**fROMÂNIA**

**MINISTERUL APĂRĂRII NAŢIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**FACULTATEA DE SISTEME INFORMATICE ȘI**

**SECURITATE CIBERNETICĂ**

**Specializarea: Inteligență Artificială pentru**

**Apărare și Securitate**



**SISTEM DE ANALIZĂ MALWARE FOLOSIND TEHNICI DE INTELIGENȚĂ ARTIFICIALĂ**

CONDUCĂTOR ŞTIINŢIFIC:

**Col. conf. univ. dr. ing. Ion BĂDOI**

ABSOLVENT:

**Slt. ing. Radu-Ilie HEREȘANU**

Conţine\_\_\_\_\_\_\_\_\_\_\_\_file Inventariat sub nr. \_\_\_\_\_\_

Poziţia din indicator: \_\_\_

Termen de păstrare: \_\_\_\_

**BUCUREŞTI**

**2025**

**ABSTRACT**

The exponential evolution of artificial intelligence, supported by significant advancements in IT hardware, has generated multiple benefits in terms of user experience but has also brought major challenges to the field of cybersecurity. Cyber attackers are exploiting AI to develop autonomous malware capable of adapting in real time to the detection mechanisms implemented by their victims. At the same time, phishing and spoofing campaigns are becoming increasingly sophisticated, using machine learning algorithms to generate highly personalized and convincing messages.

As cyber threats grow more complex and dynamic, security professionals are compelled to adopt AI-based solutions capable of covering a wide spectrum of vulnerabilities. Both initial detection and post-incident analysis are being fundamentally transformed through the integration of modern machine learning techniques.

This paper, motivated by the current context, proposes the development of a system for malware threat detection and preliminary analysis using artificial intelligence techniques. The proposed solution focuses on identifying abnormal network traffic, detecting malicious files, and classifying email messages, thereby offering a comprehensive initial assessment of potential threats.

In the implementation process, machine learning (ML) classification algorithms were used, along with deep learning (DL) neural networks, applied to data derived from executable files, network traffic, and textual content. Well-established datasets in the field, such as NSL-KDD and Malimg, were analyzed to validate the performance of the models in scenarios that closely resemble real-world conditions.

The system’s performance was evaluated using standard metrics such as accuracy, precision, and recall, and the results indicated that deep neural networks deliver the best performance in detecting malicious behavior and traffic anomalies. This approach reinforces the need to integrate AI into modern cybersecurity infrastructures to effectively respond to an increasingly threatening digital environment.

The final results highlighted both the high potential of the proposed solution and the challenges posed by the complexity of implementing a system capable of addressing a wide range of vulnerabilities. Furthermore, the importance of using larger datasets and better-calibrated neural architectures was emphasized to ensure robust and scalable detection in real-world contexts.

**REZUMAT**

Evoluția exponențială a inteligenței artificiale, susținută de progresele semnificative în domeniul hardware-ului IT, a generat multiple beneficii în ceea ce privește experiența utilizatorului, dar a adus totodată și provocări majore în sfera securității cibernetice. Atacatorii cibernetici exploatează inteligența artificială pentru a dezvolta malware autonom, capabil să se adapteze în timp real la mecanismele de detecție implementate de victime. În același timp, campaniile de phishing și spoofing devin tot mai sofisticate, folosind algoritmi de învățare automată pentru a genera mesaje personalizate și extrem de convingătoare.

Pe măsură ce amenințările cibernetice devin tot mai complexe și dinamice, specialiștii în securitate sunt nevoiți să adopte soluții bazate pe inteligență artificială, capabile să acopere un spectru larg de vulnerabilități. Atât detecția inițială, cât și analiza post-incident sunt profund transformate prin integrarea tehnicilor moderne de învățare automată.

Această lucrare, motivată de contextul actual, propune realizarea unui sistem de detecție și analiză sumară a amenințărilor malware, utilizând tehnici de inteligență artificială. Soluția se axează pe identificarea traficului de rețea anormal, detecția fișierelor malițioase și clasificarea mesajelor de e-mail, oferind astfel o evaluare inițială cuprinzătoare a potențialelor amenințări.

În procesul de implementare au fost utilizați algoritmi de învățare automată (ML) pentru clasificare, precum și rețele neuronale specifice învățării profunde (DL), aplicate asupra datelor provenite din fișiere executabile, trafic de rețea și conținut textual. Au fost analizate seturi de date recunoscute în domeniu, precum NSL-KDD și Malimg, pentru a valida performanța modelelor în scenarii cât mai apropiate de realitate.

Performanța sistemului a fost evaluată utilizând metrici standard precum acuratețea, precizia și recall-ul, iar rezultatele au indicat că rețelele neuronale profunde oferă cele mai bune rezultate în identificarea comportamentelor malițioase și a anomaliilor din trafic. Această abordare confirmă necesitatea integrării AI în infrastructurile de apărare cibernetică moderne, pentru a răspunde eficient unui mediu tot mai amenințător.

Rezultatele finale au evidențiat atât potențialul ridicat al soluției propuse, cât și dificultățile legate de complexitatea implementării unui sistem capabil să acopere o gamă largă de vulnerabilități. De asemenea, s-a subliniat importanța utilizării unor seturi de date mai extinse și a unor arhitecturi neuronale mai bine calibrate, pentru a asigura o detecție robustă și scalabilă în contextul real.

**CUPRINS**

[1 Introducere 7](#_Toc197538232)

[1.1 Importanța temei alese 9](#_Toc197538233)

[1.2 Definirea problemei 10](#_Toc197538234)

[1.3 Soluții similare 10](#_Toc197538235)

[2 Concepte de bază 13](#_Toc197538236)

[2.1 Domeniul Inteligenței Artificiale 13](#_Toc197538237)

[2.2 Intrusion Detection System și Intrusion Prevention System 17](#_Toc197538238)

[2.2.1 Knoledge-Based IDS și Behavior-Based IDS 19](#_Toc197538239)

[2.2.2 Host-Based IDS și Network-Based IDS 20](#_Toc197538240)

[2.2.3 Intrusion Prevention Systems 20](#_Toc197538241)

[2.3 Phishing și spoofing 21](#_Toc197538242)

[2.4 Contramăsuri împotriva atacurilor cibernetice 22](#_Toc197538243)

[3 Implementarea practică a sistemului de programe 23](#_Toc197538244)

[3.1 Environment 24](#_Toc197538245)

[3.2 Fluxul aplicației 25](#_Toc197538246)

[3.3 Seturi de date folosite 26](#_Toc197538247)

[3.4 Detecţia e-mailurilor frauduloase 30](#_Toc197538248)

[3.5 Analiza fişierelor online/offline 34](#_Toc197538249)

[3.6 Analiza traficului de date 43](#_Toc197538250)

[3.7 Analiza malware și modelele LLM 47](#_Toc197538251)

[4 Concluzii și direcții viitoare de cercetare 48](#_Toc197538252)

[5 Bibliografie 49](#_Toc197538253)

LISTĂ DE ABREVIERI

|  |  |  |
| --- | --- | --- |
|  | **IDS** | Intrusion Detection System |
|  | **IPS** | Intrusion Prevention System |
|  | **AI** | Artificial Inteligence |
|  | **IoT** | Internet of Things |
|  | **CNN** | Convolutional Neural Network |
|  | **SDLC** | Software Development Life Cycle |
|  | **SIEM** | Security Information and Event Management |
|  | **2FA** | Two-factor authentication |
|  | **DDoS** | Distributed Denial-of-service |
|  | **API** | Application Programming Interface |
|  | **VGG** | Visual Geometry Group |
|  | **LLM** | Large Language Model |
|  | **NTAM** | Network Traffic prediction model based on Attention Mechanism |
|  | **ML** | Machine Learning |
|  | **DL** | Deep Learning |

LISTĂ DE FIGURI

[Figura 1 - Metoda propusă de detectare a malware-ului prin ML 9](#_Toc197538173)

[Figura 2 - Domeniile AI 14](#_Toc197538174)

[Figura 3 - Procesul de clasificare 15](#_Toc197538175)

[Figura 4 - Schema simplificată a unui sistem de detectare/prevenire intruziuni 18](#_Toc197538176)

[Figura 5 - Ciclul de viață al dezvoltării software 24](#_Toc197538177)

[Figura 6 - Meniul principal al aplicației desktop 26](#_Toc197538178)

[Figura 7 - Word Cloud pentru a vizualiza cele mai frecvente cuvinte 27](#_Toc197538179)

[Figura 8 - Distribuția lungimii mesajelor HAM vs Spam 27](#_Toc197538180)

[Figura 9 - Phishing Email Dataset după sanitizare 28](#_Toc197538181)

[Figura 10 - Repartiția claselor în setul de date Malimg 29](#_Toc197538182)

[Figura 11 - Distribuția tipurilor de atacuri în setul de antrenare 30](#_Toc197538183)

[Figura 12 - Clasificatori folosiți în clasificarea binară a e-mailurilor (spam/ham) 31](#_Toc197538184)

[Figura 13 - Arhitectura modelului LSTM folosită în clasificarea e-mailurilor 33](#_Toc197538185)

[Figura 14 - Acuratețea și funcția de cost pentru arhitectura LSTM 33](#_Toc197538186)

[Figura 15 - Metadatele afișate în interfața meniului File Info 35](#_Toc197538187)

[Figura 16 - Funcționalitatea Online File Analyser 36](#_Toc197538188)

[Figura 17 - Etapele de procesare ale fișierului (funcția process\_file) 37](#_Toc197538189)

[Figura 18 - Funcţionalitatea Offline File Analyser 37](#_Toc197538190)

[Figura 19 - Arhitectura convoluțională folosită inițial 38](#_Toc197538191)

[Figura 20 - Proiecție a celor 13 straturi convoluționale din arhitectură 39](#_Toc197538192)

[Figura 21 - Graficul funcției de cost și acurateții - 14 epoci - arhitectură VGG 40](#_Toc197538193)

[Figură 22 - Grafic acuratețe și cost - arhitectură simplă 41](#_Toc197538194)

[Figura 23 - Matricea de confuzie calculată 42](#_Toc197538195)

[Figura 24 - Încărcarea și preprocesarea datasetului Malimg 42](#_Toc197538196)

[Figura 25 - Curba acurateții și curba de cost 46](#_Toc197538197)

[Figura 26 - Matricea de confuzie (dreapta), distribuția claselor (stânga) 46](#_Toc197538198)

[Figura 27 - Interfața chatbotul-ui bazat pe modelul Mistral 47](#_Toc197538199)

LISTĂ DE TABELE

[Tabelul 1 - Exemple de soluții comerciale antivirus și antimalware 12](#_Toc197538200)

[Tabelul 2 - Exemple de soluții pentru analiză malware 13](#_Toc197538201)

[Tabelul 3 - Formula Acurateții 16](#_Toc197538202)

[Tabelul 4 - Formula Preciziei 16](#_Toc197538203)

[Tabelul 5 - Formula Senzitivității 16](#_Toc197538204)

[Tabelul 6 - Configurația mediului experimental 25](#_Toc197538205)

[Tabelul 7 - Pregătirea mediului de lucru și a pachetelor necesare 25](#_Toc197538206)

[Tabelul 8 - Distribuția claselor reprezentând atacuri în setul KDDtrain+.txt 30](#_Toc197538207)

[Tabelul 9 - Performanța clasificatorilor obținută pe setul de testare 32](#_Toc197538208)

[Tabelul 10 - Performanța clasificatorilor și modelului LSTM pe setul de testare 34](#_Toc197538209)

# Introducere

Într-un mediu digital în permanentă transformare, rețelele de calculatoare devin constant ținta unor atacuri tot mai sofisticate. Fiecare port deschis în rețea reprezintă o posibilă poartă de intrare pentru atacatorii cibernetici, care utilizează tehnici malițioase din ce în ce mai avansate pentru a compromite sistemele. În contextul acestor amenințări, învățarea automată a devenit o tehnică răspândită și eficientă, contribuind semnificativ la îmbunătățirea sistemelor de detectare a intruziunilor și identificarea traficului de rețea rău intenționat.

Eficiența modelelor machine learning în detecția amenințărilor depinde în mod esențial de calitatea seturilor de date folosite pentru antrenarea acestora, precum și de algoritmul implementat. Sistemele de detectare a intruziunilor (IDS) sunt critice în identificarea potențialelor atacuri asupra rețelei și în informarea administratorului cu privire la natura și tipul acestora. Aceste IDS-uri se împart în două categorii principale: IDS-uri bazate pe gazdă (HIDS), focalizate pe atacuri din interiorul unui computer, și IDS-uri bazate pe rețea (NIDS), specializate în monitorizarea atacurilor din cadrul unei rețele.

Mai mult, IDS-urile pot utiliza abordări diferite, cum ar fi IDS bazate pe anomalii, care se sprijină pe tehnici de învățare automată și învățare profundă pentru a identifica modele de atac, și IDS bazate pe reguli, care utilizează reguli derivate din datele istorice pentru detectarea atacurilor. IDS-urile bazate pe anomalii sunt capabile să identifice atacurile de tip zero-day, oferind astfel o protecție avansată.

Ca atare, un sistem IDS ideal ar trebui să fie capabil să recunoască o varietate extinsă de vulnerabilități provenite din surse diverse. Acesta ar trebui să poată identifica amenințări precum spam-ul/phishing-ul malițios prin e-mail, fișiere conținând payload-uri malițioase și să detecteze traficul malițios, evoluând și auto-perfecționându-se în mod constant și automat.

La această lucrare de disertație, intenționez să dezvolt în limbajul de programare Python un sistem de detecție a intruziunilor capabil să îndeplinească diverse sarcini. Acestea includ clasificarea email-urilor frauduloase, analiza și detecția traficului de date rău intenționat, precum și analiza fișierelor malițioase. Pentru a atinge aceste obiective, voi parcurge mai multe etape:

1. Culegere de informații de specialitate: Voi investiga posibilii vectori de amenințare, seturile de date disponibile, soluțiile existente de detecție/analiză malware de pe piață și algoritmii de învățare automată utilizați în implementarea diferitelor tipuri de sisteme de detecție a intruziunilor (IDS).

2. Analiză și prelucrare a datelor de interes: Voi analiza și procesa datele relevante pentru a obține informații esențiale în implementarea sistemului.

3. Implementarea efectivă presupune concretizarea fiecărei sarcini componente din cadrul IDS-ului propus. Această etapă poate implica, la rândul său, mai multe faze, precum curățarea datelor, procesarea acestora, clusterizarea și analiza datelor, pregătirea modelului și ajustarea acestuia.

4. Realizarea unei interfețe grafice (opțional): În funcție de necesități, voi explora posibilitatea dezvoltării unei interfețe grafice care să integreze modulele și să faciliteze interacțiunea cu sistemul.

Mai mult, intenționez să dezvolt o modalitate de salvare a log-urilor și de avertizare a utilizatorului atunci când traficul este unul malițios. Pe baza traficului captat, am în plan să îmbunătățesc modelele și algoritmii utilizați. Acest lucru va contribui la o detecție mai precisă a amenințărilor și la oferirea unei notificări rapide în cazul unor activități suspecte.

De asemenea, voi optimiza rețeaua folosind diferite tehnici, precum calcul evolutiv - Genetical Artificial Neural Network (GANN). Această abordare va contribui la optimizarea performanței rețelei și la adaptarea acesteia la schimbările din mediul digital în continuă evoluție.

