# 

# Universitatea de Medicină, Farmacie, Științe și Tehnologie „George Emil Palade” din Târgu Mureș

# LUCRARE DE DISERTAȚIE

# Sistem multi-agent de gestionare a analizelor pentru prezicerea pietrelor la rinichi

Coordonator proiect: Student: Prof. univ. dr. abil. IANTOVICS László-Barna ing. PASCAL Vasile-Marian

**Cuprins**

[1. Rezumat 4](#_Toc168861507)

[2. Summary 5](#_Toc168861508)

[3. Introducere 6](#_Toc168861509)

[3.1. Obiectivul Cercetării 7](#_Toc168861510)

[3.2. Metodologii 7](#_Toc168861511)

[3.3. Structura lucrării 8](#_Toc168861512)

[4. Concepte și tehnologii utilizate 9](#_Toc168861513)

[4.1. Paradigma de programare orientată pe agenți 9](#_Toc168861514)

[4.2. Programarea orientată pe agenți vs. programarea orientată pe obiecte 10](#_Toc168861515)

[4.3. Java Agent DEvelopment Framework 12](#_Toc168861516)

[4.4. Învățare supervizată 15](#_Toc168861517)

[4.4.2 Aspecte ce trebuie luate în considerare în învățarea supervizată 16](#_Toc168861518)

[4.4.3 Algoritmi folosiți în învățarea supervizată 19](#_Toc168861519)

[4.4.4 Rețele neuronale - perceptronul multistrat 20](#_Toc168861520)

[5. Medicină 24](#_Toc168861521)

[5.1. Pietrele la rinichi (nefrolitiaza) 25](#_Toc168861522)

[5.2. Ce sunt pietrele la rinichi? 25](#_Toc168861523)

[5.3. Cauzele aparițiilor pietrelor la rinichi 26](#_Toc168861524)

[5.4. Simptomele pietrelor la rinichi 27](#_Toc168861525)

[5.5. Metode de diagnosticare al pietrelor la rinichi 27](#_Toc168861526)

[5.6. Tratament pietre la rinichi 28](#_Toc168861527)

[6. Implementarea sistemului multi-agent 28](#_Toc168861528)

[6.1 Structura proiectului 28](#_Toc168861529)

[6.2. Setul de date 30](#_Toc168861530)

[6.3. Arhitectura aplicației 32](#_Toc168861531)

[6.4. Agentul manager 33](#_Toc168861532)

[6.5. Agentul pentru adăugarea unor noi analize 33](#_Toc168861533)

[6.6. Agentul pentru selecția analizelor 34](#_Toc168861534)

[6.7. Agentul pentru predicția pietrelor la rinichi cu rețeaua neuronală 35](#_Toc168861535)

[6.8. Agentul Sniffer 39](#_Toc168861536)

[7. Rezultate obținute 40](#_Toc168861537)

[8. Discuții 44](#_Toc168861538)

[9. Concluzie 44](#_Toc168861539)

[10. Bibliografie 45](#_Toc168861540)

## 1. Rezumat

Această lucrare prezintă o metodă de prezicere a pietrelor la rinichi care utilizează un sistem multi-agent ce utilizează tehnologii avansate precum Java Agent Development Framework (JADE) și rețele neuronale. Scopul acestui studiu este de a crea un sistem care, pe baza datelor clinice și a rezultatelor analizelor, ar putea prezice prezența sau absența pietrelor la rinichi la pacienți atunci când este adăugat în sistem un nou set de analize ale urinei.  
  
Pentru a îndeplini acest obiectiv, s-a folosit un set de date alcătuit din 79 de analize ale pacienților; aceste analize au inclus proprietățile urinei pacienților:

* densitatea specifică, sau gravitatea urinei în raport cu apa;
* pH-ul, care este logaritmul negativ al ionului de hidrogen;
* osmolaritatea (mOsm), o unitate care este folosită în chimia fizică, dar nu în biologie și medicină. Osmolaritatea este proporțională cu concentrația de molecule în soluție;
* conductivitatea. Un Mho este egal cu un Ohm. Conducția este proporțională cu concentrația de substanțe încărcate cu ioni prezente în soluție;
* nivelul de uree în milimoli/litru;
* nivelul de calciu (CALC) în milimoli/litru.

Un sistem multi-agent format dintr-un număr de patru agenți

* un agent pentru adăugarea unei noi analize;
* un agent pentru managementul celorlalți agenți;
* un agent pentru selectarea unor analize cu valori alese de către utilizator;
* un agent pentru predicția pietrelor la rinichi care utilizează o rețea neuronală;

este folosit pentru a gestiona și a genera predicția pietrelor la rinichi în baza analizelor. Agenții colaborează între ei pentru gestionarea analizelor.

Metodele de învățare automată bazate pe rețele neuronale sunt utilizate în sistemul propus pentru a analiza datele și pentru a identifica modele sau caracteristici care sunt relevante pentru prezența pietrelor la rinichi. Rezultatele experimentelor au arătat că sistemul propus funcționează bine și are o acuratețe semnificativă în predicția prezenței pietrelor la rinichi. Acest lucru demonstrează utilitatea și eficacitatea metodei sugerate pentru diagnosticarea acestei afecțiuni. O direcție promițătoare în domeniul diagnosticului medical este integrarea tehnologiilor avansate precum rețelele neuronale și JADE într-un sistem multi-agent. Cu toate acestea, sunt necesare eforturi continue pentru a valida sistemul și pentru a extinde eșantionul de pacienți pentru a confirma că este fiabil și generalizabil.

## 2. Summary

This paper presents a method for kidney stone prediction using a multi-agent system employing advanced technologies such as Java Agent Development Framework (JADE) and neural networks. The aim of this study is to create a system that, based on clinical data and test results, could predict the presence or absence of kidney stones in patients when a new set of urine tests is added to the system.

To meet this objective, a dataset consisting of 79 patient analyses was used; these analyses included the properties of the patients' urine:

* Specific gravity, or the gravity of urine in relation to water;
* pH, which is the negative logarithm of the hydrogen ion;
* osmolarity (mOsm), a unit that is used in physical chemistry but not in biology and medicine. Osmolarity is proportional to the concentration of molecules in solution;
* conductivity. One Mho is equal to one Ohm. Conductivity is proportional to the concentration of ionically charged substances present in solution;
* urea level in millimoles/litre;
* calcium level (CALC) in millimoles/litre.

A multi-agent system consisting of four agents

* one agent for adding a new analysis;
* an agent for managing the other agents;
* an agent for selecting analyses with values chosen by the user;
* an agent for kidney stone prediction using a neural network;

is used to manage and generate kidney stone prediction based on analyses. The agents collaborate with each other to manage the analyses.

Machine learning methods based on neural networks are used in the proposed system to analyze the data and identify patterns or features that are relevant to the presence of kidney stones. The experimental results showed that the proposed system works well and has significant accuracy in predicting the presence of kidney stones. This demonstrates the usefulness and effectiveness of the suggested method for diagnosing this condition. A promising direction in the field of medical diagnosis is the integration of advanced technologies such as neural networks and JADE into a multi-agent system. However, continued efforts are needed to validate the system and expand the sample of patients to confirm that it is reliable and generalizable.

## 3. Introducere

Ciclul de dezvoltării unei aplicații software cuprinde patru faze de bază: planificarea, analiza, proiectarea și implementarea. O metodologie bine pusă la punct economisește efort și timp. O metodologieacționează ca o "rețetă", ajutând proiectantul să implementeze soluția prin specificarea unora dintre etapele procesului, lăsându-le pe altele la creativitatea proiectantului [1]. Importanța unei metodologii în dezvoltareaunei aplicații software nu poate fi supraestimată.Tehnologiile bazate pe agenți sunt în prezent considerate niște mijloace alternative viabile (uneori chiar și mai fezabile) de implementare a aplicațiilor la nivel de întreprindere care să opereze împreună cu alte sisteme eterogene. Această paradigmă de programare orientată pe agenți oferă abstracțiile software de nivel înalt care sunt necesare pentru gestionarea de aplicații cu o complexitate medie dar, mai ales, și pentru că au fost inventate pentru a face față interoperabilității și distribuției [2].

Ingineria software bazată pe agenți este relativ nouă și poate fi considerată ca o extensie a programării orientate pe obiecte [3] . Deși această paradigmă oferă un mijloc eficient de rezolvare a problemelor în unele aplicații unde alte tehnici pot fi ineficiente sau greoaie, există în prezent o lipsă de aplicații complexe bazate pe agenți. Această deficiență a fost subliniată ca fiind una dintre principalele bariere în calea adoptării pe scară largă a tehnologiei agenților [4]. Prin urmare, continuarea dezvoltării și îmbunătățirii metodelor pentru dezvoltarea de aplicații multi-agent este imperativă, meritând o atenție sporită [5].

Marea majoritate a metodelor actuale încearcă să adapteze programarea orientată pe obiecte la programarea bazată pe agenți [6] și urmează o abordare de sus în jos. Adaptarea metodologiilor de proiectare orientate pe obiecte la dezvoltarea de sisteme multi-agent are mai multe dezavantaje ce rezultă în principal din faptul că obiectele și agenții oferă abstracțiuni diferite care trebuie gândite din puncte de vedere difeite.

Scopul acestei lucrări este de a studia și a aplica funcționalitățile puse la dispoziție de platforma JADE pentru a face design-ul, proiectarea și implementarea a unei aplicații medicale bazate pe agenți care integrează concepte de inteligență artificială pentru a gestiona analize legate de urină și pentru a prezice existența sau nu a unor pietre la rinichi. Design-ul aplicației se concentrează pe platforma JADE. JADE este abrevierea de la Java Agent DEvelopment Framework și a fost dezvoltat de Telecom Italia Lab (TILAB) din Italia, în conformitate cu FIPA (Foundation for Intelligent Physical Agents) [7]. FIPA este o organizație non-profit organizație orientată spre producerea de standarde pentru interoperarea de sisteme eterogene de agenți eterogeni. În esență, JADE este un framework scris în întregime în limbajul Java ce ușurează implementarea sistemelor multi-agent prin furnizarea unui set de instrumente grafice specifice sistemelor multi-agent. Mai multe informații despre JADE pot fi găsite la [8]. Prin concentrarea specifică asupra pe platforma JADE în faza de design și proiectare, proiectantul poate trece direct la implementare, acest lucru ducând la câștigarea de timp oferind o imagine mult mai clară asupra modului în care progresează implementare.

Programarea orientată pe agenți (POA) este o paradigmă a programării oarecum nou apărută,ideea de a centra software-ul în jurul conceptului de agent fiind introdusă de Yoav Shoham în cadrul studiilor sale de inteligență artificială în 1990. Acesta propune metoda de a crea software folosind un set de agenți care interacționează între ei pentru a rezolva probleme care depășesc capacitățile lor individuale, deci este de înțeles că această paradigmă este centrată pe conceptul de agenți software, comparativ cu programarea orientată pe obiecte, paradigmă care este centrată pe obiecte.

