graficul arată modul în care fiecare caracteristică a influențat predicțiile modelului. Importanța fiecărui parametru este reprezentată prin lungimea barei corespunzătoare. Conform graficului SHAP, dificultatea la înghițire (Swallowing_difficulty) a avut cea mai mare influență asupra predicțiilor modelului, urmată de tuse (Coughing), consum de alcool (Alcohol_consuming), alergie (Allergy) și oboseală (Fatigue).

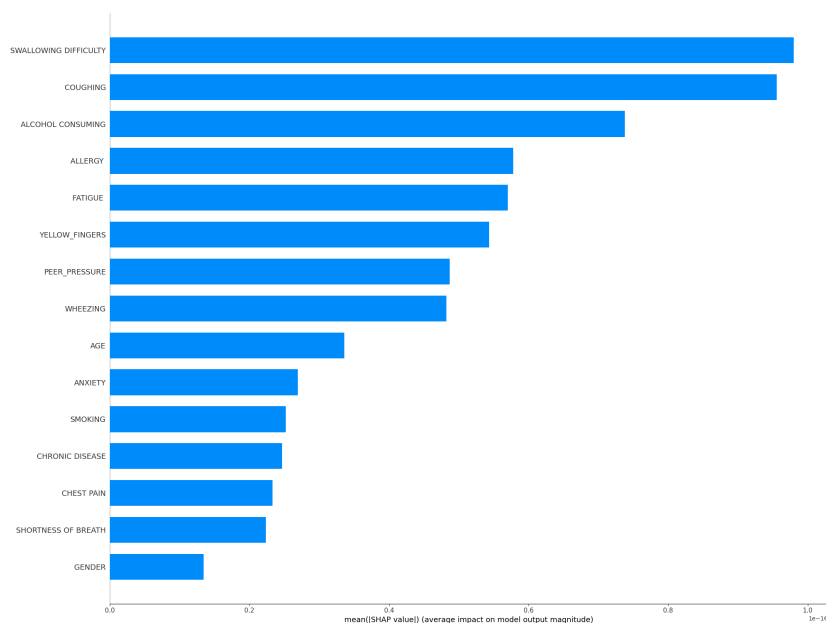


Figura 30: Graficul de importanță SHAP pentru setul de date real

4.4.2.3 Validarea rezultatelor

Setul de date real public nu permite aplicarea tehnicii de validare a importanței parametrilor deoarece corelațiile exacte dintre parametri nu sunt cunoscute, fiind un set de date real necontrolat. Cu toate acestea, demonstrația validării realizate pe setul de date artificial asigură că rezultatele obținute în cazul real sunt valide. În setul de date real, modelul Random Forest a obținut o acuratețe de 94%, ceea ce indică o performanță excelentă în prezicerea prezenței cancerului pulmonar.

Astfel, se poate concluziona că modelul Random Forest identifică cu succes parametrii relevanți și face predicții precise chiar și în contextul unui set de date real și necontrolat. Validarea realizată pe setul de date artificial asigură că tehnicile și algoritmi utilizați sunt fiabili și aplicabili în practică.

4.4.3 Setul de date real privat

Setul de date real privat a fost oferit de către Siemens, în cadrul proiectului IHelp, în colaborare cu TMU, sub supravegherea domnului Gabriel Danciu. Acest set de date conținea informații despre pacienții de la diferite spitale din Taipei, suspectați sau confirmați ca având cancer pancreatic. În cadrul acestui proiect, algoritmul Random Forest a avut o acuratețe de identificare a parametrilor de

peste 90%, rezultatele obținute fiind validate împreună cu experți din domeniul medical.

5 Concluzii

Proiectul a investigat performanța algoritmului Random Forest în diagnosticarea și identificarea corelațiilor între parametri clinici specifici bolilor complexe, utilizând seturi de date artificiale și reale. Abordarea a fost structurată în mai multe etape, fiecare contribuind la evaluarea și validarea modelului propus.

Setul de date artificial: Pentru a evalua într-un mediu controlat capacitatea algoritmului Random Forest de a identifica corelații și importanțe ale variabilelor, a fost generat un set de date artificial. Acest set a simulat un scenariu realist de diagnosticare a diabetului de tip 2, incluzând variabile precum vârsta, indicele de masă corporală (BMI), nivelul de glucoză, tensiunea arterială, colesterolul și markeri de inflamație și stres oxidativ. Testele au arătat că modelul poate atinge o acuratețe de 85%, identificând corect parametrii relevanți. Această validare demonstrează capacitatea modelului de a funcționa eficient într-un mediu controlat.

