Documentație finală

Proiect software ce vizează analiza experimentală a unui set de date medical

Realizat de ↓

Nume Student	*Grupa*
1. Avramescu Andrei	1411B
2. Ciobanu Maria-Denisa	1411B
3. Fortoeș Codrin	1411B
4. Hrițcu Petronela-Adelina	1411B
5. Lupu George	1411B
6. Musteață Raluca-Elena	1411B
7. Popa Șerban Alexandru	1411B
8. Tăbușca Codrina	1411B

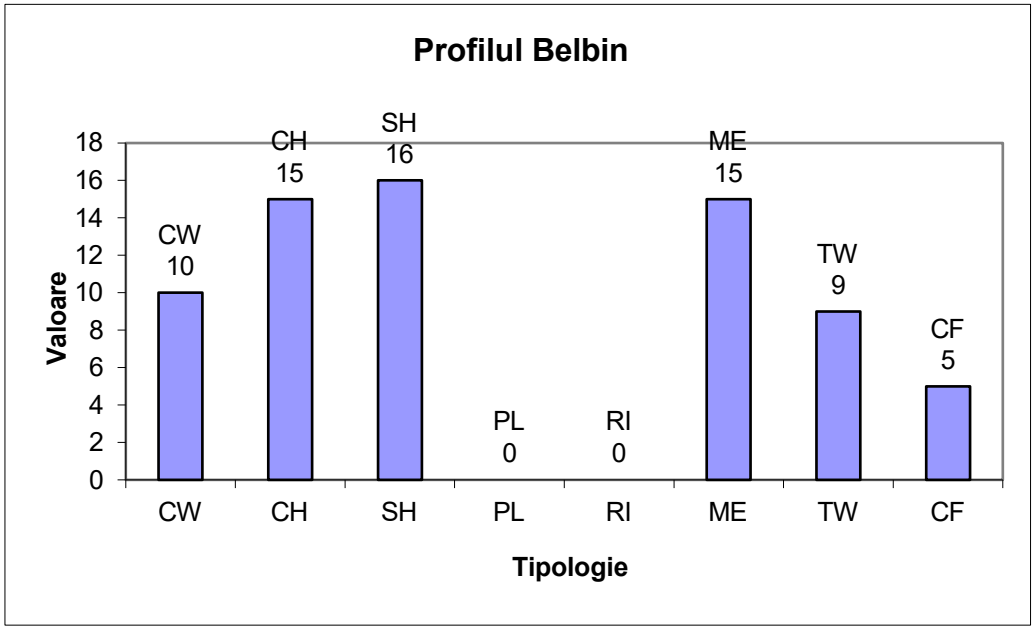
Cuprins

1. Etapa_1 : Organizarea echipei

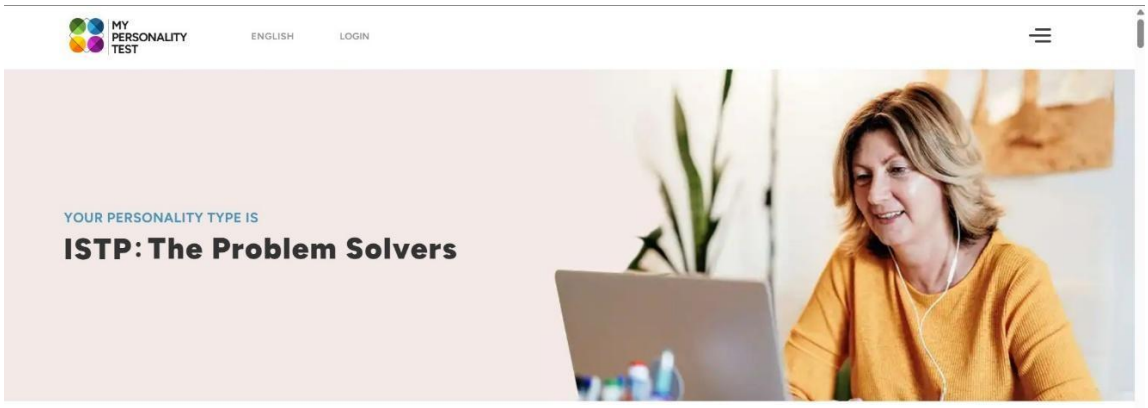
1.1 Rezultatele testelor :

1) Avramescu Andrei *(Manager) :*

- *rezultate test Belbin :*

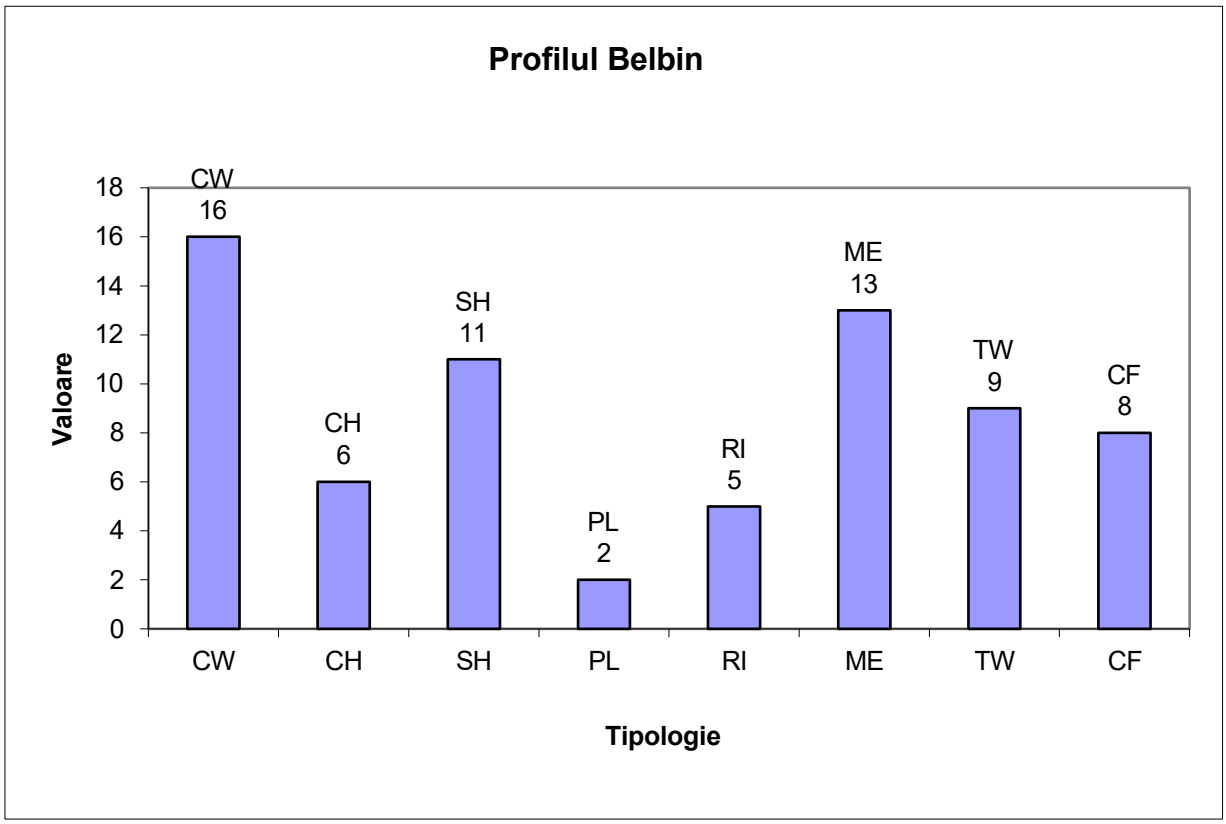


- *rezultate test personalitate :*

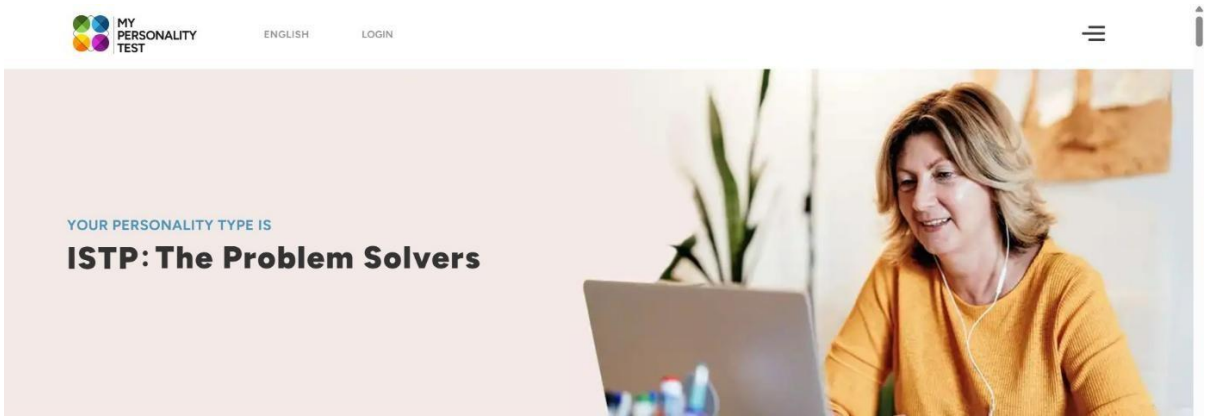


2) Ciobanu Maria-Denisa :

- *rezultate test Belbin :*

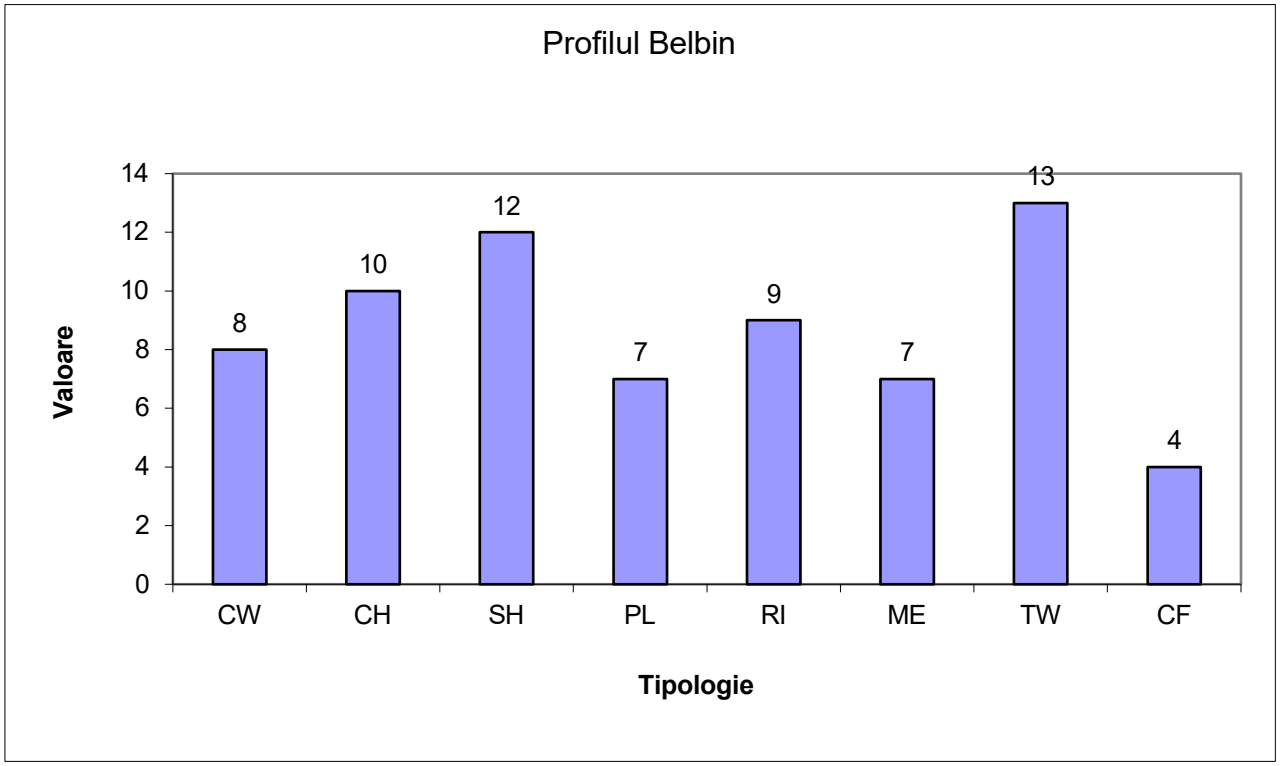


- *rezultate test personalitate :*

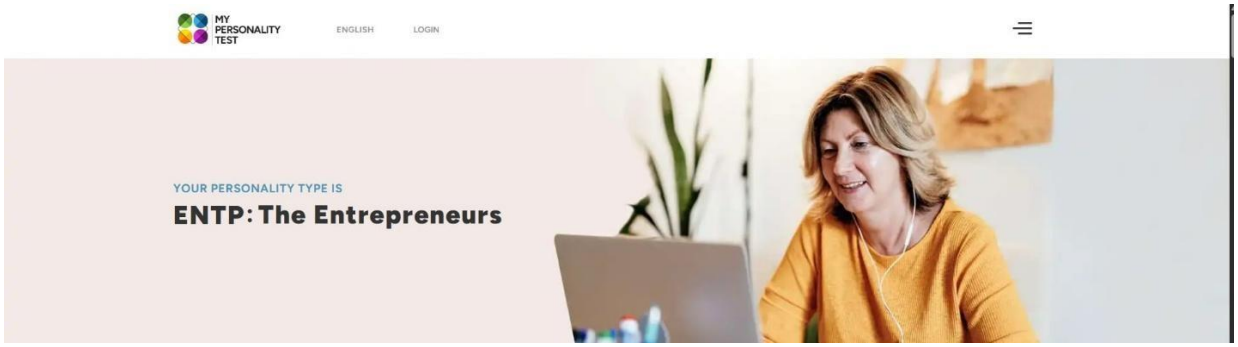


3) Fortoeș Codrin :

- *rezultate test Belbin :*

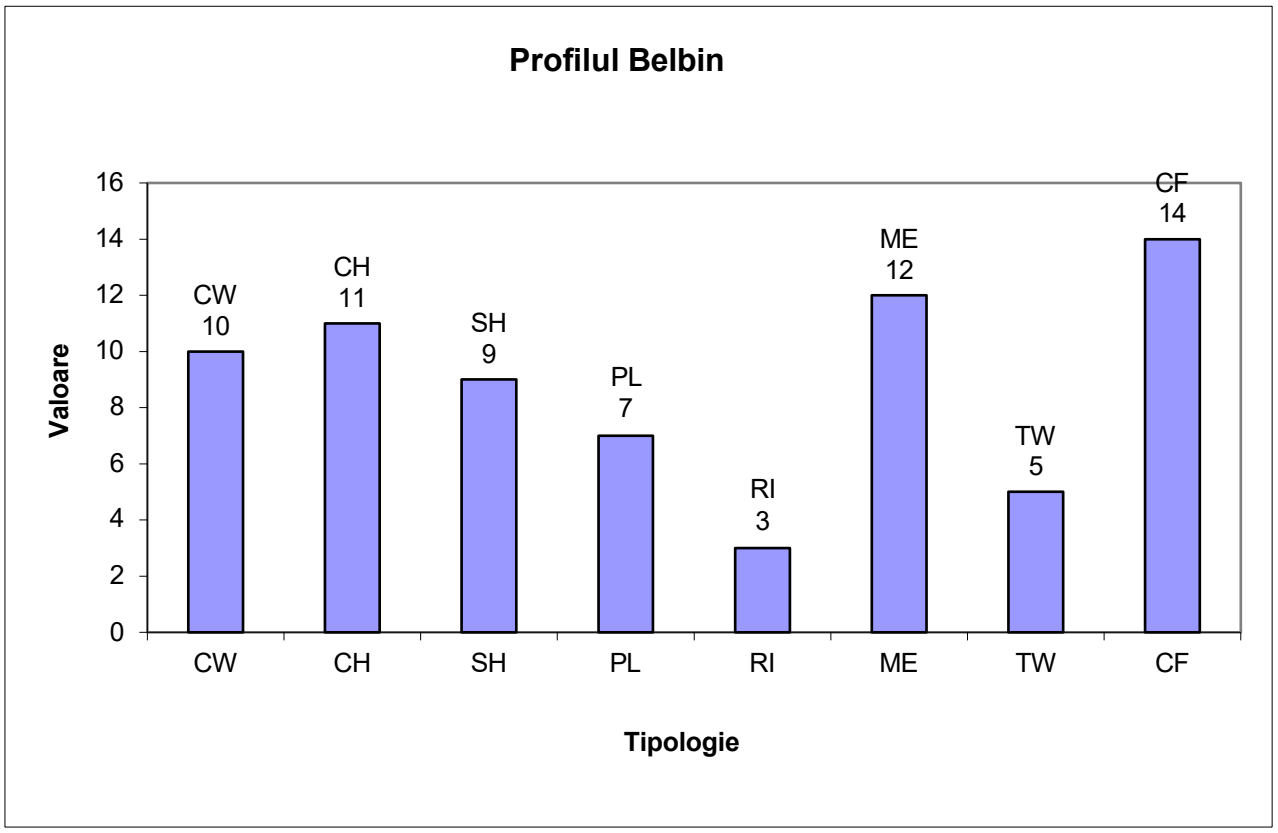


- *rezultate test personalitate :*



4) Hrițcu Petronela-Adelina :