Există mai multe framework-uri de POA ce implementează paradigma de programare a lui Shoham, una dintre ele fiind și unul dintre subiectele acestei lucrări, care este un software ce ușurează dezvoltarea de aplicații de tip multi-agent fiind conform cu specificațiile FIPA (Foundation for Intelligent Physical Agents). JADE utilizează implementare în Java ce permite o eficiență decentă la execuție, flexibilitatea agenților, reutilizarea software-ului și realizarea de arhitecturi diferite de tipul agenților multipli [9].

### 3.1. Obiectivul Cercetării

Obiectivul principal al cercetării este acela de a explora și de a dezvolta câteva abordări eficiente pentru programarea orientată pe agenți în sisteme cu agenți multipli utilizând platforma JADE, folosind funcționalitățile framework-ului pentru a analiza modul de interacțiuniune între agenți, punându-I într-un mediu multi-agent pentru a observa și facilita comunicația și coordonarea lor astfel încât dezvoltarea aplicației pentru predicția pietrelor la rinichi folosind și rețele neuronale care sunt “învățate” folosind 79 de analize ale urninei, cu 6 caracteristici să respecte conceptele paradigmei de programare pe agenți și să folosească ce este potrivit dintre funcționalitățile JADE în combinație cu un concept al inteligenței artificiale, învățarea supervizată, folosind rețele neuronale și date gata clasificate de către experți umani.

### 3.2. Metodologii

Metodologiile propuse pentru această lucrare au fost concepute pentru a aborda în mod eficient următoarele obiective:

1. Analiza Paradigmei de Programare Orientată pe Agenți:

* Realizarea unei analize amănunțite a paradigmei de programare orientată pe agenți prin studiul materialelor disponibile.
* Compararea acestei paradigme cu programarea orientată pe obiecte pentru identificarea diferențelor și asemănărilor.

1. Analiza Framework-ului JADE:

* Evaluarea detaliată a caracteristicilor și a funcționalităților cheie ale framework-ului JADE.
* Înțelegerea modului de funcționare a JADE și capacitatea sa de integrare în sistemele multi-agent.

1. Descrierea pietrelor la rinichi:
   * Înțelegerea a ce sunt pietrele la rinichi.
   * Cauzele aparițiilor pietrelor la rinichi.
   * Simptomele cauzate de pietrele la rinichi.
   * Metode de diagnosticare a pietrelor la rinichi.
   * Metode de a trate apariția pietrelor la rinichi.
2. Descrierea conceptului de învățare supervizată
   * Explicarea paradigmei de învățare supervizată
   * Rețele neuronale
   * Aspecte ce trebuie luate in considerare
3. Proiectarea și Implementarea unui sistem multi-agent:

* Dezvoltarea unei simulări pentru o aplicație medicală utilizând JADE și concepte de inteligență artificială bazate pe învățarea supervizată și rețele neuronale ce folosesc date deja clasificate de experți umani.
* Implementarea unui sistem cu mai mulți agenți cu scopul de a prezice, pe baza analizelor de urină, dacă pacientul suferă sau nu de pietre la rinichi.

Prin îndeplinirea acestor etape, se urmărește atingerea obiectivelor propuse și contribuirea la corelarea și îmbunătățirea cunoștințelor în domeniul programării orientate pe agenți în sistemele multi-agent. Aceasta va aduce soluții practice și perspective inovatoare prin intermediul studiului de materiale relevante și implementarea aplicației propuse.

### 3.3. Structura lucrării

Înainte de a explora detaliile și conținutul fiecărui capitol, este esențial să înțelegem cum este organizată această lucrare. Astfel, acest subcapitol va prezenta o privire de ansamblu asupra structurii generale a lucrării și a fluxului de informații. Este conceput pentru a furniza cititorului o orientare clară în ceea ce privește direcția și scopul fiecărei secțiuni a lucrării.

* **Introducere**: În acest capitol, sunt prezentate obiectivele cercetării, metodologiile utilizate și structura generală a lucrării;
* **Concepte și Tehnologii Utilizate**: Acest capitol explorează paradigmele de programare orientată pe agenți, alături de alte concepte și tehnologii relevante pentru proiect.
* **Medicină**: Aici sunt discutate aspectele medicale legate de pietrele la rinichi, inclusiv cauzele, simptomele, diagnosticul și tratamentul.
* **Implementarea Sistemului Multi-Agent**: Această secțiune detaliază structura proiectului, setul de date utilizat și arhitectura aplicației, alături de rolurile și funcționalitățile fiecărui agent.
* **Rezultate Obținute**: În acest capitol sunt prezentate rezultatele obținute în urma cercetării și implementării sistemului multi-agent.
* **Discuții**: Aici sunt discutate implicațiile, limitările și interpretările rezultatelor, alături de eventualele direcții pentru lucrările viitoare.
* **Concluzie**: Această secțiune reiterează principalele concluzii și contribuții ale lucrării.
* **Bibliografie**: O listă a tuturor surselor utilizate în cercetare, prezentată în conformitate cu standardele bibliografice.

## 4. Concepte și tehnologii utilizate

În acest capitol vor fi prezentate conceptele și tehnologiile principale ce au fost folosite pentru elaborarea acestei lucrări.

### 4.1. Paradigma de programare orientată pe agenți

O nouă paradigmă în programare este definită de termenul programare orientată pe agenți (AOP). Un agent este conceptul compozițional central al acestui cadru computațional și este considerat un element software care are abilități mentale și capacități de comunicare. Orice lucru care are capacitatea de a utiliza senzori pentru a detecta mediul înconjurător și efectoare pentru a manipula acest mediu este considerat un agent.Un agent uman posedă diferite organe, inclusiv mâini, picioare și gură, pentru efectoare, și organe senzoriale, inclusiv ochi, urechi, nas, limbă și piele, în paralel cu senzorii. Un agent robotic utilizează numeroase motoare și actuatori pentru efectoare, camere de luat vederi și telemetre cu infraroșu pentru senzori. Șirurile de biți codificate alcătuiesc programele și comportamentele unui agent software.AOP este văzută ca o specializare a programării orientate-obiect (OOP). Cu toate acestea, există o distincție semnificativă între AOP și OOP, în special faptul că obiectele și agenții au un grad diferit de autonomie.

În AOP, lucrurile cunoscute sub numele de agenți lucrează împreună pentru a atinge obiective specifice. Un agent poate fi o entitate independentă care își alege singur următorul pas fără intervenția unui utilizator, sau poate fi controlabil, funcționând ca intermediari între utilizator și un alt agent. Programarea AOP se face într-un mod abstract. Ingineria software orientată pe agenți (AOSE) este considerată o nouă paradigmă în cercetarea ingineriei software. Cu toate acestea, sunt necesare instrumente și metodologii puternice și ușor de utilizat pentru a crea o nouă paradigmă pentru industria software. Shoham [10] a propus un sistem AOP care are trei părți. În primul rând, un sistem logic care descrie starea psihologică a agenților; apoi, un limbaj de program interpretat pentru programarea agenților; și, în cele din urmă, un proces de „agentificare”, care este folosit pentru a compila programele agenților în sisteme executabile de nivel inferior [10].

Agenții își exprimă dorința de a face o acțiune. Acest lucru îi diferențiază de obiectele care invocă direct acțiunile altor obiecte. De asemenea, agenții au deseori interese diferite, astfel încât ar putea fi dăunător pentru un agent să execute o cerere de acțiune de la un alt agent [11].

Un agent, conform lui Jennings [11], este un obiect care are capacitatea de a prezenta următoarele:

* Autonomia este capacitatea de a funcționa fără intervenția directă a altora și de a avea un anumit grad de control asupra stării interne și a acțiunilor.
* Reactivitatea (structurală) este capacitatea de a percepe și de a răspunde în mod regulat la schimbările din mediul nostru înconjurător.
* Abilitatea socială se referă la capacitatea de a interacționa cu alți agenți, precum și cu indivizi, folosind un anumit tip de limbaj de comunicare între agenți.
* Proactivitate: Capacitatea de a avea un comportament care este orientat spre un obiectiv, luând inițiativa în loc să acționezi doar ca răspuns [12].

Conform cercetătorilor din domeniul inteligenței artificiale, un agent este un sistem informatic care posedă caracteristicile menționate mai sus și care poate fi fie conceptualizat, fie implementat folosind concepte care se referă la oameni, cum ar fi cunoașterea, credința, intenția și obligația [13]. Caracteristicile suplimentare ale agentului includ următoarele:

* Mobilitate: capacitatea de a se mișca într-o rețea electronică.
* Veridicitate: Ideea că nu ar trebui să transmiteți informații false în cunoștință de cauză.
* Benevolența este presupusul că nu ai scopuri contradictorii.
* Raționalitatea este ideea de a acționa pentru a-și atinge scopurile, mai degrabă decât să le împiedice.

Agenții inteligenți pot fi utilizați în multe sisteme complexe pentru a îmbunătăți capacitățile și eficiența acestora. Dezvoltarea continuă a sistemelor complexe utilizate precum internetul obiectelor, aplicațiile software a telefonului, etc. necesită programarea orientată pe agent și domeniul agenților. Pentru a crește potențialul agenților inteligenți și pentru a permite dezvoltarea agenților inteligenți mai complexi în termen scurt, este esențială crearea unui limbaj specific de programare în domeniul agenților. Acest lucru permite crearea sistemelor care oferă utilizatorilor finali mai multe funcții.

### 4.2. Programarea orientată pe agenți vs. programarea orientată pe obiecte

În această secțiune, abordăm în detaliu diferențele dintre POA și POO folosind tabelul 1 [10].

|  |  |  |
| --- | --- | --- |
| **Caracteristici** | **POO** | **POA** |
| Unitatea de bază | obiect | agent |
| Parametrii care definesc starea unității de bază | fără constrângeri | convingeri, angajamente, capacități, alegeri… |
| Procesul de calcul | metode de transmitere a mesajelor și de răspuns | metode de transmitere a mesajelor și de răspuns |
| Tipuri de mesaje | fără constrângeri | a informa, a cere, a oferi, a promite, a refuza… |
| Constrângeri asupra metodelor | fără constrângeri | onestitate, consistență, coerență... |

Tabel 1. POA vs POO

Aici sunt prezentate diferențiatorii care disting agenții de obiecte: În timp ce agenții pot vorbi folosind o limbă de comunicare a agenților ACL sau ICL, obiectele pot comunica folosind interfețe fixe de metodă. Clasele de obiecte sunt de obicei proiectate pentru a face previzibile cumpărarea și vânzarea de piese care pot fi folosite din nou.

De obicei, agenții își stabilesc comportamentul în funcție de obiectivele, circumstanțele și conversațiile lor cu alți agenți. Gândirea bazată pe agenți nu tolerează acest lucru, dar implementările OO pot include comportament nedeterminist. Deoarece agenții sunt interactivi și autonomi, nu este necesară integrare pentru lansarea fizică a unei aplicații. În schimb, obiectele sunt tradițional pasive, iar procedurile lor sunt invocate de un fir de control al apelantului. Atunci când o entitate se adresează doar altor părți ale sistemului, cuvântul „autonomie” este rar folosit [15].

