Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Facultatea de Automatică şi Calculatoare
Domeniul: Calculatoare şi Tehnologia Informaţiei
Specializarea: Securitatea Spaţiului Cibernetic

Analiza automată a emailurilor pentru identificarea phishing-ului prin intermediul Machine Learning

LUCRARE DE DISERTAȚIE

Absolvent:
ing. **Bogdan Iacob**

Coordonator:
Ș.l. dr. ing. **Cristian-Mihai Amarandei**

Iași, 2023

Cuprins

Rezumat

Această lucrare propune dezvoltarea a două modele de învățare automată (AI) pentru analiza automată a emailurilor și a URL-urilor, cu scopul de a identifica și a preveni atacurile de phishing. Prin integrarea acestor modele AI într-o extensie pentru clientul de email, se urmărește furnizarea unui avertisment în timp real utilizatorilor atunci când sunt detectate indicii de phishing.

Pentru analiza automată a conținutului e-mailurilor, se utilizează un model AI antrenat pe un set de date extins, care cuprinde exemple de emailuri malițioase și legitime. Modelul utilizează tehnici de învățare automată, cum ar fi învățarea supervizată, pentru a identifica indicii specifice ale phishing-ului, precum: link-uri suspecte, cereri de informații personale sensibile sau mesaje de amenințare. Prin analizarea conținutului e-mailurilor, modelul poate determina probabilitatea de a fi un atac de phishing și poate genera avertismente pentru utilizatorii aflați în pericol.

Pentru analiza automată a URL-urilor, se dezvoltă un al doilea model AI care se concentrează pe identificarea URL-urilor malițioase ce ar putea conduce spre pagini de phishing. Acest model utilizează tehnici de învățare automată pentru a extrage caracteristici relevante ale URL-urilor și pentru a determina probabilitatea de a fi malițioase. Modelul este antrenat pe un set de date extins, care conține exemple de URL-uri legitime și malițioase. Atunci când utilizatorii deschid un email extensia pentru clientul de email analizează URLurile-ul în timp real, iar modelul AI furnizează o evaluare a riscului și, în cazul unui posibil atac de phishing, se afișează un avertisment în interfața clientului de email.

Prin dezvoltarea acestor două modele AI integrate într-o extensie pentru clientul de email, se oferă utilizatorilor o protecție suplimentară împotriva atacurilor de phishing. Avertismentele în timp real permit utilizatorilor să fie conștienți de potențialele amenințări și să evite accesarea paginilor malițioase sau furnizarea de informații personale sensibile.

Această lucrare contribuie la dezvoltarea de soluții eficiente și automate pentru detectarea phishing-ului și protecția utilizatorilor împotriva atacurilor cibernetice. Prin integrarea modelelor AI într-o extensie pentru clientul de email, se asigură o protecție activă și o experiență mai sigură în utilizarea serviciilor online.

Introducere

În acest capitol se va face o descriere introductivă a aplicației, prin prezentarea contextului acesteia, motivând conceptul care a stat la baza studierii și dezvoltării acestui proiect. De asemenea, va fi definită și structura acestei lucrări, care, conține documentarea bazelor teoretice ale temei alese, pașii parcurși și rezultatele obținute după implementarea soluției dezvoltate.

Contextul proiectului

Atacurile de phishing reprezintă una dintre cele mai comune forme de atac de inginerie socială, care vizează emailurile utilizatorilor pentru a fura informații confidențiale și sensibile. Aceste atacuri pot fi utilizate ca parte a unor atacuri mai ample, lansate pentru a obține un punct de acces într-o instituție. În ultimul deceniu, s-au propus numeroase tehnici anti-phishing pentru detectarea și limitarea acestor atacuri. Cu toate acestea, aceste tehnici sunt încă ineficiente și inexacte.[1] Prin urmare, există o mare nevoie de tehnici eficiente și precise de detectare pentru a face față acestor atacuri.

Motivația alegerii temei

Motivația realizării acestui proiect este reprezentată de dorința de a oferi o soluție parțială a acestei probleme, detectarea phishing-ului este extrem de relevantă și importantă în contextul actual al creșterii amenințărilor cibernetice și al vulnerabilității utilizatorilor online.

Utilizarea machine learning și a tehnologiilor de inteligență artificială în detectarea phishing-ului oferă numeroase avantaje. Aceste metode pot analiza modelele și caracteristicile specifice ale mesajelor de phishing, identificând indicii și semnături ale atacurilor.

Prin antrenarea unui model pe seturi mari de date de phishing, algoritmul poate deveni din ce în ce mai precis în identificarea amenințărilor și poate furniza alerte în timp real pentru utilizatori.[2]

Implementarea unei soluții bazate pe machine learning pentru detectarea phishing-ului poate contribui la protejarea utilizatorilor și la reducerea riscului de a deveni victime ale atacurilor cibernetice.

Structura lucrării

În continuare, va fi introdus conținutul lucrării și structura sa ce se bazează pe o listă de obiective stabilite în vederea documentării și rezolvarea problemei asociate acestui tip de atac, după cum urmează:

- În capitolul de *Introducere*, se prezintă contextul și motivația care stau la baza elaborării acestui studiu.
- Capitolul I, intitulat *Fundamentarea teoretică* și documentarea bibliografică pentru tema propusă, prezintă tehnologiile utilizate în implementarea soluției și face referire la alte aplicații similare.

- Capitolul II, intitulat *Proiectarea aplicației*, descrie cerințele aplicației și componentele principale ale acesteia, prezentând în același timp justificarea utilizării anumitor tehnologii.

- În capitolul III, *Implementarea aplicației*, se prezintă implementarea efectivă a aplicației, evidențiind dificultățile întâmpinate în procesul de dezvoltare.

- Capitolul IV, *Testarea aplicației* și rezultate experimentale, prezintă metodologia de testare a aplicației și ilustrează parțial rezultatele obținute în cadrul experimentelor.

- Capitolul final, *Concluzii*, analizează lucrarea în ansamblu, evidențiind principalele obiective atinse și oferind posibile direcții de dezvoltare viitoare.

I. Fundamentarea teoretică şi documentarea bibliografică pentru tema propusă

În acest capitol se vor prezenta concepte de bază din domeniul temei alese, se vor acoperi elementele esențiale ale infrastructurii de bază din spatele tehnicilor de machine learning. Scurta descriere este furnizată pentru a introduce termenii folosiți pe parcursul lucrării.

1.1. Phishing-ul

Phishing-ul reprezintă o formă de atac cibernetic în care atacatorii încearcă să obțină informații confidențiale, cum ar fi parole, numere de card de credit sau date personale, prin intermediul manipulării sociale și a înșelăciunii. Atacatorii pretind adesea că sunt o entitate de încredere, cum ar fi o instituție bancară, un furnizor de servicii sau o organizație cunoscută, în încercarea de a câștiga încrederea și cooperarea utilizatorilor.[2]

Scopul principal al phishing-ului este de a exploata utilizatorii prin inducerea în eroare și obținerea accesului la informații sensibile. Phishing-ul reprezintă o amenințare serioasă pentru utilizatorii de internet, având consecințe financiare și de securitate semnificative. Utilizatorii pot deveni victime ale fraudelor bancare, furtului de identitate sau atacurilor cibernetice, dacă cad în capcana phishing-ului.[1][2]

Pentru a se proteja împotriva atacurilor de phishing, utilizatorii trebuie să fie conștienți de semnele de avertizare și să adopte măsuri preventive. Acestea pot include verificarea atentă a URL-urilor și a expeditorilor de email, evitarea furnizării informațiilor personale sensibile prin intermediul emailurilor sau mesajelor și utilizarea unor soluții de securitate cibernetică actualizate.