Prin urmare, aceste etape vor contribui la conceperea și implementarea unui sistem eficient de detecție a intruziunilor, cu scopul de a asigura securitatea și integritatea sistemelor informatice. Un posibil workflow al aplicației ar putea fi:

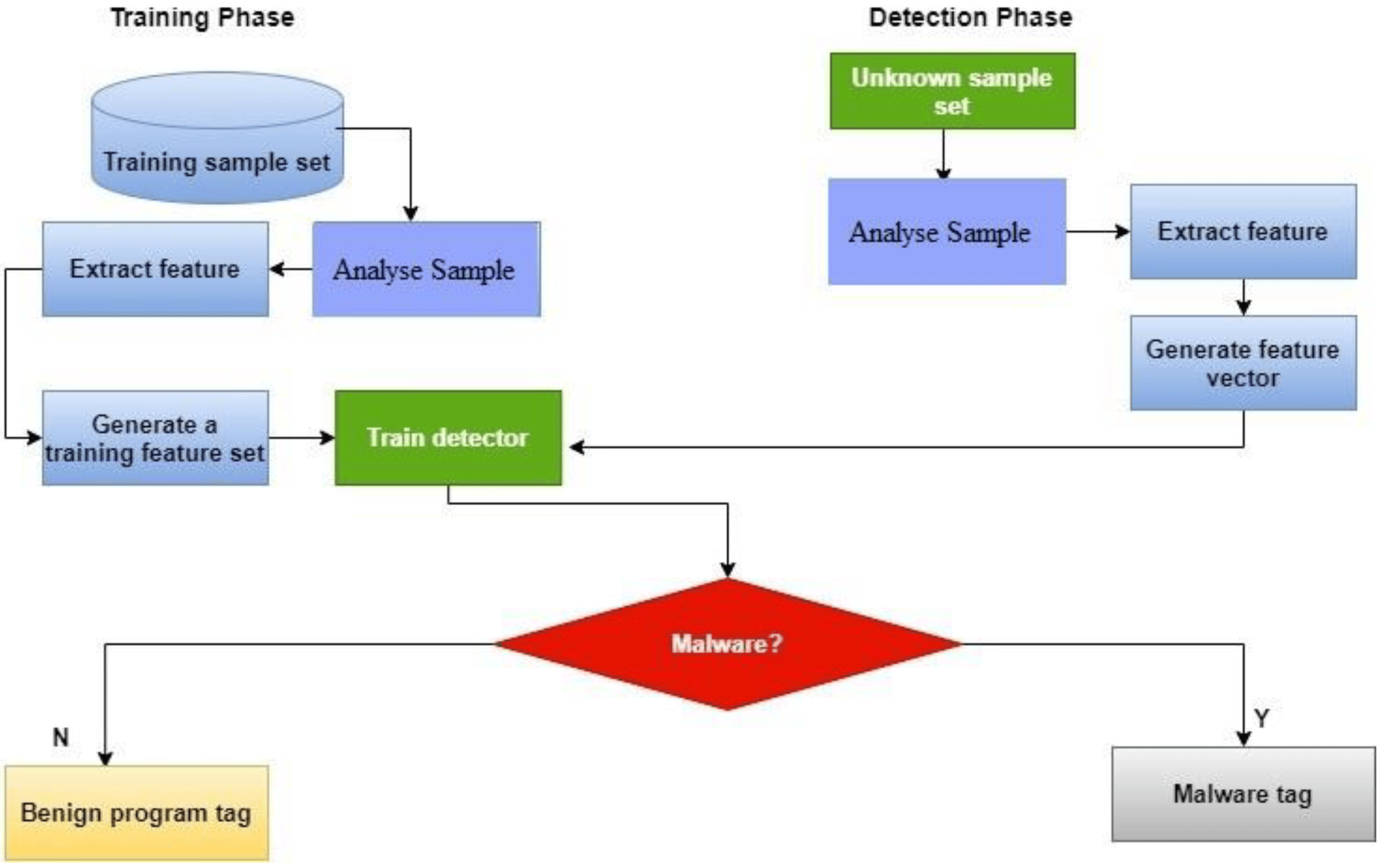


Figura 1 - Metoda propusă de detectare a malware-ului prin ML

<Sursa: [**https://www.mdpi.com/2073-8994/14/11/2304**](https://www.mdpi.com/2073-8994/14/11/2304) >

## Importanța temei alese

Sistemele de detecție și analiză malware sunt utilizate tot mai frecvent în viața de zi cu zi, nu doar de utilizatorii de calculatoare, tablete și smartphone-uri, ci și în cadrul sistemelor de comandă pentru instalații fotovoltaice, televizoare inteligente și alte dispozitive conectate. Majoritatea dispozitivelor moderne, sunt echipate din fabrică cu soluții de protecție împotriva malware-ului. Aceste soluții sunt integrate fie la nivelul sistemului de operare, fie prin aplicații preinstalate, și oferă funcții avansate precum scanarea automată a aplicațiilor sau fișierelor descărcate, detectarea comportamentelor suspecte și protecția în timp real împotriva amenințărilor cibernetice. Astfel de măsuri sunt fundamentale pentru a proteja securitatea datelor utilizatorilor și pentru a preveni atacurile care pot compromite aceste dispozitive, indiferent de categoria din care fac parte.

Soluția propusă este esențială în contexte în care se urmărește implementarea unui sistem multifuncțional, capabil să detecteze malware-ul atât în timp real, cât și în mod local, oferind în același timp utilizatorului o experiență educativă și interactivă. Aceasta contribuie la îmbunătățirea securității cibernetice și la creșterea gradului de conștientizare și educare a utilizatorilor prin intermediul unor funcționalități avansate și intuitive.

În acest moment se găsesc diferite soluții software ce îndeplinesc funcția de detecție/prevenire a malware-ului. Ceea ce le diferențiază este scopul de utilizare (sisteme industriale, casnice), aria de folosire (publicul larg, sistemul militar), tipul de dispozitiv informatic pe care trebuie să îl protejeze de atacurile cibernetice (computere personale, servere de date, dispozitive mobile, echipamente IoT etc.).

## Definirea problemei

Problema detectării malware-ului constă în identificarea și clasificarea activităților malițioase dintr-o mare cantitate de date. Atacatorii folosesc tehnici tot mai sofisticate pentru a ascunde activitățile lor, cum ar fi:

* Malware polimorfic și metamorf - codul se modifică pentru a evita detecția bazată pe semnături.
* Zero-day attacks - exploatarea vulnerabilităților necunoscute anterior.
* Atacuri bazate pe machine learning invers - tehnici prin care atacatorii încearcă să păcălească modelele de învățare automată.

Detectarea tradițională se bazează pe reguli statice sau baze de semnături, ceea ce o face ineficientă împotriva amenințărilor necunoscute sau adaptabile [1].

Totodată, atacatorii caută să compromită un sistem informatic sau să obțină beneficii materiale necuvenite prin campanii bine orchestrate de phishing, în care colectează informații despre victime pentru a exploata vulnerabilitățile identificate în etapele ulterioare ale atacului. Aceste metode pot include ingineria socială, spoofing-ul identității sau livrarea de fișiere malițioase ascunse în documente aparent legitime.

În acest context, problema detectării malware-ului devine una multifațetată, necesitând o abordare complexă, care să combine analiza comportamentală, detecția anomaliilor și clasificarea precisă a amenințărilor cibernetice. Provocarea majoră constă în identificarea unui atac înainte ca acesta să compromită sistemul, în condițiile unui volum de date tot mai mare și a unui peisaj de amenințări în continuă schimbare.

## Soluții similare

În domeniul analizei și detectării malware-ului, soluțiile existente pe piață sunt deja foarte sofisticate și oferă un grad ridicat de acuratețe în identificarea amenințărilor. Cu toate acestea, soluția propusă în această lucrare nu are scopul de a depăși performanțele acestor soluții comerciale, ci de a aborda o gamă largă de probleme existente în utilizarea și gestionarea securității pe un calculator. Prin caracteristicile sale specifice, aceasta își propune să ofere o alternativă mai accesibilă și adaptabilă pentru utilizatorii care necesită o protecție robustă în condiții de resurse limitate și costuri reduse.

Exemple de soluții comerciale recunoscute pe piață, în ceea ce privește detecția și stoparea malware (*Tabel 1*), includ: Bitdefender, Avast, Kaspersky, Norton, McAfee, Trend Micro etc. Aceste soluții oferă protecție antivirus**[[1]](#footnote-1)** și antimalware**[[2]](#footnote-2)** de înaltă calitate și includ, pe lângă aceasta, funcționalități suplimentare precum detectarea emailurilor suspecte, protecția împotriva phishing-ului, servicii VPN, protecție pentru browser, blocarea trackerelor și opțiuni de cleanup contra cost. În general, aceste soluții prezintă funcționalități similare, diferența majoră constând în costul abonamentelor și în performanțele oferite.

|  |  |
| --- | --- |
| **Soluție comercială** | **Caracteristici particulare** |
| Avast | Scanare în timp real, protecție împotriva phishing-ului și instrumente de analiză comportamentală. Oferă un pachet complet pentru detectarea și eliminarea amenințărilor cibernetice. |
| Bitdefender | Utilizează tehnologii avansate de inteligență artificială și analiză comportamentală pentru detectarea amenințărilor emergente. Pune accent pe reducerea alarmele false și optimizarea performanței sistemului. |
| Kaspersky | Renumit pentru rata ridicată de detecție a malware-ului. Combină analiza statică și dinamică a fișierelor, oferind protecție extinsă împotriva virusurilor, troienilor și altor programe malițioase. |
| Norton Antivirus | Oferă o suită integrată de securitate cibernetică care include protecție antivirus, managementul parolelor, backup în cloud și protecție a identității. |
| McAfee | Se remarcă prin soluțiile de securitate complete: protecție antivirus, firewall-uri și instrumente de prevenire a atacurilor cibernetice, potrivite atât pentru uz casnic cât și pentru medii corporative. |
| Trend Micro | Pune accent pe tehnologiile de protecție bazate pe cloud și pe analiza proactivă a amenințărilor, fiind o alegere populară pentru companiile ce doresc o securitate integrată și ușor de gestionat. |

Tabelul 1 - Exemple de soluții comerciale antivirus și antimalware

În ceea ce privește analiza malware și securitatea cibernetică, există o gamă variată de soluții avansate care necesită expertiză specializată pentru a fi utilizate eficient. Aceste instrumente oferă funcționalități complexe, de la analiza statică și dinamică a codului la monitorizarea traficului de rețea și evaluarea securității aplicațiilor web. Tabelul 2 prezintă o selecție de astfel de soluții utilizate frecvent de analiștii de securitate pentru identificarea și înțelegerea amenințărilor cibernetice:

|  |  |  |  |
| --- | --- | --- | --- |
| **Soluție** | Tip de analiză | Platformă | Descriere scurtă |
| IDA Pro | Analiză statică (disassembler), Dinamică (debugger) | Windows, Linux, macOS | Disassembler comercial avansat, oferă reverse engineering la nivel de assembly și un sistem de scripting extins; include și componentă de debugging. |
| Ghidra | Statică (disassembler) | Cross-platform (Java-based) | Dezvoltată de NSA, oferă un disassembler și decompiler puternice, este open-source și include instrumente colaborative pentru echipe de analiști. |
| Wireshark | Analiză trafic rețea, Captură pachete | Windows, Linux, macOS | Wireshark este unul dintre cele mai populare instrumente de analiză a protocolului de rețea, permițând capturarea și inspectarea traficului de rețea în timp real. Suportă o gamă largă de protocoale și oferă funcționalități avansate de filtrare și decriptare. Este recomandat pentru utilizatorii avansați datorită complexității sale și a curbei de învățare asociate. |
| Burp Suite | Analiză securitate web, Proxy HTTP/HTTPS | Windows, Linux, macOS | Burp Suite este un set de instrumente pentru testarea securității aplicațiilor web, permițând interceptarea, analizarea și modificarea traficului HTTP/HTTPS, fiind utilizat pentru identificarea vulnerabilităților în aplicațiile web. |
| Suricata | Detectare/prevenire intruziuni, Monitorizare rețea | Windows, Linux, macOS | Suricata este un motor open-source pentru detectarea și prevenirea intruziunilor, monitorizarea securității rețelei și procesarea PCAP, capabil să identifice și să blocheze rapid atacuri sofisticate, oferind suport pentru captura completă de pachete și detectarea anomaliilor în trafic. |

Tabelul 2 - Exemple de soluții pentru analiză malware

# Concepte de bază

Înainte de a implementa soluția propriu-zisă, este esențială o bună înțelegere a conceptelor teoretice fundamentale din domeniile inteligenței artificiale și securității cibernetice. În acest capitol, voi prezenta principalele concepte utilizate în abordarea problemei de față, precum și noțiunile relevante dobândite pe parcursul orelor din cadrul programului de master.

## Domeniul Inteligenței Artificiale

Domeniul Inteligenței Artificiale (AI) este unul în continuă expansiune, vizând crearea de sisteme capabile să îndeplinească sarcini ce necesită, în mod obișnuit, inteligență umană, precum recunoașterea vorbirii, învățarea, traducerea și luarea deciziilor. Spre deosebire de afirmația potrivit căreia actualele modele AI „întrec” deja inteligența umană în toate aspectele, este mai corect să spunem că multe sisteme AI depășesc performanțele umane în sarcini specifice (de exemplu, recunoaștere de imagini, jocuri strategice, traduceri automate), însă obținerea unei inteligențe generale comparabile cu a omului rămâne un obiectiv de cercetare pe termen lung.

Acest progres tehnologic este considerat un pas esențial pentru dezvoltarea unor aplicații inovatoare, de la asistență medicală până la conducere autonomă și analiză complexă de date. În același timp, utilizarea AI implică și responsabilități etice și de securitate, fiind necesare eforturi continue pentru a asigura transparența, corectitudinea și protecția datelor în cadrul sistemelor inteligente.

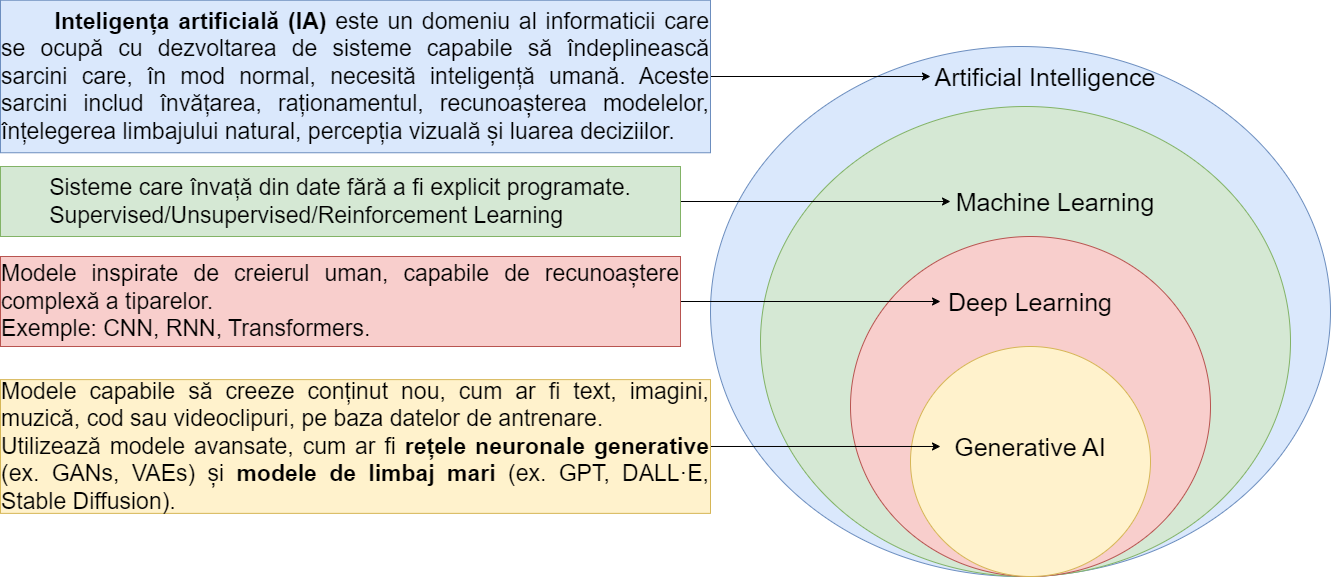


Figura 2 - Domeniile AI

Așa cum se observă și în figura de mai sus, AI poate fi împărțită în mai multe subdomenii:

* Machine Learning (ML): Sisteme care învață din date fără a fi explicit programate, folosind metode precum învățarea supervizată, nesupervizată sau prin consolidare (reinforcement learning).
* Deep Learning (DL): Un subset al ML care utilizează rețele neurale artificiale complexe, inspirate de structura creierului uman, capabile să recunoască tipare foarte complexe. Exemple de arhitecturi: CNN (Convolutional Neural Networks), RNN (Recurrent Neural Networks), Transformers etc.
* Generative AI: O ramură a Deep Learning-ului care se concentrează pe crearea de conținut nou (text, imagini, muzică, cod). Se bazează pe rețele neurale generative (GAN, VAE) și pe modele de limbaj de mari dimensiuni (precum GPT, DALL-E, Stable Diffusion).