Un strat suplimentar de componente software în sistemele multiagent poate fi descris în mod natural ca obiecte și grupuri de obiecte. Acesta este cadrul de bază care reprezintă asistența pentru agenții alcătuiți din componente obiect.

O parte din mecanismele și ideologiile utilizate de obiecte sunt aplicate de agenți. De fapt, mulți ingineri software susțin cu fervoare crearea de agenți din obiecte, bazând arhitectura sistemelor bazate pe agenți pe aceleași tipuri de mecanisme de suport întâlnite în sistemele software OO. De exemplu, este rezonabil să se reprezinte multe dintre componentele și structurile agenților ca obiecte. Acestea ar putea include politici de discuție, codificări, ontologii și elemente de vocabular în limbajul de comunicare al agenților, precum și nume ale agenților [16].

Agenții, în cadrul programării orientate pe agenți, sunt importanți deoarece:

* aceștia oferă un mecanism care produce un comportament emergent în cadrul unei arhitecturi altfel statice;
* oferă un mijloc de raționalizare a fluxului de control într-un sistem distribuit masiv;
* codifică cele mai bune practici de organizare a obiectelor care cooperează concomitent.

### 4.3. Java Agent DEvelopment Framework

JADE este un cadru software care este complet implementat în Java. Printr-un middle-ware care respectă standardele FIPA și printr-un set de instrumente grafice care sprijină fazele de depanare și implementare, acesta facilitează implementarea sistemelor multi-agent. Platforma de agenți poate fi instalată pe mai multe mașini, cu diferite sisteme de operare, iar configurația poate fi gestionată de la distanță prin intermediul unei interfețe grafice. Chiar și în timpul execuției, configurația poate fi modificată prin mutarea agenților de pe o mașină pe alta după cum este necesar. JADE este dezvoltat în limbajul Java și are nevoie de versiunea 1.4 a JAVA (mediul de execuție sau JDK) pentru a funcționa [15].

Câteva dintre caracteristicile principale ale structurii JADE:

* Este compatibil cu Java deci aplicațiile dezvoltate cu JADE pot fi rulate pe o gamă largă de platforme hardware și software, datorită platformei Java.
* JADE oferă o gamă largă de facilități și instrumente pentru gestionarea agenților, comunicarea între aceștia și organizarea activităților lor, ajutând la dezvoltarea sistemelor care includ mai mulți agenți.
* JADE oferă un model robust și adaptabil pentru definirea și implementarea agenților software. Acest model abordează concepte precum comportamente, agenți, mesaje și servicii, ceea ce facilitează dezvoltarea și gestionarea agenților într-un sistem care include mai mulți agenți.
* Comunicare între agenți: JADE oferă mecanisme avansate de comunicare între agenți, cum ar fi gestionarea agenților, serviciile de căutare și descoperire și transmiterea de mesaje. În cadrul unui sistem multi-agent, agenții pot lucra împreună și interacționa eficient cu ajutorul acestor facilități.
* Există protocoale de comunicare care sunt suportate FIPA (Fundația pentru agenți fizici inteligenți) este unul dintre protocoalele de comunicare standardizate pe care JADE le susține. Aceste protocoale permit agenților să interacționeze și să lucreze în funcție de standarde bine definite.
* Mesajele ACL sunt transportate efectiv în cadrul aceleiași platforme de agent. De fapt, în loc să fie trimise sub formă de șiruri de caractere, mesajele sunt codificate ca obiecte Java pentru a evita necesitatea proceselor de marshalling și unmarshalling. În informatică, marshalling este procesul de transformare a reprezentării în memorie a unui obiect într-un format de date adecvat pentru stocare sau transmitere, în special între diferiți timpi de execuție. Mesajul este transformat automat în/din protocolul de transport, codificarea și sintaxa conforme cu FIPA atunci când traversează granițele platformei. Cei care implementează agentul trebuie pur și simplu să se ocupe de obiecte Java, prin urmare această schimbare este transparentă pentru ei.
* API-ul și plugin-urile JADE permit dezvoltatorilor să extindă și să personalizeze funcționalitățile [15].

În centrul modelului AOP al JADE se află faptul că API-ul său suportă limbajul standard de comunicare a agenților FIPA.Specificațiile FIPA sunt un set de orientări concepute pentru a facilita comunicarea între diverși agenți și serviciile pe care aceștia le pot reprezenta.Tehnologia bazată pe agenți și interoperabilitatea standardelor cu alte tehnologii sunt promovate de FIPA, un organism de standardizare al IEEE Computer Society.În scopul de a crea specificații pentru standardele de software pentru agenți eterogeni și interactivi și pentru sistemele bazate pe agenți, FIPA a fost înființată în 1996 ca organizație elvețiană. De la înființarea sa, FIPA a contribuit la crearea de standarde pentru agenți și a sprijinit mai multe proiecte și reuniuni care au contribuit la progresul și aplicarea tehnologiei agenților. În plus, noile generații de tehnologie Web/Internet și specificațiile aferente aduc în prim plan multe dintre conceptele care au fost articulate pentru prima dată în cadrul FIPA.

JADE este un software Java compus din mai multe pachete Java care oferă programatorilor de aplicații funcționalități gata făcute, precum și interfețe abstracte pentru sarcini unice, specifice. Datorită numeroaselor sale caracteristici atractive, printre care se numără serializarea obiectelor, Reflection API și Remote Method Invocation (RMI), care sunt deosebit de potrivite pentru programarea orientată pe obiecte în contexte eterogene distribuite, Java a fost limbajul de programare preferat.

JADE este compus din următoarele pachete principale:

* *jade.core* este utilizat pentru a implementa nucleul sistemului. Acesta conține clasa Agent, pe care programatorii de aplicații trebuie să o extindă. De asemenea, subpachetul *jade.core.behaviors* are o ierarhie de comportamente. Comportamentele realizează obiectivele sau sarcinile unui agent. Acestea sunt unități logice de activitate care pot fi executate concomitent și asamblate în diferite moduri pentru a obține modele de execuție complicate. Programatorii de aplicații creează comportamentele și căile de execuție ale agentului care conectează operațiunile definite ale agentului;
* *jade.lang.acl*pentru a procesa limbajul de comunicare al agenților în conformitate cu specificațiile standard FIPA;
* *jade.content* conține un set de clase pentru a susține ontologiile definite de utilizator și limbi de conținut definite de utilizator;
* *jade.domain* conține toate clasele Java care reprezintă Agentul de gestionare a agenților definite de standardul FIPA, în special agenții AMS și DF, care furnizează servicii de ciclu de viață Subpachetul jade.domain.FIPAAgentManagement conține ontologia FIPA-Agent-Management și toate clasele care reprezintă conceptele sale. Subpachetul jade.domain.JADEAgentManagement conține, în schimb, extensiile JADE pentru AgentManagement (adulmecarea mesajelor, controlul ciclului de viață al agenților), inclusiv ontologia și toate clasele care reprezintă conceptele sale. Subpachetul *jade.domain.introspection* conține conceptele utilizate pentru domeniul de discurs dintre instrumentele JADE (de exemplu, Sniffer și Introspector) și nucleul JADE. Subpachetul *jade.domain.mobility* conține toate conceptele utilizate pentru a comunica despre mobilitate;
* *jade.gui* oferă o colecție de clase generice care poate fi utilizată pentru a crea interfețe grafice care afișează și modifică identificatorii agentului, descrierile agentului și mesajele ACL;
* *jade.mtp* conține o interfață Java pe care fiecare protocol de transport de mesaje ar trebui să îl implementeze pentru a fi ușor de integrat în cadrul JADE;
* *jade.wrapper* oferă un “wrapper” al pachetului JADE de nivel superior care permite utilizarea JADE ca o bibliotecă, în care aplicațiile Java externe lansează agenți JADE și containere de agenți.

Câteva utilitare care facilitează administrarea platformei și dezvoltarea de aplicații sunt incluse în JADE. Fiecare instrument este găzduit într-un subpachet jade.tools diferit. Sunt accesibile următoarele instrumente:

* *Agentul de gestionare la distanță, sau RMA (Remote Management Agent)*, acționează ca o consolă grafică pentru gestionarea și controlul platformei. Un argument în linia de comandă („-gui”) poate iniția un RMA în prima instanță, dar activează și multe interfețe grafice;
* *Dummy Agent,* compus dintr-un agent JADE sub-ierarhic și o interfață grafică de utilizator, este un instrument de monitorizare și de depanare. Pentru a permite înregistrarea și repetarea conversațiilor agentului, interfața grafică permite utilizatorilor să creeze și să trimită mesaje ACL către alți agenți. De asemenea, aceasta afișează o listă a tuturor mesajelor ACL trimise și primite, completată cu informații privind marcajul de timp;
* *Un agent numit Sniffer* are capacitatea de a intercepta mesajele ACL în timp ce sunttrimise și de a le reprezenta vizual folosind o notație asemănătoare cu diagramele de secvență UML. Văzând modul în care comunicația dintre cei patruagenți din aplicația creată în această lucrare schimbă mesaje ACL, snifferul ajută la depanare și observarea comportamentelor agenților din cadrul aplicației;
* *DF (Directory Facilitator)* GUI este o interfață grafică completă prin intermediul căreia baza de cunoștințe a unui DF poate fi gestionată cu ușurință, poate fi federată cu alți DF și poate fi înregistrată, radiată, modificată și căutată de la distanță. Acest lucru permite DF să implementeze o rețea de domenii și subdomenii;
* *SocketProxyAgent* este un agent simplu, care servește ca un gateway bidirecțional între o conexiune TCP/IP standard și o platformă JADE. Mesajele ACL sunt transformate în șiruri ASCII de bază înainte de a fi transmise printr-o conexiune socket folosind serviciul de transport proprietar JADE. Invers, această conexiune TCP/IP poate fi utilizată pentru a introduce mesaje ACL în platforma JADE. De exemplu, gestionarea firewall-urilor de rețea sau facilitarea interacțiunii platformei cu applet-urile Java dintr-un browser web sunt două utilizări ale acestui agen [16].

Exemplu de cod JADE din cadrul aplicației:



Figura 1. Cod JADE

### 4.4. Învățare supervizată

În domeniul învățării automate, învățarea supervizată implică utilizarea seturilor de date etichetate pentru a antrena algoritmi de recunoaștere a tiparelor și de predicție a rezultatelor. Algoritmii de învățare supervizată primesc o datele etichetate de către experți umani pentru a înțelege legătura dintre datele de intrare și cele de ieșire, spre deosebire de învățarea nesupravegheată care constă în învățareastructurii datelor de intrarefără a primi instrucțiuni explicite.