1.2. Mecanisme și tehnicile utilizate în Phishing

Phishing-ul utilizează o varietate de mecanisme și tehnici pentru a obține informații personale și de autentificare de la utilizatori. Acestea sunt concepute pentru a manipula utilizatorii și a-i convinge să dezvăluie informații sensibile sau să acționeze în mod nedorit. Iată câteva dintre principalele mecanisme și tehnici utilizate în phishing:

- E-mailuri de phishing: Atacatorii trimit e-mailuri false care imită mesajele legale și de încredere, cum ar fi cele de la bănci, instituții sau companii cunoscute. Aceste e-mailuri pot solicita utilizatorilor să furnizeze informații personale, cum ar fi numele de utilizator, parola sau numerele de card de credit. De asemenea, pot conține link-uri către pagini web malițioase sau atașamente infectate cu malware[3].

- Site-uri web de phishing: Atacatorii creează site-uri web falsificate care imită aspectul și funcționalitatea site-urilor legitime. Aceste site-uri sunt proiectate pentru a apărea cât mai autentice și convincente, în scopul de a induce utilizatorii în eroare. Utilizatorii sunt adesea direcționați către aceste site-uri prin intermediul link-urilor din e-mailuri sau prin intermediul căutărilor pe internet. Scopul este de a-i determina pe utilizatori să introducă informații personale sau de autentificare, care sunt apoi colectate de către atacatori[3].

- Inginerie socială: Ingineria socială reprezintă o tehnică de manipulare psihologică prin care atacatorii încearcă să câștige încrederea și cooperarea utilizatorilor. Aceștia pot utiliza pretexte credibile, cum ar fi actualizări ale contului sau avertismente privind securitatea, pentru a convinge utilizatorii să dezvăluie informații sensibile sau să efectueze acțiuni nedorite. De exemplu, atacatorii pot pretinde că sunt reprezentanți ai unei instituții bancare și pot solicita utilizatorilor să-și actualizeze informațiile de autentificare[3].

- Phishing prin telefon (vishing) și mesaje text (smishing): În afară de e-mailuri, atacatorii pot utiliza și alte canale de comunicare, cum ar fi telefonul și mesajele text, pentru a efectua atacuri de phishing. Acestea pot implica apeluri telefonice în care atacatorii pretind că sunt reprezentanți ai unor organizații cunoscute și cer utilizatorilor să furnizeze informații confidențiale. Mesajele text pot conține link-uri sau instrucțiuni care îi îndeamnă pe utilizatori să acceseze pagini web malițioase sau să furnizeze informații personale[3].

Pe lângă metodele de phishing obișnuite, există și tehnici avansate de phishing, cum ar fi spear phishing și whaling, care vizează ținte specifice și utilizează informații personalizate pentru a părea mai credibile și mai autentice cât și mecanisme sofisticate precum DNS Poisoning și Pharming-ul utilizate de atacatori pentru a amplifica eficacitatea atacurilor și pentru a evita detectarea.

- Spear phishing reprezintă o formă avansată de phishing în care atacatorii se concentrează asupra unor ținte specifice, cum ar fi angajații unei companii sau utilizatori cu un nivel ridicat de acces la informații sensibile. În cadrul acestui atac, atacatorii colectează informații detaliate despre țintele lor, cum ar fi nume, titluri de job, conexiuni sociale și alte informații personale relevante. Aceste informații sunt utilizate pentru a personaliza mesajele de phishing și pentru a le face să pară mai credibile și mai autentice. De exemplu, atacatorii pot trimite un e-mail personalizat către un angajat al unei companii, pretinzând că sunt colegi sau superiori. E-mailul poate conține informații interne specifice, precum proiecte în desfășurare, termene limită sau detalii despre activități recente ale companiei. Scopul este de a induce în eroare ținta și de a-i

determina să dezvăluie informații confidențiale sau să efectueze acțiuni nedorite, cum ar fi transferul de fonduri sau accesarea unor sisteme sensibile.

- Whaling reprezintă o formă de phishing care se concentrează asupra unor ținte de mare importanță într-o organizație, cum ar fi directori executivi, manageri de nivel înalt sau alte persoane cu putere de decizie. Atacatorii utilizează informații personalizate și cunoștințe despre industrie și organizație pentru a crea mesaje de phishing credibile și convingătoare.În cazul whaling-ului, atacatorii pot trimite e-mailuri false către un director executiv, pretinzând că sunt de la o bancă sau o instituție financiară și solicitând informații confidențiale, cum ar fi numerele de card de credit sau parolele conturilor. Deoarece aceste ținte de înaltă importanță au adesea acces la informații sensibile și putere de decizie, succesul unui atac de whaling poate avea consecințe grave, inclusiv pierderi financiare sau acces neautorizat la sistemele organizației.[4]

- DNS Poisoning este un alt atac eficient utilizat de către atacatori, DNS (Domain Name System) reprezintă un protocol esențial în infrastructura internetului, având rolul de a traduce numele de domenii (ex: www.exemplu.com) în adrese IP (ex: 192.168.0.1). Atacul de DNS poisoning constă în modificarea în mod intenționat a bazei de date DNS, astfel încât să furnizeze informații greșite sau să direcționeze utilizatorii către site-uri web falsificate. Această tehnică permite atacatorilor să redirecționeze traficul către servere controlate de ei, în scopul de a colecta informații confidențiale.[3]

- Pharming-ul este o tehnică de atac care exploatează vulnerabilități în infrastructura DNS. Atacatorii preiau controlul asupra serverelor DNS sau manipulează adresele IP asociate cu un anumit domeniu, astfel încât utilizatorii să fie redirecționați către site-uri web malițioase, în mod neobservat. Utilizatorii sunt direcționați către aceste site-uri prin introducerea greșită a URL-urilor sau prin redirecționare automată.[5]

Aceste tehnici avansate cum ar fi DNS poisoning și pharming, pot face ca atacurile să fie mai dificil de detectat și de prevenit, deoarece nu depind de acțiunile utilizatorilor sau de manipularea lor directă. Detectarea și contracararea acestor metode implică abordări de securitate avansate și soluții inteligente de analiză și protecție.

Este important să înțelegem aceste mecanisme și tehnici utilizate în phishing, deoarece aceasta ne permite să dezvoltăm metode de detecție și prevenire eficiente. În următoarele capitole ale lucrării, vom explora modul în care tehnologiile de machine learning și analiza automată pot fi utilizate pentru a identifica și contracara acest tip de atac.

1.3. Machine Learning

Capacitatea de a învăța este o trăsătură centrală a inteligenței și reprezintă o preocupare deosebită atât în psihologia cognitivă, cât și în inteligența artificială. Domeniul învățării automate, care se intersectează cu aceste discipline, se ocupă de studiul procesului computațional care caracterizează atât învățarea umană, cât și învățarea automată.

Există două aspecte principale pentru care învățarea automată reprezintă o parte integrală a acestor domenii extinse, cu toate că au identități separate. În primul rând, problemele legate de reprezentarea cunoștințelor, organizarea memoriei și performanța nu pot fi ignorate de cercetători, deoarece acestea sunt preocupări majore în ambele domenii.În al doilea rând, învățarea are loc în orice domeniu care implică inteligență, indiferent dacă sarcinile de bază se referă la diagnosticare, planificare, limbaj natural sau altceva. Prin urmare, învățarea automată poate fi considerată mai mult decât un subdomeniu al inteligenței artificiale și al științelor cognitive, fiind o paradigmă pentru cercetare și dezvoltare.