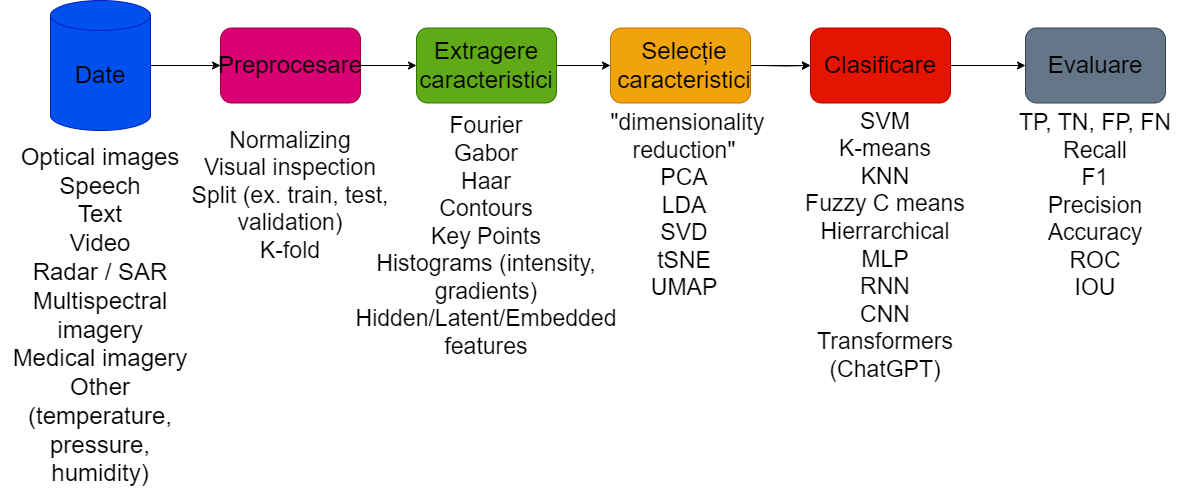


Figura 3 - Procesul de clasificare

Clasificarea reprezintă o problemă de învățare supervizată, în care setul de antrenament conține exemple etichetate, fiecare instanță având asociată clasa corectă. Printre cei mai utilizați algoritmi de clasificare se numără: regresia logistică (sau liniară și polinomială, în funcție de complexitatea relațiilor), arborii decizionali, Random Forest, Support Vector Classifier (SVC), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), algoritmii bayesieni (Naive Bayes), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), AdaBoost, Gradient Boosting, Bagging și metode moderne ca XGBoost sau CatBoost.

Înainte de aplicarea efectivă a algoritmului, datele sunt preprocesate prin curățare, normalizare și selecția trăsăturilor relevante (feature selection), pentru a reduce dimensiunea și a crește performanța clasificatorului. Această etapă este esențială deoarece elimină zgomotul și asigură un model mai precis și mai eficient (vezi Figura 3).

Clasificarea este o problemă de învăţare supervizată, unde datele de antrenare includ exemple etichetate cu clasa corespunzătoare. Metode și algoritmi de clasificare: regresie liniară/logistică/polinomială, random forest, arbori decizionali (tree-based methods), boosting (AdaBoost), bagging trees (Boostrap aggreagation), SVC (Support Vector Classifier), SVM, LDA, QDA, Nayve Bayes, KNN (K-Nearest Neighbors). Premergător etapei de clasificare propriu-zisă, datele sunt preprocesate (), iar din caracteristicile extrase sunt păstrate doar cele relevante pentru ca rezultatul să aibe performanțe maxime.

În schimb, clusterizarea este o problemă de învățare nesupervizată, unde datele nu sunt etichetate, iar scopul este descoperirea automată a structurilor sau a pattern-urilor ascunse. Algoritmi populari includ K-Means, Hierarchical Clustering, DBSCAN (Density-Based Spatial Clustering), Gaussian Mixture Models (GMM), Spectral Clustering, Mean-Shift și OPTICS. Acești algoritmi sunt adesea folosiți pentru detecția de anomalii sau gruparea tiparelor necunoscute anterior.

Evaluarea performanței algoritmilor de învățare automată este o etapă crucială în procesul de dezvoltare. Pentru algoritmii de clasificare, se folosesc metrici standard precum:

* Acuratețea (Accuracy): proporția predicțiilor corecte din totalul datelor.

|  |  |
| --- | --- |
|  | (2.1) |

Tabelul 3 - Formula Acurateții

* Precizia (Precision): proporția predicțiilor pozitive care sunt corecte.

|  |  |
| --- | --- |
|  | (2.2) |

Tabelul 4 - Formula Preciziei

* Senzitivitatea (Recall sau True Positive Rate): capacitatea modelului de a detecta corect cazurile pozitive.

|  |  |
| --- | --- |
|  | (2.2) |

Tabelul 5 - Formula Senzitivității

În acest proiect, voi folosi, în principal, cele trei metrici prezentate (Tabelul 3, *Tabelul 4* și *Tabelul 5*) pentru a evalua clasificatorii, dar mai există și altele precum: F1-Score (media armonică între precizie și recall), curba ROC (Receiver-Operating Characteristic) și AUC (Area Under the Curve), coeficientul Kappa (esențial în contextul dezechilibrului de clase), MCC (Matthews Correlation Coefficient) etc.

Reducerea dimensională este procesul de reducere a numărului de variabile ale unui set de date, păstrând totuși cât mai multă informație semnificativă posibil. Scopul principal al reducerii dimensionale este de a simplifica datele fără a pierde prea mult din caracteristicile lor importante, permițând astfel analiza și vizualizarea mai eficientă a acestora. Motive pentru reducerea dimensională: eliminarea redundanței și a zgomotului (variabile care nu aduc informație utilă), îmbunătăţirea performanţei algoritmilor (reduc timp de execuţie), vizualizare mai eficientă (înţeleg mai bine structurile). Tehnici şi algoritmi pentru reducerea dimensională: PCA (Principal Component Analysis), MDS (Multidimensional Scaling), LDA (Linear Discriminant Analysis), t-SNE (t-Distributed Stochastic Neighbor Embedding), iSOMAP (Isometric Mapping), UMAP(Uniform Manifold Approximation and Projection), QDA (Quadratic Discriminant Analysis) etc.

Tipuri de rețele neuronale: Feedforward Neural Network (FNN), Convolutional Neural Network (CNN), Time Delay Neural Network (TDNN) Recurrent Neural Network (RNN), Long Short-Term Memory Network (LSTM), Radial Basis Function Network (RBFN), Autoencoder Neural Network, Generative Adversarial Network (GAN), Transformers Network, Modular Neural Network (MNN) etc.

Aplicațiile IA sunt extrem de variate și se regăsesc în aproape toate domeniile. Câteva exemple reprezentative sunt:

* + - 1. Recunoașterea și procesarea imaginilor (Computer Vision): detecția și clasificarea obiectelor, diagnostic medical, recunoaștere facială, segmentarea zonelor urbane, detecția de scurgeri de petrol, clasificarea culturilor agricole, reconstrucție sau înlăturare ceață/nori, eliminarea zgomotului din imagini, detecția hazardelor naturale și a zonelor de risc, sisteme radar cu apertură sintetică;
      2. Procesarea limbajului natural (NLP): traduceri automate, analiză de sentiment, recunoștere vorbire/vorbitor, chatbots și asistenți virtuali;
      3. Sisteme de recomandare de produse/reclame/postări folosind algoritmi de profilare a utilizatorului și prezente în e-commerce, rețele sociale și streaming;
      4. Conducere autonomă și transport inteligent prin optimizarea rutelor;
      5. Educație și e-learning: platforme care adaptează conținutul și dificultatea lecțiilor în funcție de progresul și stilul de învățare al fiecărui elev.

## Intrusion Detection System și Intrusion Prevention System

Sistemele de detectare a intruziunilor (IDS) nu fac altceva decât să detecteze și să avertizeze administratorii cu privire la orice activitate anormală în rețea, în timp ce sistemele de prevenire a intruziunilor (IPS) funcționează în timp real și opresc automat traficul rău intenționat. În timp ce IDS avertizează, dar nu rezolvă problema, IPS adoptă o poziție proactivă pentru a atenua încălcarea securității. Alegerea unui IDS, a unui IPS sau a ambelor depinde de toleranța la risc, de buget și de necesitatea unui răspuns imediat la amenințări. Aceste sisteme au roluri complementare în cadrul unui plan cuprinzător de securitate cibernetică [2]. În Figura 4 este reprezentată schema simplificată a unui ansamblu format din IDS și IPS ce asigură monitorizarea constantă a sistemelului informatic, implicit a traficului rețelei, și intervenția automată în momentul detectării activităților suspecte.

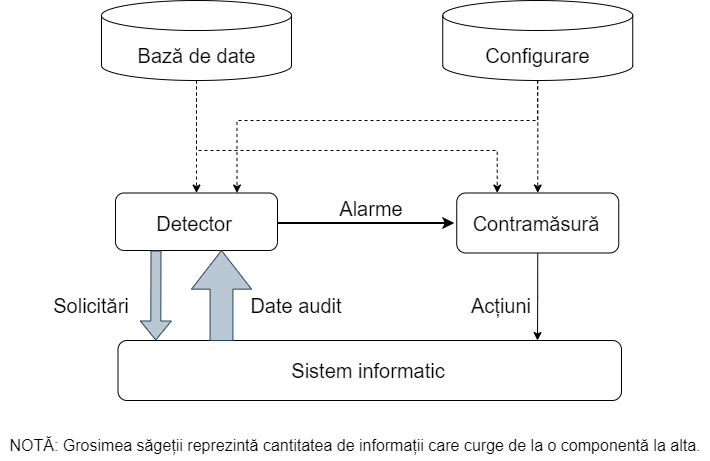


Figura 4 - Schema simplificată a unui sistem de detectare/prevenire intruziuni

Un sistem de detectare a intruziunilor (IDS) colectează informații despre un sistem informațional pentru a realiza o diagnoză asupra stării de securitate a acestuia din urmă. Scopul este de a descoperi breșe de securitate, tentative de încălcare sau vulnerabilități existente care ar putea duce la posibile încălcări de securitate [3].

Un IDS poate fi descris, la un nivel foarte macroscopic, ca un detector care procesează informațiile provenite din sistemul ce trebuie protejat. Acest detector poate, de asemenea, să lanseze solicitări pentru a declanșa procesul de audit, de exemplu, prin solicitarea numerelor de versiune ale aplicațiilor. El utlizează trei tipuri de informații: informații pe termen lung legate de tehnica folosită pentru detectarea intruziunilor (o bază de cunoștințe a atacurilor, de exemplu), informații de configurare despre starea actuală a sistemului și informații de audit care descriu evenimentele ce se petrec în sistem.

Rolul detectorului este de a elimina informațiile inutile din jurnalul de audit. Ulterior, acesta prezintă fie o vedere sintetică a acțiunilor legate de securitate întreprinse în timpul utilizării normale a sistemului, fie o vedere sintetică a stării curente de securitate a sistemului. Se ia apoi o decizie pentru a evalua probabilitatea ca aceste acțiuni sau această stare să fie considerate simptome ale unei intruziuni sau ale unor vulnerabilități. O componentă de contramăsură poate apoi să intervină, fie prevenind executarea acestor acțiuni, fie schimbând starea sistemului pentru a reveni la o stare sigură. În acest context, pentru a evalua eficienţa unui sistem de detectare a intruziunilor, Porras şi Valdes [4] au propus următorii trei parametri: acurateţea (detectarea corectă a atacurilor comparativ cu numărul alarmelor false), performanţa (rata la care evenimentele de audit sunt procesate) și completitudinea (capacitatea unui IDS de a detectecta toate atacurile).

### Knoledge-Based IDS și Behavior-Based IDS

Există două tendințe complementare în detectarea intruziunilor: (1) utilizarea cunoștințelor acumulate despre atacuri și căutarea de dovezi ale exploatării acestor atacuri (knowledge-based IDS) și (2) construirea unui model de referință al comportamentului obișnuit al sistemului informațional monitorizat și căutarea abaterilor de la modul de utilizare observat (behavior-based IDS).

Tehnicile de detectare a intruziunilor bazate pe cunoaștere aplică informațiile acumulate despre atacuri specifice și vulnerabilitățile sistemului. Sistemul de detectare a intruziunilor conține informații despre aceste vulnerabilități și caută încercări de exploatare a acestora. Când o astfel de încercare este detectată, se declanșează o alarmă. Avantajele abordărilor bazate pe cunoaștere sunt că, teoretic, au rate foarte scăzute de alarme false și că analiza contextuală propusă de sistemul de detectare a intruziunilor este detaliată, facilitând astfel înțelegerea problemei de către ofițerul de securitate care utilizează acest sistem și adoptarea de măsuri preventive sau corective. Dezavantajele includ dificultatea de a colecta informațiile necesare despre atacurile cunoscute și de a le menține actualizate cu noile vulnerabilități și medii. Menținerea bazei de cunoștințe a sistemului de detectare a intruziunilor necesită o analiză atentă a fiecărei vulnerabilități și este, prin urmare, o sarcină consumatoare de timp.

Tehnicile de detectare a intruziunilor bazate pe comportament presupun că o intruziune poate fi identificată prin observarea unei abateri de la comportamentul normal sau așteptat al sistemului sau al utilizatorilor. Modelul comportamental normal sau valid este extras din informații de referință colectate prin diverse mijloace. Ulterior, sistemul de detectare a intruziunilor compară acest model cu activitatea curentă. Când se observă o abatere, se generează o alarmă. Cu alte cuvinte, orice acțiune care nu corespunde unui comportament învățat anterior este considerată intruzivă. Prin urmare, sistemul de detectare a intruziunilor poate fi complet, însă acuratețea acestuia reprezintă o problemă dificilă. Avantajele abordărilor bazate pe comportament sunt că pot detecta încercările de exploatare a vulnerabilităților noi și neașteptate. Ele pot chiar contribui la descoperirea (parțial) automată a acestor noi atacuri. Sunt mai puțin dependente de mecanismele specifice ale sistemului de operare și ajută la detectarea atacurilor de tip abuz de privilegii, care de fapt nu implică exploatarea vreunei vulnerabilități de securitate. Rata ridicată de alarme false este, în general, citată ca principalul dezavantaj al tehnicilor bazate pe comportament, deoarece nu întreaga gamă a comportamentului unui sistem informațional poate fi acoperită în timpul fazei de învățare.

### Host-Based IDS și Network-Based IDS

Detectarea intruziunilor bazată pe gazdă (HIDS) a fost primul domeniu explorat în detectarea intruziunilor. Când au fost proiectate primele sisteme de detectare a intruziunilor, mediul țintă era un calculator mainframe, iar toți utilizatorii erau locali în sistemul respectiv. Acest aspect a simplificat foarte mult sarcina de detectare a intruziunilor, deoarece interacțiunea din exterior era rară. Sistemul de detectare a intruziunilor analiza informațiile de audit furnizate de mainframe, fie local, fie pe o mașină separată, și raporta evenimente suspecte din punct de vedere al securității.

Odată cu utilizarea pe scară largă a Internetului, sistemele de detectare a intruziunilor s-au concentrat pe atacurile asupra rețelei în sine. Atacurile de rețea (DNS spoofing, TCP hijacking, port scanning, ping of death etc.) nu pot fi detectate prin examinarea jurnalului de audit al gazdei, sau cel puțin nu așa de ușor. Prin urmare, au fost dezvoltate instrumente specifice care analizează pachetele de rețea în timp real, căutând astfel de atacuri de rețea. Astfel, în contextul actual, sistemele de detectare a intruziunilor bazate pe rețea (NIDS) sunt indispensabile pentru sistemelor informatice.

Pentru a asigura o securitate cibernetică robustă, multe organizații implementează soluții hibride care combină funcționalitățile HIDS și NIDS. Această abordare integrată permite monitorizarea atât a activităților interne ale dispozitivelor individuale, cât și a traficului de rețea, oferind o perspectivă cuprinzătoare asupra potențialelor amenințări și vulnerabilități. Prin corelarea datelor provenite din ambele surse, sistemele hibride pot detecta și răspunde mai eficient la o gamă largă de atacuri cibernetice, îmbunătățind astfel postura generală de securitate a organizației.

### Intrusion Prevention Systems

Un sistem de prevenire a intruziunilor (IPS) este conceput pentru a analiza continuu traficul și activitățile sistemelor, identificând rapid comportamente neobișnuite sau potențial dăunătoare. Acesta compară fluxurile de date cu modele predefinite de atacuri sau cu tipare normale de funcționare și, în cazul detectării unor anomalii, poate interveni automat pentru a bloca sursa amenințării și a minimiza efectele negative. IPS poate fi implementat la nivelul rețelelor sau pe dispozitive individuale, oferind o protecție în timp real și alertând personalul tehnic pentru a putea lua măsuri suplimentare. Această abordare proactivă ajută la prevenirea incidentelor majore, menținând integritatea operațională și reducând impactul atacurilor asupra infrastructurii.