Paradigma de învățare numită învățare supervizată antrenează un model folosind date de intrare (cum ar fi un analizele urinei în cadrul aplicației din această lucrare) și o valoare de ieșire dorită, care este etichetată de către om. După procesarea setului de date de intrare, este creată o funcție de clasificare a datelor noi către valorile de ieșire anticipate (în cazul aplicației din această lucrare, valoarea de ieșire va fi predicția de prezență a pietrelor la rinichi) [20]. Algoritmul va fi capabil să determine valorile de ieșire pentru evenimente pe care nu le-a întâlnit în setul de date de antrenare (cum ar fi un nou set de analize ale urinei). Aceasta înseamnă că sistemultrebuie să facă generalizări din datele de instruire la scenarii necunoscute.Principalele tipuri de învățare supervizată:

* Clasificare: în clasificare, un algoritm este instruit să atribuie o clasă sau o etichetă fiecărei instanțe de date dintr-un set predeterminat de categorii discrete. Scopul este de a dobândi un model care să poată generaliza și face predicții precise pentru date noi, necunoscute. Prin urmare, clasificarea implică atribuirea unei etichete specifice unui set de date pe baza caracteristicilor pe care le-a remarcat (în cazul aplicației din această lucrare, va fi prezisă prezența pietrelor la rinichi).
* Regresia: în regresie,un algoritm este instruit să prezică valori continue pe baza datelor de intrare. Scopul este de a descoperi relația matematică care există între variabilele de intrare și de ieșire, astfel încât să poată prezice corect valoarea de ieșire pentru date necunoscute. În general, rezultatul este o funcție continuă care are capacitatea de a prezice un număr real.

### 4.4.2 Aspecte ce trebuie luate în considerare în învățarea supervizată

Dezvoltarea unor modele de învățare supravegheată de succes care să aibă performanțe bune pe date nevăzute necesită abordarea acestor probleme din subcapitolele următoare:

#### Deviație versus variație

Termenul „bias” descrie greșeala care rezultă din utilizarea unui model simplist pentru a aproxima o problemă complicată din lumea reală. Subadaptarea apare atunci când un model cu un grad ridicat de distorsiune nu reușește să surprindă relațiile importante dintre caracteristici și rezultatele dorite.

Sensibilitatea modelului la variațiile infime din setul de instruire se numește varianță. Atunci când un model are o varianță ridicată, acesta se poate supraadapta și poate reprezenta zgomotul din setul de instruire mai degrabă decât rezultatele dorite.

Să presupunem că avem la dispoziție mai multe seturi de date la fel de bune. Dacă un algoritm de învățare prezice în mod constant o ieșire incorectă pentru o anumită intrare 𝑥 după ce a fost antrenat pe toate aceste seturi de date, atunci acesta este distorsionat pentru acea anumită intrare 𝑥. În cazul în care un algoritm de învățare este antrenat pe seturi de instruire distincte și prezice valori de ieșire variate, acesta are o varianță ridicată pentru acea anumită intrare 𝑥. Suma varianței și a prejudecății algoritmului de învățare determină eroarea de predicție a unui clasificator instruit[21]. În general, prejudecata și varianța sunt compromisuri. Pentru ca un algoritm de învățare să se potrivească în mod eficient datelor, acesta trebuie să fie „flexibil” și să aibă un bias minim.Cu toate acestea, un algoritm de învățare prea flexibil va avea ca rezultat o variație ridicată, deoarece se va adapta diferit la fiecare set de date de instruire. Capacitatea de a modifica acest compromis între bias și varianță - fie automat, fie prin oferirea unui parametru de bias/varianță pe care utilizatorul îl poate modifica - este o caracteristică esențială a multor tehnici de învățare supervizată.

Este important să se ia în considerare compromisul dintre bias și varianță. Este ideal să avem un model cu o varianță scăzută și un bias minim, dar acest lucru este adesea dificil de realizat.

#### Cantitatea și complexitatea datelor de instruire

Un algoritm de învățare „inflexibil”, cu un bias ridicat și o varianță redusă, poate învăța funcția reală dintr-o cantitate minimă de date, dacă aceasta este simplă. Cu toate acestea, funcția poate învăța doar cu o cantitate mare de date de instruire combinate cu un algoritm de învățare „flexibil” care are o prejudecată scăzută și o varianță ridicată dacă funcția adevărată este foarte complexă (de exemplu, implică interacțiuni complexe între multe caracteristici de intrare diferite și se comportă diferit în diferite părți ale spațiului de intrare).

* **Calitatea setului de date de învățare** - în general, mai multe date de instruire permit modelului să învețe mai bine și să generalizeze mai eficient la date noi. Ajută la reducerea supraadaptării prin furnizarea unei reprezentări mai cuprinzătoare a distribuției datelor subiacente.După un anumit punct, beneficiile adăugării de date scad. Acest lucru este valabil mai ales dacă datele suplimentare sunt redundante sau nu aduc informații noi.Un număr mai mare de cazuri dintr-un set de date mai mare poate contribui la capacitatea modelului de a discerne între zgomot și modelele reale de bază.Cu mai multe date, modelul este expus la o varietate mai mare de scenarii, ceea ce îi îmbunătățește capacitatea de a generaliza la date inedite. Mai multe date înseamnă o putere de procesare și un timp de instruire mai mare. Pentru a gestiona seturi masive de date, sunt adesea necesare acceleratoare hardware, cum ar fi GPU, și algoritmi eficienți.Integritatea datelor trebuie menținută prin tratarea adecvată a datelor lipsă (imputare, ștergere sau utilizarea unor modele care pot gestiona datele lipsă).În procesul de învățare supervizată, calitatea etichetelor este extrem de importantă. Etichetele zgomotoase sau incorecte pot induce în eroare procesul de instruire, ceea ce duce la o performanță slabă a modelului.
* **Complexitatea setului de date de învățare -** Datele cu dimensionalitate mare pot fi mai informative, dar există o șansă mai mare de supraadaptare, în special dacă anumite aspecte sunt duplicate sau neimportante.Crearea unor caracteristici semnificative prin cunoașterea domeniului și tehnici precum caracteristicile polinomiale sau interacțiunile pot îmbunătăți performanța modelului.Datele provenite din diverse surse sau care prezintă caracteristici diferite pot face ca instruirea să fie mai dificilă, dar pot duce, de asemenea, la un model mai robust.Modelul poate suferi din cauza existenței unor valori aberante (puncte de date ciudate) și a zgomotului (erori aleatorii). Este imperativ să se efectueze o preprocesare adecvată, cum ar fi reducerea zgomotului și detectarea valorilor aberante.Pentru a surprinde modele în date cu conexiuni neliniare între caracteristici și variabilele țintă, sunt necesare modele mai sofisticate (cum ar fi rețelele neuronale).
* **Tehnici de gestionare a complexității:**
  + Reducerea dimensionalității: Metode precum analiza componentelor principale (PCA) sau t-SNE pot ajuta la reducerea numărului de caracteristici, menținând în același timp datele esențiale.
  + Regularizarea: Prin penalizarea coeficienților mari și încurajarea modelelor mai simple, termenii de regularizare sunt adăugați la funcția de pierdere pentru a ajuta la prevenirea supraajustării.

Doi factori cheie care afectează crearea și funcționalitatea modelelor de învățare supravegheată sunt volumul și complexitatea datelor de instruire. Pentru a maximiza învățarea, generalizarea și eficiența, trebuie să se găsească un echilibru.

#### Dimensionalitatea spațiului de intrare

Dimensionalitatea spațiului de intrare reprezintă o a treia problemă. Chiar dacă funcția corectă depinde de un număr limitat de caracteristici, învățarea funcției poate fi dificilă dacă vectorii de caracteristici de intrare au dimensiuni uriașe. Acest lucru se datorează posibilității ca algoritmul de învățare să devină confuz din cauza numeroaselor dimensiuni „suplimentare”, ceea ce duce la o varianță ridicată. Prin urmare, este de obicei necesar să se regleze clasificatorul pentru a avea o varianță scăzută și o polarizare ridicată atunci când se lucrează cu date de intrare de dimensiuni mari. În utilizarea reală, precizia funcției învățate va crește probabil dacă inginerul este capabil să excludă manual caracteristicile străine din datele de intrare. În plus, există numeroși algoritmi de selecție a caracteristicilor care au ca scop separarea caracteristicilor importante de cele neimportante.

#### Zgomot în valorile de ieșire

Nivelul de zgomot în valorile de ieșire preconizate (variabilele țintă) reprezintă o a patra problemă. Algoritmul de învățare ar trebui să se abțină de la încercarea de a găsi o funcție care să se potrivească cu precizie instanțelor de învățare dacă valorile de ieșire preconizate sunt frecvent eronate (din cauza unei greșeli umane sau a unor probleme ale senzorilor). Supraadaptarea apare atunci când se încearcă cu prea multă atenție adaptarea datelor. În cazurile în care funcția este prea complexă pentru modelul de învățare, se poate produce o supraadaptare chiar și în absența erorilor de măsurare (zgomot stocastic). În acest scenariu, porțiunea nemodelabilă a funcției țintă „corupe” datele de instruire; acest lucru este cunoscut sub numele de zgomot determinist.În cazul în care este prezent oricare dintre aceste tipuri de zgomot, este mai bine să se opteze pentru un estimator cu bias mai mare și varianță mai mică.

### 4.4.3 Algoritmi folosiți în învățarea supervizată

Cei mai utilizați algoritmi de învățare sunt:

* Mașini de suport-vector (SVM) sunt utilizate pentru clasificare și regresie. Ele găsesc hiperplanul care separă cel mai bine diferitele clase de date într-un spațiu de caracteristici de dimensiuni mari. SVM-urile sunt eficiente în spații de dimensiuni mari și se descurcă bine când raportul dintre dimensiunea caracteristicilor și numărul de eșantioane este mare. SVM-urile sunt folosite în recunoașterea imaginii, bioinformatică (ex: clasificarea proteinelor), detectarea spamului și clasificarea textului.
* Regresie liniară modelează relația dintre o variabilă dependentă și una sau mai multe variabile independente printr-o linie dreaptă. Aceasta este folosită pentru a face predicții bazate pe relații liniare între variabile. Regresia liniară este utilizată în economie pentru modelarea comportamentului pieței, în finanțe pentru predicția prețurilor acțiunilor și în științele sociale pentru analiza datelor experimentale.
* Regresie logistică este utilizată pentru clasificare binară. Aceasta estimează probabilitatea ca un eșantion să aparțină unei clase specifice, utilizând o funcție logistică (sigmoidă) pentru a transforma rezultatele liniare într-un interval între 0 și 1. Este folosită în medicină pentru predicția bolilor (ex: riscul de atac de cord), marketing pentru segmentarea clienților și analiză politică pentru previziuni electorale.
* Naive Bayes este un algoritm de clasificare probabilistic bazat pe teorema Bayes, presupunând independența între predictorii. Este simplu și eficient, în special pentru seturi mari de date. Este utilizat în filtrarea spamului, clasificarea textului și analiza sentimentelor.
* Analiza discriminantă liniară (LDA) este folosit pentru a găsi o combinație liniară de caracteristici care separă două sau mai multe clase de obiecte sau evenimente. Este un algoritm de clasificare liniară și de reducere a dimensiunilor. Este folosit în recunoașterea feței, diagnostic medical și clasificarea documentelor.
* Arbori de decizie modelează deciziile și consecințele lor printr-o structură de arbore. Fiecare nod intern reprezintă un "test" pe o caracteristică, fiecare ramură reprezintă rezultatul testului, iar fiecare nod terminal reprezintă o etichetă de clasă. Sunt utilizați în diagnoza medicală, analiza riscului financiar și în luarea deciziilor de marketing.
* Algoritmul K-nearest neighbor (KNN) este un algoritm de clasificare simplu, care atribuie o etichetă de clasă unui eșantion nou pe baza majorității claselor celor mai apropiați K vecini din setul de antrenament. Este un algoritm de învățare bazat pe instanțe. Este folosit în recunoașterea patternurilor, sisteme de recomandare și în detecția anomaliilor.
* Rețele neuronale (Perceptron multistrat) sunt modele inspirate de creierul uman, compuse din straturi de neuroni artificiali. Perceptronul multistrat (MLP) include un strat de intrare, unul sau mai multe straturi ascunse și un strat de ieșire, utilizând funcții de activare pentru a rezolva probleme complexe. Sunt folosite în recunoașterea imaginii și a vorbirii, traducerea automată și învățarea profundă pentru diverse aplicații AI. Acest concept a fost folosit și în această lucrare pentru a face o predicție folosind analizele urinei.
* Învățarea prin similitudine implică compararea noilor date cu datele cunoscute și luarea deciziilor bazate pe similaritățile identificate. Algoritmii de clustering, cum ar fi K-means, sunt adesea folosiți pentru această abordare. Este folosită în segmentarea pieței, bioinformatică pentru clasificarea genelor și în detecția fraudelor.