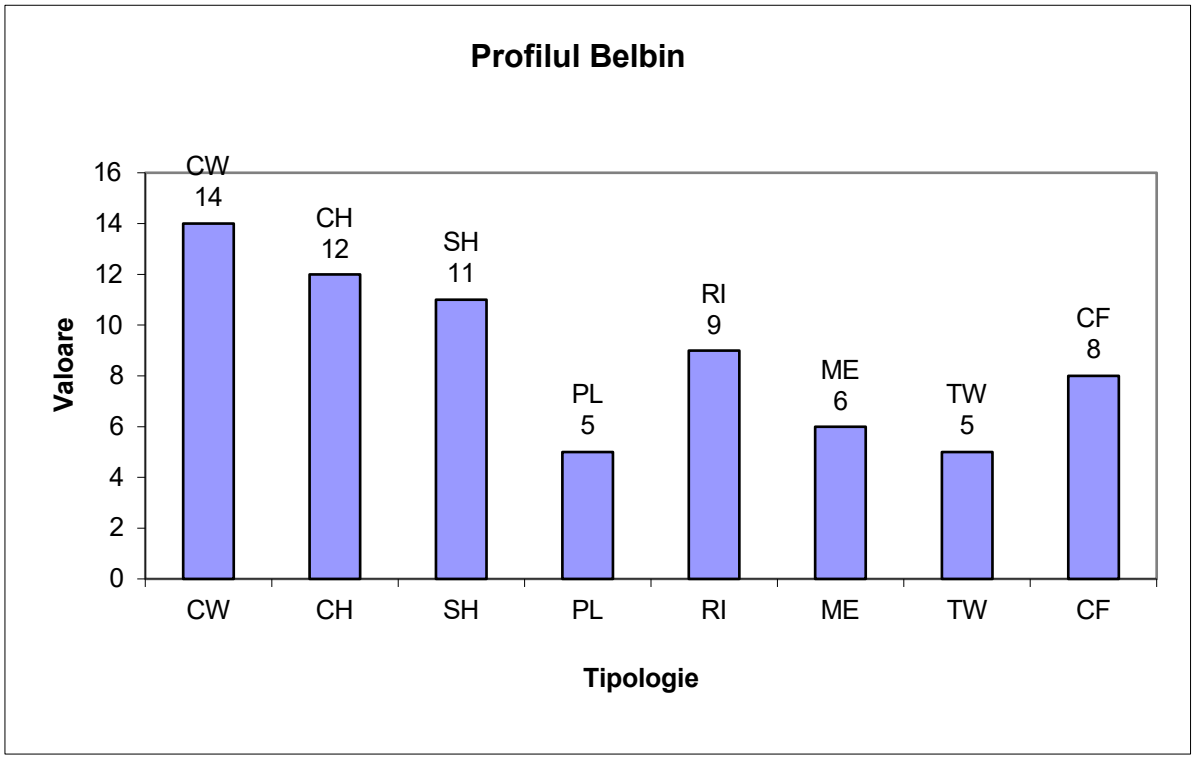
- *rezultate test Belbin :*



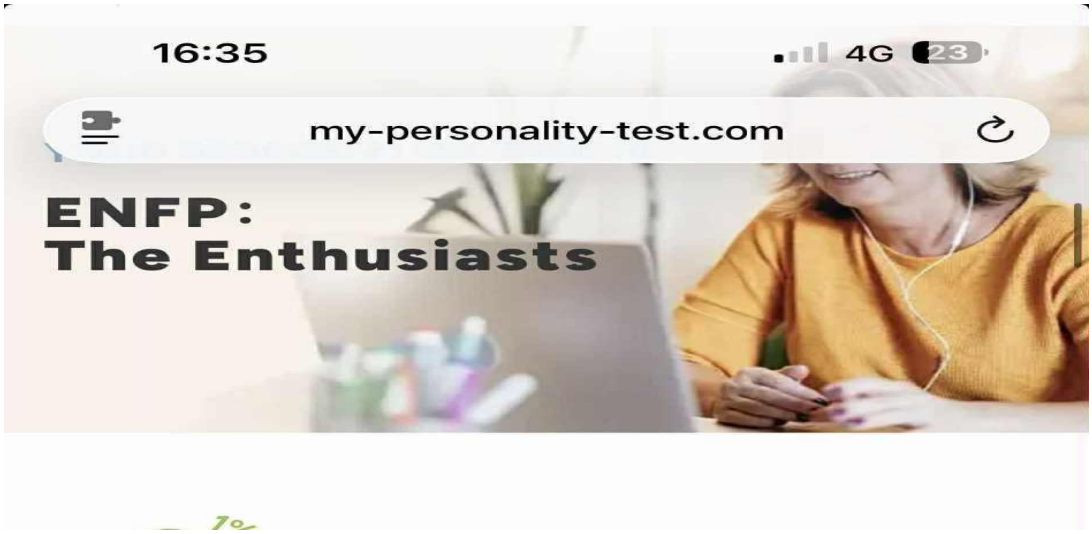
- *rezultate test personalitate :*

5) Lupu George :

- *rezultate test Belbin :*

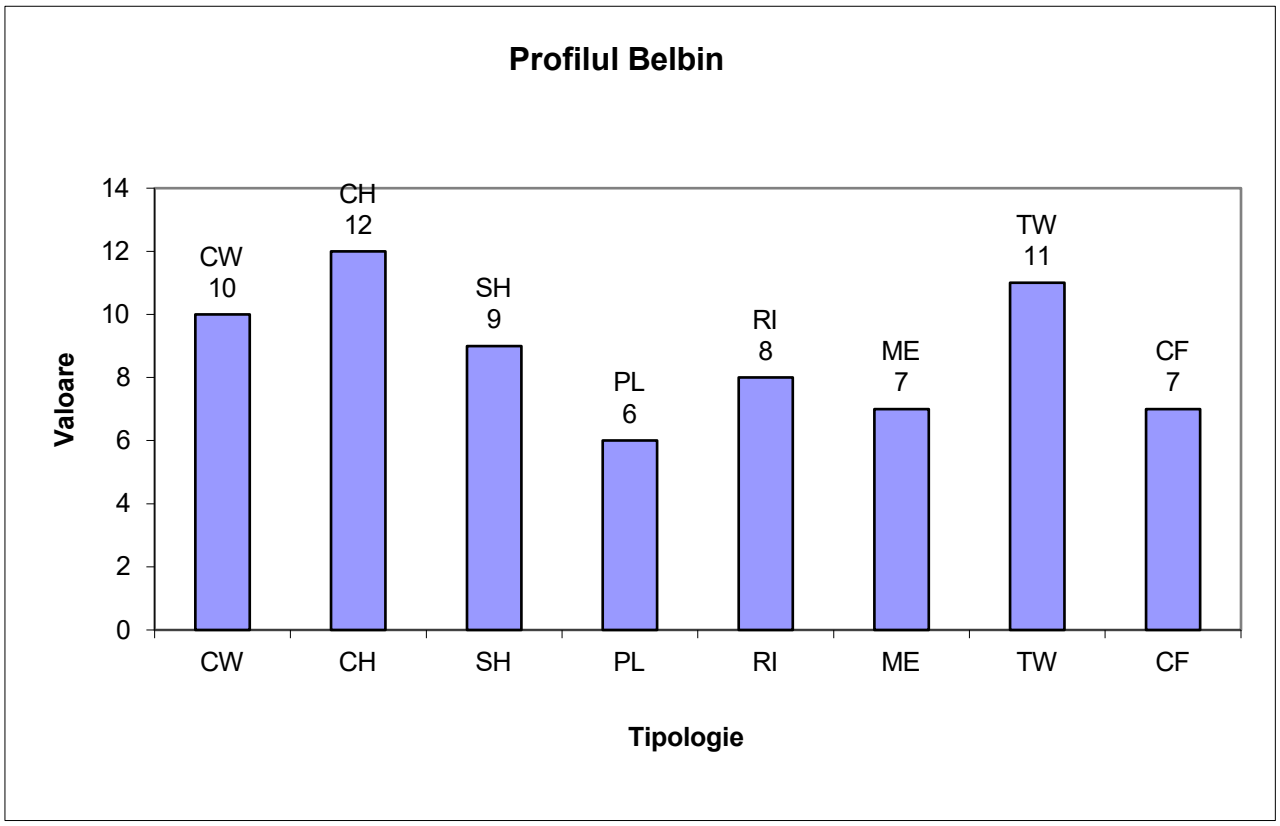


- *rezultate test personalitate :*

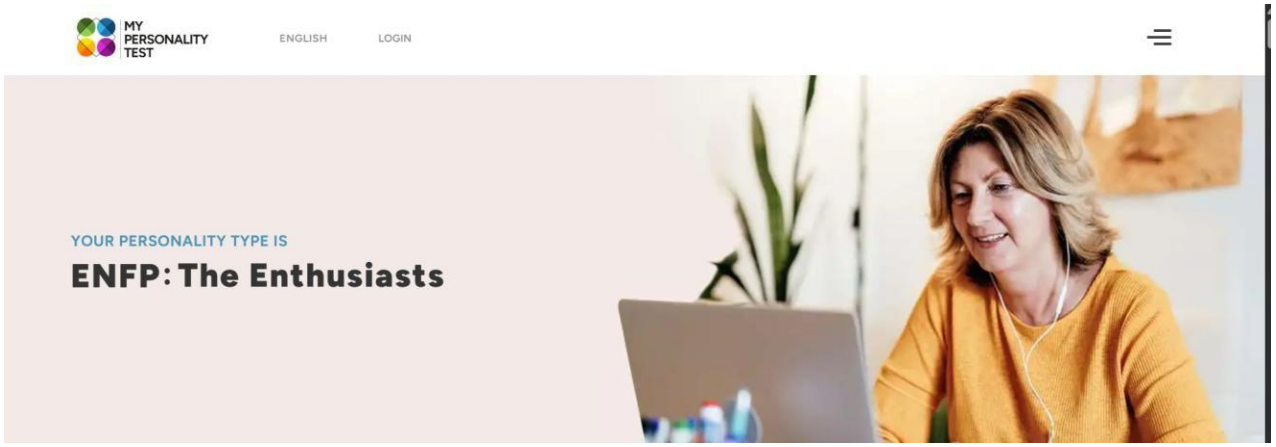


6) Musteață Raluca-Elena :

- *rezultate test Belbin :*

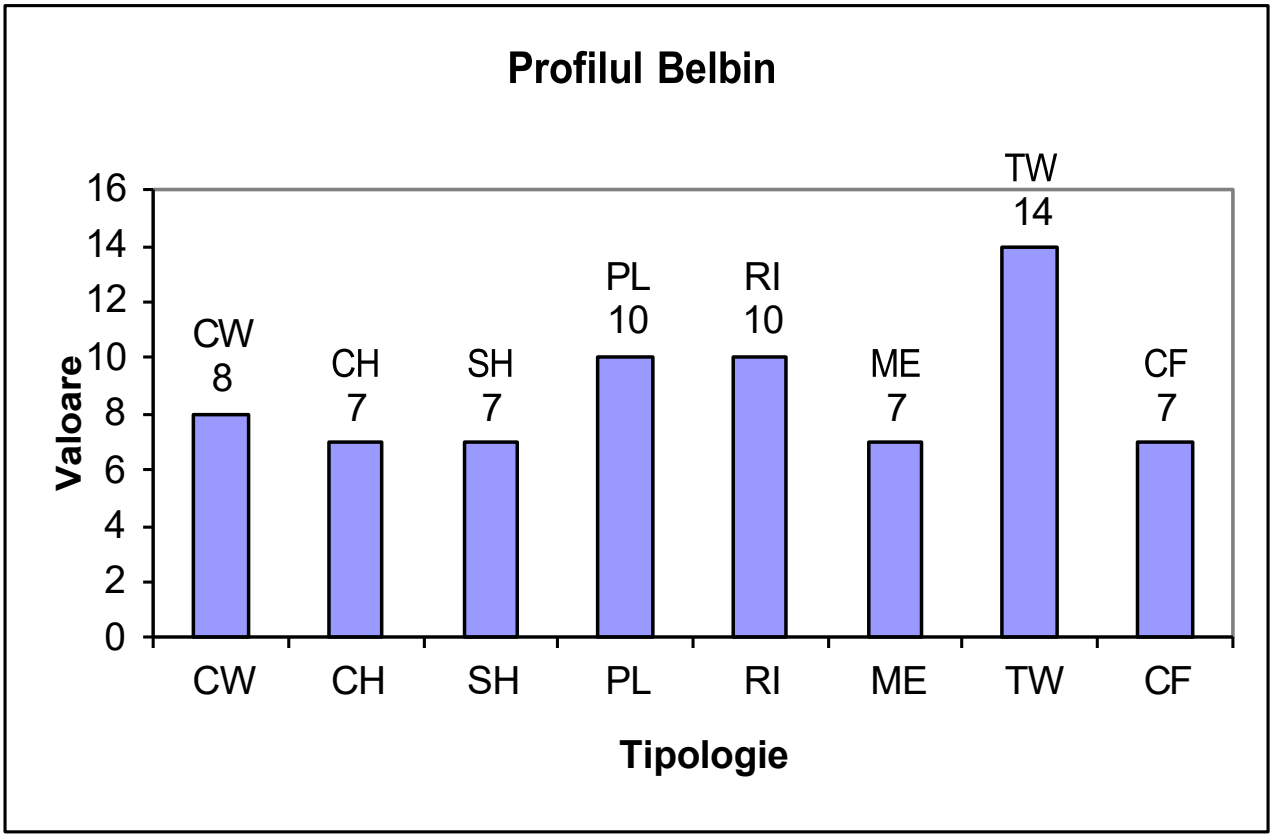


- *rezultate test personalitate :*

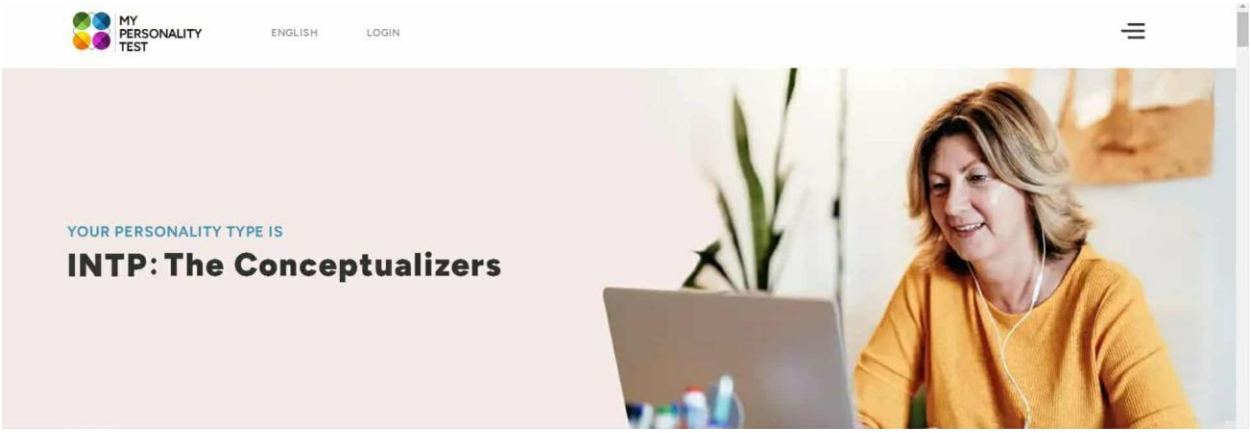


7) Popa Șerban Alexandru :