Există mai multe categorii de sisteme de prevenire a intruziunilor, fiecare adresând un aspect specific al monitorizării și protejării infrastructurii informatice:

* **IPS bazat pe rețea (NIPS):** Monitorizează traficul de rețea la nivelul întregii infrastructuri, analizând pachetele de date pentru a detecta și bloca fluxurile suspecte.
* **IPS bazat pe gazdă (HIPS):** Se instalează direct pe dispozitive (servere, stații de lucru) și supraveghează activitatea locală pentru a identifica semnele unor comportamente anormale sau atacuri asupra sistemului respectiv.
* **IPS pentru rețele fără fir (WIPS):** Specializat pe rețelele wireless, acesta detectează activități neautorizate sau periculoase în mediul fără fir, asigurând protecția comunicațiilor wireless.
* **Sisteme bazate pe analiza comportamentului rețelei (NBA):** Aceste soluții monitorizează fluxurile de date și compară activitatea curentă cu modele predefinite de comportament normal, identificând anomalii care ar putea indica un atac sau o scurgere de informații [5].

În lucrarea practică elaborată se abordează partea de HIDS și NIDS. În ceea ce privește etapa prevenției proactive, aceasta necesită studiu amănunțit și modele mult mai complexe.

## Phishing și spoofing

Atacul de tip phishing reprezintă o metodă prin care atacatorii obțin acces la informații sensibile și restricționate ale utilizatorilor finali, utilizând ingineria socială și tehnologia, fiind totodată și cea mai utilizată metodă de către actorii cibernetici pentru a exploata confidențialitatea utilizatorilor de internet [6].

Printre tipuri cunoscute de phishing se numără următoarele:

* Email Phishing - Atacatorii trimit e-mailuri frauduloase care par să provină de la organizații legitime, solicitând utilizatorilor să furnizeze informații confidențiale.
* Spear Phishing - O formă mai avansată de phishing, care vizează un individ sau o organizație specifică, folosind informații personalizate pentru a face mesajele mai convingătoare.
* Whaling - O tehnică de spear phishing direcționată către persoane cu funcții de conducere, cum ar fi CEO sau CFO, cu scopul de a obține acces la resurse financiare sau informații critice.
* Vishing (Voice Phishing) - Atacatorii folosesc apeluri telefonice pentru a manipula victimele să furnizeze date confidențiale.
* Smishing (SMS Phishing) - Atacurile sunt realizate prin mesaje SMS frauduloase, prin care se solicită accesarea unor linkuri malițioase.
* Clone Phishing - Crearea unor copii identice ale e-mailurilor legitime pentru a convinge utilizatorii să acceseze linkuri periculoase.
* Malvertising - Utilizarea anunțurilor publicitare online pentru a redirecționa utilizatorii către site-uri de phishing.
* Pharming - Manipularea setărilor DNS ale unui utilizator pentru a redirecționa cererile legitime către site-uri frauduloase.

Pe de altă parte, spoofing-ul (falsificarea) este un tip de atac cibernetic care implică falsificarea identității unei surse de comunicare pentru a obține acces neautorizat sau pentru a manipula victimele. Spoofing-ul este utilizat în combinație cu alte tehnici de atac, cum ar fi phishing-ul, pentru a crește eficiența acestuia. Spre exemplu, un atacator poate folosi email spoofing pentru a trimite un mesaj de spear phishing unui angajat dintr-o companie, solicitând transferul unor fonduri către un cont fals. Tipurile de spoofing includ:

* Email Spoofing - Atacatorii falsifică adresa expeditorului unui e-mail pentru a părea că mesajul provine de la o sursă legitimă.
* IP Spoofing - Manipularea pachetelor de date pentru a ascunde identitatea reală a atacatorului și a eluda mecanismele de securitate.
* DNS Spoofing - Redirecționarea utilizatorilor către site-uri frauduloase prin modificarea răspunsurilor serverelor DNS.
* Caller ID Spoofing - Falsificarea numărului de telefon afișat pentru a convinge victimele că apelul provine de la o entitate de încredere.

## Contramăsuri împotriva atacurilor cibernetice

În fiecare zi, suntem expuşi, atât acasă cât şi la locul de muncă, la ameninţări ce îşi au originea în spaţiul virtual. În majoritatea cazurilor nici măcar nu suntem conştienţi de acest lucru, sau dacă-l realizăm, nu reacţionăm la aceste ameninţări într-o manieră adecvată. În media apar zilnic articole referitoare la incidente de securitate şi la impactul pe care acestea îl au asupra noastră, ca indivizi sau organizaţii deopotrivă.

În urma infectării sistemului informatic, acesta poate deveni, de exemplu, parte a unui botnet - o rețea de calculatoare infectate prin diverse metode de către o persoană/entitate rău-intenţionată în vederea utilizării acesteia în folosul celui care controlează reţeaua (botmaster), pentru sustragerea de date confidenţiale sau bancare, pentru iniţierea de atacuri de tip DDoS, pentru spargerea parolelor sau pentru căutarea şi exfiltrarea de informaţii [7].

Un utilizatorul obișnuit se poate preveni a fi victima unui atac cibernetic, în primul rând, prin prudență în navigarea pe Internet, dar și în offline prin protejarea datelor personale și evitarea utilizării dispozitivelor necunoscute sau nesigure. Este esențial ca utilizatorii să nu descarce fișiere sau să acceseze linkuri din surse necunoscute, să actualizeze constant sistemul de operare și aplicațiile utilizate și să folosească soluții de securitate, precum antivirus și firewall. De asemenea, utilizarea unor parole puternice și unice pentru fiecare cont, împreună cu activarea autentificării în doi pași (2FA), poate reduce considerabil riscul de compromitere a datelor personale. Educația și conștientizarea riscurilor cibernetice sunt, de asemenea, factori esențiali în prevenirea atacurilor, deoarece mulți atacatori exploatează lipsa de informare a utilizatorilor pentru a obține acces la sistemele informatice.

În plus, organizațiile trebuie să implementeze politici stricte de securitate cibernetică, să instruiască angajații cu privire la amenințările existente și să monitorizeze constant rețelele pentru a detecta activități suspecte. Numai printr-o abordare proactivă și printr-un comportament responsabil în mediul digital se poate reduce semnificativ riscul unui atac cibernetic

# Implementarea practică a sistemului de programe

Dezvoltarea unei aplicații software presupune parcurgerea unui ciclu de viață al dezvoltării software (SDLC), un proces continuu de optimizare și perfecționare. Implementarea practică a sistemului de programe presupune transpunerea conceptelor și specificațiilor tehnice într-un produs funcțional, prin scrierea codului, integrarea componentelor, testarea și optimizarea performanței aplicației. Acest proces asigură conformitatea cu cerințele utilizatorilor și pregătirea soluției pentru implementare și utilizare reală. Etapele SDLC, prezentate în Figura 5, oferă un plan ce trebuie urmat pentru a dezvolta un produs software de calitate, într-un buget și timp alocat [8]. Acest plan fiind format din 11 etape: comunicare, colectarea cerințelor, studiu de fezabilitate, analiza sistemului, design software, programare/coding, testare, integrare, implementare, mentenanţă și, în final, dispunerea produsului software.

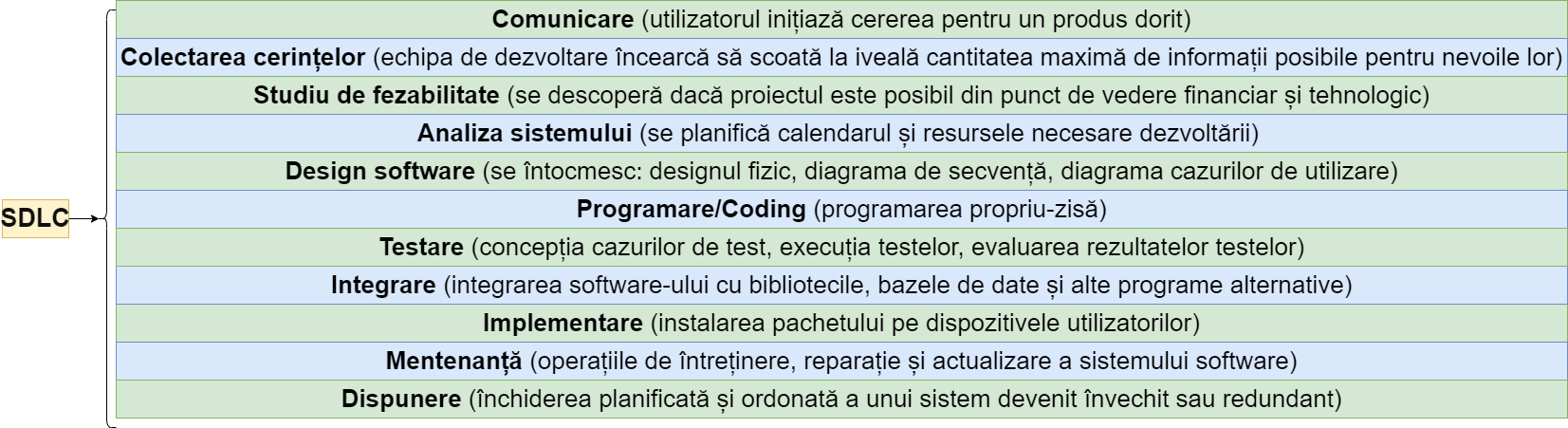


Figura 5 - Ciclul de viață al dezvoltării software

În cadrul proiectelor complexe este folosită o metodologie de dezvoltare software. Câteva exemple uzuale sunt: Modelul cascadă (Waterfall model), Modelul incremental, Modelul în V (V-model), Modelul ESA (European Space Agency model), Dezvoltarea “agilla” (Agile development), Modelul SCRUM, Dezvoltarea pe bază de prototip (Prototyping), Modelul în spirarlă (Spiral model). În cadrul acestui proiect am folosit modelul incremental care livrează un produs executabil la fiecare etapă e dezvoltării, îmbunătățirile fiind aduse pe parcursul dezvoltării.

## Environment

Experimentările au fost realizate folosind TensorFlow (versiunea 2.18.0) pe platforma Windows (Windows 10 Pro), implementarea fiind realizată cu ajutorul Keras (versiunea 3.8.0) și Python (versiunea 3.10.16). Configurația mediului experimental este prezentată în Tabelul 6, de mai jos:

|  |  |
| --- | --- |
| Proiect | Versiune/Mediu |
| Sistem de operare | Windows 10 Pro, versiune 22H2 |
| CPU | i7-11800H |
| Unitate de procesare grafică (GPU) | GeForce RTX 3050 Ti |
| Memorie | 16 GB (15.8 GB utilizabili) |
| Mediu de dezvoltare | Visual Studio Code |
| Limbaj de programare utilizat | Python - versiunile 3.7.16 și 3.10.16 |
| Tensorflow | Versiunile 2.11.0 și 2.18.0 |
| Keras | Versiunile 2.10.0 și 3.8.0 |

Tabelul 6 - Configurația mediului experimental

Pentru acest proiect, mediul izolat necesar a fost realizat utilizând exclusiv linia de comandă și salvat sub forma unui fișier YAML. O listă a comenzilor folosite pentru a instala pachetele necesare este expusă în Tabelul 7, de mai jos:

|  |
| --- |
| conda create -n gputf3.10 python=3.10.16 |
| conda install jupyter |
| conda install -c conda-forge tensorflow-gpu |
| pip install tensorflow |
| pip install pandas |
| pip install scikit-learn |
| pip install nbformat |
| pip install nbconvert |
| conda install -c conda-forge capstone |
| pip install Pillow |
| pip install matplotlib |
| conda env export > environment.yml |
| conda env create --name myenv -f environment.yml |

Tabelul 7 - Pregătirea mediului de lucru și a pachetelor necesare

## Fluxul aplicației

Aplicația implementată are ca scop prezentarea succintă a unor metode de bază pentru analiza preliminară și detectarea fișierelor sau a traficului malițios. Spre deosebire de soluțiile antivirus existente pe piață, care utilizează modele AI avansate și baze de date extinse de semnături pentru detectarea și prevenirea malware-ului, această aplicație se concentrează pe abordări fundamentale, oferind o perspectivă simplificată și cuprinzătoare asupra procesului de securitate cibernetică. Totodată, s-a încercat acoperirea unui spectru cât mai larg în ceea ce privește prevenirea breșelor de securitate, fiind implementate metode de filtrare/clasificare a email-urilor, a fișierelor și a traficului de internet care să ajute un specialist din domeniu.

Fluxul aplicației este destul de intuitiv pentru un utilizator care are cunoștințe de bază în ceea ce privește domeniul IT. Meniul principal este format din șapte butoane: Spam Analyser, Phishing Analyser, File Info, Online File Analyser, Offline File Analyser Image Based, Traffic Live Analyser, LLM Analyser.

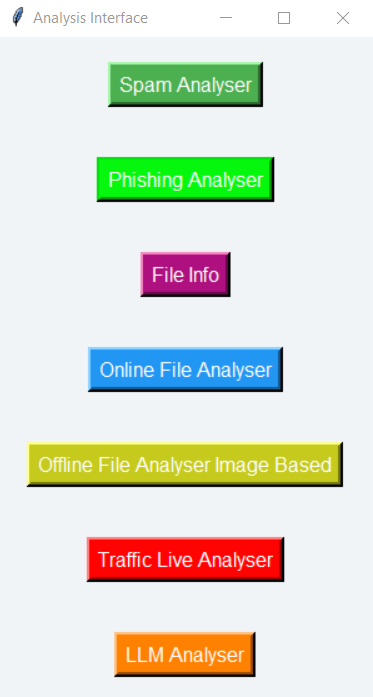


Figura 6 - Meniul principal al aplicației desktop

Fiecare funcţionalitate, reprezentată prin butoanele specife ale interfeţei principale (Figura 6), are o implementare modulară, folosind fișiere Jupyter Notebook (extensie .ipynb) independente, care sunt apelate prin butoanele specifice din meniul principal. Totodată, am folosit rularea metodelor folosind thread-uri separate, pentru a evita eventualele blocări ale interfeței grafice - “threading.Thread(target=phishing\_analyser).start()”. Tot pentru a îmbunătăți timpul de răspuns, modelele antrenate au fost salvate în foldere specifice pentru ca rulările ulterioare să nu necesite reluarea procesului de antrenarea.

## Seturi de date folosite

**SMS Spam Collection** [9] este un set public de mesaje SMS etichetate, care au fost colectate pentru cercetarea spamului de pe telefonul mobil. Corpusul, reprezentat de fişierul spam.csv, este alcătuit din 5.572 instanțe unice etichetate “spam/ham(legitim)”, din surse gratuite de cercetare: 425 de mesaje spam SMS a fost extrase manual de pe site-ul web Grumbletext, un subset de 3.375 de mesaje SMS au fost alese aleatoriu din National University of Singapore SMS Corpus, o listă de 450 de mesaje colectate din teza de doctorat a lui Caroline Tagg disponibilă online [[Web Link](https://etheses.bham.ac.uk/id/eprint/253/1/Tagg09PhD.pdf)] și 1.332 mesaje SMS (dintre care 322 mesaje spam) încorporate din corpusul creat de José María Gómez Hidalgo - SMS Spam Corpus v.0.1 Big.

În Figura 7 sunt prezentate cuvintele cele mai frecvente în mesajele legitime și, respectiv, cele spam. Totodată, în Figura 8, se poate observa că mesajele legitime au un număr mai mic de cuvinte, iar disproporțioanlitatea setului de date este evidentă.