Interesul pentru abordarea computațională a inteligenței a apărut în momentul începuturilor inteligenței artificiale, în anii '50. Diversitatea problemelor abordate include rezolvarea jocurilor, recunoașterea literelor, concepte abstracte și memoria verbală. Învățarea era considerată o trăsătură definitorie a sistemelor inteligente, iar îmbunătățirea acesteia era strâns legată de dezvoltarea mecanismelor generale pentru cunoaștere, percepție și acțiune.

Aceste definiții prezintă mai multe limitări deoarece este dificil să se testeze în mod absolut dacă învățarea a fost cu adevărat dobândită. Conform dicţionarului, învăţarea poate fi definită ca procesul de dobândire a cunoştinţelor prin studiu, experienţă sau educare; dobândirea conştiinţei prin informaţie sau observaţie; angajamentul memoriei. Aceste definiții prezintă multiple limitări, deoarece nu este posibil să se testeze în mod absolut dacă învățarea a fost complet dobândită.

Totuşi, o definiţie practică, cu ajutorul căreia să păstrăm subiectul sub studiu, este următoarea:
“Învăţarea este îmbunătăţirea performanţei într-un mediu prin achiziţionarea cunoştinţelor rezultate din experienţa obţinută în acel domeniu”. Astfel, putem spune că obiectele învață atunci când sunt capabile să-și modifice comportamentul într-un mod care îi permite să obțină o performanță mai bună în viitor.

Deși învățarea automată se concentrează asupra procesului de învățare, literatura de specialitate evidențiază patru scopuri principale, fiecare cu propriile metodologii, abordări de evaluare și exemple de succes.

Primul scop implică modelarea mecanismelor care stau la baza învățării umane. În acest context psihologic, cercetătorii dezvoltă algoritmi de învățare care sunt în concordanță cu cunoștințele despre arhitectura cognitivă umană și care sunt proiectați să explice anumite comportamente observate în procesul de învățare. Această abordare conduce la o varietate largă de modele computaționale, unele dintre ele oferind explicații calitative ale comportamentului, în timp ce altele se concentrează pe ajustarea ratei de eroare și a timpului de răspuns al subiecților umani. O problemă tipică de învățare poate fi definită utilizând diagrama următoare (Fig. 1.1) :

Unul dintre punctele centrale ale acestor abordări este focalizarea asupra dezvoltării, înțelegerii și evaluării algoritmilor de învățare. Dacă considerăm învățarea automată ca fiind o știință, atunci cu siguranță aceasta reprezintă o știință a algoritmilor.[7]

Unul dintre avantajele învățării automate constă în capacitatea sa de a căuta într-un spațiu extins de ipoteze posibile, în vederea determinării celei mai bune ipoteze care se potrivește cu datele observate, pe baza cunoștințelor deținute a priori.

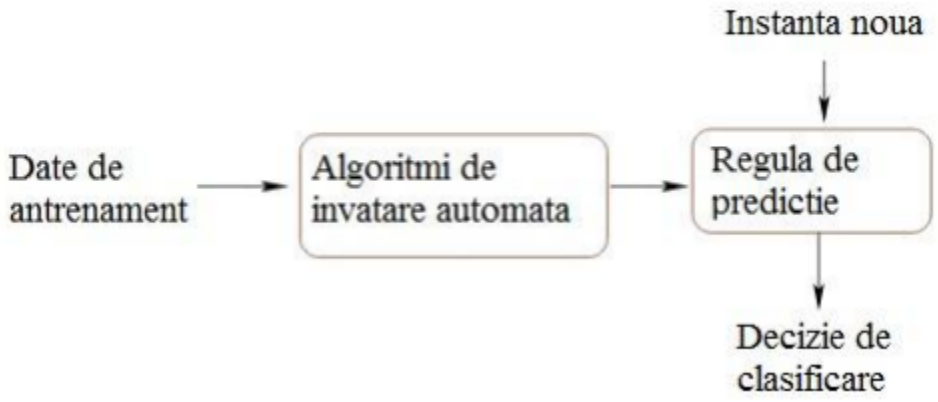


Figura 1.1. Diagrama unei probleme de învăţare, preluat din [6]

Al doilea scop al învățării automate constă în dezvoltarea de algoritmi și modele care pot face predicții precise și generalizate pe baza datelor de antrenare. Acest aspect este deosebit de important în domenii precum recunoașterea de obiecte, clasificarea textului sau diagnosticarea medicală. Prin utilizarea unor tehnici avansate de învățare automată, cum ar fi rețelele neuronale profunde, se poate obține o acuratețe ridicată în predicțiile făcute de modelele de învățare.

Al treilea scop al învățării automate constă în identificarea și extragerea automată a caracteristicilor relevante din datele de intrare. Acest lucru este esențial pentru a dezvolta modele eficiente și performante. Prin aplicarea unor algoritmi de învățare automată, precum selecția de

caracteristici și reducerea dimensionalității, se poate obține o reprezentare compactă a datelor, eliminând caracteristicile redundante sau irelevante.

Ultimul scop al învățării automate constă în adaptarea și învățarea în timp real. Acest aspect este crucial în domenii precum robotica sau sistemele de recomandare, unde modelele trebuie să se adapteze și să învețe din datele noi sau din interacțiunea cu utilizatorii. Prin aplicarea algoritmilor de învățare online și a tehnicilor de învățare prin întărire, modelele pot fi actualizate în timp real pentru a reflecta schimbările și noile informații.

1.4. Taxonomia învățării automate

În funcție de obiectivul dorit al algoritmului sau de disponibilitatea datelor de intrare, putem utiliza următoarea taxonomie pentru tehnicile de învățare automată:

În cadrul învățării supervizate, algoritmul generează o funcție care realizează o mapare între datele de intrare și rezultatul dorit. Sarcinile de învățare supervizată se formulează în mod obișnuit ca probleme de clasificare: algoritmul trebuie să învețe sau să aproximeze comportamentul unei funcții care atribuie o instanță în una dintre clasele disponibile, pe baza unor exemple de intrare-ieșire ale funcției.

De exemplu, să presupunem că avem un set de imagini și dorim să dezvoltăm un algoritm care să poată clasifica automat imaginile în câteva categorii predefinite, cum ar fi "pisici" și "câini". În acest caz, algoritmul de învățare supervizată ar fi antrenat cu un set de imagini etichetate, adică imagini pentru care știm deja că aparțin categoriei "pisici" sau "câini". Algoritmul învață să generalizeze aceste exemple și să clasifice corect și imagini noi care nu au fost văzute anterior.