- *rezultate test Belbin :*

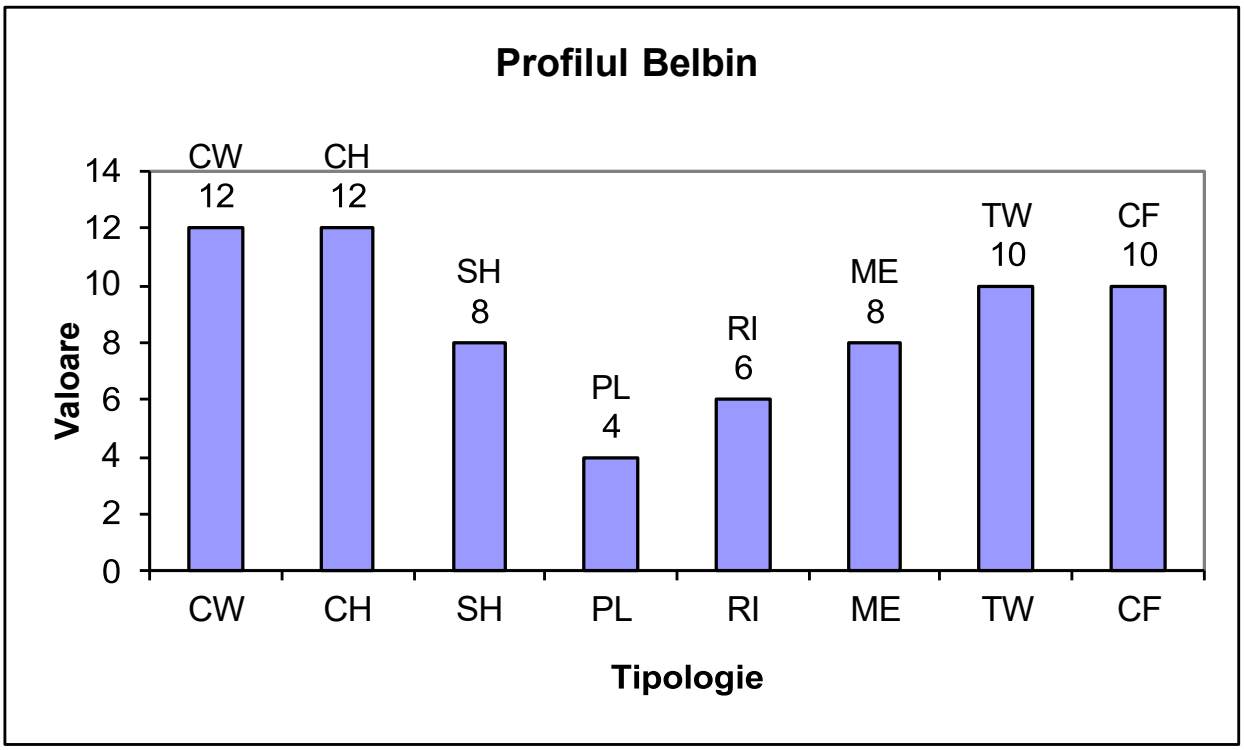


- *rezultate test personalitate :*

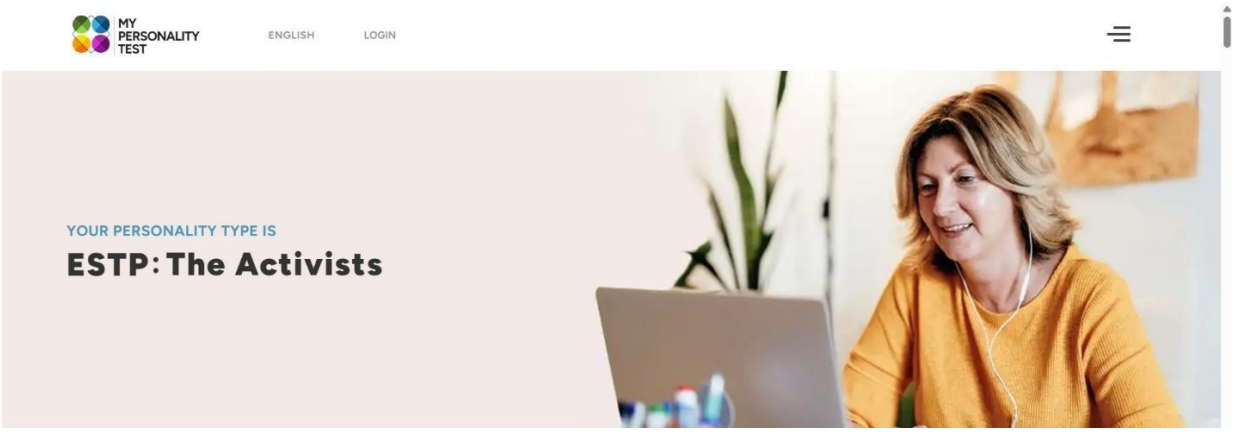


8) Tăbușca Codrina :

- *rezultate test Belbin :*



- *rezultate test personalitate :*

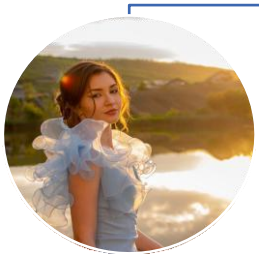


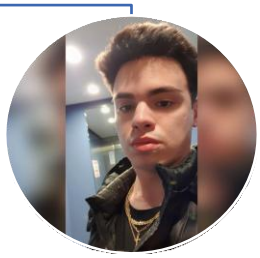
1.2 Schema de structurare a echipei :

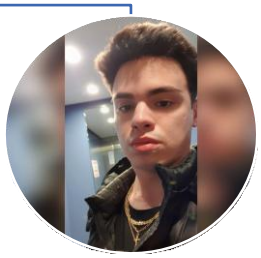
Avramescu Andrei - *Manager*

Lupu George – *Lead Developer*

Hrițcu Petronela-Adelina – *Lead QA*







Ciobanu Maria-Denisa

Tăbușca Codrina

Popa Șerban Alexandru

- ***Developers***

Musteață Raluca-Elena

Fortoeș Codrin

Popa Șerban Alexandru

- ***Quality Engineers***

1.3 Rolul, funcția ocupată și atribuțiile fiecărui membru în cadrul echipei (rolurile membrilor au fost atribuite pe baza testelor Belbin) :

- ***Avramescu Andrei***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Modelator***
    2) ***Coordonator***
    3) ***Monitor/Evaluator***
  - ***Rol in echipa : Coordonator.***
  - ***Funcția ocupată în echipă : Manager***
  - ***Atribuții :***
    - ***organizarea echipei***
    - ***împărțirea sarcinilor astfel încât fiecare membru să își folosească talentul și calitățile la potențialul maxim***

- ***Ciobanu Maria-Denisa :***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Finalizator***
    2) ***Monitor/Evaluator***
    3) ***Modelator***
  - ***Rol in echipa : Monitor/Evaluator.***
  - ***Funcția ocupată în echipă : Developer***
  - ***Atribuții :***
    - ***analizarea ideilor și a obiectivelor pe partea de software într-un mod cât mai perspicace***
    - ***evaluarea tuturor sugestiilor primite***
- ***Fortoeș Codrin :***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Lucrător al echipei***
    2) ***Modelator***

3) ***Coordonator/Investigator de resurse***

- ***Rol in echipa : Investigator de resurse.***
- ***Funcția ocupată în echipă : Quality Engineer***
- ***Atribuții :***
  - ***explorarea a cât mai multor idei pentru testarea completă a produsului***
  - ***păstrarea a unui nivel de entuziasm cât mai mare în echipă***

- ***Hrițcu Petronela-Adelina :***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Implementator***
    2) ***Monitor/Evaluator***
    3) ***Coordonator***
  - ***Rol in echipa : Implementator.***
  - ***Funcția ocupată în echipă : Lead QA***
  - ***Atribuții :***
    - ***transpunerea planurilor primite de la manager și lead developer în scheme cât mai concise***
    - ***folosirea a unei abordări sistematice care se conformează indicațiilor primite de la manager***

- ***Lupu George :***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Finalizator***
    2) ***Coordonator***
    3) ***Modelator***
  - ***Rol in echipa : Modelator.***
  - ***Funcția ocupată în echipă : Lead Developer***
  - ***Atribuții :***
    - ***modelarea activității echipei proprii***
    - ***producerea urgentarea acțiunilor atunci când este cazul necesar.***

- ***Musteață Raluca-Elena :***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Coordonator***
    2) ***Lucrător al echipei***
    3) ***Finalizator***
  - ***Rol in echipa : Lucrător al echipei.***
  - ***Funcția ocupată în echipă : Quality Engineer***
  - ***Atribuții :***
    - ***promovarea unui mod de lucru cooperant și eficient prin sprijinirea celorlalți***
    - ***punerea în evidență a propriilor calități (flexibilitatea, adaptibilitatea, sociabilitatea etc.) într-un mod cât mai valoros în perioade de criză***
- ***Popa Șerban Alexandru :***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Lucrător al echipei***
    2) ***Inovator***
    3) ***Investigator de resurse***
  - ***Rol in echipa : Inovator.***
  - ***Funcția ocupată în echipă : Developer & Quality Engineer***
  - ***Atribuții :***
    - ***generarea a cât mai multor idei pentru partea de software, cât și pe partea de testing***
    - ***furnizarea a cât mai multor soluții alternative în rezolvarea problemelor***
- ***Tăbușca Codrina :***
  - ***Cele mai bune roluri rezultate în urma testelor :***
    1) ***Finalizator***
    2) ***Coordonator***
    3) ***Lucrător al echipei / Implementator***
  - ***Rol in echipa : Finalizator.***

- ***Funcția ocupată în echipă : Developer***
- ***Atribuții :***
    - ***verificarea proiectului la final, asigurându-se ca nimic nu a fost scapat din vedere***
    - ***menținerea unei atenții deosebite pentru detalii pe tot parcursul proiectului.***

2.Etapa_2 : Documentul de specificații sofware [SRS]

2.1 Scopul documentului

Acest document este destinat să descrie cu exactitate capabilitățile proiectului software pentru analiza și clasificarea datelor referitoare la Hepatita, utilizând algoritmul Random Forest (RF) în mediul Matlab.

Clarifică specificațiile proiectului, obiectivele și constrângerile, ajutând la înțelegerea modului în care produsul va îndeplini cerințele funcționale și nefuncționale, facilitând astfel o implementare precisă și o evaluare corectă a performanței în procesarea datelor (a biomarkerilor) și clasificarea rezultatelor.

2.2 Descriere generală

*Această aplicație software, realizată în **Matlab**, este destinată clasificării pacienților cu hepatită utilizând setul de date **Hepatitis (UCI ML Repository),** care conține biomarkeri și categorii de diagnostic.*

Proiectul implică:

- ***Încărcarea și Curățarea** setului de date, verificând și corectând valorile lipsă.*
- ***Analiza Statistică** a variabilelor de intrare (media, dispersia, min/max) pentru detectarea anomaliilor.*
- *Construirea a diverse **Partiții** ale seturilor de date (80%-20%, 70%-30%, 60%-40%, 50%-50%).*
- *Clasificarea comparativă cu RF și SVM*
- ***Evidențierea selecției de trăsături relevante și analiza avantajelor/dezavantajelor abordării.***

2.3 Contextul și motivația implementării

2.3.1 Situația curentă

Setul de date conține valori de laborator și date demografice. Datele brute conțin valori lipsă și potențiale anomalii care pot afecta rezultatele analizei. Fără o preprocesare adecvată și un model de clasificare eficient, aceste date nu pot fi utilizate corespunzător pentru a sprijini clasificarea sau diagnosticarea bolii hepatice.

2.3.2 Scopul produsului

Algoritmul software are ca scop analiza experimentală și clasificarea datelor medicale, utilizând tehnici de învățare automată (ML) pentru a sprijini luarea deciziilor. Acesta va permite preprocesarea datelor brute, identificarea trăsăturilor relevante și aplicarea modelului Random Forest pentru a oferi predicții.

2.3.3 Contextul produsului și motivarea implementării

Bolile hepatice reprezintă o problemă majoră. Diagnosticarea corectă în timp util este crucială. Aplicațiile bazate pe ML pot analiza rapid volume mari de date, identificând tipare care pot sprijini deciziile medicale. Motivarea principală ***este îmbunătățirea acurateței și vitezei clasificării*** *pacienților pe baza biomarkerilor existenți, utilizând un algoritm robust precum Random Forest.*

2.3.4 Beneficii

- ***Diagnostice îmbunătățite:*** *Clasificarea datelor medicale cu Random Forest oferă o diagnosticare mai rapidă și mai precisă.*
- ***Costuri și timp de analiză reduse:*** *Utilizarea algoritmilor automați reduce semnificativ timpul necesar pentru a obține o predicție.*
- ***Validarea și analiza datelor:*** *Analiza statistică (medii, dispersii, valori minime și maxime) oferă o imagine de ansamblu asupra caracteristicilor setului de date.*

2.4 Specificații Funcționale

2.4.1 Actori

2.4.1.1 USER

Utilizatorul este actorul principal al sistemului, fiind persoana care inițiază și execută întregul flux de analiză, de la încărcarea datelor până la evaluarea finală a modelului.

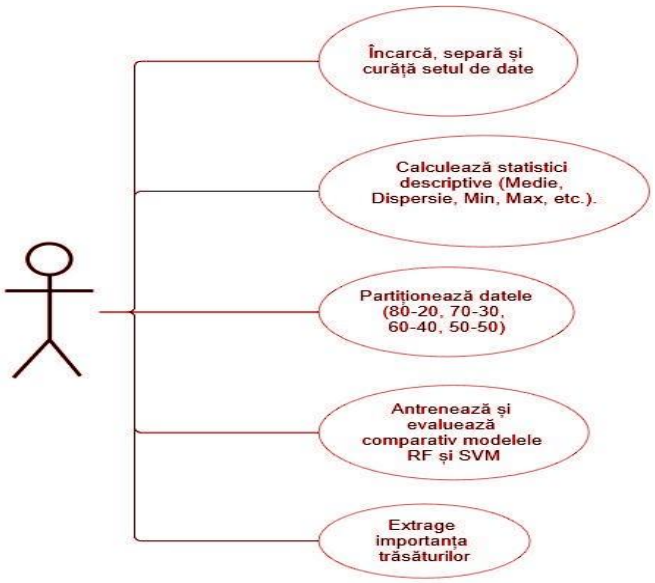
Utilizatorul realizează oricare dintre următoarele operațiuni:

- Încarcă, separă și curăță Setul de Date.
- Calculează statistici descriptive (Medie, Dispersie, Min, Max, etc.).
- Partiționează datele (80-20, 70-30, 60-40, 50-50).
- Antrenează și evaluează comparativ modelele RF și SVM
- Extrage importanța trăsăturilor.

2.4.1.2 System Boundary

Limita sistemului este definită de mediul software care contine și execută toate modulele necesare pentru preprocesare, analiză statică, modelare (RF și SVM) și evaluarea performanței. Sistemul primește ca intrare setul de date Hepatitis și produce ca ieșire rezultate statistice, tabelele de performanță și vizualizările necesare.

2.4.2 Diagrama cazurilor de utilizare



2.4.3 Descrierea cazurilor de Utilizare (USE-CASE)

2.4.3.1 Încarcă,separă și curăță Setul de Date.

Nume	Pregătirea și Curățarea Datelor
Descriere	Operațiune care încarcă setul de date Hepatitis, separă variabilele X și Y, și aplică imputarea valorilor lipsă.
Prioritate	Esențială
Trigger	Utilizatorul inițiază scriptul de preprocesare.
Precondiție	Setul de date Hepatitis este accesibil.
Flux Basic	1. Sistemul încarcă datele și le mapeaza. 2. Sistemul separă X de Y 3. Acesta aplică imputarea cu mediană pentru variabilele numerice și imputarea cu modul pentru variabilele categorice. 4. Sistemul va confirma că nu mai sunt valori lipsă.
Cale alternativă	Se trece la calcularea statisticilor.
Postcondiție	Vectorii X și Y sunt completi și gata de analiză.
Cale Excepție	Dacă fișierul de date nu poate fi citit, sistemul returnează un mesaj de eroare critică și oprește execuția.

2.4.3.2 Calculează statistici descriptive (Medie, Dispersie, Min, Max, etc.)

Nume	Calculul Statistic Descriptiv
Descriere	Operațiune care calculează indicatorii de bază (Medie, Dispersie, Min, Max) și ilustrează caracteristicile datelor cu metode suplimentare.
Prioritate	Esențială
Trigger	Utilizatorul execută modulul de analiză statistică.
Precondiție	Setul de date X este complet curat și format din numere.
Flux Basic	1. Sistemul itereaza prin fiecare trăsătură din X. 2. Sistemul calculează Medie, Dispersie, Min și Max. 3. Sistemul afișează rezultatele într-un tabel 4. Sistemul calculeaza Matricea de Corelatie intrea toate trasaturile. 5. Sistemul genereaza o vizualizare a corelatiei.
Cale alternativă	Se trece la partiționarea datelor.
Postconditie	Tabelul statistic și toate vizualizările descriptive sunt generate și afișate.
Cale Excepție	Dacă o trăsătură numerică nu a fost convertită corect sistemul returnează un avertisment și trece la următoarea trăsătură.

2.4.3.3 Partiționează datele (80-20, 70-30, 60-40, 50-50)

Nume	Partiționarea Stratificată a Datelor
Descriere	Operațiune care creează cele patru seturi de antrenare-testare solicitate (80-20, 70-30, 60-40, 50-50) și verifică distribuția clasei țintă.
Prioritate	Esențială
Trigger	Utilizatorul inițiază funcția de partiționare a datelor.
Precondiție	Setul de date (X, Y) este disponibil și curat.
Flux Basic	1. Sistemul preia lista de proporții de partiționare (80/20, 70/30, 60/40, 50/50). 2. Pentru fiecare proporție, Sistemul aplică funcția de partiționare aleatoare stratificată. 3. Sistemul afișează un tabel de verificare a distribuției claselor.
Cale alternativă	Se trece la antrenare.
Postcondiție	Cele 4 perechi de seturi sunt generate și stocate.
Cale Excepție	Dacă o proporție de partiționare este invalidă (e.g., 90-10, 10-90), Sistemul returnează un mesaj de eroare și ignoră acea proporție.