Figura 7 - Word Cloud pentru a vizualiza cele mai frecvente cuvinte

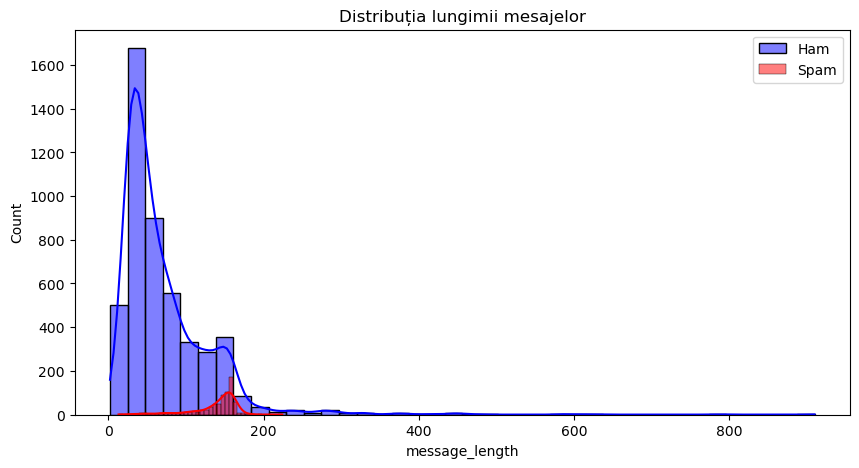


Figura 8 - Distribuția lungimii mesajelor HAM vs Spam

**Phishing Email Dataset,** creat de **zefang-liu** este un set de date utilizat pentru detectarea email-urilor de phishing. Acest set de date conține 18.650 e-mailuri etichetate ca fiind "Phishing Email" sau "Safe Email", și este utilizat pentru antrenarea și testarea modelelor de machine learning în detectarea fraudei prin email. După sanitizarea acestui set de date (eliminarea valorilor nule, și duplicatelor) rămân 17.538 de mostre în dataset (10.980 Safe și 6558 Phishing), lucru evidențiat și în Figura 9.

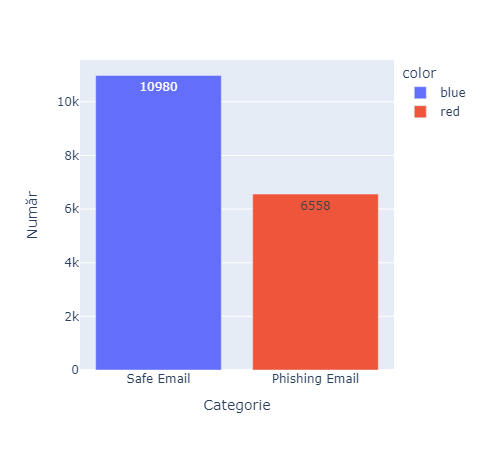


Figura 9 - Phishing Email Dataset după sanitizare

Setul de date **Malimg** constă din 9.348 mostre de malware din 25 de familii diferite. Malimg a fost creat convertind binarele malware în întregi pe 8 biți fără semn, formând o matrice , ce poate fi vizualizată ca o imagine în tonuri de gri. Clasele de malware sunt: Adialer.C, Agent.FYI, Allaple.A, Allaple.L, Alueron.gen!J, Autorun.K, C2LOP.P, C2LOP.gen!g, Dialplatform.B, Dontovo.A, Fakerean, Instantaccess, Lolyda.AA1, Lolyda.AA2, Lolyda.AA3, Lolyda.AT, Malex.gen!J, Obfuscator.AD, Rbot!gen, Skintrim.N, Swizzor.gen!E, Swizzor.gen!I, VB.AT, Wintrim.BX, Yuner.A [10]. Din Figura 10 se poate sesiza că acest set de date este destul de dezechilibrat: o mulțime de mostre aparțin clasei 2 (Allaple.A) și clasei 3 (Allaple.L).

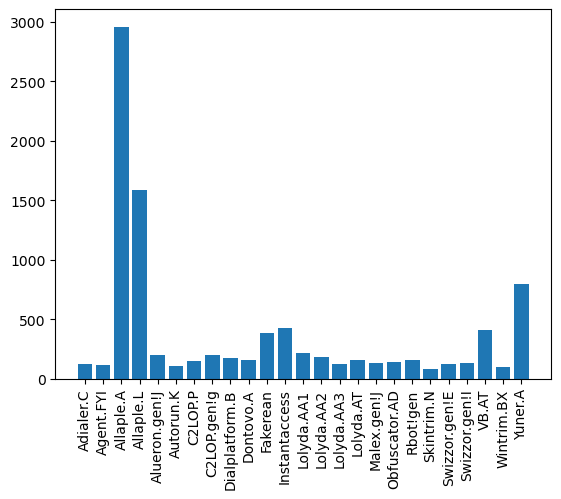


Figura 10 - Repartiția claselor în setul de date Malimg

**NSL-KDD** este un set de date utilizat pe scară largă pentru cercetare în domeniul detectării intruziunilor în rețelele de calculatoare. Acesta este o versiune îmbunătățită a celebrului set de date KDD Cup 1999, care a fost criticat pentru problemele sale, precum redundanța și distribuția dezechilibrată a claselor. Față de setul original, în NSL-KDD sunt eliminate datele duplicat și selectate un număr echilibrat de instanțe pentru antrenare și testare. NSL-KDD este format din 125,973 înregistrări, fiecare cu 41 de caracteristici care descriu conexiunea – durată, număr de pachete trimise/primate, protocoale utilizate etc.

Setul de date are 23 de clase: back, buffer\_overflow, ftp\_write, guess\_passwd, imap, ipsweep, land, loadmodule, multihop, neptune, nmap, normal, perl, phf, pod , portsweep, rootkit, satan, smurf, spy, teardrop, warezclient, warezmaster. Cele 23 de clase pot fi împărțite în patru categorii de atacuri și trafic normal: **DoS** (Denial of Service - atacatorul încearcă să împiedice utilizatorii legitimi să acceseze resursele), **R2L** (Remote to Local - atacatorul încearcă să obțină acces la un sistem de la distanță), **U2R** (User to Root – atacatorul încearcă să obțină privilegii administrative), **Probe** (Probing - atacuri care investighează rețeaua pentru a găsi vulnerabilități), **Normal** (traffic legitim) [11].

|  |  |
| --- | --- |
| Categorie atac | Tipuri de atac și număr mostre din setul de antrenare |
| Probe | satan (3633), portsweep (2931), nmap (1493), jpsweep (3599) |
| R2L | spy (2), phf (4), multihop (7), imap (11), guess\_passwd (53), ftp\_write (8), warezmaster (20), warezclient (890) |
| U2R | rootkit (10), perl (3), loadmodule (9), buffer\_overflow (30) |
| DoS | Teardrop (892), smurf (2646), pod (201), neptune (41214), land (18), back (956) |

Tabelul 8 - Distribuția claselor reprezentând atacuri în setul KDDtrain+.txt

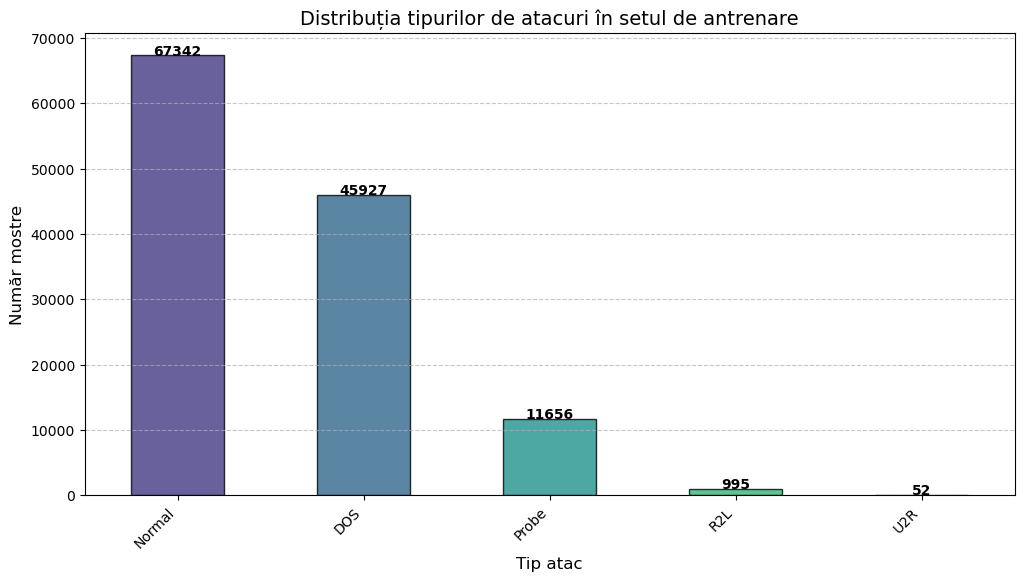


Figura 11 - Distribuția tipurilor de atacuri în setul de antrenare

## Detecţia e-mailurilor frauduloase

Pentru funcționalitatea de detecție e-mailurilor de tip spam am folosit mai întâi setul de date SMS Spam Collection. Detecția email-urilor de tip spam este realizată prin clasificarea “binară” a textului folosind mai multe tipuri de clasificatori: Naive Bayes**[[3]](#footnote-3)**, Logistic Regression**[[4]](#footnote-4)**, SVC**[[5]](#footnote-5)**, KNN**[[6]](#footnote-6)**, Decision Tree**[[7]](#footnote-7)**, Random Forest**[[8]](#footnote-8)**, Bagging Classifier**[[9]](#footnote-9)**, Extra Tree Classifier**[[10]](#footnote-10)**, Gradient Boosting**[[11]](#footnote-11)**, AdaBoost**[[12]](#footnote-12)**, XGBoost**[[13]](#footnote-13)** și CatBoost**[[14]](#footnote-14)**.

În Figura 10 sunt prezentați acești clasificatori și se pot observa câțiva dintre parametrii folosiți: learning\_rate (rata de învățare), max\_depth(adâncimea maximă a fiecărui arbore), n\_estimators (numărul de arbori în ansamblu), max\_iter (numărul maxim de iterații pentru algoritmul de optimizare), C (parametrul de regularizare invers - valori mai mici impun o regularizare mai puternică - previne supraantrenarea), eval\_metric (metrica folosită pentru evaluare).

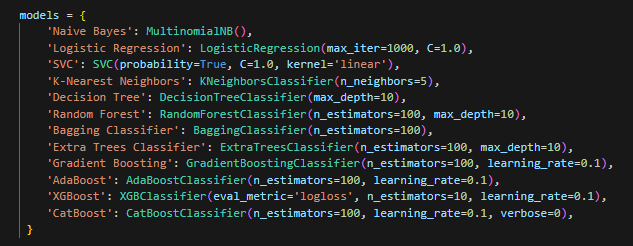


Figura 12 - Clasificatori folosiți în clasificarea binară a e-mailurilor (spam/ham)

Totodată, în Tabelul 6 este prezentată performața, pe setul de testare (20% din setul de date), a celor 12 clasificatori. Se observă performațe foarte bune în ceea ce privește atât acuratețea (peste 97%), cât și precizia și regăsirea (recall). Totuși, din experimentele efectuate pe fișiere .eml (în limba română și engleză) descărcate din căsuța de spam a Gmail, clasificatorii se comportă diferit. Cel mai bine răspunde clasificatorul SVC, separând foarte bine graniţele dintre cele două clase.

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Acuratețe | Precizie | Regăsire(Recall) |
| Naive Bayes | 0.98 | 0.99 | 0.89 |
| Logistic Regression | 0.98 | 0.99 | 0.84 |
| SVC | 0.98 | 0.97 | 0.87 |
| K-Nearest Neighbors | 0.92 | 1.00 | 0.40 |
| Decision Tree | 0.97 | 0.93 | 0.82 |
| Random Forest | 0.90 | 1.00 | 0.29 |
| Bagging Classifier | 0.97 | 0.93 | 0.87 |
| Extra Trees Classifier | 0.88 | 1.00 | 0.14 |
| Gradient Boosting | 0.97 | 0.98 | 0.79 |
| AdaBoost | 0.88 | 1.00 | 0.13 |
| XGBoost | 0.95 | 0.94 | 0.70 |
| CatBoost | 0.97 | 0.99 | 0.79 |

Tabelul 9 - Performanța clasificatorilor obținută pe setul de testare

Folosirea mai multor clasificatori ajută la verificarea și garantarea corectitudinii rezultatului clasificării. Deși este o clasificare binară, complexitatea mesajelor îngreunează siguranța predicției. Totodată, setul de date oferă predicții mai bune pe text (e-mailuri) în limba engleză, comparativ cu limba română. Dar sarcina de bază, aceea de clasificare binară, este îndeplinită cu o acuratețe foarte bună.

Pasul următor a fost detecția email-urilor de tip phishing care a fost implementată folosind setul de date Phishing Email Dataset, iar pentru clasificare următorii algoritmi: Naive Bayes, Logistic Regression, Gradient Boosting Classifier, XGB Classifier, Decision Tree, Random Forest, MLP Classifier.

În afară de acești clasificatori am folosit și o arhitectură de rețea neuronală LSTM a cărei arhitectură este prezentată în Figura 13. Model LSTM a fost antrenat pe 80% din date, iar 20% au fost date de validare și s-a folosit early stopping. Performanțele clasificatorilor și rețelei LSTM, pe setul de test, sunt expuse în Tabelul 7. Ca rezultat experimental - arhitectura neuronală a subclasat clasificatorii în ceea ce privește numărul răspunsurilor corecte pe date arbitrar alese (altele decât setul de antrenare/testare), deși graficul antrenării nu denotă acest fapt. Totodată acest fapt fiind reliefat în acuratețea și rata de regăsire calculate.

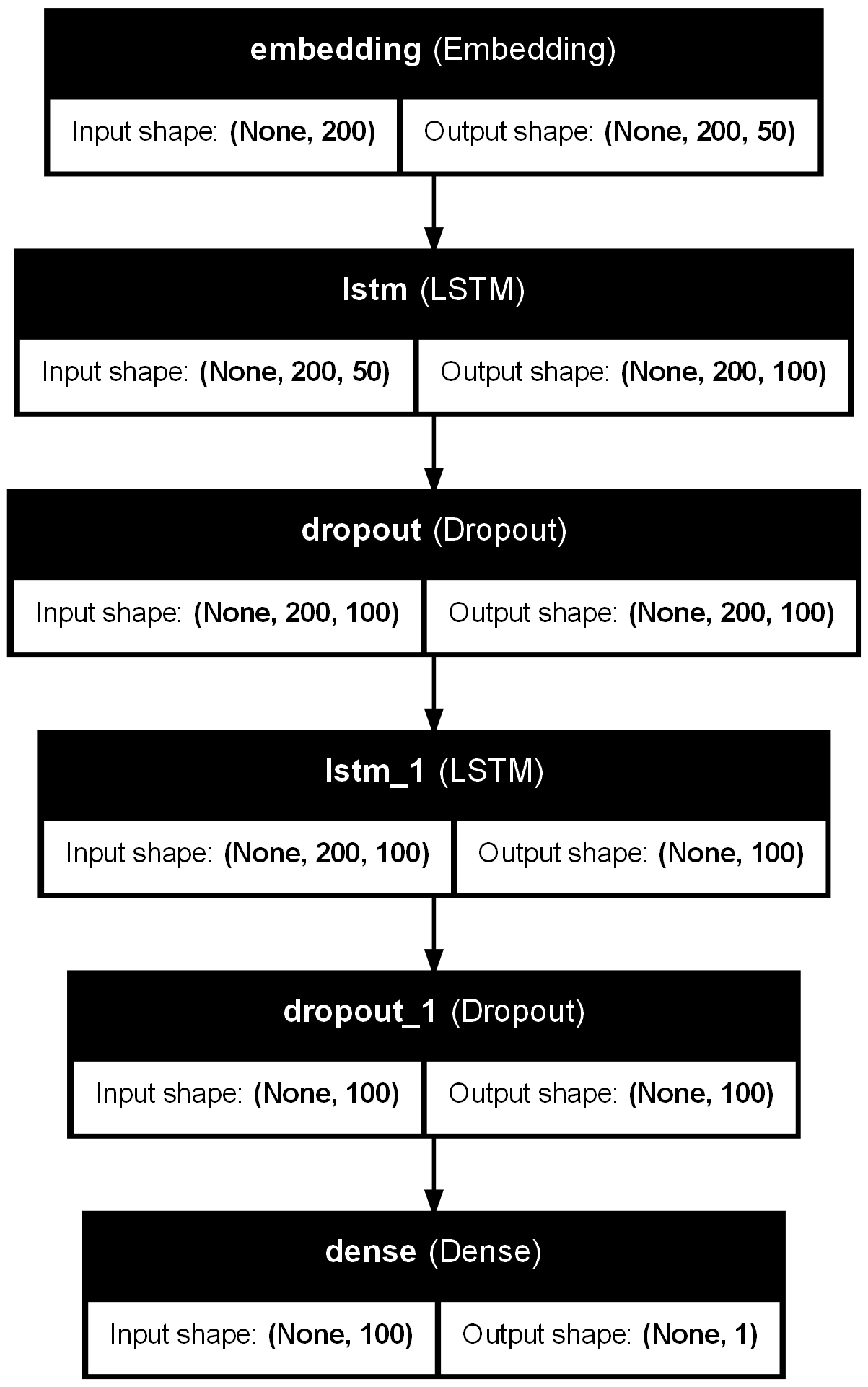


Figura 13 - Arhitectura modelului LSTM folosită în clasificarea e-mailurilor

|  |  |
| --- | --- |
|  |  |

Figura 14 - Acuratețea și funcția de cost pentru arhitectura LSTM

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Acuratețe | Precizie | Regăsire(Recall) |
| Naive Bayes | 0.96 | 0.98 | 0.96 |
| Logistic Regression | 0.96 | 0.97 | 0.96 |
| GradientBoosting Classifier | 0.93 | 0.95 | 0.94 |
| XGBoost | 0.97 | 0.98 | 0.95 |
| Decision Tree | 0.91 | 0.94 | 0.91 |
| Random Forest | 0.96 | 0.97 | 0.96 |
| MLP Classifier | 0.97 | **0.99** | 0.97 |
| LSTM model | **0.98** | 0.98 | **0.98** |

Tabelul 10 - Performanța clasificatorilor și modelului LSTM pe setul de testare

## Analiza fişierelor online/offline

Analiza de fișiere folosind resurse online/offline presupune utilizarea unor metode și tehnologii variate pentru identificarea caracteristicilor suspecte ale fișierelor într-un mod cât mai eficient și rapid. În mediul online, platformele de scanare, cum ar fi VirusTotal, oferă un avantaj prin accesul la baze de date actualizate și algoritmi de detecție din mai multe surse. Aceste soluții sunt eficiente pentru detectarea de malware cunoscut, dar pot întâmpina limitări în cazul în care fișierul conține un tip nou de amenințare sau dacă platforma este inaccesibilă.