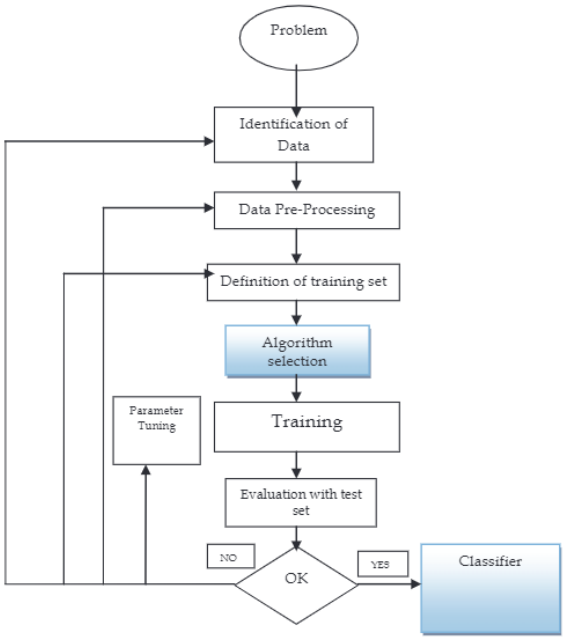


Figura 1.2: Procesul învățării automate supervizare, preluat din [8]

În contrast cu învățarea supervizată, învățarea nesupervizată nu necesită date de intrare etichetate. Scopul este să se descopere structuri sau modele ascunse în datele de intrare, fără a avea informații prealabile despre clase sau categorii. Algoritmii de învățare nesupervizată încearcă să găsească tipare sau grupuri în datele de intrare, să identifice anomalii sau să reducă dimensionalitatea datelor.

De exemplu, utilizarea învățării nesupervizate în analiza clusterizării poate fi utilă pentru gruparea documentelor similare într-o colecție de texte sau pentru identificarea grupurilor de utilizatori cu comportamente similare într-o rețea socială.

Învățarea semi-supervizată este o combinație a celor două tipuri de învățare menționate mai sus. Aici, algoritmul are acces la un set parțial de date etichetate și la un set mai mare de date neetichetate. Scopul este să se exploateze atât informațiile etichetate, cât și cele neetichetate pentru a obține o performanță mai bună în învățare.

Învățarea prin întărire se bazează pe interacțiunea unui agent cu un mediu și învățarea din experiență prin obținerea de recompense sau pedepse pentru acțiunile sale. Agentul ia decizii în funcție de starea mediului și primește feedback în urma acțiunilor sale. Scopul este de a învăța politici de acțiune care maximizează recompensele pe termen lung.

Învățarea prin învățare este un domeniu în care algoritmii de învățare sunt folosiți pentru a îmbunătăți sau a automatiza procesul de învățare însuși. Aceasta poate include îmbunătățirea performanței algoritmilor existenți, adaptarea parametrilor de învățare sau dezvoltarea de noi metode și tehnici de învățare.[8]

Această taxonomie oferă o structură pentru înțelegerea diferitelor paradigme de învățare automate și permite cercetătorilor și practicienilor să identifice și să aplice tehnicile potrivite în funcție de problemele și seturile de date cu care lucrează.

Algoritmul k-Nearest Neighbors (KNN) este o tehnică utilizată în domeniul învățării automate și al recunoașterii de tipar pentru clasificarea și regresia datelor. Este un algoritm de învățare supervizată care se bazează pe ideea că obiectele similare sunt adesea vecini într-un spațiu dimensional.

Funcționarea algoritmului KNN presupune identificarea k vecini cei mai apropiați ai unui obiect de test într-un set de date de antrenare. Vecinii sunt selectați pe baza unei măsuri de similaritate sau de distanță, cum ar fi distanța euclidiană sau distanța Manhattan . Odată ce vecinii au fost identificați, algoritmul poate realiza o predicție pentru obiectul de test pe baza etichetelor vecinilor. În cazul clasificării, obiectul de test va fi atribuit clasei majoritare a vecinilor săi, în timp ce în cazul regresiei, se poate utiliza o medie ponderată a valorilor vecinilor pentru a obține o valoare estimată.[10]

Pentru alegerea valorii lui k, numită și numărul de vecini, se poate aplica o metodă de optimizare sau se poate utiliza o valoare fixă. Dacă k este prea mic, algoritmul poate fi sensibil la zgomot și la variațiile locale ale datelor. În schimb, dacă k este prea mare, algoritmul poate subestima variațiile locale și poate afecta performanța generală a predicțiilor.

Algoritmul KNN poate fi utilizat într-o varietate de domenii, cum ar fi recunoașterea de imagini, filtrarea colaborativă, analiza de sentiment și clasificarea textelor. Este un algoritm simplu și ușor de implementat, dar poate fi costisitor în ceea ce privește timpul de calcul în cazul unui set de date mare.[9]

Este important de menționat că KNN este un algoritm de învățare leneș (lazy learning), ceea ce înseamnă că în faza de antrenare, nu se realizează nicio prelucrare asupra datelor. Toată informația este stocată, iar procesarea are loc în faza de testare.

Weighted Nearest Neighbor (WNN) reprezintă un rafinament al algoritmului K-Nearest Neighbors (KNN) prin care se introduce ponderarea contribuției fiecărui vecin în funcție de

distanța față de noua instanță ce trebuie clasificată. Acest lucru înseamnă că vecinii mai apropiați vor avea o pondere mai mare în procesul de clasificare.

Metoda de ponderare a voturilor a fost introdusă pentru prima dată de Dudani și este denumită regula distanței ponderate a celor mai apropiați k-vecini.

Pentru a calcula pondera wi' corespunzătoare celui de-al i-lea vecin al noii instanţe x', se aplică formula [11] următoare :

$$w_i' = \begin{cases} \frac{d(x', x_k^{NN}) - d(x', x_i^{NN})}{d(x', x_k^{NN}) - d(x', x_1^{NN})} & , \text{daca } d(x', x_k^{NN}) \neq d(x', x_1^{NN}), \\ 1 & , \text{daca } d(x', x_k^{NN}) = d(x', x_1^{NN}). \end{cases}$$

Ecuatia 1.1. preluat din [11]

Pentru a trata situația în care valorile atributelor unui vecin sunt identice cu cele ale noii instanțe de catalogat, iar numitorul ponderii devine zero, se clasează noua instanță în aceeași clasă cu prima instanță și nu se mai realizează alte calcule. În cazul în care există mai multe astfel de instanțe, se alege clasa majoritară.
Majoritatea voturilor ponderate vor determina rezultatul clasificarii:

$$y' = \arg\max_y \sum_{(x_i^{NN}, y_i^{NN}) \in T'} w_i' \times \delta(y = y_i^{NN}).$$

Ecuatia 1.2. preluat din [11]

Cu toate că algoritmul KNN ia în considerare doar primii k vecini pentru clasificare, introducerea ponderării distanțelor reduce riscul de obținere a unui rezultat greșit prin faptul că vecinii mai îndepărtați primesc o pondere mai mică.

Weighted Nearest Neighbor (WNN) îmbunătățește astfel precizia clasificării în comparație cu KNN clasic, prin luarea în considerare a distanțelor ponderate. Acest rafinament contribuie la reducerea erorilor și creșterea performanței algoritmului în situații în care toate instanțele sunt luate în considerare, oferind o atenție mai mare vecinilor mai apropiați.

1.5. Referințe la teme/subiecte conceptual similare

Detectarea phishing-ului beneficiază de o atenție sporită în cercetarea din ultimii ani, atât în mediul academic, cât și în industria tehnologică. Pe baza abordării metodologice, se disting două categorii de metode: metode bazate pe reguli și metode bazate pe învățare automată (ML).

Una dintre cele mai comune metode utilizate în detectarea tentativelor de phishing este abordarea bazată pe un set de reguli create de experții în securitate. Un sistem bazat pe un set de mai multe caracteristici care funcționează la nivelul URL-urilor este prezentat în [26], împreună cu un component care utilizează o potrivire aproximativă a URL-urilor cu un set de înregistrări dintr-o listă neagră existentă.