Setul de date real public: Un set de date real referitor la cancerul pulmonar, disponibil pe platforma Kaggle, a fost utilizat pentru a valida modelul în condiții reale. Acest set de date a inclus variabile precum dificultatea la înghițire, tusea, consumul de alcool, alergiile, oboseala, degetele îngălbenite și altele. Algoritmul Random Forest a atins o acuratețe de 94%, demonstrându-și eficiența în identificarea parametrilor relevanți și a corelațiilor într-un mediu real necontrolat. Validarea acestui model, chiar și în lipsa unor corelații cunoscute, a fost susținută de rezultatele obținute pe setul de date artificial, garantând astfel validitatea concluziilor.

Setul de date real privat: În cadrul proiectului IHelp, în colaborare cu TMU și Siemens, a fost accesat un set de date confidențial referitor la pacienți suspectați sau confirmați cu cancer pancreatic din diverse spitale din Taipei. Algoritmul Random Forest a obținut o acuratețe de peste 90% în identificarea parametrilor relevanți, confirmându-și astfel utilitatea și aplicabilitatea practică. Rezultatele obținute au fost validate împreună cu experți din domeniul medical, asigurându-se astfel acuratețea și relevanța acestora.

Implementarea platformei web: Pe lângă analiza datelor și dezvoltarea algoritmilor, a fost implementată cu succes o platformă web accesibilă medicilor. Această platformă permite utilizarea descoperirilor făcute în cadrul proiectului, facilitând astfel accesul rapid și eficient la informațiile esențiale pentru diagnosticarea și monitorizarea pacienților.

Contribuții și impact: Proiectul a demonstrat capacitatea algoritmului Random Forest de a funcționa eficient atât în medii controlate, cât și în scenarii reale, necontrolate. Acesta poate fi utilizat în diagnosticarea și monitorizarea bolilor complexe, oferind o unealtă robustă pentru profesioniștii din domeniul

medical. Validarea modelului în diverse contexte și pe diferite tipuri de date a subliniat versatilitatea și robustețea acestuia.

5.1 Dezvoltări viitoare

Pe baza rezultatelor obținute și a succesului demonstrat al algoritmului Random Forest în acest proiect, există mai multe direcții posibile pentru dezvoltări viitoare:

- **Integrarea datelor multi-omice:** Extinderea seturilor de date pentru a include informații genomice, proteomice și metabolomice ar putea îmbunătăți semnificativ acuratețea și utilitatea modelului în diagnosticarea bolilor complexe.
- **Dezvoltarea unei aplicații mobile:** Crearea unei aplicații mobile care să permită medicilor să introducă datele pacienților și să primească instantaneu predicții și analize privind riscurile și stadiile bolilor.
- **Învățare transferabilă:** Implementarea tehnicilor de învățare transferabilă (transfer learning) pentru a adapta modelele pre-antrenate pe seturi de date mari și diverse la seturi de date specifice instituțiilor medicale individuale.
- **Monitorizarea în timp real:** Dezvoltarea unor sisteme de monitorizare în timp real care să utilizeze senzori și dispozitive portabile pentru a colecta date în mod continuu, oferind astfel o evaluare dinamică și actualizată a stării de sănătate a pacienților.
- **Automatizarea diagnosticării:** Explorarea tehnologiilor de inteligență artificială pentru a automatiza procesul de diagnosticare, reducând astfel timpul și resursele necesare și crescând eficiența sistemului medical.

Aceste direcții de dezvoltare viitoare au potențialul de a revoluționa modul în care sunt diagnosticate și monitorizate bolile complexe, contribuind semnificativ la îmbunătățirea calității vieții pacienților și eficienței sistemului medical.

Rezumat

Această lucrare își propune cercetarea utilizării algoritmului Random Forest pentru identificarea factorilor bolilor și importanța acestora, precum și integrarea acestor descoperiri într-o platformă accesibilă personalului medical. Proiectul implică dezvoltarea întregului pipeline, de la generarea și preprocesarea datelor, până la implementarea și evaluarea modelului, și dezvoltarea unei platforme web.

Pentru a evalua performanța algoritmului, au fost utilizate seturi de date artificiale și reale. Setul de date artificial a fost generat pentru a simula un scenariu realist de diagnosticare, incluzând variabile clinice relevante. Algoritmul a atins o acuratețe de 85% pe acest set de date, demonstrând capacitatea de a identifica corelații și importanțe ale variabilelor într-un mediu controlat.

Pe setul de date real referitor la cancerul pulmonar, algoritmul a atins o acuratețe de 94%, validându-și astfel eficiența într-un mediu necontrolat. În cadrul proiectului IHelp, în colaborare cu TMU și Siemens, algoritmul a obținut o acuratețe de peste 90% pe un set de date confidențial referitor la cancerul pancreatic, confirmându-și utilitatea practică. De asemenea, a fost implementată o platformă web care permite medicilor să acceseze și să utilizeze descoperirile făcute.