### 4.4.4 Rețele neuronale - perceptronul multistrat

O rețea neuronală, cunoscută și sub numele de rețea neuronală artificială sau rețea neuronală este un algoritm/mod de învățare automată care se inspiră din arhitectura și funcționarea rețelelor neuronale biologice din creierul animalelor. Neuronii artificiali, care sunt unități sau noduri de rețea care seamănă cu neuronii cuplați din creier, alcătuiesc o rețea neuronală artificială. Aceștia sunt legați între ei prin muchii care seamănă cu sinapsele creierului. După ce procesează semnale de la alți neuroni conectați, fiecare neuron artificial transmite un semnal către un alt neuron conectat. Ieșirea fiecărui neuron este determinată de o funcție neliniară, cunoscută sub numele de funcție de activare, care ia în considerare totalul intrărilor sale. „Semnalul” este o valoare reală. O pondere care variază pe parcursul învățării determină intensitatea semnalului la fiecare conexiune.

Neuronii sunt, în general, grupați în straturi. Este posibil ca diferitele straturi să își modifice intrările în moduri diferite. De la primul strat, cunoscut sub numele de strat de intrare, până la ultimul strat, cunoscut sub numele de strat de ieșire, semnalele pot trece printr-un număr de niveluri intermediare, cunoscute și sub numele de straturi ascunse. În cazul în care o rețea conține două sau mai multe straturi ascunse, aceasta este denumită în mod obișnuit o rețea neuronală profundă [22].



Figura 2. Perceptron multistrat

Metoda obișnuită de învățare a rețelelor neuronale este minimizarea empirică a riscului. Principiul din spatele acestei abordări este de a reduce riscul empiric - diferența dintre valorile obiectivului dintr-un set de date dat și rezultatul anticipat - prin optimizarea parametrilor rețelei. În mod obișnuit, pentru a estima parametrii rețelei se utilizează tehnici bazate pe gradient, cum ar fi backpropagation. Pentru a minimiza o funcție de pierdere predeterminată, rețeaua neuronală își actualizează iterativ parametrii în timp ce învață din datele de instruire etichetate în timpul fazei de învățare, rețeaua putând generaliza la date necunoscute anterior folosind acest mod [24]. Intrările pot fi ieșirile altor neuroni sau valorile caracteristice ale unui eșantion de date externe, cum ar fi documente sau imagini. Sarcina, cum ar fi identificarea unui obiect într-o imagine, este îndeplinită de către neuronii de ieșire finali ai rețelei neuronale. Pentru a obține ieșirea neuronului, folosim totalul ponderat al tuturor intrărilor, ajustat în funcție de ponderile conexiunilor dintre intrări și neuron. Majorăm această sumă cu un termen de polarizare. Activarea este un alt nume pentru acest total ponderat. Ieșirea se obține apoi prin trecerea acestei sume ponderate printr-o funcție de activare, care este adesea neliniară. Datele externe, cum ar fi documentele și fotografiile, servesc drept primele intrări. Ieșirile finale completează obiectivul, cum ar fi identificarea unui obiect într-o fotografie [25].

Un hiperparametru este un parametru constant, ceea ce înseamnă că valoarea sa este predeterminată înainte de a avea loc învățarea. Valorile parametrilor sunt obținute prin învățare. Numărul de straturi ascunse, dimensiunea lotului și rata de învățare sunt câteva exemple de hiperparametri. Valorile anumitor hiperparametri pot fi influențate de cele ale altor hiperparametri. De exemplu, numărul total de straturi poate afecta dimensiunea anumitor straturi [26]. Învățarea reprezintă adaptarea rețelei pentru a îndeplini o sarcină mai eficient, luând în considerare un eșantion de observații. Pentru a crește acuratețea rezultatului, învățarea presupune modificarea ponderilor și a hiperparametrilor rețelei. Minimizarea greșelilor observate realizează acest lucru. Atunci când examinarea mai multor date nu ajută la scăderea semnificativă a ratei de eroare, se spune că învățarea este completă. De obicei, rata de eroare nu scade la zero nici măcar după învățare. De obicei, rețeaua trebuie să fie modificată dacă, după învățare, rata de eroare este încă prea mare.În practică, acest lucru se realizează prin dezvoltarea unei funcții de cost care este evaluată în mod continuu pe parcursul procesului de învățare. Învățarea continuă atâta timp cât producția sa nu se îmbunătățește. Se obișnuiește să se definească costul ca o statistică a cărei valoare poate fi doar aproximată. Majoritatea modelelor de învățare pot fi considerate ca o simplă aplicație a teoriei estimării statistice și a optimizării [27].

#### Gradient descrescător stocastic

În învățarea automată și învățarea profundă, Stochastic Gradient Descent (SGD) este o metodă de optimizare foarte apreciată care este utilizată pentru a minimiza funcțiile de pierdere și pentru a determina parametrii ideali ai unui model.

#### Rata de învățare și optimizatorul ADAM

Numărul de acțiuni de corectare pe care modelul le efectuează pentru a ține cont de erorile din fiecare observație este determinat de rata de învățare. În timp ce o rată de învățare mai mică necesită mai mult timp, dar are potențialul de a obține o precizie mai mare, o rată de învățare mai mare reduce timpul de învățare, dar duce la o precizie finală mai slabă. Rata de învățare reprezintă, în esență, viteza cu care modelul își actualizează parametrii de învățare (ponderi și bias).

Atunci când se antrenează modele de învățare automată, rata de învățare este o componentă esențială. Eficacitatea procesului de instruire și performanța modelului pot fi influențate în mare măsură de ajustarea corectă a acestui hiperparametru. Optimizarea instruirii modelelor necesită o înțelegere și o capacitate de a modifica rata de învățare [28], un optimizator întâlnit foarte des pentru eficiența sa este ADAM. Bazată pe estimarea adaptivă a momentelor de ordinul întâi și doi, optimizarea Adam este o tehnică stocastică de coborâre a gradientului. Metoda este "*bine adaptată pentru probleme care sunt mari din punct de vedere al datelor/parametrilor, este eficientă din punct de vedere computațional, are un necesar redus de memorie și este invariantă la redimensionarea diagonală a gradienților*", potrivit *Kingma et al. (2014)*.

Cunoașterea caracteristicilor rețelelor neuronale artificiale este necesară pentru a le utiliza. Selectarea modelului se bazează pe aplicație și pe reprezentarea datelor. Numărul, tipul și conectivitatea straturilor de rețea, precum și dimensiunea și tipul de conexiune (completă, de tip pooling etc.) ale fiecăruia sunt exemple de parametri ai modelului. Învățarea modelelor complexe necesită mult timp. Există o mulțime de compromisuri între algoritmii de învățare. Pentru învățarea pe un set de date specific, aproape orice metodă va avea performanțe satisfăcătoare atunci când hiperparametrii sunt ajustați corect [28]. Cu toate acestea, este necesară o mare cantitate de încercări și erori atunci când se alege și se ajustează fin un algoritm pentru învățarea pe date necunoscute. Rețeaua neuronală rezultată poate deveni robustă dacă modelul, funcția de cost și metoda de învățare sunt alese cu atenție.

#### Metoda de inițializare Xavier

Metoda Xavier, denumită uneori inițializare Glorot sau inițializare Xavier, este o metodă frecvent utilizată pentru a inițializa greutățile unei rețele neuronale înainte de instruire. Evitând probleme precum dispariția sau extinderea gradienților, tehnica Xavier abordează dificultatea de a inițializa greutățile unei rețele neuronale într-un mod care să favorizeze o instruire eficientă. Aceste probleme pot apărea din cauza inițializării aleatorii a unor greutăți prea mari sau prea mici, ceea ce face ca antrenarea să fie dificilă. Tehnica de inițializare Xavier modifică scala de inițializare în funcție de cantitatea de neuroni de intrare și de ieșire din fiecare strat. Pentru a se asigura că gradienții nu dispar sau explodează pe măsură ce se deplasează prin rețea în timpul antrenamentului, se încearcă să se mențină variația activărilor destul de constantă pe diferitele niveluri ale rețelei. Eșantionarea ponderilor dintr-o distribuție gaussiană cu o medie 0 și o varianță definită de numărul de neuroni de intrare și de ieșire este procedura standard pentru metoda de inițializare Xavier. Indiferent dacă inițializarea Xavier este aplicată unei funcții de activare tangentă hiperbolică (tanh) sau unei funcții de activare liniară (cum ar fi ReLU), formula se schimbă semnificativ. Cu toate acestea, ideea fundamentală rămâne aceeași: menținerea constantă a gradienților în întreaga rețea [29].

#### Regularizarea L2

Regularizarea L2, denumită uneori „scădere a ponderii”, este o metodă de prevenire a supraajustării în formarea rețelelor neuronale. Se spune că un model este supraadaptat dacă învață să memoreze datele de instruire în loc să facă deducții din acestea, ceea ce duce la performanțe inferioare pe date neinstruite. Pentru a descuraja ponderile mari în rețea în timpul instruirii, regularizarea L2 adaugă un termen de penalizare la funcția de pierdere. Termenul de penalizare se bazează pe pătratul mărimilor ponderilor. Funcția de pierdere inițială (cum ar fi pierderea de entropie încrucișată pentru problemele de clasificare) plus termenul de regularizare se adaugă la pierderea totală în timpul instruirii:

*Pierderea inițială plus termenul de regularizare este egal cu pierderea totală.*

Ponderile mari sunt penalizate în timpul instruirii, pe lângă încercarea de a reduce eroarea pe datele de instruire, datorită adăugării acestui termen de regularizare la funcția de pierdere. Ca urmare, modelul este încurajat să preia tipare mai ușoare care se transpun mai eficient în noile date [28].