2.4.3.4 Calculează statistici descriptive (Medie, Dispersie, Min, Max, etc.)

Nume	Calculul Statistic Descriptiv
Descriere	Operațiune care calculează indicatorii de bază (Medie, Dispersie, Min, Max) și ilustrează caracteristicile datelor cu metode suplimentare.
Prioritate	Esențială
Trigger	Utilizatorul execută modulul de analiză statistică.
Precondiție	Setul de date X este complet curat și format din numere.
Flux Basic	1. Sistemul itereaza prin fiecare trăsătură din X. 2. Sistemul calculează Medie, Dispersie, Min și Max. 3. Sistemul afișează rezultatele într-un tabel 4. Sistemul calculează Matricea de Corelație între toate trăsăturile. 5. Sistemul generează o vizualizare a corelației.
Cale alternativă	Se trece la partiționarea datelor.
Postcondiție	Tabelul statistic și toate vizualizările descriptive sunt generate și afișate.
Cale Excepție	Dacă o trăsătură numerică nu a fost convertită corect sistemul returnează un avertisment și trece la următoarea trăsătură.

2.4.3.5 Antrenează și evaluează comparativ modelele RF și SVM

Precondiție	Cele 4 seturi de antrenare/testare sunt disponibile și valide.
Flux Basic	1. Sistemul iterează prin cele 4 partiții. 2. Pentru fiecare, antrenează modelul Random Forest. 3. Pentru fiecare model, face predicții pe setul de testare. 4. Calculează și stochează cele 4 metrici de performanță. 5. Afișează rezultatele într-un tabel comparativ.
Cale alternativă	Se trece la extragerea importanței trăsăturilor.
Postcondiție	Toate metricile de performanță sunt colectate, iar tabelul comparativ RF vs. SVM pentru cele 4 partiții este afișat.
Cale Exepție	Dacă antrenarea unuia dintre modele eșuează din cauza unei erori interne, sistemul returnează un mesaj de eroare pentru modelul eșuat și continuă cu celălalt model.

2.4.3.6 Extrage importanța trăsăturilor.

Nume	Extragerea Importanței Trăsăturilor
Descriere	Operațiune care utilizează modelul Random Forest pentru a determina și vizualiza trăsăturile cele mai relevante pentru predicția rezultatului.
Prioritate	Esențială
Trigger	Sistemul finalizează antrenarea modelului Random Forest (de obicei, pe setul 80/20 sau cel mai bun set).
Precondiție	Cel puțin un model Random Forest a fost antrenat cu succes.
Flux Basic	***1.*** Sistemul extrage scorurile de importanță a trăsăturilor din modelul RF. ***2.*** Sistemul sortează trăsăturile după scorul de importanță ***3.*** Sistemul generează o vizualizare de tip Bar Chart cu trăsăturile clasate ***4.*** Sistemul afișează o analiză textuală care evidențiază avantajele și dezavantajele RF și SVM, bazate pe performanța comparativă și capacitatea de *feature selection*.
Cale alternativă	N/A
Postcondiție	O analiză clară a selecției de trăsături relevante și a avantajelor/dezavantajelor abordării este prezentată.
Cale Excepție	Dacă modelul RF nu a fost antrenat, Sistemul nu poate extrage importanța trăsăturilor și returnează un mesaj de eroare.

2.5 Specificații non-funcționale

2.5.1 Specificațiile interfeței cu utilizatorul

- Aplicația trebuie să fie proiectată pentru a necesita intervenția utilizatorului doar la inițierea procesului.
- Rezultatele finale (tabelele statistice, metricile de performanță și vizualizările) trebuie prezentate direct și concis, fără a necesita prelucrare suplimentară din partea utilizatorului.
- Sistemul va fi implementat într-un mediu de calcul științific, în Matlab,fiind important ca utilizatorul să fie familiar cu limbajul de programare.

2.5.2 Specificațiile de performanță

- Timpul total necesar pentru a rula întregul flux de analiză (de la încărcare la evaluarea finală a tuturor modelelor pe toate partițiile) nu trebuie să fie foarte mare.
- Implementarea algoritmilor va utiliza biblioteci optimizate pentru a asigura viteaza antrenării.

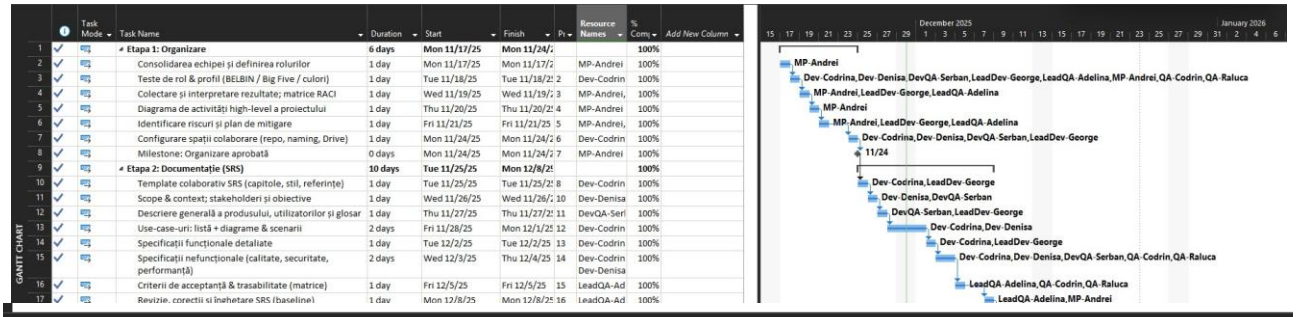
2.5.3 Disponibilitatea și fiabilitatea

- Pentru a asigura rezultate constante, Sistemul trebuie să utilizeze o sămânță aleatorie (random seed) fixă în toate etapele care implică selecția aleatorie (partiționare, inițializarea algoritmilor).
- Toate calculele statistice și metricile de performanță trebuie să fie corecte.
- Sistemul trebuie să gestioneze automat valorile lipsă prin imputare pentru a preveni blocarea execuției sau generarea de rezultate eronate.

2.5.4 Cerințe de securitate

- Codul trebuie sa fie protejat împotriva modificărilor neintenționate care ar putea compromite validitatea rezultatelor.
- Prelucrarea datelor trebuie sa se face într-un mediu sigur.

2.6 Planificarea activităților și progres



	Task Mode	Task Name	Duration	Start	Finish	Pr	Resource Names	% Com	Add New Column
1	✓	**Etapa 1: Organizare**	**6 days**	**Mon 11/17/25**	**Mon 11/24/2**			**100%**	
2	✓	Consolidarea echipei și definirea rolurilor	1 day	Mon 11/17/25	Mon 11/17/2		MP-Andrei	100%	
3	✓	Teste de rol & profil (BELBIN / Big Five / culori)	1 day	Tue 11/18/25	Tue 11/18/2	2	Dev-Codrin	100%	
4	✓	Colectare și interpretare rezultate; matrice RACI	1 day	Wed 11/19/25	Wed 11/19/2	3	MP-Andrei,	100%	
5	✓	Diagrama de activități high-level a proiectului	1 day	Thu 11/20/25	Thu 11/20/2	4	MP-Andrei	100%	
6	✓	Identificare riscuri și plan de mitigare	1 day	Fri 11/21/25	Fri 11/21/25	5	MP-Andrei,	100%	
7	✓	Configurare spații colaborare (repo, naming, Drive)	1 day	Mon 11/24/25	Mon 11/24/2	6	Dev-Codrin	100%	
8	✓	Milestone: Organizare aprobată	0 days	Mon 11/24/25	Mon 11/24/2	7	MP-Andrei	100%	
9	✓	**Etapa 2: Documentație (SRS)**	**10 days**	**Tue 11/25/25**	**Mon 12/8/2**			**100%**	
10	✓	Template colaborativ SRS (capitole, stil, referințe)	1 day	Tue 11/25/25	Tue 11/25/2	8	Dev-Codrin	100%	
11	✓	Scope & context; stakeholderi și obiective	1 day	Wed 11/26/25	Wed 11/26/2	10	Dev-Denisa	100%	
12	✓	Descriere generală a produsului, utilizatorilor și glosar	1 day	Thu 11/27/25	Thu 11/27/2	11	DevQA-Serl	100%	
13	✓	Use-case-uri: listă + diagrame & scenarii	2 days	Fri 11/28/25	Mon 12/1/25	12	Dev-Codrin	100%	
14	✓	Specificații funcționale detaliate	1 day	Tue 12/2/25	Tue 12/2/25	13	Dev-Codrin	100%	
15	✓	Specificații nefuncționale (calitate, securitate, performanță)	2 days	Wed 12/3/25	Thu 12/4/25	14	Dev-Codrin Dev-Denisa	100%	
16	✓	Criterii de acceptanță & trasabilitate (matrice)	1 day	Fri 12/5/25	Fri 12/5/25	15	LeadQA-Ad	100%	
17	✓	Revizie, corectii si înghetare SRS (baseline)	1 day	Mon 12/8/25	Mon 12/8/25	16	LeadQA-Ad	100%	

GANTT CHART

	Resource Name	Type	Material	Initials	Group	Max.	Std. Rate	Ovt.	Cost/Use	Accrue	Base
1	MP-Andrei	Work		M		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard
2	Dev-Denisa	Work		D		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard
3	Dev-Codrina	Work		D		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard
4	QA-Raluca	Work		Q		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard
5	QA-Codrin	Work		Q		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard
6	DevQA-Serban	Work		D		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard
7	LeadQA-Adelina	Work		L		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard
8	LeadDev-George	Work		L		100%	$0.00/hr	$0.00/hr	$0.00	Prorated	Standard

3.Etapa_3 : Documentul de proiectare a soluției aplicației software [SDD]

3.1 Scopul documentului

Scopul acestui document este de a descrie soluția de proiectare software pentru aplicația de analiză și clasificare a setului de date Hepatitis din UCI Machine Learning Repository.

Documentul prezintă structura datelor, arhitectura sistemului, componentele aplicației și modul de interacțiune dintre acestea, constituind un ghid pentru implementarea ulterioară a soluției.

Aplicația are ca scop preprocesarea setului de date, analiza statistică a caracteristicilor, construirea seturilor de antrenare și testare prin diferite partiționări aleatoare, precum și evaluarea comparativă a performanței algoritmilor de clasificare Random Forest (RF) și Support Vector Machine (SVM). De asemenea, soluția permite evidențierea trăsăturilor relevante și analiza avantajelor și dezavantajelor fiecărei metode de clasificare în contextul setului de date studiat.

3.2 Lista de obiective

Obiectivul principal al aplicației este analiza și evaluarea comparativă a performanței algoritmilor de clasificare aplicați asupra setului de date Hepatitis din UCI Machine Learning Repository.

Obiectivele specifice ale aplicației sunt:

- *încărcarea și organizarea setului de date în structuri de date adecvate pentru procesare;*
- *identificarea și tratarea valorilor lipsă sau nedefinite din datele de intrare;*

- *calcularea indicatorilor statistici descriptivi pentru caracteristicile setului de date;*
- *evidențierea distribuțiilor și a eventualelor discrepanțe sau valori izolate;*
- *construirea seturilor de date de antrenare și testare utilizând diferite proporții de partiționare;*
- *antrenarea și evaluarea comparativă a algoritmilor Random Forest și Support Vector Machine;*
- *identificarea trăsăturilor relevante și analiza avantajelor și dezavantajelor fiecărei metode de clasificare.*

3.3 Conținutul documentului

Documentul este structurat în următoarele capitole principale:

● Modelul datelor – descrie structurile de date utilizate, formatele fișierelor și modelul bazei de date pentru gestionarea experimentelor.

● Modelul arhitectural și modelul componentelor – prezintă arhitectura aplicației, șabloanele arhitecturale utilizate și descrierea componentelor.

● Modelul interfeței cu utilizatorul – descrie modul de utilizare al aplicației și succesiunea interacțiunilor.

● Elemente de testare – identifică componentele critice ale sistemului și posibile alternative de proiectare.

3.4 Modelul datelor

3.4.1. Structuri de date globale

*În aplicația proiectată, structura de date globală este reprezentată de o structură centrală de tip **Main**, care are rolul de nucleu al aplicației și de intermediar între modulele funcționale.*

Această structură globală conține datele încărcate din setul de date

Hepatitis *și permite accesul controlat al modulelor aplicației la informațiile necesare procesării. Prin intermediul acestei structuri, modulele de preprocesare, analiză statistică, partiționare și clasificare pot accesa în mod coerent aceleași date, evitând duplicarea informației.*

Structura globală include următoarele elemente principale:

- *matricea de intrare **X**, care conține valorile caracteristicilor pentru fiecare eșantion;*
- *vectorul țintă **Y**, care conține etichetele de clasă asociate eșantioanelor;*
- *parametrii globali ai aplicației, precum seed-ul aleator, tipurile de partiționare utilizate și setările generale ale algoritmilor de clasificare;*
- *referințe către rezultatele intermediare și finale obținute în urma analizei.*

Prin utilizarea acestei structuri de date globale cu rol de intermediar, modulele aplicației pot colabora fără a depinde direct unele de altele, asigurând o separare clară a responsabilităților și o organizare coerentă a datelor procesate.

3.4.2. Structuri de date de legătură

Structurile de date de legătură sunt utilizate pentru a permite comunicarea între modulele aplicației și pentru a transmite informațiile necesare realizării procesărilor specifice fiecărei componente.

*În cadrul aplicației, modulul de preprocesare comunică cu structura globală Main prin intermediul datelor încărcate din setul de date Hepatitis, transmițând matricea de intrare **X** și vectorul țintă **Y**, rezultate în urma separării caracteristicilor și etichetelor de clasă. De asemenea, modulul de preprocesare transmite informații privind pozițiile valorilor lipsă și rezultatele corectării acestora.*

Modulul de analiză statistică comunică cu structura globală Main prin

intermediul indicatorilor calculați pentru variabilele de intrare, precum media, dispersia, valorile minime și maxime, precum și prin intermediul altor indicatori utilizați pentru caracterizarea setului de date. Aceste informații sunt transmise sub forma unor structuri de date agregate, utilizate ulterior pentru raportare.

Modulul de partiționare comunică cu structura globală Main prin transmiterea seturilor de date de antrenare și testare rezultate în urma aplicării partiționărilor aleatoare 80%–20%, 70%–30%, 60%–40% și 50%–50%. Pentru fiecare partiționare sunt transmise atât datele efective, cât și informații privind distribuția claselor în seturile obținute.

Modulul de clasificare comunică cu structura globală Main prin intermediul modelelor de clasificare antrenate (Random Forest și Support Vector Machine) și al predicțiilor generate pe seturile de testare. Rezultatele obținute sunt transmise către evaluare sub forma vectorilor de predicție.

Modulul de evaluare transmite către structura globală Main metricile de performanță calculate pentru fiecare algoritm și fiecare partiționare, precum și matricele de confuzie asociate. Aceste structuri de date de legătură sunt utilizate ulterior pentru compararea performanțelor și pentru generarea rapoartelor finale.

3.4.3 Structuri de date temporare

În cadrul aplicației se utilizează structuri de date temporare pe durata execuției, necesare etapelor de preprocesare, analiză statistică, partiționare și evaluare a clasificatorilor. Aceste structuri au rol de suport pentru calcule și nu sunt persistate după finalizarea rulării.

În etapa de preprocesare, se utilizează structuri temporare precum:

- *măști logice pentru identificarea valorilor lipsă sau nedefinite (ex. pozițiile valorilor NaN / lipsă);*
- *valori intermediare rezultate din imputare sau din transformări aplicate variabilelor (de exemplu, rezultatele conversiilor/encodării variabilelor categorice).*

În etapa de analiză statistică, se utilizează structuri temporare pentru:

● *calculul indicatorilor descriptivi pe variabile (medii, dispersii, minime, maxime);*

● *construirea distribuțiilor și a structurilor intermediare necesare generării reprezentărilor grafice (histograme, boxplot, corelații).*

În etapa de partiționare, se utilizează structuri temporare pentru:

● *generarea indecșilor de antrenare/testare pentru fiecare proporție (80/20, 70/30, 60/40, 50/50);*

● *calculul distribuției clasei în subseturi (train/test).*

În etapa de clasificare și evaluare, se utilizează structuri temporare precum:

● *modele antrenate în memorie pentru fiecare split (RF și SVM) și predicțiile asociate;*

● *matricea de confuzie și valorile intermediare utilizate pentru calculul metricilor (Accuracy, Precision, Recall, F1).*

Aceste structuri de date temporare există doar pe durata execuției unei rulări și sunt eliberate după finalizarea procesării, rezultatele finale fiind păstrate prin mecanismele de salvare (fișiere și/sau baza de date de experimente).

3.4.4 Formatul fișierelor utilizate

*Aplicația utilizează fișiere de tip tabular ca mod de intrare a datelor în sistem. Setul de date utilizat este setul **Hepatitis**, preluat din depozitul **UCI Machine Learning Repository**, și este utilizat sub forma unui fișier în format **CSV** sau **ARFF**.*

Fișierul de intrare conține valorile caracteristicilor asociate fiecărui pacient, precum și variabila țintă ce indică starea pacientului. Datele pot fi de tip numeric sau categoric, iar valorile lipsă sau nedefinite sunt marcate conform specificației setului de date și sunt identificate și tratate în etapa de preprocesare a aplicației.