Un exemplu de implementare strâns legat de tema aleasă este articolul ”An Experimental Study of Machine Learning for Phishing Detection” [12] autorii acestei lucrări descriind dezvoltarea unei soluții bazate pe 3 modele de analiza separate pentru URL, Continutul HTML al emailului și ScreenShot-ul paginii la care acesta face referință.

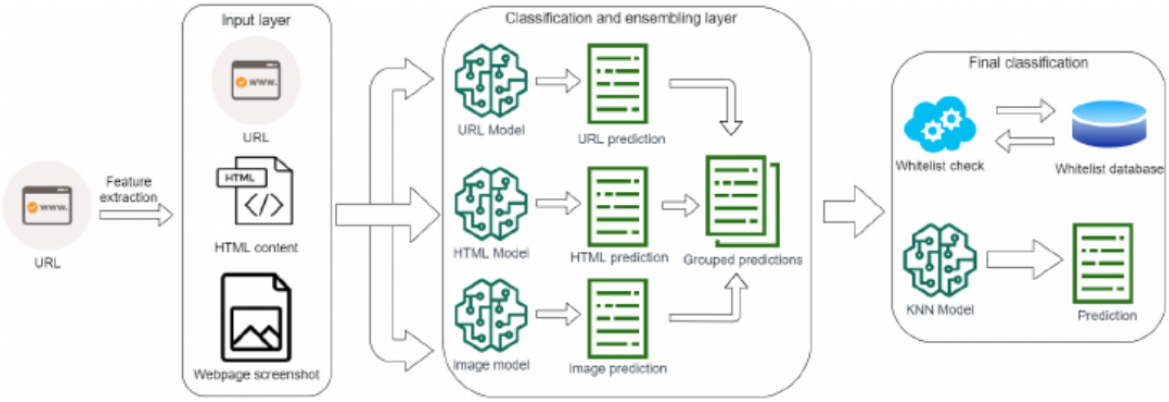


Figura 1.3. Arhitectura ansamblului modelului, preluat din [12]

Spre deosebire de autorii acestei lucrări, soluția descrisă de mine prezintă implementare în timp real prin afișarea mesajelor de avertisment după fiecare deschidere a unui email.

II. Proiectarea aplicației

În cadrul acestui capitol, se vor descrie cerințele aplicației pe baza cărora se vor proiecta componentele principale ale acesteia. De asemenea, se vor prezenta și motiva tehnologiile folosite.

2.1. Cerințele aplicației

Scopul aplicației constă în dezvoltarea unei soluții care să protejeze utilizatorii împotriva riscului de phishing prin intermediul unei extensii pentru browser care să faciliteze clasificarea email-urilor primite pe baza unor predicții. În acest sens, aplicația trebuie să îndeplinească următoarele cerințe:

- Model de inteligență artificială: Aplicația trebuie să includă dezvoltarea unui model de inteligență artificială robust, capabil să prezică cu precizie dacă un mesaj este de tip phishing sau nu. Acest model va fi antrenat pe un set de date relevant și va fi actualizat periodic pentru a menține performanța în fața noilor tehnici de phishing.

- Extensie pentru browser: Aplicația trebuie să ofere o extensie pentru browser (de exemplu, Chrome, Firefox) care să permită utilizatorului să primească în timp real rezultatele clasificării email-urilor. Aceasta ar trebui să funcționeze ca o unealtă de filtrare și să ofere notificări vizuale pentru a avertiza utilizatorul cu privire la posibilele mesaje de phishing.

Analizand aceste cerinte, am decis sa proiectez aplicația utilizand un modul frontend și un modul de backend folosind tehnologii precum:

HyperText Markup Language (HTML) este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser (sau navigator). Scopul HTML este mai degrabă prezentarea informațiilor – paragrafe, fonturi, tabel, descrierea semanticii documentului.[13]

CSS sau Cascading Style Sheets este un standard pentru formatarea elementelor unui document HTML. CSS se poate utiliza și pentru formatarea elementelor XHTML, XML și SVG. Acesta permite separarea și prezentarea vizuală a conținutului unei pagini web, inclusiv culorile și fonturile disponibile. Separarea elementelor unei pagini îmbunătățește accesibilitatea paginii și permite o mai bună flexibilitate și un control în specificațiile caracteristicilor de prezentare.[14]

JavaScript (JS) este un limbaj de programare orientat obiect bazat pe conceptul prototipurilor. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul JavaScript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea site-urilor web, dar este folosit și pentru accesul la obiecte încapsulate (embedded objects) în alte aplicații, folosit adesea pe server, utilizând un mediu de rulare, cum ar fi Node.js.

Cea mai des întâlnită utilizare a JavaScript este în scriptarea paginilor web. Programatorii web pot îngloba în paginile HTML script-uri pentru diverse activități cum ar fi verificarea datelor introduse de utilizatori sau crearea de meniuri și alte efecte animate.

Browserele reține în memorie o reprezentare a unei pagini web sub forma unui arbore de obiecte și pun la dispoziție aceste obiecte script-urilor JavaScript, care le pot citi și manipula. Arborele de obiecte poartă numele de Document Object Model sau DOM.[15]

Node.js este o platformă de dezvoltare open source pentru rularea javascript pe partea de server. Nodul este util pentru dezvoltarea de aplicații care necesită o conexiune permanentă de la browser către server. Node.js este proiectat să funcționeze pe un server dedicat HTTP și să utilizeze un fir cu un proces pe o singură unitate de timp, este bazată pe eveniment și rulează asincron. Codul construit pe această platformă nu corespunde modelului tradițional de recepționare, procesare, trimitere, așteptare și primire. În schimb, Nodul procesează cererile primite în stivă de evenimente persistente, trimite cereri mici unul câte unul și nu așteaptă răspunsuri.[16]

Câteva dintre caracteristicile importante care fac din Node.js prima alegere a arhitecților software sunt:

- Asincron și event-driven - Toate API-urile bibliotecii Node.js sunt asincrone, adică neblocante. În esență, un server bazat pe Node.js nu așteaptă niciodată ca un API să returneze date. Serverul trece la următorul API, și folosește un mecanism de notificare a evenimentelor Node.js, care ajută serverul să obțină un răspuns din apelul API anterior.
- Single-threaded, dar foarte scalabil - Node.js folosește un model single-thread, event looping. Mecanismul de evenimente ajută serverul să răspundă într-un mod care nu blochează și face ca serverul să fie puternic scalabil, spre deosebire de serverele tradiționale care creează fire limitate pentru a gestiona cererile.
- Foarte rapid - Fiind construit pe motorul JavaScript V8 al Google Chrome, biblioteca Node.js este foarte rapidă în executarea codului.
- Fără buffering - aplicațiile Node.js nu „tamponează” niciodată date. Aceste aplicații pur și simplu emit datele în bucăți.

Axios este o bibliotecă JavaScript pentru efectuarea de cereri HTTP din mediul browser sau din cadrul platformei Node.js. Aceasta oferă o interfață simplă și intuitivă pentru a comunica cu servicii web și a obține date de la acestea.

Axios poate fi utilizat pentru a realiza cereri HTTP de tip GET, POST, PUT, DELETE și altele către servere web. Această bibliotecă facilitează trimiterea datelor către server și primirea de răspunsuri într-un mod eficient și sigur.[17]

Una dintre caracteristicile principale ale Axios este posibilitatea de a trata și manipula date în diferite formate, cum ar fi JSON, XML, text sau fișiere binare. Această flexibilitate permite dezvoltatorilor să proceseze și să interpreteze răspunsurile primite în funcție de nevoile aplicației.