#### ReLU

ReLU este acronimul de la Rectified Linear Unit (unitate liniară rectificată) și este una dintre cele mai utilizate funcții de activare în rețelele neuronale, în special în modelele de învățare profundă. Este o funcție matematică simplă definită astfel:

f(x) = max(0,x);

Cu alte cuvinte, ieșirea este zero dacă valoarea de intrare este mai mică decât zero, iar ieșirea este egală cu valoarea de intrare dacă este mai mare sau egală cu zero. Această funcție de activare introduce neliniaritatea în rețea, permițându-i acesteia să învețe modele și relații complexe în cadrul datelor. ReLU oferă o serie de beneficii.

* Eficient din punct de vedere computațional: ReLU necesită doar operația de pragare, care este un calcul simplu. ReLU se calculează mult mai rapid decât alte funcții de activare precum sigmoid sau tanh.
* Activare dispersată: Doar o parte din intrări pot determina neuronii ReLU să devină activi sau să producă valori diferite de zero. În anumite circumstanțe, această dispersie poate fi utilă și poate ajuta la regularizare.
* Reduce problema gradientului de dispariție: Capacitatea ReLU de a evita saturarea în zona pozitivă ajută la reducerea problemei gradienților care dispar în timpul propagării inverse. Acest lucru face posibilă antrenarea cu mai mult succes a unor rețele mai profunde.

ReLU are totuși anumite dezavantaje, mai ales problema „ReLU muribund”, care face ca neuronii să intre permanent într-o stare de inactivitate dacă producția lor este constant zero. Acest lucru se poate întâmpla atunci când un neuron ReLU primește o intrare negativă constantă, ceea ce face ca gradientul care trece prin el să fie zero și împiedică neuronii să își actualizeze ponderile [29].

#### Funcția de pierdere a entropiei încrucișate (XENT)

La antrenarea modelelor de clasificare, funcția de pierdere a entropiei încrucișate - adesea prescurtată XENT - este o opțiune comună, în special atunci când rezultatele pot fi interpretate ca probabilități. Aceasta calculează diferența dintre două distribuții de probabilități: probabilitățile reale, care sunt adesea reprezentate ca vectori codificați pentru sarcinile de clasificare, și probabilitățile prezise produse de model. Modelul este încurajat să producă predicții mai sigure și mai precise prin minimizarea pierderii de entropie încrucișată în timpul instruirii, în special atunci când este asociat cu funcții de activare adecvate, cum ar fi sigmoid (pentru clasificarea binară din contextul aplicației multi-agent din această lucrare) [30].

## 5. Medicină

În acest capitol se explică problema medicinală care a motivat crearea acestei lucrări și a unui sistem multi-agent ce folosește și rețele neuronale pentru diagnosticarea prezenței pietrelor la rinichi. Următoarele subcapitole prezintă problema în amănunt pentru o înțelegere mai bună, alături de metode alternative de diagnosticare.

### 5.1. Pietrele la rinichi (nefrolitiaza)

Formarea calculilor renali în tractul urinar este o afecțiune numită nefrolitiază. Agregatele minerale cunoscute sub numele de calcuri renali se formează la nivelul rinichilor. Este o boală mai frecventă la bărbați. Apariția microcristalelor minerale începe procesul de formare a calculilor renali la nivelul tubilor distali colectori. Suprasaturarea urinei cu saruri de calciu, oxalat și acid uric este următoarea etapă în formarea pietrelor la rinichi. În condiții favorabile, aceste saruri se combină și formează pietre la rinichi.

### 5.2. Ce sunt pietrele la rinichi?

În majoritatea situatiilor, calculii renali sunt constituiti din:

* oxalat de calciu;
* fosfat de calciu;
* acid uric;
* fosfat amoniaco-magnezian.

Aproximativ 75% dintre acestea conțin calciu în componenta lor. Calculii care conțin cistină, care sunt rezultatul unei tulburări ereditare de excreție a cistinei din organism, precum și calculii care conțin agregate medicamentoase (atazanavir, indinavir, sulfamide) sunt mai puțin întâlniți în practica medicală. Infecțiile bacteriene persistente ale tractului urinar cauzate de patogeni precum Proteus, Klebsiella și E. coli sunt asociate cu formarea calculilor renali care conțin fosfat amoniacomagnezian, cunoscut și sub numele de struvit.

În ceea ce privește patologiile renale, nefrolitiaza reprezintă între 10 și 40% din toate afecțiunile renale și se clasează pe locul trei în clasamentul patologiilor urologice, după infectiile urinare și afecțiunile prostatei. Femeile din decada a 4-a pana la a 6-a de varsta sunt cele mai afectate de calculi renali, care sunt rar intalniti la tineri. Pietrele la rinichi pot fi rotunde, ascuțite sau cu aspect ramificat neregulat de tip coraliform, în funcție de locul în care se formează. Agregatele renale ocupă întregul bazinet al pietrei [18].

A diagram of a human body

Description automatically generated

Figura 3. Pietre la rinichi

### 5.3. Cauzele aparițiilor pietrelor la rinichi

Atunci când anumite substanțe chimice devin suficient de concentrate în urină pentru a forma cristale, se formează pietre la rinichi. Creșterea cristalelor duce la formarea de mase mai mari (pietre), care au capacitatea de a trece prin tractul urinar. Pietrele pot provoca durere dacă se blochează într-un loc și împiedică fluxul de urină.

Majoritatea pietrelor sunt create atunci când calciul este amestecat cu una sau două substanțe diferite: fosfor sau oxalat. Acidul uric, care este produs atunci când organismul metabolizează proteina, este o altă sursă de formare a pietrelor [18].

Factorii care pot provoca procese în formarea calculilor renali sunt:

* Aportul insuficient de lichide
* Concentrația crescută de ioni din urina
* pH-ul urinar scăzut
* Reducerea substantelor care inhiba in mod fiziologic agregarea cristalelor - citrat, magneziu si mucoproteinele Tamm-Horsfall.

Următorii factori de risc contribuie la apariția calculilor renali:

* Antecedentele personale sau familiale de litiază renală: persoanele care au o astfel de boală în familie sunt de trei ori mai susceptibile de a dezvolta această afecțiune.
* Deshidratarea: riscul de a dezvolta calculi renali crește dacă consumă o cantitate insuficientă de lichide pe parcursul zilei.
* Obezitatea: studiile de-a lungul timpului au demonstrat că există o relație de cauzalitate între indicele de masa corporală crescut și riscul de a dezvolta calculi renali; în plus, persoanele care locuiesc în zone geografice cu temperaturi ambientale crescute si clima uscata, deoarece pierderea de apa prin transpiratie, care provoacă deshidratare, crește riscul de a dezvolta calculi renali.
* Interventiile chirurgicale gastrice: bypassul gastric, intestinul iritabil si diareea cronica pot determina aparitia modificarilor la nivelul proceselor de reabsorbtie intestinala ale sarurilor de calciu, crescand riscul de aparitie al litiazei renale
* Alte afectiuni medicale asociate in mod frecvent cu nefrolitiaza: acidoza tubulara, cistinuria, hiperparatiroidismul si infectiile urinare recurente, favorizeaza initierea proceselor de formare ale calculilor renali
* Administrarea anumitor tratamente medicamentoase si a suplimentelor vitaminice asa cum sunt laxativele (cand sunt utilizate in exces), antiacide cu continut crescut de calciu, substantele medicamentoase utilizate pentru tratarea depresiei si migrenelor sau suplimentele alimentare cu vitamina C [18].

### 5.4. Simptomele pietrelor la rinichi

Până când calculii încep să migreze de-a lungul tractului urinar (prin uretere, vezică și uretră), nefrolitiaza nu dă simptome. După ce calculii urinari ajung la nivelul ureterului, musculatura netedă ureterală este obstrucționată, dilatată și spastică. Ca urmare, durerea lombară devine mai intensă, iar radiațiile ajung la micul bazin și la organele genitale externe.

Migrarea calculilor renali poate provoca, pe lângă sindromul algic (durere):

* nevoie urgentă de urinare, greață și vărsături;
* febră care poate fi însoțită de frisoane;
* emisii de urini hiperchrome cu miros și aspect tulbure[18].

### 5.5. Metode de diagnosticare al pietrelor la rinichi

Este recomandată adresarea unui medic pentru examinare dacă simptomele indică posibilitatea prezenței pietrelor la rinichi. Următoarele metode de diagnosticare pot da roade:

* analize ale urinei: analiza urinei necesită recoltarea urinei pe o perioadă de 24 de ore. În urma analizei, se determină dacă organismul elimină prea multe minerale, ceea ce va provoca formarea pietrelor. Ocazional, este necesară recoltarea de urină timp de 48 de ore;
* testele de sange pot determina dacă există cantități mari de calciu sau acid uric în sange. În cazul în care acest nivel crește, vor fi necesare teste suplimentare pentru a diagnostica pietrele la rinichi sau alte afecțiuni;
* testele de imagistica: oferă o imagine detaliată a pietrelor și a locului în care se află în tractul urinar. O imagine asupra pietrelor de mici dimensiuni poate fi furnizată prin testarea cu tomografia computerizată (CT);
* radiografia este un test neinvaziv care este folosit frecvent pentru a diagnostica pietrele la rinichi [19].

Lucrarea de față propune un nou mod de diagnosticare ce folosește analize ale urinei care apoi sunt folosite drept set de date pentru antrenarea unei rețele neuronale astfel încât să se găsească modele ce pot ajuta la recunoașterea prezenței pietrelor la rinichi atunci când adăugăm un nou set de analize ale unui pacient ce nu cunoaște dacă are sau nu nefrolitiază.

### 5.6. Tratament pietre la rinichi

Cel mai bun tratament pentru pietrele la rinichi, cunoscute și sub numele de litiază renală, este dificil de determinat. În timp ce pietrele renale mai mari necesită medicamente sau chiar intervenții chirurgicale, calculii renali mai mici pot fi eliminați cu lichide [19].

## 6. Implementarea sistemului multi-agent

Folosind mediul de dezvoltare integrat numit Eclipse, am integrat librăria JADE pentru dezvoltarea agenților și librăria DL4J pentru a avea metode de a crea rețeaua neuronală pentru predicția prezenței pietrelor la rinichi. Pentru interfața grafică am folosit librăria JAVA Swing, din pachetul Window Builder.

### 6.1 Structura proiectului

Proiectul este împărțit în 4 sub-directoare, **AgentsBehaviour** folosit pentru a implementa comportamentele agenților și funcționalitățile acestora, **AgentsGUI** pentru a integra tot ce ține de interfața fiecărui agent în parte, **AgentsStart** pentru a defini configurațiile fiecărui agent și a-l integra în mediul JADE. Directorul **Helpers** este folosit pentru funcții adiacente ce au fost folosite în definirea comportamentelor agenților și pentru gestionarea setului de date aflat într-un fișier denumit *urineAnalyses.csv*.