*În urma execuției aplicației, sunt generate automat fișiere de ieșire ce conțin rezultatele procesărilor efectuate. Acestea includ fișiere de tip **CSV** pentru statisticile descriptive și pentru metricile de performanță obținute în urma clasificării cu algoritmii **Random Forest** și **Support Vector Machine**, precum și fișiere grafice utilizate pentru reprezentarea distribuțiilor și a rezultatelor comparative.*

Fișierele de ieșire sunt utilizate exclusiv în scop de analiză și raportare și nu sunt reutilizate ca date de intrare pentru alte execuții ale aplicației.

3.4.5. Descrierea bazei de date

3.4.5.1. Diagrama schemei bazei de date

*Baza de date are rolul de a stoca informațiile rezultate în urma rulărilor aplicației asupra setului de date **Hepatitis**, astfel încât acestea să poată fi consultate și comparate ulterior. În mod particular, baza de date reține configurațiile de rulare (de exemplu seed-ul aleator și proporțiile de partiționare), modelele antrenate (**Random Forest și Support Vector Machine**), metricile de performanță obținute pentru fiecare partiție și, respectiv, scorurile de importanță ale trăsăturilor extrase din modelul Random Forest.*

*Modelul bazei de date este format din următoarele tabele **interrelaționate**, așa cum este ilustrat în **Figura 1**.*

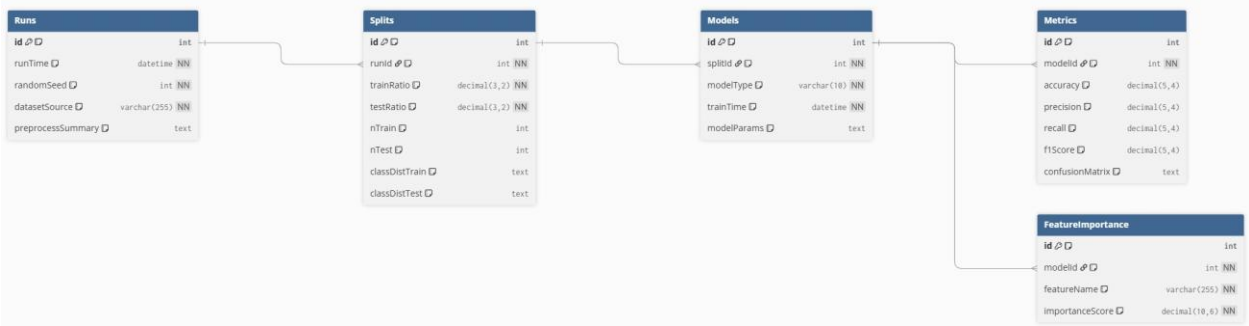


Figura 1. Diagrama schemei bazei de date

3.4.5.2. Descrierea tabelelor

Schema bazei de date este alcătuită din următoarele tabele, utilizate pentru stocarea și corelarea informațiilor rezultate în urma execuțiilor aplicației.

Tabelul Runs

Tabelul Runs reține informațiile generale despre o rulare completă a aplicației. Fiecare înregistrare corespunde unei execuții asupra setului de date Hepatitis.

- ***id –*** *identificator unic al rulării (cheie primară);*
- ***runTime –*** *momentul în care a fost efectuată rularea;*
- ***randomSeed –*** *seed-ul aleator utilizat pentru operațiile stocastice;*
- ***datasetSource –*** *sursa setului de date utilizat;*
- ***preprocessSummary –*** *descriere succintă a etapelor de preprocesare aplicate.*

Tabelul Splits

Tabelul Splits reține informațiile despre partiționările realizate în cadrul unei rulări.

- ***id –*** *identificator unic al partiționării (cheie primară);*
- ***runId –*** *identificatorul rulării asociate (cheie externă către Runs);*
- ***trainRatio –*** *proporția setului de antrenare;*
- ***testRatio –*** *proporția setului de testare;*
- ***nTrain –*** *numărul de eșantioane din setul de antrenare;*
- ***nTest –*** *numărul de eșantioane din setul de testare;*
- ***classDistTrain –*** *distribuția claselor în setul de antrenare;*

● ***classDistTest –*** *distribuția claselor în setul de testare.*

Tabelul Models

Tabelul Models conține informații despre modelele de clasificare antrenate pentru fiecare partiționare.

● ***id –*** *identificator unic al modelului (cheie primară);*

● ***splitId –*** *identificatorul partiționării asociate (cheie externă către Splits);*

● ***modelType –*** *tipul modelului utilizat (Random Forest sau Support Vector Machine);*

● ***trainTime –*** *momentul antrenării modelului;*

● ***modelParams –*** *parametrii utilizați la antrenarea modelului.*

Tabelul Metrics

Tabelul Metrics reține metricile de performanță obținute pentru fiecare model antrenat.

● ***id –*** *identificator unic al setului de metrici (cheie primară);*

● ***modelId –*** *identificatorul modelului asociat (cheie externă către Models);*

● ***accuracy –*** *acuratețea obținută pe setul de test;*

● ***precision*** *– precizia obținută pe setul de test;*

● ***recall –*** *valoarea recall-ului;*

● ***f1Score –*** *scorul F1;*

● ***confusionMatrix –*** *matricea de confuzie asociată modelului.*

Pentru fiecare model există o singură înregistrare corespunzătoare în acest tabel.

Tabelul FeatureImportance

Tabelul FeatureImportance reține scorurile de importanță ale trăsăturilor calculate pe baza modelului Random Forest.

- ***id –*** *identificator unic al înregistrării (cheie primară);*
- ***modelId*** *– identificatorul modelului asociat (cheie externă către Models);*
- ***featureName –*** *denumirea trăsăturii;*
- ***importanceScore –*** *scorul de importanță al trăsăturii.*

Pentru un model de tip Random Forest pot exista mai multe înregistrări în acest tabel, câte una pentru fiecare trăsătură.

Concluzie

Structura bazei de date permite urmărirea completă a rulărilor aplicației, a partiționărilor utilizate, a modelelor antrenate și a performanțelor obținute, facilitând analiza comparativă a rezultatelor.

3.5 Modelul arhitectural și modelul componentelor

3.5.1 Arhitectura sistemului

Arhitectura aplicației este una de tip modular, în care funcționalitățile sunt organizate în componente distincte, fiecare având un rol bine definit în cadrul fluxului de analiză și clasificare a setului de date Hepatitis. Această abordare permite separarea clară a responsabilităților și facilitează dezvoltarea, testarea și extinderea aplicației.

Aplicația este structurată în jurul unei componente centrale de coordonare, care gestionează fluxul de execuție și asigură comunicarea dintre modulele funcționale. Componentele sistemului sunt organizate astfel încât datele să fie prelucrate secvențial, începând cu încărcarea și preprocesarea setului de date, continuând cu analiza statistică și partiționarea datelor, și finalizând cu antrenarea și evaluarea modelelor de clasificare.

Arhitectura adoptată permite rularea aplicației într-un mediu local, fără

a necesita componente distribuite sau servicii externe. Comunicarea dintre componente se realizează prin intermediul structurilor de date definite în capitolul anterior, iar rezultatele obținute sunt stocate în baza de date a aplicației și, respectiv, în fișierele de ieșire generate automat.

3.5.1.1 Șabloane arhitecturale folosite

Aplicația utilizează un model arhitectural de tip layered (stratificat), în care funcționalitățile sunt organizate pe niveluri logice. Fiecare nivel are responsabilități clare și comunică cu celelalte niveluri prin interfețe bine definite.

Nivelurile arhitecturale principale sunt:

● ***nivelul de prezentare/execuție,*** *responsabil cu inițierea rulării aplicației și coordonarea fluxului;*

● ***nivelul de logică a aplicației,*** *care include modulele de preprocesare, analiză statistică, partiționare, clasificare și evaluare;*

● ***nivelul de persistență a datelor,*** *responsabil cu stocarea rezultatelor în baza de date și în fișierele de ieșire.*

Acest șablon arhitectural a fost ales datorită simplității și clarității sale, fiind adecvat pentru o aplicație de analiză de date cu flux secvențial bine definit.

3.5.1.2 Diagrama de arhitectură

Diagrama de arhitectură evidențiază componentele principale ale aplicației și relațiile dintre acestea. Componenta centrală coordonează fluxul de execuție și interacționează cu modulele funcționale responsabile de prelucrarea datelor și de evaluarea modelelor de clasificare. De asemenea, diagrama evidențiază legătura dintre componentele de procesare și baza de date utilizată pentru stocarea rezultatelor (Figura 2).

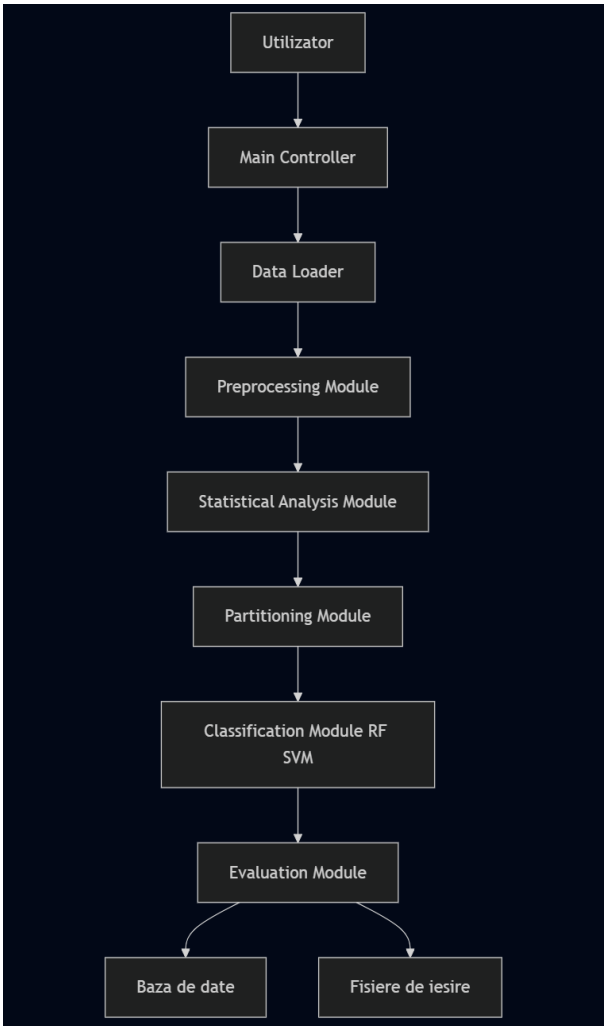


Figura 2. Diagrama de arhitectură a aplicației

3.5.2 Descrierea componentelor

Componenta Main / Controller

Componenta Main / Controller are rolul de a coordona execuția aplicației și de a controla fluxul general de procesare. Aceasta inițiază rularea aplicației la cererea utilizatorului și orchestrează apelurile către celelalte componente funcționale, asigurând transmiterea corectă a datelor între module.

Componenta Data Loader

Componenta Data Loader este responsabilă de încărcarea setului de

date Hepatitis din fișierele de intrare și de organizarea acestuia în structuri de date interne. Aceasta separă datele de intrare în matricea caracteristicilor și vectorul țintă, pregătind datele pentru etapele ulterioare de procesare.

Componenta Preprocessing Module

Componenta Preprocessing Module realizează preprocesarea datelor încărcate. Aceasta identifică valorile lipsă sau nedefinite și aplică metodele de corectare stabilite, precum imputarea valorilor, asigurând consistența și calitatea datelor de intrare utilizate în analiză și clasificare.

Componenta Statistical Analysis Module

Componenta Statistical Analysis Module este responsabilă de calcularea indicatorilor statistici descriptivi pentru caracteristicile setului de date. Aceasta determină valori precum media, dispersia, valorile minime și maxime, precum și alte informații utile pentru caracterizarea distribuției datelor.

Componenta Partitioning Module

Componenta Partitioning Module realizează împărțirea aleatoare a setului de date în seturi de antrenare și testare. Aceasta suportă diferite proporții de partiționare (80%–20%, 70%–30%, 60%–40% și 50%–50%) și asigură păstrarea informațiilor privind distribuția claselor în subseturile obținute.

Componenta Classification Module

Componenta Classification Module este responsabilă de antrenarea și utilizarea algoritmilor de clasificare. Aceasta implementează modelele Random Forest și Support Vector Machine, antrenează modelele pe seturile de antrenare și generează predicții pentru seturile de testare.

Componenta Evaluation Module

Componenta Evaluation Module evaluează performanța modelelor de

clasificare antrenate. Aceasta calculează metricile de performanță și generează matricele de confuzie, rezultatele fiind ulterior stocate în baza de date și exportate în fișierele de ieșire.

Componenta Baza de date

Componenta Baza de date asigură persistența informațiilor rezultate în urma execuțiilor aplicației. Aceasta stochează datele despre rulări, partiționări, modele și metrici de performanță, permițând consultarea și compararea ulterioară a rezultatelor.

Componenta Fișiere de ieșire

Componenta Fișiere de ieșire este utilizată pentru salvarea rezultatelor procesării sub formă de fișiere, precum statistici, metrici și reprezentări grafice, în scop de analiză și raportare.

3.5.3 Restricțiile de implementare

Aplicația este proiectată pentru a rula într-un mediu local, fără a necesita infrastructură distribuită sau servicii externe. Implementarea soluției trebuie să respecte cerințele funcționale definite în documentul SRS și structura arhitecturală descrisă în capitolele anterioare.

Restricțiile principale de implementare sunt următoarele:

● aplicația trebuie să permită procesarea setului de date Hepatitis în format tabular, fără a impune conversii suplimentare ale datelor;

● preprocesarea datelor trebuie să trateze explicit valorile lipsă sau nedefinite, conform regulilor stabilite;

● algoritmii de clasificare utilizați trebuie să fie Random Forest și Support Vector Machine, fără introducerea altor metode de clasificare;

● partiționarea datelor trebuie realizată exclusiv pentru proporțiile specificate (80%–20%, 70%–30%, 60%–40% și 50%–50%);

● rezultatele obținute trebuie să fie persistate în baza de date și/sau

exportate în fișiere de ieșire pentru analiză ulterioară;

- *implementarea trebuie să permită rulări repetate ale aplicației, utilizând configurații diferite, fără a afecta datele deja stocate.*

Aceste restricții asigură coerența implementării cu cerințele proiectului și permit evaluarea corectă și comparabilă a rezultatelor obținute.

3.5.4 Interacțiunea dintre componente

Interacțiunea dintre componentele aplicației se realizează într-un mod secvențial, conform fluxului de procesare stabilit în arhitectura sistemului. Fiecare componentă își îndeplinește rolul specific și transmite rezultatele obținute către componenta următoare din flux.

Execuția aplicației este inițiată de utilizator, care declanșează rularea prin intermediul componentei Main / Controller. Această componentă coordonează întregul proces și gestionează ordinea de apelare a modulelor funcționale.

Componenta Data Loader este apelată pentru a încărca setul de date Hepatitis din fișierele de intrare și pentru a furniza datele brute către componenta de preprocesare. Ulterior, Preprocessing Module procesează datele primite, identifică și corectează valorile lipsă sau nedefinite și transmite datele curate către componenta de analiză statistică.

Componenta Statistical Analysis Module primește datele preprocesate și calculează indicatorii statistici descriptivi, informațiile obținute fiind utilizate atât pentru raportare, cât și pentru înțelegerea caracteristicilor setului de date. După această etapă, datele sunt transmise către Partitioning Module, care realizează împărțirea setului de date în subseturi de antrenare și testare, conform proporțiilor stabilite.

Componenta Classification Module primește seturile de date rezultate din partiționare și realizează antrenarea și evaluarea modelelor de clasificare Random Forest și Support Vector Machine. Predicțiile generate sunt apoi

transmise către Evaluation Module, care calculează metricile de performanță și construiește matricele de confuzie.

Rezultatele finale sunt persistate prin intermediul componentei Baza de date, care stochează informațiile despre rulări, modele și performanțe, și sunt totodată exportate prin componenta Fișiere de ieșire pentru analiză și raportare.

3.6 Modelul interfeţei cu utilizatorul

Aplicația utilizează o interfață simplă, orientată pe rularea secvențială a proceselor de analiză și clasificare a setului de date. Interacțiunea dintre utilizator și sistem se realizează prin inițierea execuției aplicației și prin consultarea rezultatelor generate, fără a necesita o interfață grafică complexă.

3.6.1 Succesiunea interfeţelor

Succesiunea interfețelor este liniară și corespunde fluxului de execuție al aplicației. Utilizatorul inițiază rularea aplicației, iar sistemul execută automat etapele de încărcare a datelor, preprocesare, analiză statistică, partiționare, clasificare și evaluare. La finalul execuției, utilizatorul are acces la rezultatele obținute, salvate sub formă de fișiere și în baza de date.

3.6.2 Ferestrele aplicaţiei

Aplicația nu utilizează ferestre grafice dedicate. Conceptual, pot fi identificate următoarele „ferestre” logice:

- *o fereastră de inițiere a execuției, în care utilizatorul pornește aplicația;*
- *o fereastră de procesare, în care sunt executate automat etapele de analiză;*
- *o fereastră de rezultate, în care sunt disponibile statisticile și metricile de performanță generate.*

Aceste ferestre sunt logice și nu implică o interfață grafică explicită,

interacțiunea fiind realizată prin rularea aplicației și consultarea rezultatelor generate.