În mediul offline, analiza fișierelor devine o provocare din cauza lipsei unei baze de date centralizate și a resurselor externe, ceea ce poate necesita implementarea unor algoritmi locali pentru detecția semnelor de comportament malițios. Această abordare offline este esențială pentru organizații care trebuie să opereze în medii izolate de rețea sau pentru aplicații ce necesită răspunsuri imediate fără a depinde de o conexiune la internet. Astfel, integrarea unui model de învățare automată poate oferi o soluție complementară, capabilă să analizeze structura și caracteristicile fișierelor, identificând posibile amenințări chiar și în absența unei conexiuni la resurse online.

Având în vedere cele spuse mai sus, rezolvarea sarcinii de analiză preliminară a fișierelor a presupus implementarea mai multor funcționalități complementare. Un specialist în securitate cibernetică analizează un fișier, în primul rând, din punct de vedere al metadatelor. Pe sistemul de operare linux, există comenzi specifice în acest sens: file, strings, exiftool, stat etc. Aceste instrumente permit extragerea informațiilor esențiale, precum determinarea tipului de fișier, extragerea de string-uri, extrage metadatele ascunse din fișiere multimedia, data creării, ultimele modificări, drepturile de acces, precum și detalii despre proprietarul fișierului și structura internă a acestuia.

Astfel, în funcționalitatea “File Info”, am introdus calcularea și afișarea acestor metadate referitoare la un fișier, ce pot fi folosite mai departe în analiza malware folosind tehnici de AI. Informațiile afișate în interfața prezentată în Figura 14 sunt: tipul fișierului, dimensiune, permisiuni, proprietar, grup, dată creării, ultima modificare și accesare, hash-ul fișierului (MD5 şi SHA256), vizualizare hexa, string-uri extrase, importuri de librării (în cazul fișierelor executabile din linux).



Figura 15 - Metadatele afișate în interfața meniului File Info

În etapa a doua a dezvoltării, am integrat API-urile VirusTotal[[15]](#footnote-15) și MetaDefender[[16]](#footnote-16) în cadrul funcționalității “Online File Analyser” pentru a analiza fișierele încărcate. Practic, am folosit aceste servicii online pentru a compara hash-urile și semnăturile fișierelor cu bazele de date de malware existente și pentru a verifica eventualele comportamente suspecte, asigurând astfel o evaluare rapidă și precisă a riscului de securitate asociat fiecărui fișier. Totodată am comparat hash-urile MD5 şi SHA256 calculate pentru fiecare fişier şi le-am comparat cu liste de hash-uri malițioase găsite online.

Aceste API-uri analizează fișierele folosind multiple motoare antivirus și raportează rezultatele detaliate - cum ar fi numărul de engine-uri care au identificat activitate suspectă, data scanării și alte informații relevante. În final, utilizatorul primește un raport comprehensiv (Figura 15) care indică dacă fișierul este sigur sau prezintă riscuri, facilitând astfel luarea unor decizii informate în materie de securitate.

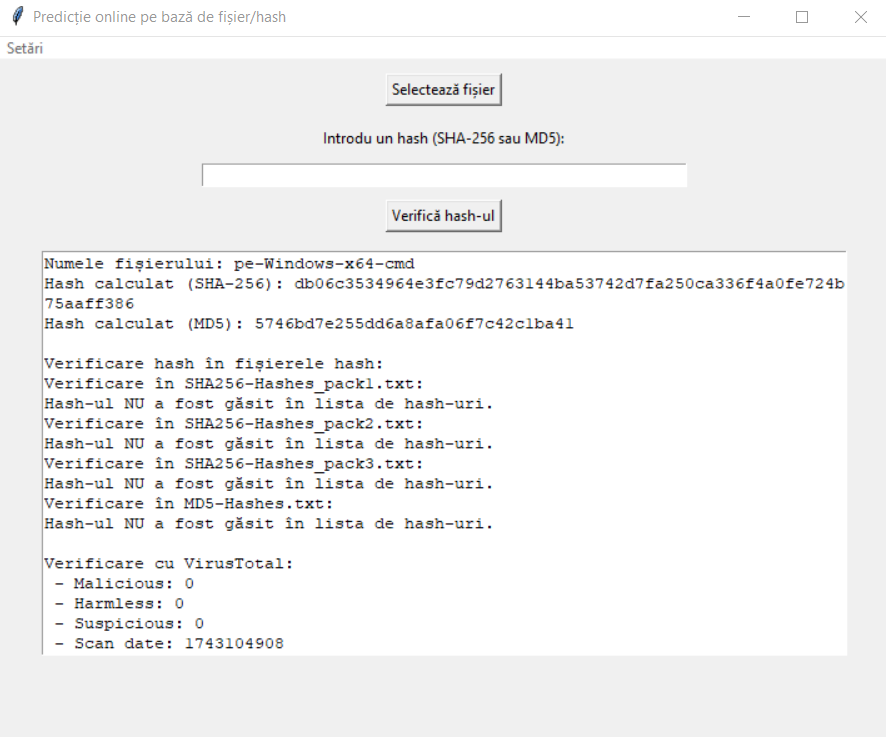


Figura 16 - Funcționalitatea Online File Analyser

În ceea ce privește analiza offline a unui fișier, funcționalitatea se bazează pe prelucrarea datelor binare printr-o serie de pași automatizați, care transformă fișierul într-o reprezentare vizuală care poate fi analizată de un model de învățare automată.

Mai întâi, fișierul este parcurs în bucăți de câte 16 octeți și convertit fiecare octet într-o reprezentare hexazecimală - rezultatul fiind salvat într-un fișier text temporar (temp.bytes) şi afişat în interfaţă (Figura 16). Tabloul numeric este apoi reformatat pentru a construi o imagine. Se determină o lățime optimă, calculată astfel încât să se apropie de o structură pătrată, iar înălțimea este dedusă astfel încât produsul dintre lățime și înălțime să corespundă numărului total de valori. Imaginea monocronă obținută este convertită într-o imagine color, replicând datele pe cele trei canale RGB. Pentru a standardiza intrarea modelului de învățare automată, imaginea RGB este redimensionată la o dimesiune fixă. Această redimensionare asigură că toate imaginile folosite la predicție au aceleași dimensiuni, facilitând procesarea și compararea între ele. Etapele acestui proces sunt reprezentate în Figura 17. Imaginea preprocesată este apoi furnizată modelului pre-antrenat pentru clasificare. Modelul, familiar cu un set de clase de malware, analizează imaginea și produce o distribuție de probabilități, indicând cât de probabil este ca fișierul analizat să aparțină fiecarei categorii.

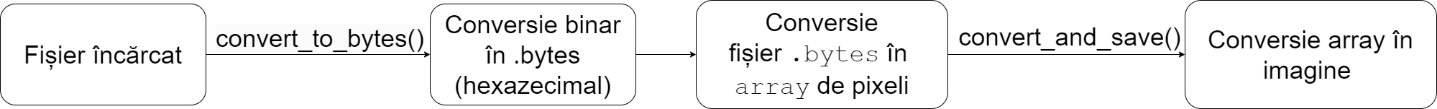


Figura 17 - Etapele de procesare ale fișierului (funcția process\_file)

În final, rezultatul analizei este afișat în interfața grafică (Figura 18), unde se prezintă atât imaginea generată, cât și informațiile procesate (reprezentarea hexazecimală și șirurile extrase din datele binare). Totodată, predicțiile modelului sunt afișate sub forma unor procente pentru fiecare clasă de malware, oferind utilizatorului o imagine clară a analizelor efectuate. Rezultatul este, de asemenea, salvat într-un istoric pentru a putea fi consultat ulterior.

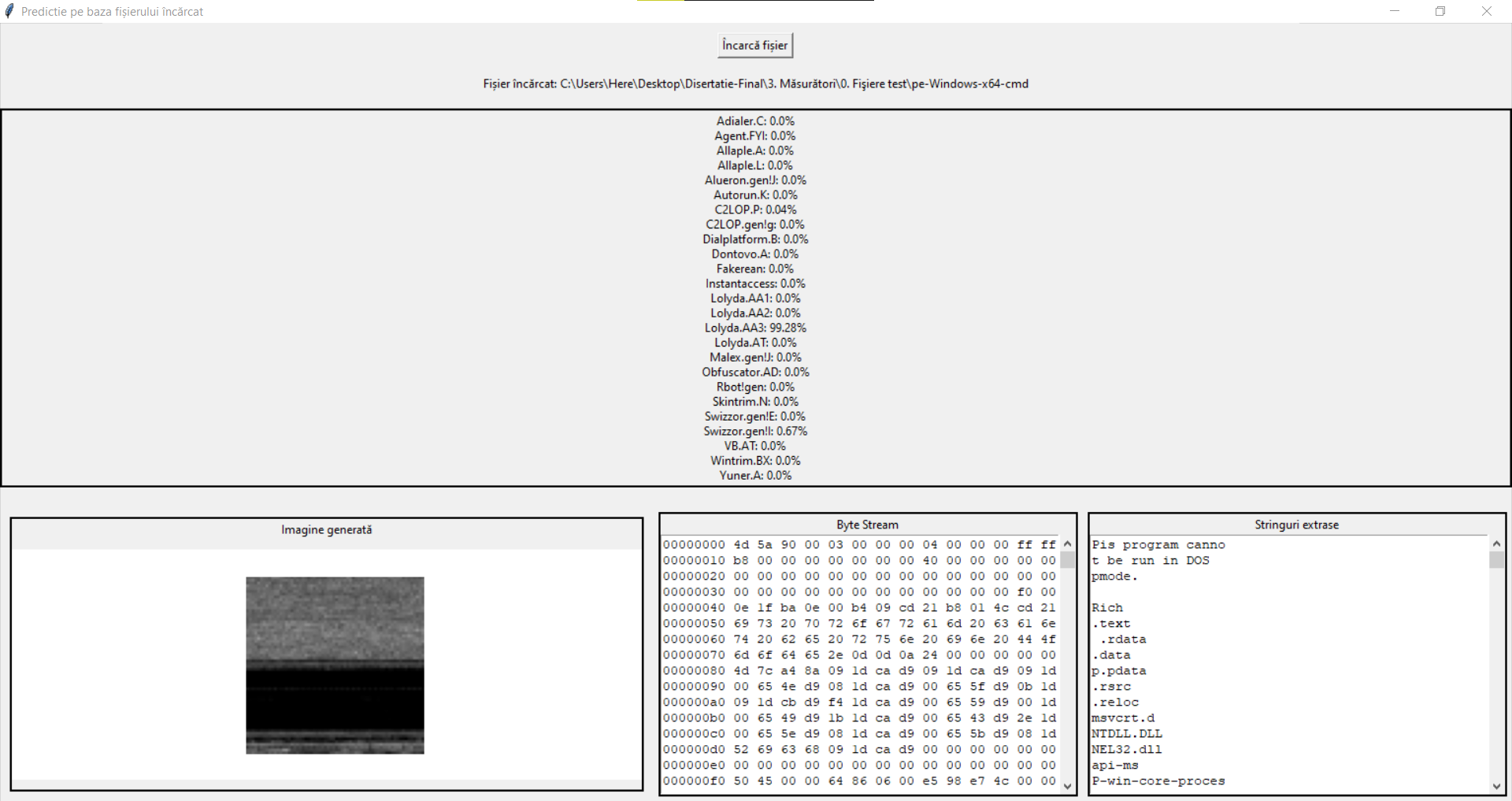


Figura 18 - Funcţionalitatea Offline File Analyser

Arhitectura neuronală utilizată iniţial, petru a testa clasificarea imaginilor malware, este una destul de simplă. Aceasta este formată din două straturi convoluționale urmate de un strat de max pooling (Figura 19). Am folosit optimizatorul Adam și ca funcție de cost entropia încrucișată categorică. Performanțele obținute pe setul de testare, cu această arhitectură, sunt mulțumitoare, cel puțin pe datasetul Malimg.



Figura 19 - Arhitectura convoluțională folosită inițial

Ultima îmbunătățire a fost folosirea unei arhitecturi VGG16[[17]](#footnote-17).

Modificările aplicate designului original, propus de Simonyan și Zisserman, modelului VGG16 [12] sunt următoarele:

* Forma de intrare a fost modificată de la 224 × 224 × 3 la 64 x 64 x 3 (sau 256 × 256 × 3, în funcție de dimensiunea imaginii generate);
* Dropout a fost adăugat după straturile convoluționale: 1, 3, 5, 6, 8, 9, 11, 12 și 13;
* Primul strat complet conectat și următorul de dropout au fost eliminate pentru a simplifica modelul și a evita supraadaptarea(overfitting);
* Numărul de unități din al doilea strat dens a fost modificat de la 4096 la 512;
* Normalizarea lotului (batch normalization) a fost adăugată după fiecare strat, cu excepția stratului de ieșire;
* Lungimea stratului de ieșire a fost redusă de la 1000 la 25 (numărul de clase).

În final am 13 straturi convoluționale, reprezentate sumar în Figura 20, primul cu dimensiunea de ieșire (256, 256, 64) iar ultimul (16, 16, 512). Funcția de activare finală este Softmax cu 25 de clase pentru clasificare.

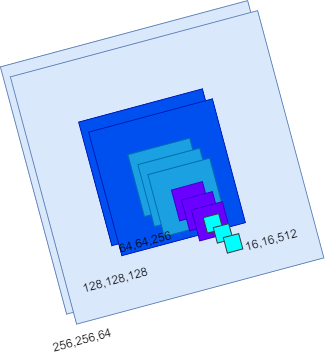


Figura 20 - Proiecție a celor 13 straturi convoluționale din arhitectură

Folosind această arhitectură am observat că timpul de antrenare diferă destul de mult de la un input la altul. Antrenând modelul, de-a lungul a 50 de epoci (folosind, totuși, early stopping) cu imagini 64x64x3, o epocă durează între 138s și 156s, iar folosind ca input imagini de dimensiune 256x256x3, timpul pentru parcurgerea unei epoci este de aproximativ două ore jumătate - o diferență foarte mare. Numărul total de parametri antrenabili fiind, în cel de-al doilea caz, fiind 31.514.713 (120.26 MB), iar dimensiunea modelului de 360MB. Timpul crescut de antrenare face destul de dificile testele multiple folosind diferiți parametri de antrenare sau configurare a rețelei.

Pentru un număr de 14 epoci (timp de antrenare aproximativ nouă ore), graficul acurateții și costului în cazul folosirii unu input 256 x 256 și arhitecturii VGG este repezentat în Figura 22. Costul de antrenare scade constant și stabil pe parcursul epocilor, ceea ce sugerează că modelul învață eficient pe setul de antrenare. Costul de validare are fluctuații mari (mai ales la început și la sfârșit), ceea ce indică instabilitate în generalizarea pe date noi. După epoca 7-8, costul de validare începe să crească, ceea ce este un idiciu clar de overfitting (modelul învață prea bine datele de atrenare și nu generalizează bine). Acuratețea de antrenare crește rapid și se stabilizează la valori ridicate (apx. 0.9), semn că modelul se potrivește bine pe datele de antrenare. Acuratețea de validare este extrem de variabilă. Deși atinge uneori valori comparabile cu cele de antrenare (epocile 7 și 10), aceasta scade dramatic spre final (epoca 12-13), indicând din nou overfitting și o posibilă problemă cu datele de validare (puține sau distribuite diferit față de cele de antrenare).