A screenshot of a computer

Description automatically generated

Figura 4. Structura proiectului

### 6.2. Setul de date

Pe baza analizei urinei, setul de date utilizat poate fi folosit pentru a prezice prezența pietrelor la rinichi.Au fost examinate 79 de probe de urină în încercarea de a stabili dacă anumite caracteristici fizice ale urinei ar putea fi asociate cu formarea de cristale de oxalat de calciu.  
Urina are șase caracteristici fizice folosite pentru a învăța rețeaua neuronală și a găsi forme pentru predicție:

1. densitatea specifică, sau gravitatea urinei în raport cu apa;
2. pH-ul, care este logaritmul negativ al ionului de hidrogen;
3. osmolaritatea (mOsm), o unitate care este folosită în chimia fizică, dar nu în biologie și medicină. Osmolaritatea este proporțională cu concentrația de molecule în soluție;
4. conductivitatea. Un Mho este egal cu un Ohm. Conducția este proporțională cu concentrația de substanțe încărcate cu ioni prezente în soluție;
5. nivelul de uree în milimoli/litru;
6. nivelul de calciu (CALC) în milimoli/litru.

Un capitol din seria Springer in Statistics, „Physical Characteristics of Urines With and Without Crystals”, a furnizat datele [17], analizele pot fi observate în figura 2. Datele au fost obținute de la laboratorul doctorului James S. Elliot M.D. de la secția de urologie, Centrul medical al Administrației Veteranilor, Palo Alto și la Divizia de Urologie, Stanford University School of Medicine, Stanford.

A table of numbers and symbols

Description automatically generated

Figura 5. Analize ale urinii

Din moment ce avem analizele clasificate în baza de date de către un expert și știm exact care analize corespund sau nu cu pietrele la rinichi, putem folosi o metodă de predicție bazată pe inteligență artificială în care modul de învățare să fie mod de învățare supervizat. Această metodologie se pretează perfect problemei de față.

### 6.3. Arhitectura aplicației

Arhitectura aplicației este creată cu ajutorul website-ului Draw.io și include cei patru agenți și funcționalitățile necesare, săgețile reprezentând modul în care comunică agenții.

A diagram of a process

Description automatically generated

Figura 6. Arhitectura aplicației

Aplicația este formată din patru agenți care îndeplinesc anumite funcționalități și au diferite comportamente, comunicând între ei pentru a facilita gestionarea analizelor și a face o predicție asupra existenței sau nu a pietrelor la rinichi. Toți agenții dispun de o interfață grafică pentru interacțiunea cu utilizatorul aplicației. Cei patru agenți sunt descrisi în subcapitolele următoare.

### 6.4. Agentul manager

Acest agent este centrul aplicației, prin el se desfășoară toate comunicațiile dintre agenți, acesta având rolul de a gestiona fișierul cu analize și de a oferi informațiile necesare celorlalți agenți. Agentul are definite trei comportamente:

1. adaugă noile analize în fișier atunci când primește mesaj de la agentul pentru adăugarea unor noi analize cu valorile analizelor;
2. verifică analizele selectate și fișierul în care sunt adăugate atunci când primește mesaj de la agentul pentru selecția analizelor;
3. oferă setul de date necesar pentru a antrena rețeaua neuronală din comportamentul agentului pentru predicția pietrelor la rinichi.

Pentru a putea interacționa cu utilizatorul aplicației, acest agent dispune de o interfață grafică ce afișează tabelul cu setul de date întreg sau setul de date selectat de utilizator prin intermediul agentului pentru selecția anizelor (figura 6).

A screenshot of a computer

Description automatically generated

Figura 7. Interfață grafică agent manager

### 6.5. Agentul pentru adăugarea unor noi analize

Acest agent oferă o interfață grafică pentru interacțiunea cu utilizatorul aplicației prin care expertul uman va putea introduce un nou set de analize. Agentul preia datele din interfață și le comunică agentului manager pentru a adăuga analizele în setul de date. În figurile de mai jos se pot observa interfața și mesajele trimise de agenți în comunicarea lor.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figura 10. Mesaj confirmare adăugare

Figura 9. Mesaj adăugare

Figura 8. Interfață grafică agent manager

### 6.6. Agentul pentru selecția analizelor

Acest agent oferă o interfață grafică pentru interacțiunea cu utilizatorul aplicației prin care expertul uman poate selecta intervale dorite pentru fiecare dintre cei șase parametri ai analizelor și poate genera un nou set de date care conține doar analizele care au intervalele alese de expertul uman. Acest nou set de analize va putea fi setul de date pentru antrenarea rețelei neuronale de către agentul pentru predicția pietrelor la rinichi care este descris în subcapitolul următor. În figura de mai jos se poate observa interfața grafică a agentului selector.

A screenshot of a computer

Description automatically generated

Figura 11. Interfață grafică agent selector

### 6.7. Agentul pentru predicția pietrelor la rinichi cu rețeaua neuronală

Acest agent oferă o interfață grafică pentru interacțiunea cu utilizatorul aplicației prin care expertul uman poate introduce un nou set de analize al unui pacient care nu știe dacă are sau nu pietre la rinichi. După introducerea setului de analize, se selectează setul de date cu care se va realiza antrenarea rețelei neuronale din comportamenul agentului predictor, fie cu setul întreg de date fie cu setul care a fost selectat prin intermediul agentului selector. După ce antrenarea este relizată, predicția existenței sau nu a pietrelor la rinichi poate fi văzută în interfața grafică. Acest agent comunică cu agentul manager pentru a primi setul de date selectat.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figura 12. Interfață grafică agent predictor

#### Fluxul de lucru de învățare supervizată pentru predicție

**Încărcarea și preprocesarea datelor:**

* este utilizat CSVRecordReader pentru a citi fișierul CSV care conține setul de date;
* se împart datele în seturi de instruire (80%) și de testare (20%);
* se normalizează datele folosind NormalizerStandardize.

**Crearea modelului de predicție**:

* se construiește o rețea neuronală cu două straturi dense și un strat de ieșire folosind Deeplearning4j;
* se configurează cu parametri specifici (algoritm de optimizare – stochastic gradient descent, inițializarea greutății - XAVIER, funcții de activare – RELU și SIGMOID și funcție de pierdere - XENT).

**Învățare:**

* se antrenează modelul pe un număr definit de epoci (NUM\_EPOCHS = 10.000).
* se colectează valorile pierderilor de antrenament și se actualizează graficul în timp real.

**Evaluare:**

* se evaluează modelul pe setul de date de testare;
* se afișează acuratețea modelului.

**Predicție**:

* se utilizează modelul instruit pentru a prezice dacă noile date de analiză indică sau nu prezența pietrelor la rinichi.
* afișează rezultatul predicției în interfața grafică a agentului de predicție.

**Comunicarea agentului de predicție**

* agentul comunică cu agenții manager pentru a solicita setul de date și a trimite rezultatul predicției.
* se folosesc mesaje ACL (ACLMessage) pentru comunicare.

**Integrarea GUI**

* interfața AgentPredictGUI permite utilizatorilor să introducă noi date de analiză și declanșează procesul de predicție;

**Exemplu de utilizare**

* agentul este configurat și se înregistrează la DF;
* utilizatorul introduce noi date de analiză prin intermediul interfeței grafice (GUI);
* agentul solicită setul de date de la agenții manager;
* odată ce setul de date este primit, agentul îl încarcă și îl preprocesează;
* agentul creează, antrenează și evaluează un model de rețea neuronală;
* agentul face predicții pe baza noilor date de analiză și trimite rezultatul agenților manageri;

#### Arhitectura rețelei neuronale din comportamentul agentului predictor

Arhitectura definește o rețea neuronală cu cinci straturi, configurată pentru o problemă de clasificare binară.

* *Stratul de intrare* are 6 noduri, corespunzând numărului de caracteristici (features) ale setului de date (densitatea specifică; pH-ul; osmolaritatea (mOsm); conductivitatea; nivelul de uree în milimoli/litru; nivelul de calciu (CALC) în milimoli/litru).

Există trei straturi ascunse, fiecare urmat de un strat de dropout pentru regularizare. Aceste straturi folosesc funcția de activare ReLU:

* primul strat ascuns are 256 de noduri;
* al doilea strat ascuns are 128 de noduri
* al treilea strat ascuns are 64 de noduri;
* stratul de ieșire care produce rezultatele clasificării. Are 2 noduri, corespunzând celor două clase în problema de clasificare binară. Folosește funcția de activare *Sigmoid* și funcția de pierdere *Cross Entropy*.

Alte Configurări:

* optimizatorul folosit este *Adam*, cu o rată de învățare de 0.0001;
* inițializarea greutăților se face folosind metoda Xavier;
* se aplică regularizare L2 cu o putere de 0.001 pentru a preveni supraînvățarea (overfitting-ul);



Figura 13. Arhitectura rețelei neuronale



Figura 14. Cod pentru arhitectura rețelei

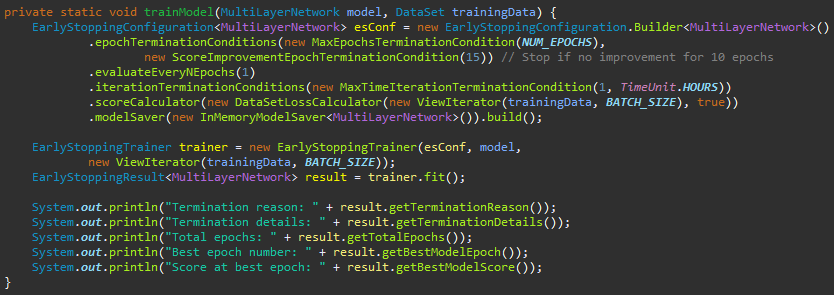


Figura 15. Cod pentru învățarea rețelei

### 6.8. Agentul Sniffer

Un agent special creat pentru a urmări și examina comunicațiile altor agenți într-un sistem multi-agent se numește agent de tip sniffer (adulmecător). Acest agent, cunoscut uneori sub numele de "sniffer" sau "observator al comunicațiilor", este esențial pentru a înțelege și depana comportamentelor și interacțiunilor celorlalți agenți din sistem. În figura de mai jos se poate observa comuncația dintre cei 4 agenți ai acestei aplicații. Putem observa fiecare mesaj trimis și primit de către fiecare agent.



Figura 16. Mesajele dintre agenți

## 7. Rezultate obținute

În figura de mai jos, putem observa relația dintre caracteristicile urinei pentru cazurile când pacientul nu prezenta pietre la rinichi și atunci când pacientul avea pietre la rinichi. Aceste relații sunt bine înțelese de algoritmul de învățare supervizată și predicția este făcută în relație cu aceste clasificări de date. Analizele pacienților ce prezintă pietre la rinichi sunt colorate cu portocaliu, pe când analizele pacienților ce nu prezintă pietre la rinichi sunt colorate cu albastru.

A screenshot of a computer screen

Description automatically generated

Figura 17. Relația între caracteristicile urinei

Harta de căldură a matricei de corelație prezintă corelațiile dintre diferitele caracteristici ale setului de date într-un mod grafic. O matrice de corelație este o tabelă care arată coeficienții de corelație între fiecare pereche de caracteristici din setul de date. O scară de culori prezintă valorile matricei, în care fiecare culoare reprezintă un grad diferit de corelație.  
  
O hartă de căldură oferă o modalitate rapidă și simplă de a vedea aceste legături. De obicei, valorile de corelație mai mari sunt ilustrate printr-o culoare mai închisă (de exemplu, roșu în schemele de culori comune), iar valorile de corelație mai mici sunt ilustrate printr-o culoare mai deschisă (de exemplu, albastru) [29].