3.7 Elemente de testare

Acest capitol identifică acele componente ale aplicației care sunt considerate critice din punct de vedere al funcționării corecte a sistemului, precum și posibile alternative de proiectare care ar putea fi luate în considerare.

3.7.1 Componente critice

Componentele critice ale aplicației sunt acele module a căror funcționare corectă influențează direct validitatea rezultatelor obținute.

Un rol critic îl are componenta de Preprocessing Module, deoarece tratarea incorectă a valorilor lipsă sau nedefinite poate conduce la rezultate eronate în etapele ulterioare de analiză și clasificare. Orice eroare apărută în această etapă se propagă în întregul flux de procesare.

De asemenea, Partitioning Module reprezintă o componentă critică, întrucât realizarea incorectă a partiționărilor poate afecta distribuția claselor în seturile de antrenare și testare, influențând în mod direct evaluarea performanței algoritmilor de clasificare.

Componenta Classification Module este considerată critică deoarece implementează algoritmii de clasificare Random Forest și Support Vector Machine. Configurarea incorectă a acestora sau antrenarea pe date necorespunzătoare poate conduce la rezultate nerelevante sau greu de comparat.

În final, componenta Evaluation Module este esențială pentru calcularea corectă a metricilor de performanță. Erori în această componentă pot duce la interpretări greșite ale rezultatelor și la concluzii eronate privind performanța modelelor utilizate.

3.7.2 Alternative

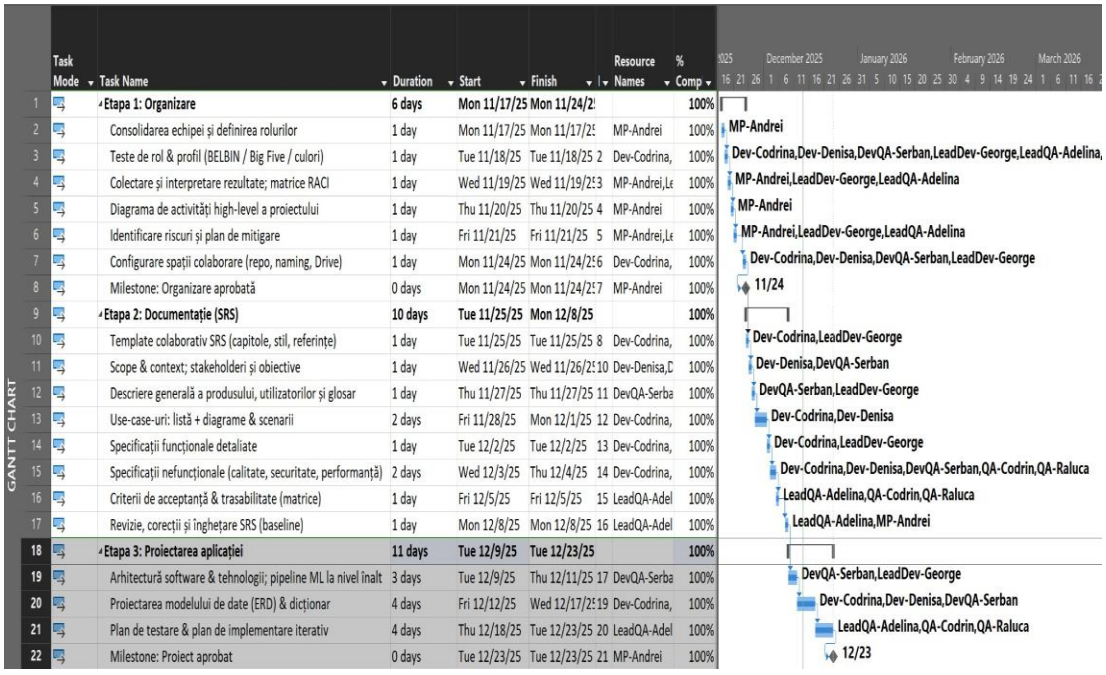
În cadrul proiectării aplicației au fost luate în considerare și alte posibile soluții, însă acestea nu au fost adoptate în varianta curentă a sistemului.

O alternativă posibilă ar fi utilizarea unor algoritmi de clasificare diferiți față de Random Forest și Support Vector Machine. Această opțiune nu a fost aleasă deoarece cerințele proiectului impun explicit analiza comparativă a acestor două metode.

O altă alternativă ar fi realizarea unei interfețe grafice pentru interacțiunea cu utilizatorul. Această soluție nu a fost adoptată deoarece aplicația este orientată pe analiză de date și rulare secvențială, iar o interfață de tip linie de comandă sau execuție automată este suficientă pentru scopul propus.

De asemenea, ar fi fost posibilă utilizarea unei alte structuri de persistență a datelor, precum stocarea exclusivă în fișiere. Varianta aleasă, care combină baza de date cu fișierele de ieșire, permite o organizare mai bună a rezultatelor și o analiză comparativă mai eficientă a rulărilor.

3.8 Planificarea activităților și progres



	Task Mode	Task Name	Duration	Start	Finish		Resource Names	% Comp
1		**Etapa 1: Organizare**	**6 days**	**Mon 11/17/25**	**Mon 11/24/2**			**100%**
2		Consolidarea echipei și definirea rolurilor	1 day	Mon 11/17/25	Mon 11/17/2		MP-Andrei	100%
3		Teste de rol & profil (BELBIN / Big Five / culori)	1 day	Tue 11/18/25	Tue 11/18/25	2	Dev-Codrina,	100%
4		Colectare și interpretare rezultate; matrice RACI	1 day	Wed 11/19/25	Wed 11/19/2	3	MP-Andrei,Le	100%
5		Diagrama de activități high-level a proiectului	1 day	Thu 11/20/25	Thu 11/20/25	4	MP-Andrei	100%
6		Identificare riscuri și plan de mitigare	1 day	Fri 11/21/25	Fri 11/21/25	5	MP-Andrei,Le	100%
7		Configurare spații colaborare (repo, naming, Drive)	1 day	Mon 11/24/25	Mon 11/24/2	6	Dev-Codrina,	100%
8		Milestone: Organizare aprobată	0 days	Mon 11/24/25	Mon 11/24/2	7	MP-Andrei	100%
9		**Etapa 2: Documentație (SRS)**	**10 days**	**Tue 11/25/25**	**Mon 12/8/25**			**100%**
10		Template colaborativ SRS (capitole, stil, referințe)	1 day	Tue 11/25/25	Tue 11/25/25	8	Dev-Codrina,	100%
11		Scope & context; stakeholderi și obiective	1 day	Wed 11/26/25	Wed 11/26/2	10	Dev-Denisa,D	100%
12		Descriere generală a produsului, utilizatorilor și glosar	1 day	Thu 11/27/25	Thu 11/27/25	11	DevQA-Serba	100%
13		Use-case-uri: listă + diagrame & scenarii	2 days	Fri 11/28/25	Mon 12/1/25	12	Dev-Codrina,	100%
14		Specificații funcționale detaliate	1 day	Tue 12/2/25	Tue 12/2/25	13	Dev-Codrina,	100%
15		Specificații nefuncționale (calitate, securitate, performanță)	2 days	Wed 12/3/25	Thu 12/4/25	14	Dev-Codrina,	100%
16		Criterii de acceptanță & trasabilitate (matrice)	1 day	Fri 12/5/25	Fri 12/5/25	15	LeadQA-Adel	100%
17		Revizie, corecții și înghețare SRS (baseline)	1 day	Mon 12/8/25	Mon 12/8/25	16	LeadQA-Adel	100%
18		**Etapa 3: Proiectarea aplicației**	**11 days**	**Tue 12/9/25**	**Tue 12/23/25**			**100%**
19		Arhitectură software & tehnologii; pipeline ML la nivel înalt	3 days	Tue 12/9/25	Thu 12/11/25	17	DevQA-Serba	100%
20		Proiectarea modelului de date (ERD) & dicționar	4 days	Fri 12/12/25	Wed 12/17/2	19	Dev-Codrina,	100%
21		Plan de testare & plan de implementare iterativ	4 days	Thu 12/18/25	Tue 12/23/25	20	LeadQA-Adel	100%
22		Milestone: Proiect aprobat	0 days	Tue 12/23/25	Tue 12/23/25	21	MP-Andrei	100%

4. Etapa_4: Implementarea aplicației

4.1 Componentele aplicației

Arhitectura software a fost divizată în module funcționale independente, orchestrate de un fișier principal (main.m). Această structură modulară asigură mentenanța codului și izolarea erorilor. Mai jos sunt prezentate implementările efective și descrierea logică a fiecărei componente.

4.1.1 Componenta de orchestrare (Main)

Acesta este punctul de intrare în aplicație. Scriptul gestionează fluxul de execuție, curățarea mediului de lucru și jurnalizarea rezultatelor într-un fișier text pentru persistența datelor.

Implementarea codului (main.m) :

```
% requirement: ADD-ON Statistics and Machine Learning Toolbox

clc; clear; close all;

nume_fisier_rezultate = 'Rezultate_Analiza_Hepatita.txt';

if exist(nume_fisier_rezultate, 'file')
    delete(nume_fisier_rezultate);
end

diary(nume_fisier_rezultate);

disp('1. Incarcare si prelucrare date...');
[X, Y, nume_coloane] = incarcare_date('hepatitis/hepatitis.data');

disp('2. Calcul statistici descriptive...');
calcul_statistici(X, nume_coloane);

disp('3. Generare grafice...');
grafice(X, Y);

disp('4. Analiza comparativa RF vs SVM pe diferite partitii...');
analiza_clasificare(X, Y, nume_coloane);

diary off;

disp(['Rezultatele au fost salvate in: ', nume_fisier_rezultate]);
```

Descriere tehnică:

• Secvența clc; clear; close all; asigură inițializarea unui mediu de lucru curat, eliminând variabile reziduale sau ferestre grafice anterioare.

• Funcția diary este utilizată pentru a redirecționa output-ul din consolă (inclusiv erorile și rezultatele clasificării) către fișierul Rezultate_Analiza_Hepatita.txt.

• Scriptul apelează secvențial funcțiile specializate pentru cele 4 etape majore: încărcare, statistică, vizualizare și analiză Machine Learning.

4.1.2.Componenta de achiziție și pregătire a datelor

Această componentă gestionează citirea fișierului brut și tratarea valorilor l ipsă, un pas critic pentru algoritmii de clasificare care nu acceptă valori NaN.

Implementarea codului (incarcare_date.m):

```
function [X, Y, headers] = incarcare_date(nume_fisier)
    T = readtable(nume_fisier, 'FileType', 'text', 'TreatAsMissing', '?',
'ReadVariableNames', false);

    T = fillmissing(T, 'nearest');

    data = table2array(T);

    Y = data(:, 1);
    X = data(:, 2:end);

    headers = {'Age','Sex','Steroid','Antivirals','Fatigue','Malaise','Anorexia',...

'LiverBig','LiverFirm','SpleenPalpable','Spiders','Ascites','Varices',...
                'Bilirubin','AlkPhosphate','Sgot','Albumin','Protime','Histology'};
end
```

Descriere tehnică:

• Se utilizează readtable cu parametrul 'TreatAsMissing', '?' deoarece setul de date "Hepatitis" marchează datele lipsă cu semnul întrebării.

• Funcția fillmissing cu metoda 'nearest' realizează imputarea datelor, completând valorile lipsă cu valoarea validă cea mai apropiată, evitând astfel

eliminarea rândurilor și pierderea de informație.

• Matricea rezultată este separată în vectorul de etichete Y (Clasa: DIE/LIVE) și matricea de atribute X.

4.1.3 Componenta de analiză statistică

Această funcție calculează metrici descriptive pentru a evalua distribuția datelor și a identifica necesitatea normalizării.

Implementarea codului (calcul_statistici.m):

```
function calcul_statistici(X, headers)
    media = mean(X)';
    dispersia = var(X)';
    val_min = min(X)';
    val_max = max(X)';

    T_stats = table(media, dispersia, val_min, val_max, ...
        'RowNames', headers', ...
        'VariableNames', {'Media', 'Dispersia', 'Min', 'Max'});

    disp(T_stats);

    if any(dispersia > 1000)
        disp('OBS: Exista variabile cu dispersie foarte mare, ceea ce sugereaza
necesitatea standardizarii pentru SVM...');
    end
    disp('');
end
```

Descriere tehnică:

- *Se generează un tabel sintetic (T_stats) care conține media, dispersia (varianța), minimul și maximul pentru fiecare atribut biologic.*
- *Logica condițională if any(dispersia > 1000) funcționează ca un sistem de avertizare automată. Detectarea unei dispersii mari indică faptul că atributele au scări foarte diferite (ex: vârsta vs. nivelul enzimelor), sugerând că algoritmii bazați pe distanțe (precum SVM)*

vor necesita standardizare.

4.1.4 Componenta de vizualizare grafică

Oferă o perspectivă vizuală asupra echilibrului claselor și a relațiilor dintre variabile.

Implementarea codului (grafice.m):

```
function grafice(X, Y)
    figure('Name', 'Analiza Vizuala a Datelor', 'NumberTitle', 'off');

    subplot(1, 2, 1);
    h = histogram(Y);
    title('Distributia Claselor (1=DIE, 2=LIVE)');
    xlabel('Clasa'); ylabel('Numar Pacienti');
    labels = {'DIE', 'LIVE'};
    set(gca, 'XTick', [1 2], 'XTickLabel', labels);

    subplot(1, 2, 2);
    imagesc(corrcoef(X));
    colorbar;
    title('Matricea de Corelatie a Atributelor');
    xlabel('Index Atribut'); ylabel('Index Atribut');
end
```

Descriere tehnică:

- *Se utilizează histogram pentru a verifica dacă setul de date este echilibrat între pacienții care au supraviețuit și cei decedați.*
- *Funcția imagesc(corrcoef(X)) generează o hartă termică (heatmap) a matricei de corelație, utilă pentru a identifica redundanța între parametrii clinici (multicoliniaritate)*

.

4.1.5 Componenta de analiză și clasificare (Machine Learning)

Nucleul aplicației care realizează antrenarea comparativă a modelelor Random Forest și SVM pe multiple partiții de date.

Implementarea codului (analiza_clasificare.m):

```
function analiza_clasificare(X, Y, headers)
    procente = [0.8, 0.7, 0.6, 0.5];

    fprintf('%-10s %-15s %-15s %-15s\n', 'Split %', 'Train Samples', 'Acc RF (%)',
'Acc SVM (%)');
    disp('');

    importanta_trasaturi = zeros(1, size(X, 2));

    for i = 1:length(procente)
        p = procente(i);

        cv = cvpartition(Y, 'HoldOut', 1 - p);
        XTrain = X(training(cv), :);
        YTrain = Y(training(cv), :);
        XTest = X(test(cv), :);
        YTest = Y(test(cv), :);

        if p == 0.5
            disp('Verificare distributie clase pentru split 50-50:');
            tabulate(YTrain);
        end

        rf = TreeBagger(50, XTrain, YTrain, 'Method', 'classification', ...
            'OOBPredictorImportance', 'on');
        predRF = str2double(predict(rf, XTest));
        accRF = sum(predRF == YTest) / length(YTest) * 100;

        if i == 1
            importanta_trasaturi = rf.OOBPermutedPredictorDeltaError;
        end

        svm = fitcsvm(XTrain, YTrain, 'Standardize', true, 'KernelFunction',
'linear');
        predSVM = predict(svm, XTest);
        accSVM = sum(predSVM == YTest) / length(YTest) * 100;

        fprintf('%-10.0f %-15d %-15.2f %-15.2f\n', p*100, length(YTrain), accRF,
accSVM);
    end

    figure('Name', 'Importanta Trasaturi (RF)', 'NumberTitle', 'off');
    bar(importanta_trasaturi);
    set(gca, 'XTick', 1:length(headers), 'XTickLabel', headers, 'XTickLabelRotation',
45);
    title('Selectia Trasaturilor Relevante (Random Forest)');
    ylabel('Importanta');
```

```
    grid on;

    disp('CONCLUZII COMPARATIVE:');
    disp('1. Random Forest: Avantaj - Ofera automat importanta variabilelor (ex:
Albumin, Protime).');
    disp('   Dezavantaj - Poate face overfitting pe seturi mici daca nu e limitat.');
    disp('2. SVM: Avantaj - Functioneaza bine pe date putine dar cu dimensiuni
multe.');
    disp('   Dezavantaj - Sensibil la date nestandardizate (rezolvat aici prin
Standardize=true).');
end
```

Descriere tehnică:

• Splitarea datelor: Se utilizează cvpartition pentru a împărți datele în seturi de Antrenare și Testare conform procentelor [80%, 70%, 60%, 50%], permițând observarea stabilității modelelor.