Axios oferă, de asemenea, funcționalități avansate, precum gestionarea automată a cookie-urilor și a headere-lor de autorizare, gestionarea erorilor și interceptarea cererilor și a răspunsurilor pentru a adăuga logica personalizată.

O parte importantă a funcționalității aplicației este capacitatea de a procesa și manipula date în formate populare, cum ar fi CSV (Comma-Separated Values) și JSON (JavaScript Object Notation).

În contrast, Python este un limbaj de programare dinamic multi-paradigmă, creat în 1989 de către programatorul olandez Guido van Rossum. Van Rossum, în calitatea sa de lider al comunității de dezvoltatori de software, a continuat să contribuie la îmbunătățirea limbajului Python și la dezvoltarea implementării de bază, CPython, scrisă în limbajul C.

NPM (Node Package Manager) este managerul de pachete implicit pentru platforma Node.js. Acesta oferă posibilitatea să instaleze, să gestioneze și să utilizeze biblioteci și module externe în aplicațiile lor JavaScript. Prin intermediul NPM, dezvoltatorii pot accesa un ecosistem vast de pachete și resurse dezvoltate de comunitatea globală, ceea ce facilitează dezvoltarea rapidă și extinderea funcționalității aplicațiilor.[19]

Gmail-js este o bibliotecă JavaScript care oferă abstracții și funcționalități pentru a interacționa cu interfața Gmail în cadrul unei extensii web. Aceasta permite să citească și să manipuleze e-mailurile și alte elemente ale interfeței Gmail, cum ar fi etichetele, conversațiile și contactele. Gmail-js facilitează automatizarea și personalizarea interacțiunii cu Gmail.

jQuery este o bibliotecă JavaScript populară și puternică, care simplifică procesul de manipulare a DOM-ului (Document Object Model) și de interacțiune cu HTML-ul și CSS-ul în cadrul aplicațiilor web. jQuery oferă un set bogat de funcții și metode care simplifică sarcinile comune, cum ar fi manipularea elementelor DOM, gestionarea evenimentelor, realizarea de animații și efecte vizuale, comunicarea asincronă cu serverul și multe altele. Prin intermediul jQuery, dezvoltatorii pot scrie cod JavaScript mai concis și mai eficient, reducând timpul și efortul necesare pentru dezvoltarea aplicațiilor web interactive.[21]

Pandas, NumPy și Scikit-learn (sau sklearn) sunt trei biblioteci Python extrem de populare și puternice utilizate în știința datelor și învățarea automată (machine learning).

Pandas este o bibliotecă open-source ce oferă structuri de date și instrumente pentru manipularea și analiza datelor. Aceasta furnizează obiecte de tip DataFrame, care permit organizarea și prelucrarea eficientă a datelor în tabele structurate. Pandas oferă funcționalități avansate de filtrare, sortare, agregare și manipulare a datelor, facilitând explorarea și prelucrarea eficientă a seturilor de date mari în Python.[16]

NumPy (Numerical Python) este o bibliotecă fundamentală pentru calculul științific în Python. Aceasta oferă un suport puternic pentru lucrul cu matrice multidimensionale și funcționalități de calcul numeric eficient. NumPy permite manipularea datelor și executarea operațiilor matematice și statistice pe acestea, punand la dispozitie operații avansate de algebra

liniară, generare de numere aleatoare, transformări Fourier și multe altele. NumPy servește ca bază pentru multe alte biblioteci și este utilizată extensiv în domenii precum știința datelor, inteligența artificială și simulări științifice.[20]

Scikit-learn este o bibliotecă Python de învățare automată (machine learning) care oferă un set bogat de algoritmi și funcționalități pentru a rezolva diverse probleme de învățare supervizată și nesupervizată. Scikit-learn oferă implementări eficiente ale algoritmilor de clasificare, regresie, clusterizare, reducere a dimensionalității, selecție a caracteristicilor și multe altele. Această bibliotecă facilitează utilizarea și aplicarea tehnicilor de învățare automată în proiectele Python, oferind instrumente pentru pregătirea datelor, evaluarea modelelor și selecția de parametri optimi. Scikit-learn este adesea utilizată ca sursă de încredere pentru algoritmi de învățare automată și este considerată una dintre cele mai importante biblioteci în domeniul învățării automate.[22]

Prin utilizarea axios în combinație cu Node.js și Python, dezvoltatorii au la dispoziție o gamă largă de tehnologii și instrumente pentru a crea aplicații sau extensii web și API-uri complexe și performanțe. Aceste tehnologii sunt recunoscute în industrie pentru flexibilitatea și solubilitatea lor, oferind posibilități nelimitate.

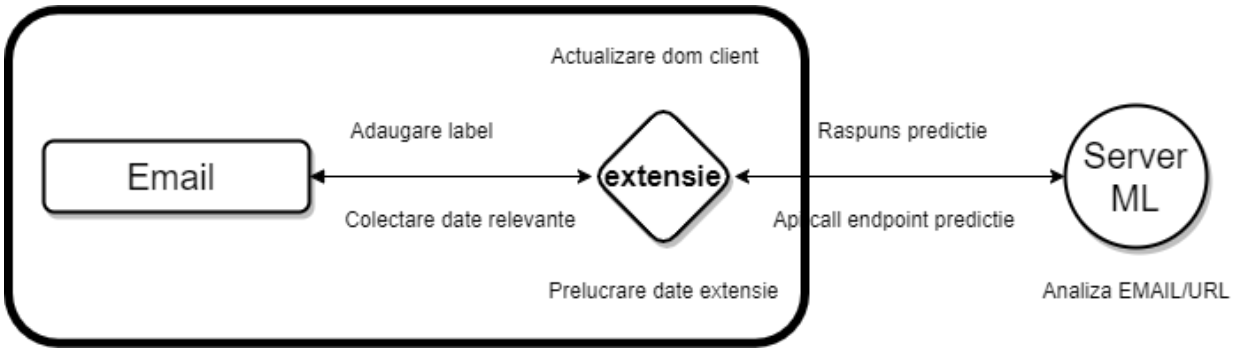


Figura 2.1. Arhitectura generală a aplicației

2.2 Proiectarea componentei de backend

2.2.1. Proiectarea API-ului

În această secțiune, voi descrie proiectarea componentei de backend pentru aplicația de detectare a emailurilor phishing. Componenta de backend va utiliza Flask, un framework web pentru Python, pentru a primi și gestiona cereri HTTP și va integra două modele de învățare automată pentru determinarea dacă URL-urile din email sunt phishing și pentru analizarea mesajului în sine.

Pentru a implementa componenta de backend, vom crea module separate pentru gestionarea URL-urilor și a mesajelor. Aceste module vor utiliza biblioteci precum Flask, pandas, numpy, sklearn pentru prelucrarea datelor și construirea modelelor de învățare automată.

Pentru gestionarea URL-urilor, vom crea un endpoint Flask care va primi cereri POST la ruta '/urlpredict'. Acest endpoint va primi un obiect JSON care conține o listă de URL-uri și va returna o predicție pentru fiecare URL, indicând dacă este phishing sau nu. Pentru a face predicția, vom utiliza modelul de învățare automată antrenat pe setul de date de URL-uri maligne și benigne. Vom utiliza un vectorizer TF-IDF pentru a transforma URL-urile într-o reprezentare numerică și un model de regresie logistică pentru clasificare.