Rezultatele obținute evidențiază eficiența și versatilitatea algoritmului Random Forest, sugerând direcții viitoare de dezvoltare, cum ar fi integrarea datelor multi-omice, dezvoltarea unei aplicații mobile, îmbunătățirea monitorizării în timp real și automatizarea procesului de diagnosticare.

Abstract

This thesis aims to investigate the use of the Random Forest algorithm for identifying disease factors and their importance, as well as integrating these findings into a platform accessible to medical personnel. The project involves the development of the entire pipeline, from data generation and preprocessing, to model implementation and evaluation, and the development of a web platform.

To evaluate the algorithm's performance, both artificial and real datasets were used. The artificial dataset was generated to simulate a realistic diagnostic scenario, including relevant clinical variables. The algorithm achieved an accuracy of 85% on this dataset, demonstrating its ability to identify correlations and variable importances in a controlled environment.

On the real dataset related to lung cancer, the algorithm achieved an accuracy of 94%, validating its effectiveness in an uncontrolled environment. In the IHelp project, in collaboration with TMU and Siemens, the algorithm achieved an accuracy of over 90% on a confidential dataset related to pancreatic cancer, confirming its practical utility. Additionally, a web platform was implemented, allowing physicians to access and utilize the findings.

The results highlight the robustness and versatility of the Random Forest algorithm, suggesting future development directions such as integrating multi-omic data, developing a mobile application, enhancing real-time monitoring, and automating the diagnostic process.

Bibliografie

- [1] Aloa, „User Interface Design for Healthcare Applications: A+ Guide”, în Aloa (2022), Accesat în Mai 2024, URL: <https://www.aloa.co/user-interface-design-for-healthcare-applications>.
- [2] Angular Team, Angular Architecture, Accesat în 12 Martie, 2024, URL: <https://angular.io/guide/architecture>.
- [3] Angular Team, Angular Documentation, Accesat în 12 Martie, 2024, URL: <https://angular.io/docs>.
- [4] Angular Team, What is Angular, Accesat în 12 Martie, 2024, URL: <https://angular.io/guide/what-is-angular>.
- [5] Auth0, „What Are Refresh Tokens and How to Use Them Securely”, în (2024), Accesat în Martie 2024, URL: <https://auth0.com/docs/tokens/refresh-tokens>.
- [6] M. Badawy, N. Ramadan și H.A. Hefny, „Healthcare predictive analytics using machine learning and deep learning techniques: a survey”, în Journal of Electrical Systems and Information Technology 10 (2023), Accesat în Martie 2024, p. 40, URL: <https://doi.org/10.1186/s43067-023-00108-y>.
- [7] Vic Barnett și Toby Lewis, „Outliers in statistical data”, în Wiley Series in Probability and Mathematical Statistics (1974), Accesat în Iunie 2024.
- [8] Sarah Mad Bhat, Lung Cancer Dataset, Accesat în Iunie 2024, 2021, URL: <https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer>.
- [9] Leo Breiman, „Random forests”, în Machine learning 45.1 (2001), Accesat în Iunie 2024, pp. 5–32.
- [10] Jason Brownlee, „Ordinal and One-Hot Encodings for Categorical Data”, în Machine Learning Mastery (2019), Accesat în Martie 2024, URL: <https://machinelearningmastery.com/ordinal-and-one-hot-encodings-for-categorical-data/>.
- [11] Jason Brownlee, „Why One-Hot Encode Data in Machine Learning?”, în Machine Learning Mastery (2018), Accesat în Martie 2024, URL: <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>.