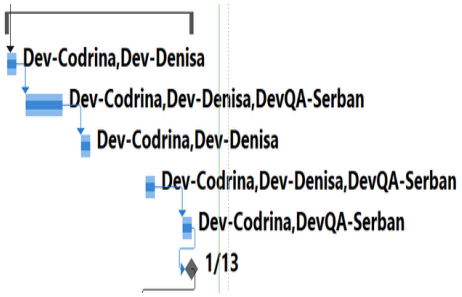
• Random Forest: Implementat prin TreeBagger cu 50 de arbori. Parametrul 'OOBPredictorImportance', 'on' este activat pentru a calcula importanța trăsăturilor (Feature Importance), generând graficul final care evidențiază cei mai relevanți biomarkeri (ex: Albumina).

• SVM: Implementat prin fitcsvm cu nucleu liniar. Parametrul 'Standardize', true este critic și a fost activat pentru a rezolva problema dispersiei mari identificate în etapa de statistică, aducând toate variabilele la aceeași scară (Z-score).

• Output: Codul afișează un tabel comparativ de acurateț̦e și generează concluzii automate bazate pe rulare.

4.2 Planificarea activităților și progres

23	✓		Etapa 4: Implementare & Integrare	14 days	Wed 12/24/25	Tue 1/13/26			100%
24	✓		Setup mediu de dezvoltare & infrastructură	1 day	Wed 12/24/25	Wed 12/24/25	22	Dev-Codrina,D	100%
25	✓		Implementare componente backend / logică principală	2 days	Fri 12/26/25	Mon 12/29/25	24	Dev-Codrina,D	100%
26	✓		Implementare componente frontend / UI	1 day	Thu 1/1/26	Thu 1/1/26	25	Dev-Codrina,D	100%
27	✓		Testare funcțională inițială	1 day	Thu 1/8/26	Thu 1/8/26		Dev-Codrina,D	100%
28	✓		Corecții & stabilizare	1 day	Mon 1/12/26	Mon 1/12/26	27	Dev-Codrina,D	100%
29	✓		Milestone: Funcționalitate de bază implementată	0 days	Tue 1/13/26	Tue 1/13/26	28	MP-Andrei	100%



5.Etapa_5 : Testarea aplicației

5.1 Plan de testare

Scopul etapei de testare este validarea corectitudinii funcționale a aplicației și analiza comportamentului acesteia în condiții experimentale, folosind setul de date ***Hepatitis (UCI ML Repository).***

Planul de testare vizează următoarele obiective:

- *verificarea încărcării și preprocesării corecte a datelor;*
- *validarea consistenței statistice a setului de date;*
- *evaluarea stabilității aplicației pentru diferite partiționări ale datelor;*
- *compararea performanței algoritmilor de clasificare Random Forest (RF) și Support Vector Machine (SVM);*
- *identificarea trăsăturilor relevante pentru clasificare.*

Testarea a fost realizată la nivel de: validare a datelor de intrare; teste experimentale de clasificare; analiză a rezultatelor obținute.

5.2 Scenarii de test

Pentru atingerea obiectivelor stabilite au fost definite următoarele scenarii de testare:

- ***TC1 –** Verificarea existenței fișierului de date și accesibilitatea acestuia;*
- ***TC2 –** Încărcarea corectă a setului de date și separarea în X (intrări) și Y (ieșiri);*
- ***TC3** – Validarea dimensiunilor setului de date (155 instanțe, 19 trăsături);*
- ***TC4** – Detectarea și corectarea valorilor lipsă utilizând isnan;*
- ***TC5 –** Calculul și validarea statisticilor descriptive (mean, var, min, max);*
- ***TC6** – Verificarea existenței a exact două clase în variabila țintă;*
- ***TC7 –** Generarea indicatorilor vizuali (boxplot și matrice de corelație);*
- ***TC8–TC11 –** Rularea experimentelor de clasificare pentru partiționări 80/20, 70/30, 60/40 și 50/50 și compararea RF vs SVM;*

- ***TC12 –*** *Determinarea importanței trăsăturilor utilizând Random Forest.*

Exemple de implementare pentru diferite scenarii

- **Încărcarea setului de date și separarea variabilelor de intrare și ieșire:**

[X, Y, headers] = incarcare_date(dataFile);

X = double(X);

Y = double(Y(:));

- **Detectarea și corectarea valorilor lipsă:**

nan_before = sum(isnan(X),'all');

X(isnan(X)) = median(X(~isnan(X)));

Fiecare scenariu de test a fost considerat valid dacă rezultatul obținut a fost conform așteptărilor și aplicația nu a generat erori.

5.3 Rapoarte cu rezultatele testelor

Rezultatele testelor au fost centralizate într-un raport de testare, care conține:

- *identificatorul testului;*
- *descrierea testului;*
- *statusul (PASS / FAIL);*
- *detalii relevante privind execuția testului.*

Toate scenariile de test definite au fost finalizate cu status PASS, ceea ce confirmă funcționarea corectă a aplicației și stabilitatea acesteia în condițiile experimentale analizate. În plus, pentru fiecare experiment de clasificare au fost raportate:

- *dimensiunea seturilor de antrenare și testare;*
- *distribuția claselor în setul de test;*
- *valorile accuracy pentru RF și SVM.*

5.3.1. Distribuția claselor

Setul de date Hepatitis prezintă un dezechilibru semnificativ între cele două clase. Din totalul de 155 de instanțe, 32 de instanțe (20.65%) aparțin clasei minoritare, iar 123 de instanțe (7G.35%) aparțin clasei majoritare.

Acest dezechilibru este menținut aproximativ și în seturile de test rezultate în urma diferitelor partiționări utilizate (80/20, 70/30, 60/40 și 50/50), după cum se observă din distribuțiile claselor raportate pentru fiecare experiment. Prezența unui dezechilibru de clasă justifică evaluarea stabilității modelelor pe mai multe împărțiri ale datelor și interpretarea atentă a valorilor de accuracy.

5.3.2 Rezultate experimentale pentru diferite partiționări

Pentru evaluarea performanței algoritmilor de clasificare Random Forest (RF) și Support Vector Machine (SVM), setul de date Hepatitis a fost împărțit aleator în seturi de antrenare și testare folosind patru partiționări diferite. Toate partiționările sunt aplicate pe același set de date, modificându-se doar proporția train/test, pentru a evalua stabilitatea rezultatelor.

PARTIȚIONARE (TRAIN/TEST)	ACCURACY RF (%)	ACCURACY SVM (%)
80/20	83.33	83.33
70/30	84.78	78.26
60/40	80.65	75.81
50/50	81.82	77.92

Rezultatele obținute indică valori ale accuracy-ului cuprinse aproximativ între 75.81% și 84.78%. Random Forest obține performanțe constante și superioare față de SVM în majoritatea partiționărilor, cu excepția cazului 80/20, unde cele două metode au avut performanțe identice. Diferențele observate pot fi explicate prin robustețea Random Forest la valori extreme și la dezechilibrul claselor, comparativ cu sensibilitatea mai ridicată a SVM la aceste caracteristici ale datelor.

5.3.3 Tabelul de testare sintetizează rezultatele tuturor scenariilor definite

	ID	Test	Status	Detalii
1	TC1	Fisier dataset disponibil	PASS	hepatitis/hepatitis.data
2	TC2	Incarcare date prin incarcare_date()	PASS	X=155x19, Y=155x1
3	TC3	Dimensiuni set date (155x19)	PASS	X=155x19, Y=155x1
4	TC4	Detectie si corectie missing values (isnan)	PASS	NaN: inainte=0, dupa=0
5	TC5	Statistici (mean/var/min/max) sunt finite	PASS	OK/NaN/Inf check
6	TC6	Exista exact 2 clase in Y	PASS	Clase: [1 2]
7	TC7	Generare indicatori vizuali (corelatii + boxplot)	PASS	Figuri generate
8	TC8	Experiment split 80/20 (RF vs SVM) a rulat	PASS	AccRF=83.33%, AccSVM=83.33%; c1=6(20.0%), c2=24(80.0%)
9	TC9	Experiment split 70/30 (RF vs SVM) a rulat	PASS	AccRF=84.78%, AccSVM=78.26%; c1=9(19.6%), c2=37(80.4%)
10	TC10	Experiment split 60/40 (RF vs SVM) a rulat	PASS	AccRF=80.65%, AccSVM=75.81%; c1=12(19.4%), c2=50(80.6%)
11	TC11	Experiment split 50/50 (RF vs SVM) a rulat	PASS	AccRF=81.82%, AccSVM=77.92%; c1=16(20.8%), c2=61(79.2%)
12	TC12	Feature selection (RF importance) calculat	PASS	1) Albumin = 0.5257 | 2) Spiders = 0.4637 | 3) Ascites = 0.4318 | 4) Varices = 0.3008 | 5) Protime = 0.2999

Figura 1 – Rezumatul testelor de validare și al experimentelor

Figura prezintă tabelul centralizat al scenariilor de test definite, incluzând identificatorul testului, descrierea acestuia, statusul execuției (PASS/FAIL) și detalii relevante privind rezultatele obținute

Testele TC1–TC7 validează corectitudinea încărcării datelor, consistența statistică și generarea indicatorilor vizuali, în timp ce testele TC8–TC11 corespund experimentelor de clasificare pentru diferite partiționări ale setului de date și compară performanța algoritmilor Random Forest și SVM. Testul TC12 evidențiază selecția trăsăturilor relevante utilizând Random Forest. Toate testele au fost finalizate cu status PASS, confirmând funcționarea corectă a aplicației și coerența rezultatelor experimentale.

5.3.4 Documentarea rezultatelor

Rezultatele experimentale au fost documentate atât numeric, cât și vizual, folosind următorii indicatori:

- ***distribuția claselor la nivel global și pentru fiecare partiționare;***
- ***boxplot-uri pentru identificarea valorilor extreme (outlieri);***
- ***matricea de corelație pentru analiza relațiilor dintre trăsături;***
- ***grafice de importanță a trăsăturilor obținute cu Random Forest.***

Analiza a evidențiat:

- ***un dezechilibru semnificativ al claselor;***
- ***prezența outlierilor în biomarkeri hepatici;***
- ***performanță ușor superioară a Random Forest față de SVM;***

- ***trăsături relevante clinic, precum Spiders, Albumin și Ascites.***

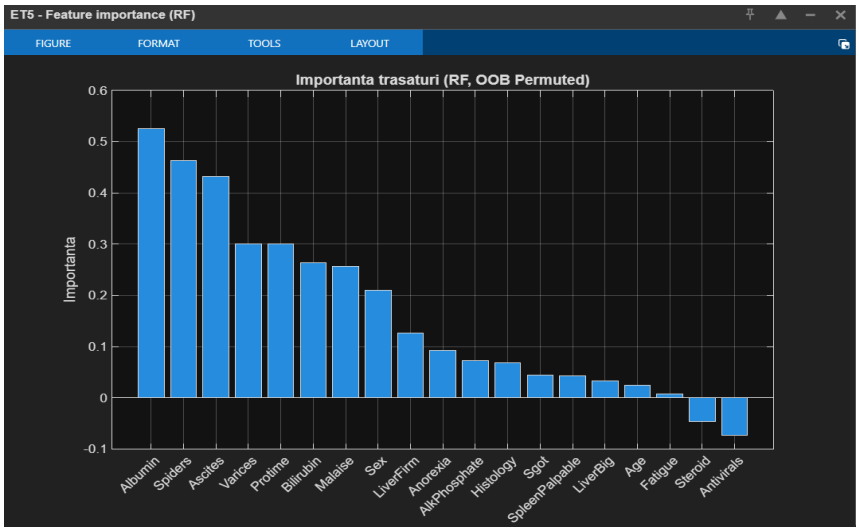


Figura 2 – Importanța trăsăturilor determinată cu Random Forest (OOB Permuted)

Graficul prezintă importanța relativă a trăsăturilor calculată folosind metoda Out-of-Bag Permuted Predictor Importance din Random Forest. Valorile mai mari indică o contribuție mai semnificativă a trăsăturii la procesul de clasificare, în timp ce valorile apropiate de zero sau negative sugerează un impact redus sau neglijabil.

Analiza importanței trăsăturilor arată că variabile precum Albumin, Spiders și Ascites au cel mai mare impact asupra clasificării pacienților cu hepatită. Aceste trăsături sunt relevante din punct de vedere clinic, fiind asociate cu severitatea afectării hepatice.

*Trăsături precum **Age**, **Steroid** sau **Antivirals** prezintă o importanță scăzută sau negativă, indicând faptul că acestea nu contribuie semnificativ la procesul de clasificare. Acest rezultat evidențiază capacitatea Random Forest de a realiza implicit selecția trăsăturilor relevante și constituie un avantaj față de SVM, care nu oferă în mod direct o măsură a importanței variabilelor. Rezultatele obținute confirmă coerența dintre analiza statistică, indicatorii vizuali și performanța modelelor de clasificare.*

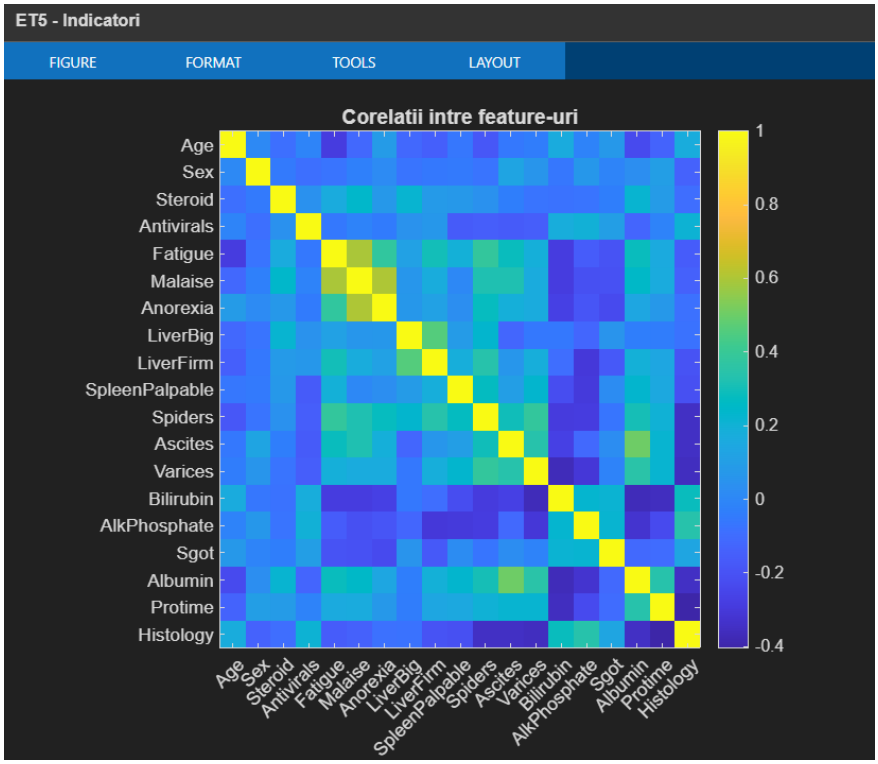


Figura 3 – Matricea de corelație a trăsăturilor setului de date Hepatitis

Figura prezintă coeficienții de corelație Pearson dintre trăsăturile setului de date. Valorile apropiate de 1 indică o corelație pozitivă puternică, valorile apropiate de −1 indică o corelație negativă, iar valorile apropiate de 0 sugerează o relație slabă sau inexistentă între trăsături.

Analiza matricei de corelație evidențiază existența unor corelații moderate între anumite trăsături, în special între variabile asociate simptomelor și biomarkerilor hepatici.

Totuși, nu se observă corelații extreme între trăsături, ceea ce indică absența multicoliniarității severe în setul de date. Acest rezultat sugerează că majoritatea trăsăturilor furnizează informații complementare și pot fi utilizate simultan în modelele de clasificare, fără a afecta semnificativ stabilitatea sau performanța acestora.

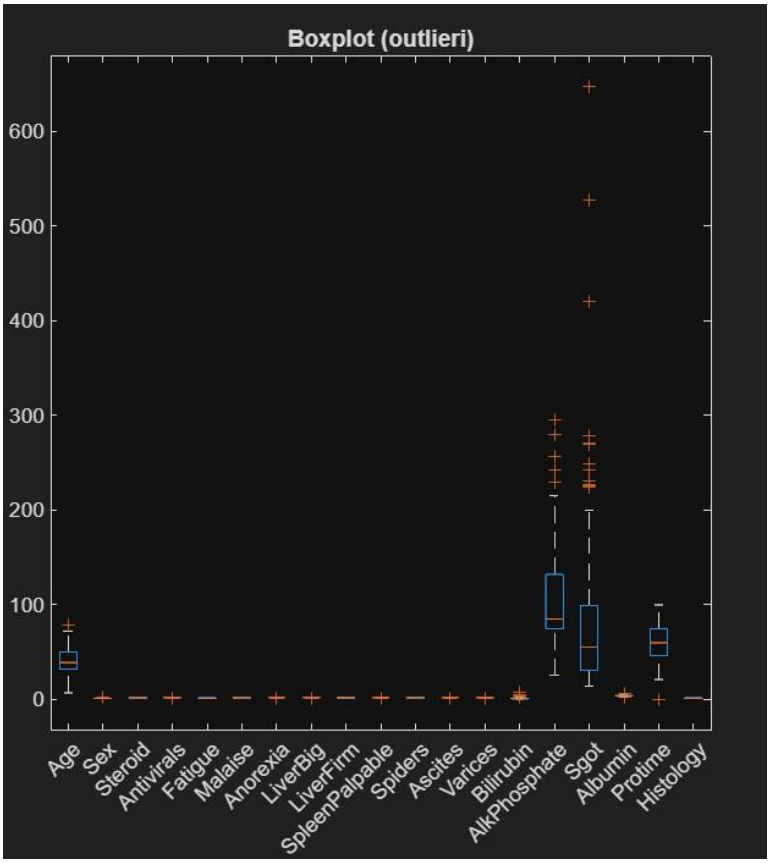


Figura 4 – Boxplot pentru trăsăturile setului de date Hepatitis

Figura prezintă distribuția valorilor pentru fiecare trăsătură, evidențiind mediana, intervalul intercuartilic și valorile extreme (outlieri). Simbolurile „+" reprezintă observații aflate în afara intervalului normal al datelor.