A red and blue squares

Description automatically generated

Figura 18. Harta de căldură a matricei de corelație

Două măsurători, acuratețea de antrenare și acuratețea de validare, sunt utilizate pentru a evalua cât de bine funcționează un model de învățare automată în timpul antrenamentului. Acestea sunt esențiale pentru a obține o înțelegere a performanței modelului și pentru a identifica problemele, cum ar fi supraantrenarea.

Acuratețea de antrenare este o măsură a cât de bine funcționează un model pe datele de antrenare. Se calculează ca raportul dintre numărul total de exemple din setul de antrenare și numărul de exemple clasificate corect. Cu alte cuvinte, acuratețea de antrenare este un indicator al capacității modelului de a înțelege datele de antrenare.

Acuratețea de validare este o măsură care măsoară cât de bine funcționează un model pe datele de validare. Folosind aceste date de validare, performanța modelului este evaluată pe date pe care nu le-a văzut în timpul antrenamentului. Acuratețea de validare este crucială pentru a verifica dacă modelul generalizează corect pe date noi și necunoscute.  
  
În timpul antrenamentului, scopul utilizării atât acurateței de antrenare, cât și a acurateței de validare este de a verifica dacă modelul se potrivește bine datelor de antrenare și dacă generalizează corect la datele de validare. O diferență semnificativă între acuratețea de antrenare și validare poate indica că modelul este supraantrenat sau subantrenat.

Supraantrenarea este atunci când modelul se potrivește foarte bine datelor de antrenare, dar nu funcționează bine pe datele de validare. Când modelul nu se potrivește suficient de bine datelor de antrenare, are subantrenare, ceea ce înseamnă că funcționează prost pe datele de validare și de antrenare. Un obiectiv este să oferim un model care se potrivește și generalizează corect pentru a obține o acuratețe ridicată atât pe datele de antrenare, cât și pe cele de validare [30].



Figura 19. Pierdere de antrenare și validare

Acuratețea modelului antrenat este de ~78.57% și rezultatul poate fi observat în Figura 8. Acest lucru înseamnă că dacă oferim modelului un nou set de analize fără a ști dacă pacientul are sau nu pietre la rinichi, probabilitatea ca modelul să prezică răspunsul corect este de ~78.57%



Figura 20. Acuratețe predicție

## 8. Discuții

Este important de menționat că compararea directă a rezultatelor între diferite studii poate fi dificilă din cauza variațiilor în seturile de date utilizate, metodele experimentale și alți factori specifici ale fiecărui studiu. Astfel, pentru a ajunge la concluzii relevante și pentru a orienta cercetarea în acest domeniu, este necesară o evaluare și o interpretare adecvate a rezultatelor.

Pietrele la rinichi sunt o afecțiune complexă, influențată de o varietate de factori, inclusiv istoricul medical al pacientului, stilul de viață și factorii genetici. Un model bazat exclusiv pe datele disponibile poate să nu captureze întreaga complexitate a acestor factori și poate duce la limitări în capacitatea sa de a face predicții precise. În continuare voi enumera câteva din limitările întalnite la scrierea acestei lucrări:

* Setul de date, care constă din doar 79 de înregistrări de pacienți, este o limitare semnificativă a acestei lucrări. Aceasta poate duce la o capacitate limitată de a antrena și evalua în mod corespunzător modelul și poate avea un impact asupra generalizabilității rezultatelor.
* Este posibil ca performanța modelului și interpretarea rezultatelor să fie afectată dacă datele sunt incomplete, inexacte sau conțin erori. Verificarea și curățarea atentă a datelor poate fi o provocare, în special atunci când vine vorba de cantități mici de informații.
* În ciuda eforturilor de a crea un model precis pe baza setului de date disponibil, poate fi dificil să aplicăm rezultatele la alte tipuri de pacienți. Capacitatea modelului de a face predicții precise pe datele noi poate fi afectată de variabilitatea caracteristicilor pacienților și a metodelor de diagnosticare a pietrelor la rinichi.
* Interpretarea rezultatelor modelului poate fi dificilă și necesită o înțelegere profundă a modului în care funcționează rețelele neuronale. Este esențial să se țină cont de limitările modelului și să se interpreteze rezultatele cu atenție, luând în considerare contextul medical și natura problemei studiate.

În ciuda beneficiilor aduse de utilizarea rețelelor neuronale în prezicerea prezenței pietrelor la rinichi, este important să se recunoască și să se abordeze în mod corespunzător limitările menționate mai sus pentru a obține rezultate fiabile și relevante clinic.

## 9. Concluzie

Studiul, care a folosit un sistem multi-agent care utilizează platforma JADE și rețelele neuronale pentru a prezice prezența pietrelor la rinichi în rândul pacienților, pe baza datelor din 79 de analize, a concluzionat că metoda propusă are potențialul de a fi folosită în diagnosticarea și predicția acestei afecțiuni. Rezultatele arată că sistemul funcționează cu o acuratețe de 78,57% în identificarea pacienților cu pietre la rinichi. Integrarea tehnologiilor precum rețelele neuronale și JADE într-un sistem multi-agent oferă o bună perspectivă în diagnosticul medical, care ar putea îmbunătăți eficacitatea și precizia diagnosticului clinic. Cu toate acestea, pentru a confirma fiabilitatea și validitatea rezultatelor pe un eșantion mai mare de pacienți, este necesară continuarea cercetărilor și validarea constatărilor.

## 10. Bibliografie

1. S. Bussmann, N. R. Jennings, and M. Wooldridge, Multiagent Systems for Manufacturing Control, Springer-Verlag, 2004
2. Bellifemine, F., Poggi, A., Rimassa, G. (2001). Developing Multi-agent Systems with JADE. In: Castelfranchi, C., Lespérance, Y. (eds) Intelligent Agents VII Agent Theories Architectures and Languages. ATAL 2000. Lecture Notes in Computer Science(), vol 1986. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/3-540-44631-1_7>
3. 24. J. Odell, „Objects and agents: how do they differ?,” Journal of Object-Oriented Programming, October 2000
4. M. Luck, P. McBurney, and C. Preist. “Agent Technology: Enabling Next Generation Computing,” AgentLink, 2003, see: <http://www.agentlink.org/admin/docs/2003/2003-48.pdf>
5. Magid Nikraz1a, Giovanni Caireb , and Parisa A. Bahria, „A Methodology for the Analysis and Design of Multi-Agent Systems using JADE”, a School of Engineering Science and Parker Center, Murdoch University, Dixon Road, Rockingham, Western Australia, 2006
6. M. Wooldridge, An Introduction to Multiagent Systems, John Wiley and Sons, 2002
7. Foundation for Intelligent Physical Agents (FIPA), see: <http://www.fipa.org/>
8. JADE – Java Agent DEvelopment Framework, see: <http://jade.tilab.com/>
9. Mascardi, V.: Coo-BDI: Extending the BDI Model with Cooperativity, In Declarative Agent Languages and Technologies, Vol. 2990, pp.109-134 (2004)
10. Y.Shoham, "Agent-Oriented Programming"(Technical Report STAN-CS-90-1335). Stanford University: Computer Science Department, UK, 1990
11. N.R.Jennings, "On Agent-Based Software Engineering. Artificial Intelligence", vol. 117, pp.277-296, 2000
12. H.S. Nwana ," Software Agents: An Overview" ,Intelligent Systems Research , Advanced Applications & Technology Dep. , Cambridge university U.K. Knowledge Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996.
13. Al azawi, Rula. (2013). COMPARING AGENT–ORIENTED PROGRAMMING VERSUS OBJECT-ORIENTED PROGRAMMING. 6th International Conference on Information Technology ICIT'13.
14. H.V. Parunak, "’Go to the Ant’: Engineering Principles from Natural Agent Systems," Annals of Operations Research, 75, 1997, pp. 69-101
15. JADE – Java Agent DEvelopment Framework, see: <https://jade.tilab.com/documentation>
16. https://jade.tilab.com/doc/programmersguide.pdf
17. Odell, James. (2000). Objects and Agents: how do they differ?
18. [Pietre la rinichi (nefrolitiaza): simptome, cauze si tratament (medicover.ro)](https://www.medicover.ro/despre-sanatate/pietre-la-rinichi-nefrolitiaza-simptome-cauze-si-tratament,920,n,295)
19. [Pietre la rinichi: cauze, simptome, tratament | Reginamaria.ro | Reginamaria.ro](https://www.reginamaria.ro/utile/dictionar-de-afectiuni/pietre-la-rinichi-calculi-renali)
20. Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) Foundations of Machine Learning, The MIT Press
21. G. James (2003) Variance and Bias for General Loss Functions, Machine Learning 51, 115-135. (<http://www-bcf.usc.edu/~gareth/research/bv.pdf>)
22. Bishop CM (17 August 2006). Pattern Recognition and Machine Learning. New York: Springer.
23. Andrews, D.F., Herzberg, A.M. (1985). Physical Characteristics of Urines With and Without Crystals. In: Data. Springer Series in Statistics. Springer, New York, NY. <https://doi.org/10.1007/978-1-4612-5098-2_45>
24. Russell SJ, Norvig P (2021). Artificial intelligence: a modern approach. Pearson Series in Artificial Intelligence. Ming-wei Chang, Jacob Devlin, Anca Dragan, David Forsyth, Ian Goodfellow, Jitendra Malik, Vikash Mansinghka, Judea Pearl, Michael J. Wooldridge (4th ed.). Hoboken, NJ: Pearson.
25. Dawson CW (1998). "An artificial neural network approach to rainfall-runoff modelling". Hydrological Sciences Journal. 43 (1): 47–66.
26. Lau S (10 July 2017). "A Walkthrough of Convolutional Neural Network – Hyperparameter Tuning"
27. Ramezanpour, A.; Beam, A.L.; Chen, J.H.; Mashaghi, A. Statistical Physics for Medical Diagnostics: Learning, Inference, and Optimization Algorithms. Diagnostics 2020
28. [L1 and L2 Regularization Methods, Explained | Built In](https://builtin.com/data-science/l2-regularization)
29. [What is Supervised Learning?  |  Google Cloud](https://cloud.google.com/discover/what-is-supervised-learning#:~:text=Supervised%20learning%20is%20a%20category,the%20input%20and%20the%20outputs.)
30. [Xavier initialization - GeeksforGeeks](https://www.geeksforgeeks.org/xavier-initialization/)
31. Brownlee, Jason (8 January 2019). "A Gentle Introduction to the Rectified Linear Unit (ReLU)". *Machine Learning Mastery*. Retrieved 8 April 2021.
32. Shoham, Ron; Permuter, Haim (2020). "Amended Cross Entropy Cost: Framework For Explicit Diversity Encouragement"
33. [How to create a seaborn correlation heatmap in Python? - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-create-a-seaborn-correlation-heatmap-in-python/)
34. [Interpreting Training/Validation Accuracy and Loss | by Frederik vom Lehn | Medium](https://medium.com/@frederik.vl/interpreting-training-validation-accuracy-and-loss-cf16f0d5329f)