- [12] R. Chandramouli și N. Memon, „Analysis of LSB Based Image Steganography Techniques”, în Proceedings of the International Conference on Image Processing, vol. 3, Accesat în Iunie 2024, IEEE, 2001, pp. 1019–1022.
- [13] Hojin Choi și Seog Seo, „Optimization of PBKDF2 using HMAC-SHA2 and HMAC-LSH Families in CPU Environment”, în IEEE Access PP (Mar. 2021), Accesat în Martie 2024, pp. 1–1.
- [14] Artur Costa, „Improving Storage and Read Performance for Free: Flat vs Structured Schemas”, în MongoDB Developer Center (Ian. 2024), Accesat în Martie 2024, URL: <https://www.mongodb.com/developer/products/mongodb/improving-storage-read-performance-free-flat-vs-structured-schemas/>.
- [15] Richard D Cutler et al., „Random forests for classification in ecology”, în Ecology 88.11 (2007), Accesat în Iunie 2024, pp. 2783–2792.
- [16] Stephanie Glen, „Spearman Rank Correlation (Spearman’s Rho): Definition and How to Calculate it”, în (2023), Accesat în Iunie 2024, URL: <https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/spearman-rank-correlation-definition-calculate/>.
- [17] Jiawei Han, Micheline Kamber și Jian Pei, Data Mining: Concepts and Techniques, Accesat în Iunie 2024, Elsevier, 2011.
- [18] Tin Kam Ho, „Random decision forests”, în Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, Accesat în Iunie 2024, IEEE, 1995, pp. 278–282.
- [19] Gareth James et al., An introduction to statistical learning, vol. 112, Accesat în Iunie 2024, Springer, 2013.
- [20] William H Kruskal și W Allen Wallis, „Use of ranks in one-criterion variance analysis”, în Journal of the American Statistical Association 47.260 (1952), Accesat în Iunie 2024, pp. 583–621.
- [21] Vladimir I. Levenshtein, „Binary codes capable of correcting deletions, insertions, and reversals”, în Soviet physics doklady 10.8 (1966), Accesat în Iunie 2024, pp. 707–710.
- [22] Andy Liaw și Matthew Wiener, „Classification and Regression by randomForest”, în R news 2.3 (2002), Accesat in Iunie 2024, pp. 18–22.
- [23] Ana Lourenço și Isabel N. Figueiredo, „Random forests for lung cancer detection”, în Computer Methods and Programs in Biomedicine 159 (2018), Accesat în Iunie 2024, pp. 105–115.

- [24] Scott M. Lundberg și Su-In Lee, „A Unified Approach to Interpreting Model Predictions”, în *Advances in Neural Information Processing Systems*, Accesat în Iunie 2024, 2017, URL: <https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html>.
- [25] George Manias et al., „Advanced Data Processing of Pancreatic Cancer Data Integrating Ontologies and Machine Learning Techniques to Create Holistic Health Records”, în *Sensors* 24.5 (2024), Accesat în Iunie 2024, p. 2072, DOI: 10.3390/sen24052072, URL: <https://www.mdpi.com/1424-8220/24/5/2072>.
- [26] Microsoft, How to guide for C#, Accesat în 15 Martie, 2024, URL: <https://learn.microsoft.com/en-us/dotnet/csharp/how-to/>.
- [27] Microsoft, Introduction to ASP.NET Core, Accesat în 15 Martie, 2024, URL: <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0>.
- [28] MinIO High Performance Object Storage for Kubernetes, <https://min.io/docs/minio/kubernetes/upstream/>, Accesat în Martie 2024.
- [29] Suraj V. Nath, „Crime pattern detection using data mining”, în 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, Accesat în Iunie 2024, IEEE, 2006, pp. 41–44.
- [30] David M W Powers, „Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”, în *Journal of Machine Learning Technologies* 2.1 (2011), Accesat în Iunie 2024, pp. 37–63.
- [31] Python Software Foundation, Python Blurb, Accesat în 12 Martie, 2024, URL: <https://www.python.org/doc/essays/blurb/>.
- [32] Fakhitah Ridzuan și Wan Mohd Nazmee Wan Zainon, „A Review on Data Cleansing Methods for Big Data”, în *Procedia Computer Science* 161 (2019), Accesat în Martie 2024, pp. 731–738, ISSN: 1877-0509, DOI: <https://doi.org/10.1016/j.procs.2019.11.177>, URL: <https://www.sciencedirect.com/science/article/pii/S1877050919318885>.
- [33] J. Sidey-Gibbons și C. Sidey-Gibbons, „Machine learning in medicine: a practical introduction”, în *BMC Medical Research Methodology* 19 (2019), Accesat în Martie 2024, p. 64, URL: <https://doi.org/10.1186/s12874-019-0681-4>.

- [34] Simplilearn, What is FastAPI, Accesat în 15 Martie, 2024, URL: https://www.simplilearn.com/what-is-fastapi-article?utm_source=frs_article_page&utm_medium=top_share_option&utm_campaign=frs_copy_share_icon.
- [35] Artsyl Technologies, „The Importance of Accuracy in Machine Learning: A Comprehensive Guide”, în Artsyl Tech Journal (2023), Accesat în Iunie 2024, URL: <https://www.artsyltech.com>.
- [36] Eric Xing, Michael Jordan și Richard Karp, „Feature Selection for High-Dimensional Genomic Microarray Data”, în Proc 18th International Conf on Machine Learning (Mai 2001), Accesat în Iunie 2024.