Boxplot-ul evidențiază o dispersie ridicată și prezența unor valori extreme pentru anumite trăsături, în special pentru biomarkerii hepatici precum Bilirubin, AlkPhosphate și Sgot. Aceste valori extreme corespund unor cazuri clinice severe și sunt caracteristice seturilor de date medicale. În contrast, trăsături precum Sex, Steroid, Antivirals sau Fatigue prezintă distribuții compacte, corespunzătoare variabilelor binare. Prezența outlierilor justifică utilizarea unor algoritmi robuști, precum Random Forest, care sunt mai puțin sensibili la astfel de valori comparativ cu SVM.

5.4 Planificarea activităților și progres

Etapa de testare a fost realizată conform planificării inițiale, după finalizarea implementării aplicației. Activitățile au fost desfășurate în următoarea ordine:

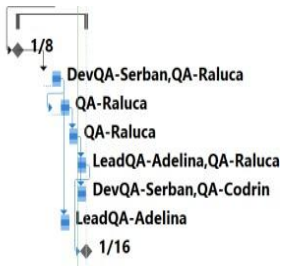
1. *validarea datelor și a precondițiilor experimentale;*
2. *definirea și rularea scenariilor de test;*

3. *colectarea și analiza rezultatelor;*

4. *documentarea concluziilor experimentale.*

Progresul a fost monitorizat prin rulări succesive ale testelor și ajustări minore ale scripturilor, fără a modifica funcționalitatea de bază a aplicației.

30	✓		**Etapa 5: Testarea aplicației – alocare resurse**	**6 days**	**Thu 1/8/26**	**Fri 1/16/26**			**100%**
31	✓		Definirea planului de testare	0 days	Thu 1/8/26	Thu 1/8/26	29	DevQA-Serbar	100%
32	✓		Definirea scenariilor de test (TC1–TC12)	1 day	Sun 1/11/26	Tue 1/13/26	31	DevQA-Serbar	100%
33	✓		Validarea încărcării și preprocesării datelor	1 day	Mon 1/12/26	Wed 1/14/26	32	QA-Raluca	100%
34	✓		Testare analize statistice & indicatori vizuali	1 day	Thu 1/15/26	Thu 1/15/26	33	QA-Raluca	100%
35	✓		Analiza rezultatelor & stabilitatea modelelor	1 day	Fri 1/16/26	Fri 1/16/26	34	LeadQA-Adelir	100%
36	✓		Analiza importanței trăsăturilor	1 day	Fri 1/16/26	Fri 1/16/26	34	DevQA-Serbar	100%
37	✓		Întocmire raport de testare	1 day	Wed 1/14/26	Wed 1/14/26	32	LeadQA-Adelir	100%
38	✓		Milestone: Testare finalizată și validată	0 days	Fri 1/16/26	Fri 1/16/26	35	LeadQA-Adelir	100%



6. Etapa_6 : Documentarea prezentării proiectului :

6.1 Introducere – starea cercetării în domeniu

În ultimii ani, utilizarea tehnicilor de analiză a datelor și a algoritmilor de învățare automată a devenit tot mai importantă în domeniul medical, oferind suport pentru diagnostic și luarea deciziilor clinice. Creșterea volumului de date medicale disponibile a impus dezvoltarea unor metode automate capabile să identifice tipare relevante și relații complexe între variabile.

Bolile hepatice, inclusiv hepatita, reprezintă o problemă majoră de sănătate publică, iar diagnosticarea timpurie este esențială pentru reducerea riscului de complicații. Datele medicale asociate acestor afecțiuni sunt adesea incomplete și eterogene, ceea ce limitează eficiența metodelor clasice de analiză. În acest context, algoritmii de clasificare automată, precum Random Forest și Support Vector Machine, sunt frecvent utilizați datorită performanței ridicate și capacității de a gestiona date complexe.

Studiile din literatura de specialitate evidențiază importanța preprocesării datelor, a analizei statistice descriptive și a evaluării comparative a modelelor de clasificare pe seturi de date standardizate, precum Hepatitis din UCI Machine Learning Repository. Soluțiile software moderne integrează aceste etape pentru a îmbunătăți acuratețea clasificării și pentru a oferi un suport decizional eficient în domeniul medical.

6.2 Metodologie

Metodologia acestui proiect vizează analiza și clasificarea datelor medicale asociate hepatitei, utilizând algoritmi de învățare automată. Procesul este organizat în etape succesive, care asigură o abordare structurată și reproductibilă.

Datele utilizate provin din setul Hepatitis, preluat din UCI Machine Learning Repository, și sunt supuse unui proces de preprocesare ce include tratarea valorilor lipsă și separarea atributelor de intrare de variabila țintă. Ulterior, se realizează o analiză statistică descriptivă pentru înțelegerea distribuției datelor și identificarea eventualelor anomalii.

Setul de date este apoi împărțit în seturi de antrenare și testare folosind mai

multe proporții de partiționare, pentru evaluarea stabilității modelelor. Sunt aplicați și comparați algoritmii Random Forest și Support Vector Machine, iar performanța acestora este evaluată pe baza unor metrici relevante de clasificare. În final, este analizată importanța trăsăturilor pentru evidențierea variabilelor cu impact semnificativ asupra rezultatelor.

6.3 Rezultate experimentale

Rezultatele experimentale obținute confirmă funcționarea corectă a aplicației și validitatea metodologiei propuse. Analiza comparativă a algoritmilor Random Forest și Support Vector Machine evidențiază performanțe stabile pentru ambele metode, cu rezultate ușor superioare în cazul Random Forest pentru majoritatea partiționărilor utilizate. Diferențele observate sunt în concordanță cu caracteristicile setului de date, precum dezechilibrul claselor și prezența valorilor extreme.

De asemenea, variația proporțiilor de antrenare și testare nu a influențat semnificativ performanța generală a modelelor, ceea ce evidențiază robustețea soluției. Analiza importanței trăsăturilor a permis identificarea variabilelor relevante pentru clasificare, susținând interpretările statistice și vizuale realizate în etapele anterioare.

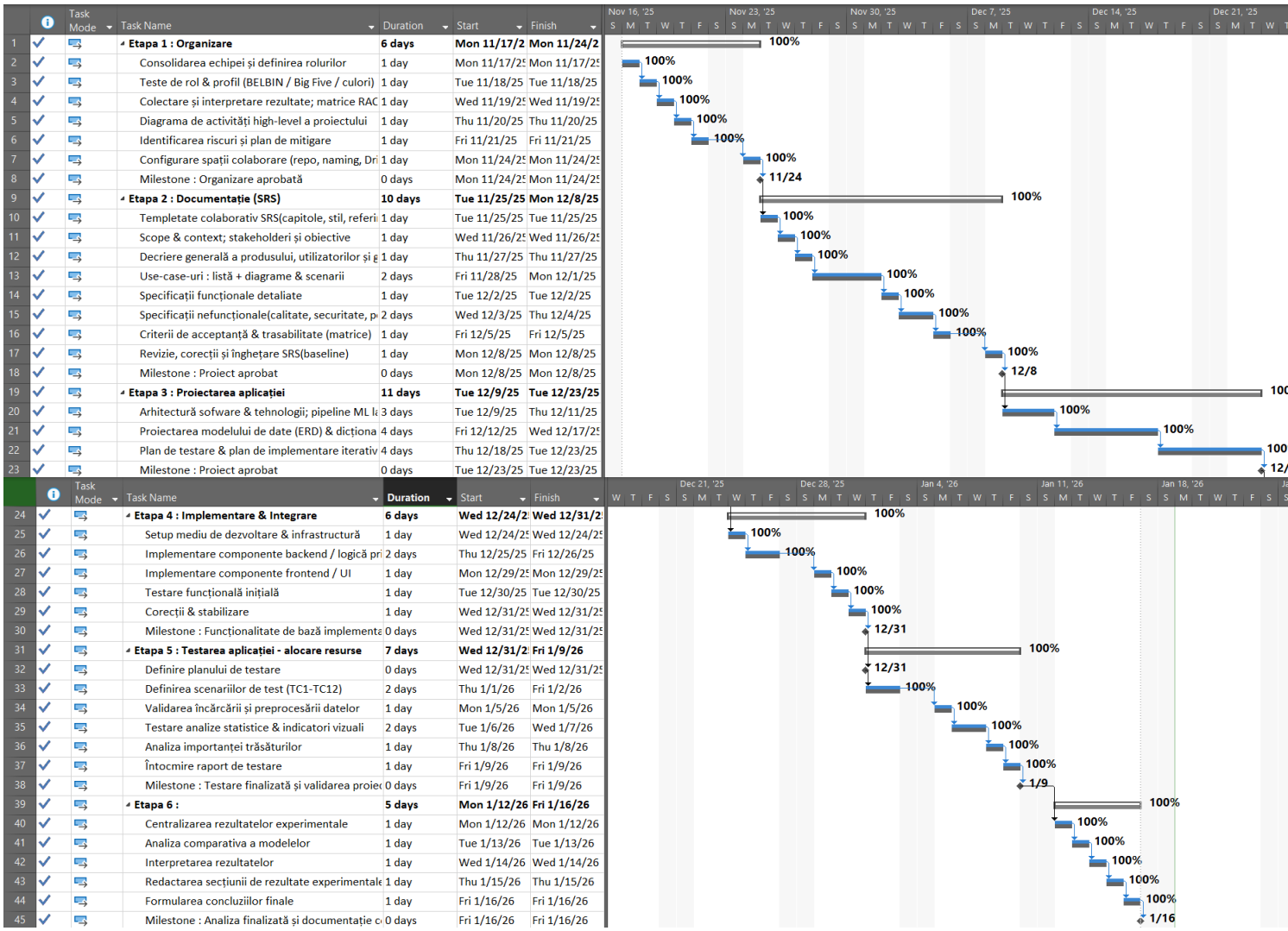
6.4 Concluzii

În cadrul acestui proiect a fost realizată o aplicație software pentru analiza experimentală și clasificarea datelor medicale asociate hepatitei, utilizând tehnici de învățare automată. Metodologia propusă, bazată pe preprocesarea datelor, analiza statistică și evaluarea comparativă a algoritmilor Random Forest și Support Vector Machine, a permis obținerea unor rezultate coerente și reproductibile.

Rezultatele experimentale confirmă eficiența soluției dezvoltate, evidențiind performanțe stabile pentru ambele modele de clasificare, cu o ușoară superioritate a algoritmului Random Forest în majoritatea scenariilor testate. Analiza importanței trăsăturilor a contribuit la interpretarea rezultatelor și la identificarea variabilelor relevante din punct de vedere clinic.

În concluzie, aplicația propusă demonstrează utilitatea utilizării algoritmilor de învățare automată în analiza datelor medicale și poate reprezenta un punct de plecare pentru extinderi viitoare, precum integrarea altor metode de clasificare sau aplicarea soluției pe seturi de date medicale mai ample și diversificate.

6.5 Planificarea proiectului



	Task Mode	Task Name	Duration	Start	Finish
1	✓	**Etapa 1 : Organizare**	**6 days**	**Mon 11/17/2**	**Mon 11/24/2**
2	✓	Consolidarea echipei și definirea rolurilor	1 day	Mon 11/17/25	Mon 11/17/25
3	✓	Teste de rol & profil (BELBIN / Big Five / culori)	1 day	Tue 11/18/25	Tue 11/18/25
4	✓	Colectare și interpretare rezultate; matrice RAC	1 day	Wed 11/19/25	Wed 11/19/25
5	✓	Diagrama de activități high-level a proiectului	1 day	Thu 11/20/25	Thu 11/20/25
6	✓	Identificarea riscuri și plan de mitigare	1 day	Fri 11/21/25	Fri 11/21/25
7	✓	Configurare spații colaborare (repo, naming, Dri	1 day	Mon 11/24/25	Mon 11/24/25
8	✓	Milestone : Organizare aprobată	0 days	Mon 11/24/25	Mon 11/24/25
9	✓	**Etapa 2 : Documentație (SRS)**	**10 days**	**Tue 11/25/25**	**Mon 12/8/25**
10	✓	Templetate colaborativ SRS(capitole, stil, referi	1 day	Tue 11/25/25	Tue 11/25/25
11	✓	Scope & context; stakeholderi și obiective	1 day	Wed 11/26/25	Wed 11/26/25
12	✓	Decriere generală a produsului, utilizatorilor și g	1 day	Thu 11/27/25	Thu 11/27/25
13	✓	Use-case-uri : listă + diagrame & scenarii	2 days	Fri 11/28/25	Mon 12/1/25
14	✓	Specificații funcționale detaliate	1 day	Tue 12/2/25	Tue 12/2/25
15	✓	Specificații nefuncționale(calitate, securitate, p	2 days	Wed 12/3/25	Thu 12/4/25
16	✓	Criterii de acceptanță & trasabilitate (matrice)	1 day	Fri 12/5/25	Fri 12/5/25
17	✓	Revizie, corecții și înghețare SRS(baseline)	1 day	Mon 12/8/25	Mon 12/8/25
18	✓	Milestone : Proiect aprobat	0 days	Mon 12/8/25	Mon 12/8/25
19	✓	**Etapa 3 : Proiectarea aplicației**	**11 days**	**Tue 12/9/25**	**Tue 12/23/25**
20	✓	Arhitectură sofware & tehnologii; pipeline ML la	3 days	Tue 12/9/25	Thu 12/11/25
21	✓	Proiectarea modelului de date (ERD) & dicționa	4 days	Fri 12/12/25	Wed 12/17/25
22	✓	Plan de testare & plan de implementare iterativ	4 days	Thu 12/18/25	Tue 12/23/25
23	✓	Milestone : Proiect aprobat	0 days	Tue 12/23/25	Tue 12/23/25
24	✓	**Etapa 4 : Implementare & Integrare**	**6 days**	**Wed 12/24/2**	**Wed 12/31/2**
25	✓	Setup mediu de dezvoltare & infrastructură	1 day	Wed 12/24/25	Wed 12/24/25
26	✓	Implementare componente backend / logică pri	2 days	Thu 12/25/25	Fri 12/26/25
27	✓	Implementare componente frontend / UI	1 day	Mon 12/29/25	Mon 12/29/25
28	✓	Testare funcțională inițială	1 day	Tue 12/30/25	Tue 12/30/25
29	✓	Corecții & stabilizare	1 day	Wed 12/31/25	Wed 12/31/25
30	✓	Milestone : Funcționalitate de bază implementa	0 days	Wed 12/31/25	Wed 12/31/25
31	✓	**Etapa 5 : Testarea aplicației - alocare resurse**	**7 days**	**Wed 12/31/2**	**Fri 1/9/26**
32	✓	Definire planului de testare	0 days	Wed 12/31/25	Wed 12/31/25
33	✓	Definirea scenariilor de test (TC1-TC12)	2 days	Thu 1/1/26	Fri 1/2/26
34	✓	Validarea încărcării și preprocesării datelor	1 day	Mon 1/5/26	Mon 1/5/26
35	✓	Testare analize statistice & indicatori vizuali	2 days	Tue 1/6/26	Wed 1/7/26
36	✓	Analiza importanței trăsăturilor	1 day	Thu 1/8/26	Thu 1/8/26
37	✓	Întocmire raport de testare	1 day	Fri 1/9/26	Fri 1/9/26
38	✓	Milestone : Testare finalizată și validarea proie	0 days	Fri 1/9/26	Fri 1/9/26
39	✓	**Etapa 6 :**	**5 days**	**Mon 1/12/26**	**Fri 1/16/26**
40	✓	Centralizarea rezultatelor experimentale	1 day	Mon 1/12/26	Mon 1/12/26
41	✓	Analiza comparativa a modelelor	1 day	Tue 1/13/26	Tue 1/13/26
42	✓	Interpretarea rezultatelor	1 day	Wed 1/14/26	Wed 1/14/26
43	✓	Redactarea secțiunii de rezultate experimentale	1 day	Thu 1/15/26	Thu 1/15/26
44	✓	Formularea concluziilor finale	1 day	Fri 1/16/26	Fri 1/16/26
45	✓	Milestone : Analiza finalizată și documentație c	0 days	Fri 1/16/26	Fri 1/16/26