Pentru analiza mesajelor, vom crea un alt endpoint Flask la ruta '/predict'. Acest endpoint va primi cereri POST cu un obiect JSON care conține detalii despre mesajul email. Vom utiliza un model separat de învățare automată pentru a face predicții privind caracterul phishing al mesajului. La fel ca și în cazul URL-urilor, vom utiliza un vectorizer TF-IDF pentru a transforma textul mesajului într-o reprezentare numerică și un model de regresie logistică pentru clasificare.

Pentru a asigura funcționarea corectă a componentei de backend, vom utiliza următorul flux de lucru:

1. Încărcarea și prelucrarea datelor:
   Vom încărca seturile de date pentru emailuri și URL-uri din fișierele CSV și le vom prelucra pentru a elimina valorile nule și a ajusta formatele.
   Vom separa datele în coloane individuale pentru fiecare atribut relevant.

2. Antrenarea modelelor:
   Vom utiliza datele prelucrate pentru a antrena modelele de învățare automată, unul pentru URL-uri și unul pentru mesaje.
   Pentru fiecare model, vom utiliza un vectorizer TF-IDF pentru a transforma datele textuale într-o reprezentare numerică și un model de regresie logistică pentru clasificare.

3. Implementarea endpoint-urilor Flask:
   Vom crea două endpoint-uri în aplicație: '/urlpredict' și '/predict'.
   Endpoint-ul '/urlpredict' va primi cereri POST cu un obiect JSON care conține URL-urile pentru a face predicții privind phishing-ul. Acesta va utiliza modelul antrenat pe URL-uri pentru a genera predicții și va returna rezultatele într-un obiect JSON.
   Endpoint-ul '/predict va primi cereri POST cu un obiect JSON care conține detalii despre mesajul email. Acesta va utiliza modelul antrenat pe mesaje pentru a face predicții privind phishing-ul și va returna rezultatele într-un obiect JSON.

Vom testa aplicația utilizând cereri curl sau alte instrumente de testare API.
Vom asigura că endpoint-urile returnează răspunsurile corecte și predicțiile dorite pentru URL-uri și mesaje.

2.2.2. Proiectarea Extensiei

O etapă importantă este validarea și prelucrarea datelor primite, pentru a se asigura că acestea respectă un format corect și nu conțin informații malițioase. Pentru aceasta, se pot utiliza diverse tehnici de validare, cum ar fi verificarea formatului adresei de email sau eliminarea caracterelor speciale cu potențial periculos.
Comunicarea cu modelele de detectare a phishing-ului:

Pentru fiecare email analizat, se va apela modelul corespunzător, iar rezultatele obținute vor fi prelucrate ulterior.

În cadrul funcțiilor de gestionare asociate rutelor, se vor procesa rezultatele obținute din modelele de detectare a phishing-ului.

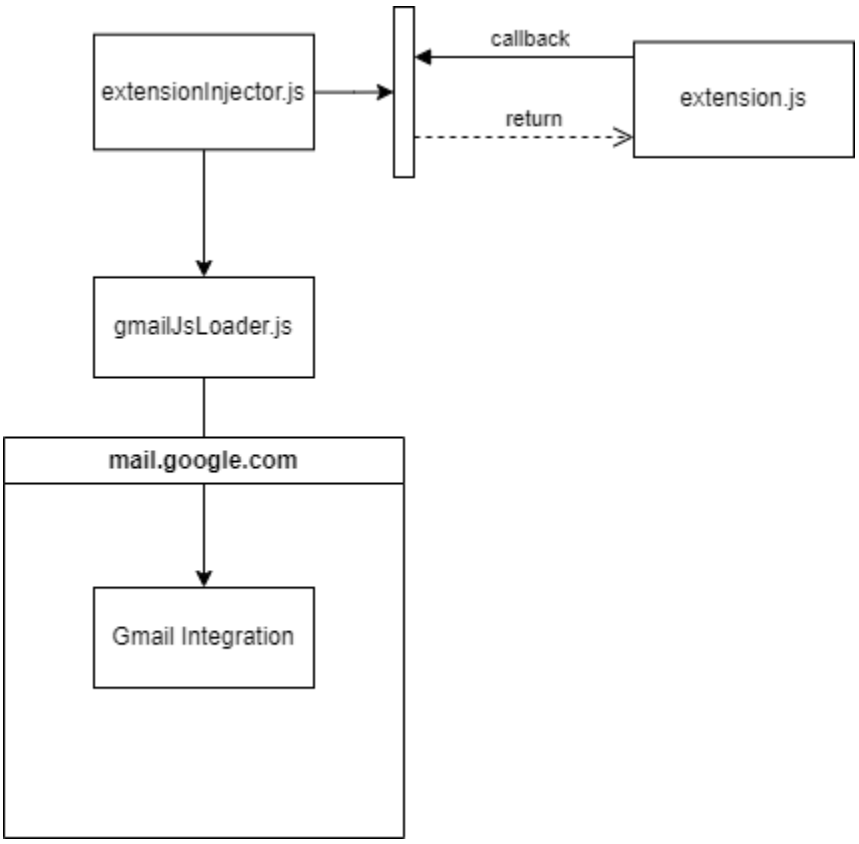


Figura 2.2 de secvență pentru procesul de parsare

2.2.3. Seturile de date

Seturile de date utilizate în cadrul proiectului au o importanță crucială în dezvoltarea și evaluarea aplicației noastre. Acestea reprezintă colecții de date structurate și etichetate, care servesc ca bază pentru antrenarea și testarea modelelor noastre de detectare a phishing-ului.

Setul de date pentru e-mailuri a fost obținut din surse interne iar datele au fost etichetate manual. Acest set de date conține clasificat individual fiecare element de date, atribuindu-i o etichetă relevantă în funcție de natura sa (phishing sau non-phishing). Această abordare asigură acuratețea și calitatea seturilor de date utilizate în proiectul nostru. Pentru a obține seturile de date, am utilizat diverse surse de informații, cum ar fi e-mailuri legitime, e-mailuri suspecte de phishing care vizează serviciul de email al Universității Tehnice Gheorghe Asachi din Iași cât și e-mailuri raportate de către Computer Emergency Response Team of Romania, rapoarte de securitate și alte resurse relevante. Acestea au fost apoi procesate și structurate într-un format adecvat, astfel încât să poată fi utilizate în procesul de antrenare și testare a modelelor noastre.

Seturile de date includ diverse caracteristici ale e-mailurilor, cum ar fi conținutul mesajului, informații despre expeditor și destinatar, subiectul și altele. Aceste caracteristici sunt esențiale în identificarea indiciilor de phishing și în construirea unor modele eficiente de detecție.

2.3. Proiectarea componentei de frontend

Componenta de frontend a fost proiectată pentru a oferi un feedback relevant și vizual utilizatorului cu privire la rezultatul predicției de detectare a phishing-ului. În cadrul acestei proiectări, extensia va aplica automat un label corespunzător asupra e-mailului deschis curent prin identificarea rezultatului analizei. Astfel, interfața utilizatorului este actualizată în consecință, afișând în mod vizibil și clar această etichetă pentru a oferi informații relevante și a transmite utilizatorului rezultatul predicției.