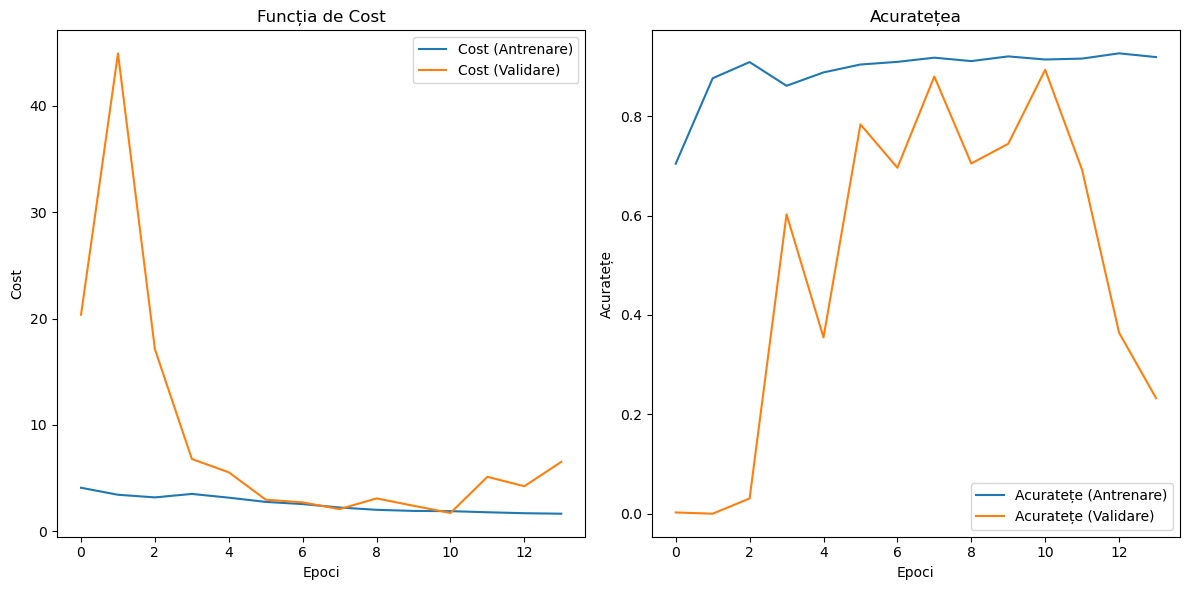
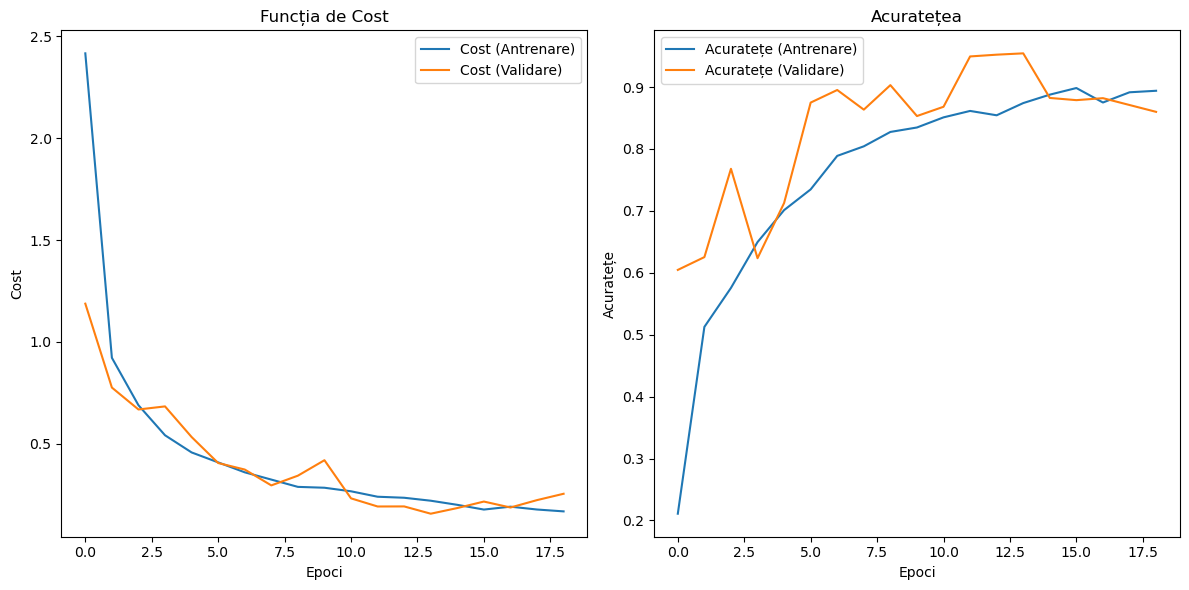


Figura 21 - Graficul funcției de cost și acurateții - 14 epoci - arhitectură VGG

Spre comparație, graficul acurateții și funcției de cost în urma antrenării cu imagini de dimensiune 64 x 64 și folosind arhitectura neuronală inițială (Figura 19), au fost mai ceva mai bune.



Figură 22 - Grafic acuratețe și cost - arhitectură simplă

Aceasta este matricea noastră de confuzie. Pentru ambele arhitecturi, rezultatele sunt asemănătoare. Putem observa că, deși majoritatea malware-urilor au fost clasificate corect, Autorun.K este confundat mereu cu Yuner.A. Acest lucru se datorează probabil faptului că avem foarte puține exemple de Autorun.K în setul nostru de antrenament. În plus, Swizzor.gen!E este adesea confundat cu Swizzor.gen!l, ceea ce poate fi explicat prin faptul că provin din familii foarte apropiate și, prin urmare, ar putea avea similitudini în codul lor.

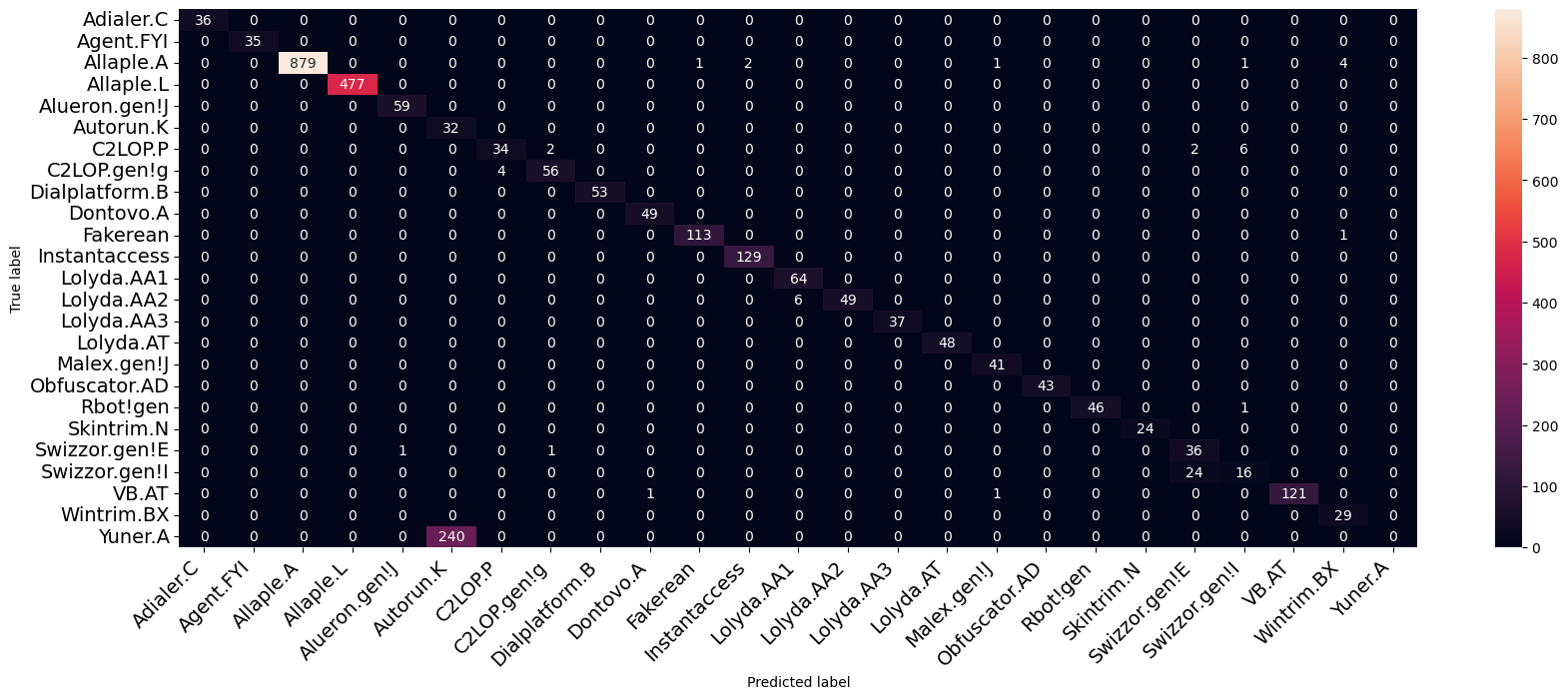


Figura 23 - Matricea de confuzie calculată

Problema care intervine în conceptul de implementare propus este că imaginile din setul de date Malimg au dimensiuni foarte variate - de la 64x216 la 798x915 pixeli. Totodată, modelul meu de rețea neuronală a fost antrenat pe imagini ale datsetului redimensionate (Figura *23*) - de dimensiune fixă (64x64x3 sau 256x256x3). Am menționat anterior că și fișierele mele ce le încarc, pentru predicție, sunt supuse unor transformări de conversie la imagini cu aceași dimensiune fixă (64x64x3 sau 256x256x3). M-am gândit că dacă redimensionarea este aplicată consecvent, nu se pierde prea multă informație critică ale semnăturii (structurii) unui anumit malware.

****

Figura 24 - Încărcarea și preprocesarea datasetului Malimg

În urma experiențelor, am concluzionat că dimensiunea ideală a imaginilor de antrenare este 64 x 64. O redimensionare a imaginilor la o dimenisune sub 64 x 64 reduce drastic rezoluția, iar precizia de inferență scade. Totodată, am testat și modele antrenate pe imagini mai mari 128x128, 256x256, dar perfomanțele graficelor de acuratețe și funcției de cost au reliefat performanțe mai slabe. Un alt aspect este cel al complexității arhitecturii. Deși mă așteptam ca VGG16 să obțină rezultate mai bune la antrenare și validare decât o arhitectură simplă, am obținut contrariul. Arhitectura inițială a obținut rezultate ceva mai bune. Probabil datorită datasetului care nu este la fel de complex ca ImageNet[[18]](#footnote-18) sau este necesară o antrenare pe parcursul a mai multor epoci.

## Analiza traficului de date

Analiza traficului de date reprezintă un element esențial în asigurarea securității rețelelor de calculatoare și în identificarea rapidă a activităților anormale. Această analiză se bazează pe monitorizarea continuă a fluxului de informații, prin intermediul unor instrumente și tehnologii avansate, care permit atât detectarea incidentelor de securitate, cât și reacția promptă în caz de atac.

În mediul online, analiza traficului de date se realizează prin colectarea unor volume mari de informații despre pachetele care traversează rețeaua. Utilizarea tehnologiilor precum NetFlow, sFlow sau captura de pachete (PCAP) permite extragerea unor date detaliate incluzând adresele IP sursă și destinație, porturile implicate, tipurile de protocoale și timpii de transmisie. Aceste informații sunt esențiale pentru a identifica tipare de trafic suspect sau anomalii, cum ar fi:

* **Atacuri de tip DDoS:** Monitorizarea intensității traficului și identificarea creșterilor bruște de volum pot semnala un atac de tip distribuție.
* **Scanări de rețea:** Detectarea fluxurilor de trafic neobișnuite sau repetitive, care indică posibile scanări de porturi sau activități de recunoaștere.
* **Comportamente anormale:** Algoritmii de învățare automată, inclusiv rețelele neuronale și tehnicile de deep learning, pot analiza statistic datele de trafic pentru a distinge între traficul legitim și cel malițios.

În plus, integrarea datelor de trafic în sisteme de management al securității, cum ar fi SIEM (Security Information and Event Management), permite corelarea evenimentelor din rețea cu alte alerte de securitate, oferind o imagine holistică a stării de securitate a infrastructurii IT. Această abordare integrată contribuie la:

* **Reducerea timpului de răspuns:** Monitorizarea în timp real și analiza automată permit identificarea rapidă a incidentelor și declanșarea unor acțiuni de apărare imediate.
* **Detecția proactivă a amenințărilor:** Prin aplicarea unor modele predictive și a analizei comportamentale, sistemele moderne pot anticipa și preveni atacurile înainte ca acestea să aibă un impact semnificativ.
* **Optimizarea performanței rețelei:** Analiza traficului nu doar că asigură securitatea, dar contribuie și la identificarea blocajelor sau a problemelor de performanță, facilitând astfel optimizarea lățimii de bandă și a configurațiilor de rețea.

În mediul offline, analiza traficului de date poate fi utilizată în cadrul testărilor de penetrare și simulărilor de atac, oferind o perspectivă detaliată asupra modului în care rețeaua răspunde la condiții de stres sau la scenarii de atac simulate. Aceste analize ajută la identificarea vulnerabilităților ascunse și la întărirea politicilor de securitate.

În acest context, codul implementat pentru rezolvarea sarcinii de analiză a traficului de date este gândit pentru a oferi funcționalități asemănătoare cu instrumentul de analiză de trafic Wireshark, dar și pentru a face predicții (live) ale acestor pachete. Pentru a putea schimba arhitectura neuronală fără a modifica întreg codul și a efectua teste specifice, am încercat să structurez codul. Setul de date folosit a fost NSL-KDD, descris anterior. Modelele antrenate și logurile testelor efectuate ulterior sunt salvate în foldere specifice.

Funcția de încărcare și preprocesare a datelor pentru antrenament încarcă datasetul, aplică encodarea etichetelor și normalizează datele. Acest proces este esențial pentru pregătirea datelor într-un format adecvat pentru antrenarea modelului de rețea neuronală. Encodarea etichetelor transformă variabilele categorice în valori numerice, iar normalizarea asigură că toate caracteristicile au aceeași scală, îmbunătățind performanța modelului.

Dacă modelul există deja, acesta este încărcat; altfel, este creat și antrenat utilizând datele de antrenament preprocesate. Antrenarea modelului implică utilizarea unui set de date de antrenament pentru ajustarea parametrilor modelului, astfel încât acesta să poată face predicții acurate pe date noi. Procesul de antrenare include utilizarea unei funcții de pierdere și a unui optimizator pentru minimizarea erorilor de predicție.

Funcțiile de preprocesare live sunt utilizate pentru preprocesarea și predicția în timp real a pachetelor de rețea. Aceste funcții extrag caracteristicile relevante din pachete și le preprocesează într-un format adecvat pentru model. Predicțiile sunt apoi utilizate pentru a determina dacă un pachet reprezintă un atac sau trafic legitim.

Clasa principală pentru interfața grafică implementează o interfață utilizator pentru analiza traficului live. Aceasta include widget-uri pentru selectarea sursei datelor (trafic live/ date de testare), a interfeței de rețea ce se dorește a fi utilizată, a pragului minim de decizie (inclusiv afișare în interfață), butoane pentru start și stop, un Treeview pentru afișarea pachetelor și o zonă de text pentru detaliile pachetelor. Interfața permite utilizatorului să interacționeze cu aplicația, să pornească și să oprească analiza traficului și să vizualizeze rezultatele în timp real.

Funcția de procesare a cozilor de mesaje gestionează afișarea în interfață a pachetelor capturate și a predicțiilor asociate. Aceasta asigură că informațiile sunt actualizate în timp real, permițând utilizatorului să monitorizeze traficul de rețea și să detecteze orice activitate suspectă. De asemenea, pentru testarea funcționalității aplicației, folosesc o funcție separată care procesează și clasifică mostre aleatorii din setul de date de testare. Aceasta simulează captarea pachetelor în timp real și permite evaluarea performanței modelului într-un mediu controlat. Totodată, folosesc threaduri pentru putea opri procesarea pachetelor și pentru a schimba modul de procesare al pachetelor live (date reale)/random (date de test) fără a bloca aplicația.

Inițial am folosit o arhitectură care se bazează pe o rețea neuronală LSTM, inspirat din lucrarea "NTAM-LSTM models of network traffic prediction" [13]. Acest model utilizează un mecanism de atenție pe termen lung și scurt pentru a prezice traficul de rețea, capturând caracteristicile complexe ale acestuia.

Această arhitectură este una minimală. Sunt definite mai multe straturi diferite, incluzând un strat convoluțional (Conv1D), un strat de reducere dimensională (MaxPooling), un strat LSTM, un strat de atenție, un strat de reducere globală, un strat de dropout și un strat complet conectat pentru clasificare. Fiecare dintre aceste straturi joacă un rol specific în procesarea și interpretarea datelor. Stratul convoluțional și cel de reducere dimensională extrag caracteristicile importante din date, în timp ce stratul LSTM modelează dependențele temporale. Stratul de atenție se concentrează pe caracteristicile relevante, iar stratul de reducere globală și cel de dropout ajută la prevenirea supraînvățării (overfitting).

În ceea ce privește performanțele, conform graficului acurateții (*Figura 25*), pot spune că rețeaua s-a antrenat destul ținând cont de numărul de dimensiuni ale setului de date NSL-KDD. Acuratețea generală este de 99,90%. Asta ținând cont că am păstrat, inițial toate cele 23 de clase, iar subsetul a fost împărțit 50% pentru antrenare 50% pentru testare. Am setat 100 de epoci de antrenare și un early\_stopping cu parametrii: monitor='val\_loss' (ce metrică urmăresc), patience=10 (numărul de epoci tolerate fără îmbunătățire) și restore\_best\_weights=True (Când antrenarea se oprește, se încarcă greutățile modelului din epoca cu cel mai bun val\_loss).

Matricea de confuzie asociată (*Figura 26*), prezintă predicțiile pentru fiecare 23 de clase de malware ale setului de date NSL-KDD, care sunt destul de bune. Totodată, pentru pentru a înțelege mai bine, în partea dreptă, a aceleiași figuri, avem prezentat distribuția claselor din setul de date.

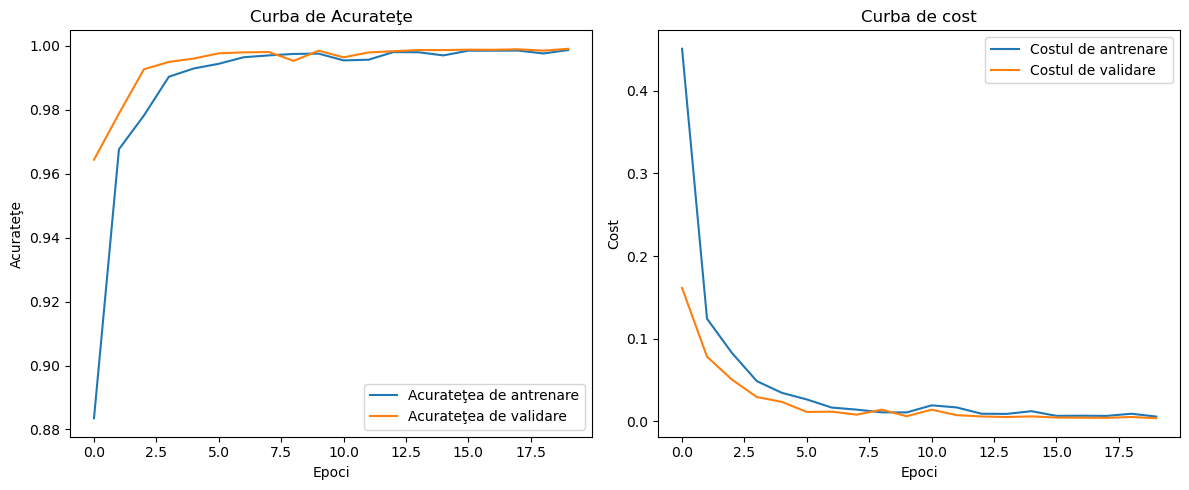


Figura 25 - Curba acurateții și curba de cost

|  |  |
| --- | --- |
|  |  |

Figura 26 - Matricea de confuzie (dreapta), distribuția claselor (stânga)

Pe datele de test, predicțiile sunt potrivite. Totuși, pe datele live (reale) arhitectura LSTM implementată cred că putea obține rezultate și mai bune. Aceasta filtrând mostrele de trafic live ca find din categoria Normal, cu un procent de încredere constant.

## Analiza malware și modelele LLM

Având în vedere popularitatea, multitudinea posibilităților de utilizare și performanțele modelelor LLM actuale, am decis să încerc să încorporez în aplicație și utilizarea unui astfel de model. Scopul este de a analiza fișiere și a obține o expertiză folosind inteligența artificială în forma ei cea mai avansată. Astfel, am instalat un manager de modele LLM, Ollama, care rulează local pe calculator. Acest lucru mi-a permis să descarc și să execut un model local, fără a trimite datele interogărilor către terți.

Folosind linia de comandă și framework-ul Ollama am descărcat modelul Mistral (dimensiune 4.1GB și 7,25 miliarde de parametri). Acesta poate fi instanțiat din linie de comandă folosind comanda “ollama run mistral”. Pentru a automatiza interogările am apelat modelul prin interogări specifice. Pe baza unui fișier încărcat, generez un prompt personalizat format din nume fișier, dimensiune, primii 256 de octeți ai acestuia și o întrebare specifică. Totodată după încărcarea unui fișier pot interoga chatbot-ul cu întrebări suplimentare. Istoricul conversației este salvat în fișiere corespunzătoare pentru a putea fi revăzut ulterior. Totodată conversația poate fi oprită în orice moment.

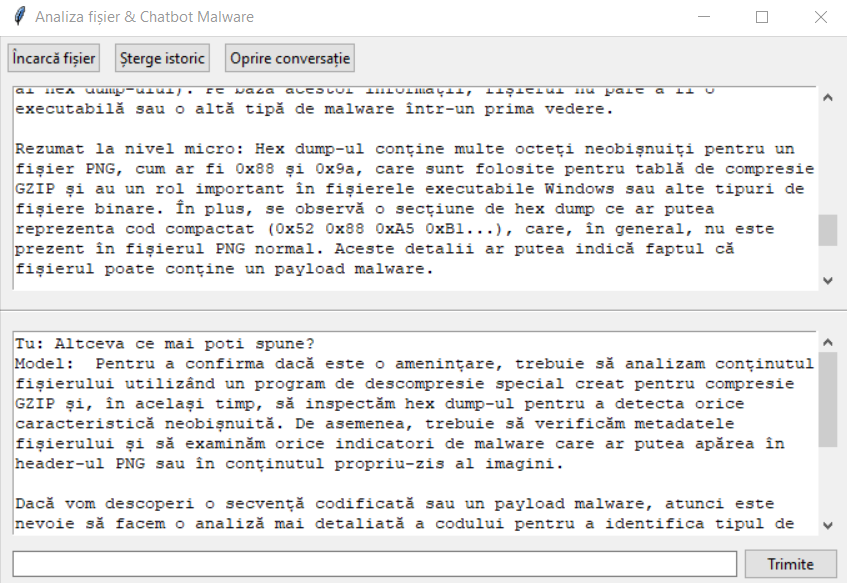


Figura 27 - Interfața chatbotul-ui bazat pe modelul Mistral

# Concluzii și direcții viitoare de cercetare

Acest proiect a reprezentat o provocare în ceea ce privește complexitatea și multitudinea de probleme care trebuie abordate în mediul cibernetic actual. Diferitele forme de malware pot fi transmise sub diferite forme și în mod distribuit pentru a compromite o întreagă rețea. Inteligența artificială trebuie să devină un atuu pentru o echipa albastră (blue team) responsabilă cu apărarea infrastructurilor informatice, oferindu-le un avantaj strategic în fața atacurilor tot mai sofisticate. Utilizarea modelelor de învățare automată, în special a rețelelor neuronale profunde, permite detectarea comportamentelor anormale și clasificarea precisă a tipurilor de atacuri cibernetice.

Prin lucrarea de faţă am încercat să prezint şi să rezolv, folosind algoritmi de AI, câteva din problemele de securitate actuale atât în ceea ce priveşte detecţia, cât şi analiza minimală malware. Acest proces a necesitat o înțelegere a tiparelor de comportament malițios, selecția seturilor de date cât mai relevante, precum și antrenarea și validarea unor modele de învățare automată capabile să distingă diferite tipuri de atacuri de traficul normal.

În general, evaluarea modelelor implementate a evidenţiat mai multe probleme comune tuturor configuraţiilor. Deşi, pe seturile de test, performanţele sunt foarte bune, pe datele reale, acurateţea diferă de la caz la caz. Trebuie ținut cont că arhitecturile AI comerciale, utilizate de marile companii, sunt antrenate pe seturi de date extrem de mari folosind resurse computaționale pe măsură, iar în finalizarea unui astfel de proiect colaborează o echipă de experți din domeniul IT.

Rezultatele obținute indică faptul că modelele de inteligență artificială, în special cele bazate pe rețele neuronale profunde, pot aduce îmbunătățiri semnificative în identificarea timpurie a amenințărilor cibernetice. Totuși, performanța acestor modele depinde în mare măsură de calitatea și diversitatea datelor de antrenament, precum și de adaptabilitatea lor la noile tipuri de atacuri care evoluează constant.

Pe viitor, cercetarea ar putea fi extinsă în mai multe direcții. Un prim pas ar fi îmbunătăţirea funcţionalităţilor aplicaţiei astfel încât să poţi realiza scanări planifcate ale sistemului de fişiere, ale traficului de date şi ale căsuţei de e-mail. Pe baza acestor scanări, urmate de o sanitizare, să reantrenez modelele în scopul creșterii performanțelor acestora folosind date reale ca input. Pe de altă parte, modelele pot fi îmbunătățite prin utilizarea unor seturi de date mai mari și testări amănunțite folosind arhitecturi noi cu parametrii potriviți.

Totodată, interfața aplicației poate fi îmbunătățită atât din perspectiva experienței utilizatorului (user experience - UX), cât și a aspectului vizual și funcțional (user interface - UI).

# Bibliografie

|  |  |
| --- | --- |
| [1] | A. T. F. Muhammad Shoaib, „Malware Analysis and Detection Using Machine Learning Algorithms,” nr. 11, 2022. |
| [2] | „Geek for geeks,” 04 07 2024. [Interactiv]. Available: https://www.geeksforgeeks.org/intrusion-detection-systems-ids-vs-intrusion-prevention-systems-ips/. [Accesat 19 03 2025]. |
| [3] | H. Debar, „An Introduction to Intrusion-Detection Systems,” IBM Research. |
| [4] | A. V. Phillip A. Porras, „Live traffic analysis of tcp/ip gateways.”. |
| [5] | P. Pedamkar, „EDUCBA,” 18 03 2023. [Interactiv]. Available: https://www.educba.com/types-of-intrusion-prevention-system/. [Accesat 19 03 2025]. |
| [6] | A. U. R. S. J. T. M. A. J. A. S. M. A. Ayesha Arshad, „A Systematic Literature Review on Phishing and Anti-Phishing Techniques,” vol. Pakistan J Engg & Tech , 2021. |
| [7] | S. R. d. Informații, *Ghid de bune practici pentru securitate cibernetică.* |
| [8] | [Interactiv]. Available: https://shahworld.wordpress.com/2015/09/24/software-development-life-cycle/. [Accesat 09 03 2025]. |
| [9] | [Interactiv]. Available: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset?resource=download. |
| [10] | S. K. G. J. B. S. M. L. Nataraj, „Malware Images: Visualization and Automatic Classification”. |
| [11] | Y. J. K. Z. J. W. H. S. Zachary Tauscher, „Learning to Detect: A Data-driven Approach for Network Intrusion Detection,” 2021. |
| [12] | A. Z. Karen Simonyan, „Very Deep Convolutional Networks for Large-Scale Image Recognition,” *ICLR 2015,* p. 14, 10 Aprilie 2015. |
| [13] | J. &. H. X. Zhao, „NTAM-LSTM models of network traffic prediction,” *MATEC Web of Conferences,* p. 10, 2022. |
| [14] | M. R. M. Talabis, R. McPherson, I. Miyamoto și J. L. Martin, Information Security Analytics Finding Security Insights, Patterns, and Anomalies in Big Data, Elsevier, 2015. |
| [15] | M. J. H. F. H. S. M. V. F. L. B. all, „Malware Detection and Prevention using Artificial,” în *2021 IEEE International Conference on Big Data*, Orlando, FL, USA, 2021. |

1. Antivirus este un tip de program software care ajută la protejarea sistemului informatic de viruși. Detectează virușii din sistemul informatic și îi distruge. Protejează sistemul informatic de malware specific. Este folosit pentru protecția împotriva unor amenințări tradiționale și simple care pot dăuna sistemului informatic. [↑](#footnote-ref-1)
2. Antimalware, este un program software care, spre deosebire de antivirus, protejează sistemele informatice de tot felul de programe malware, cum ar fi viruși, troieni, viermi etc. Acesta este folosit pentru a se proteja împotriva unor amenințări noi, sofisticate și mai periculoase care pot dăuna sistemului informatic. Este folosit mai ales în computerele organizaționale în scopuri de siguranță. [↑](#footnote-ref-2)
3. Clasificator probabilistic bazat pe teorema lui Bayes. Presupune independența caracteristicilor. Este eficient pentru probleme de clasificare textuală, cum ar fi filtrarea spamului. [↑](#footnote-ref-3)
4. Clasificator liniar care modelează probabilitatea unui rezultat binar folosind funcția logistică. Este utilizat pentru clasificare binară sau multiclase. [↑](#footnote-ref-4)
5. Algoritm de clasificare care identifică hiperplanul optim pentru separarea claselor într-un spațiu multidimensional. [↑](#footnote-ref-5)
6. Clasificator bazat pe distanțele dintre puncte în spațiul caracteristicilor, fiind utilizat pentru clasificare și regresie. [↑](#footnote-ref-6)
7. Construieste un model sub forma unui arbore de decizii, împărțind datele pe baza valorilor caracteristicilor. [↑](#footnote-ref-7)
8. Metodă de ansamblu care creează multiple arbori de decizie și agregă predicțiile pentru a spori acuratețea. [↑](#footnote-ref-8)
9. Combină predicțiile mai multor modele antrenate pe subseturi diferite ale datelor, reducând astfel variația. [↑](#footnote-ref-9)
10. Similar cu Random Forest, dar introduce o mai mare aleatorietate în construcția arborilor pentru a mări diversitatea. [↑](#footnote-ref-10)
11. Tehnică de ansamblu ce construiește modele secvențial, fiecare corectând erorile precedentului, pentru a îmbunătăți performanța generală. [↑](#footnote-ref-11)
12. Clasificator bazat pe boosting care folosește clasificatori slabi (de obicei arbori de decizie) și le atribuie ponderi. [↑](#footnote-ref-12)
13. Algoritm de gradient boosting optimizat, recunoscut pentru viteză și performanță ridicată. [↑](#footnote-ref-13)
14. Clasificator bazat pe boosting, optimizat pentru date categorice și pentru detectarea relațiilor între caracteristici. [↑](#footnote-ref-14)
15. VirusTotal este un serviciu online gratuit (deținut de Google) care analizează fișiere și URL-uri pentru a detecta viruși, malware, troieni, ransomware, spyware și alte tipuri de amenințări cibernetice. [↑](#footnote-ref-15)
16. MetaDefender este o platformă de securitate dezvoltată de compania OPSWAT, folosită pentru analizarea fișierelor, linkurilor, dispozitivelor sau aplicațiilor, în scopul detectării malware-ului și a altor amenințări. Este similar cu VirusTotal, dar oferă și funcționalități suplimentare pentru companii. [↑](#footnote-ref-16)
17. Arhitectura VGG16 a fost introdusă pentru prima dată în 2015, în cadrul lucrării de cercetare "Very Deep Convolutional Networks for Large-Scale Image Recognition", scrisă de Karen Simonyan și Andrew Zisserman de la Universitatea din Oxford. [↑](#footnote-ref-17)
18. Set de date folosit în cadrul competiției ImageNet Large Scale Visual Recognition Challenge (ILSVRC) în anul 2014. Conținea peste 1.2 milioane de imagini pentru anrenare și 1000 de clase (ex: animale, vehicule, obiecte etc.). [↑](#footnote-ref-18)