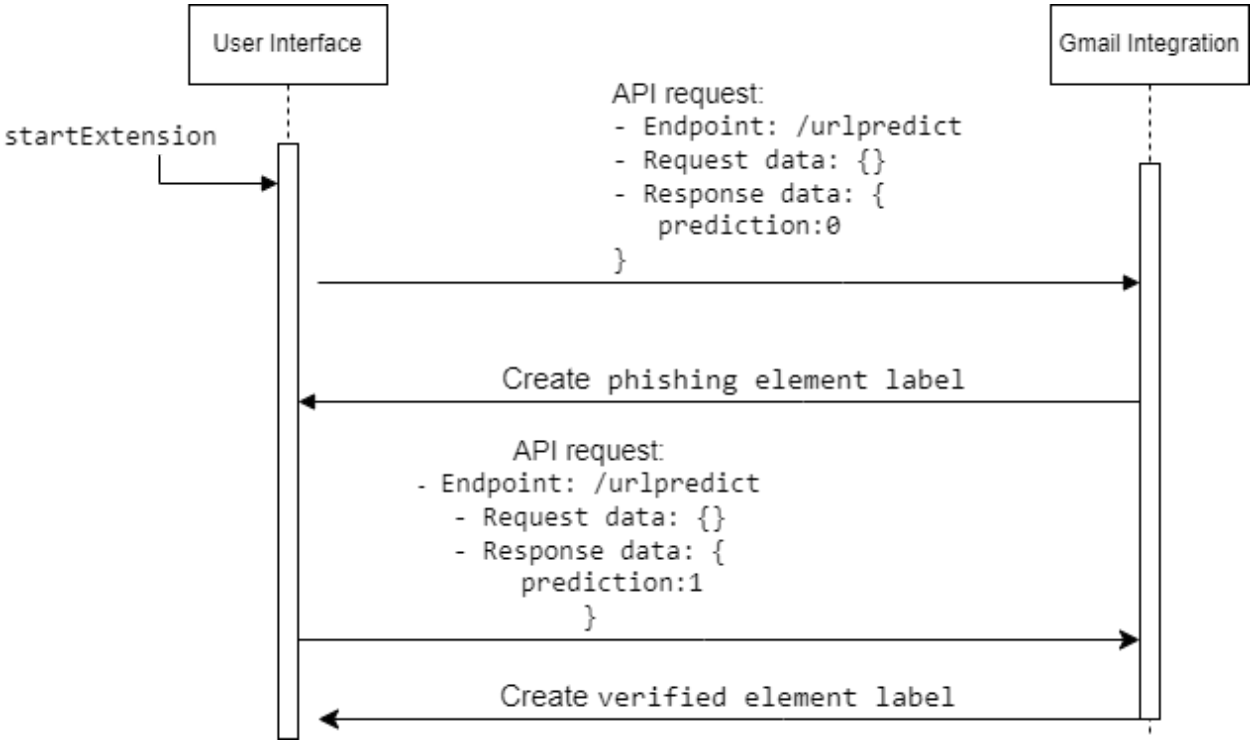


Figura 2.3. Diagrama de secvență pentru procesul de predicție

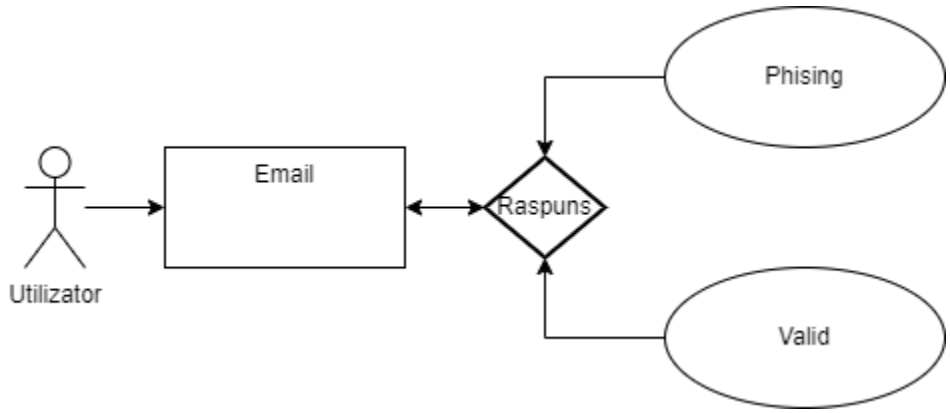


Figura 2.4. Diagrama use-case corespunzătoare funcționalității

III. Implementarea aplicației

În cadrul capitol, se vor detalia pașii implementării aplicației, urmărind metodologiile expuse anterior în capitolul Proiectarea aplicației, atât prin intermediul a mai multor ilustrații, cât și a secvențelor de cod. De asemenea, vor fi prezentate și dificultățile întâlnite pe parcursul dezvoltării aplicației.

3.1 Implementarea componentei de frontend

Pentru adaugarea etichetelor am folosit o combinație de HTML, CSS și jQuery pentru a crea și stiliza elementele din interfața Gmail. Codul de mai jos reprezinta logica pentru adăugarea a două label-uri (etichete) în interfața Gmail, în funcție de rezultatul returnat de API-ul de predicție a e-mailurilor.

```
if (response.data.prediction === 0) {
const phishingElement =
'<div style="color: white;display: flex;width: 300px;height: 25px;border-radius: 30px;background:
red;justify-content: center;align-items: center;">Emailul periculos</div>';
gmail.tools.add_toolbar_button(
phishingElement,() => {},
"custom-style-class"
);
}
if (response.data.prediction === 1) {
const verifiedElement =
'<div style="color: white;display: flex;width: 300px;height: 25px;border-radius: 30px;background:
green;justify-content: center;align-items: center;">Emailul este sigur</div>';
gmail.tools.add_toolbar_button(
verifiedElement,() => {},
"custom-style-class"
);
}
```

Listing 1. Implementarea Labelurilor

Fig.3.1 Label pentru e-mailul sigur

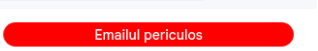


Fig.3.2 Label pentru e-mailul periculos

3.2. Agregarea setului de date

Agregarea setului de date este o etapă esențială în dezvoltarea unei aplicații bazate pe machine learning, această etapă implică colectarea și combinarea datelor relevante din diferite surse pentru a forma un set de date complet și reprezentativ.

3.2.1 Setul de date pentru email

Setul de date folosit pentru antrenarea și evaluarea modelului de ansamblu conține eșantioane pentru cele două clase de interes, respectiv phishing și benign. Setul de date conține date etichetate, care reprezintă predicțiile fiecărui submodel. Acest set este alcatuit din emailuri reale care au fost raportate și etichetate manual, astfel incat au rezultat 3225 de emailuri, dintre care 711 au fost catalogate drept phishing iar restul de 2701 emailuri benigne.

Un aspect de luat în calcul al acestui set de date este faptul că, deși este mai mic în comparație cu alte seturi de date disponibile, este special datorită conținutului său în limba română. Este dificil să găsim resurse publice cu e-mailuri etichetate și clasificate în alta limba decat engleza, ceea ce face acest set de date valoros pentru antrenarea și evaluarea modelelor de detecție a phishing-ului specifice pentru limba română.

Category	From	Fullname	To	Subject	Body	Return_Path	Delivered_to	Content_Length
phising	admin@newtarg	Agenția Național	undisclosed-reci	Plăți de impozite	O zi buna,Acest	user@example.(	user@example.(	337
phising	admin@newtarg	Agenția Național	undisclosed-reci	Factura fiscala	Vă informam că	user@example.(	user@example.(	190

Tabelul 1. Exemplu date email

Acest set de date descris în Tabelul 1. conține următoarele câmpuri: Category care indica tipul fiecărui e-mail din setul de date, From indica adresa de e-mail de la care a fost expediat mesajul, Fullname reprezinta numele expeditorului, To indică destinatarii, Subject este subiectul e-mailului, Body corpul mesajului, Return_Path este adresa de returnare, Delivered_to: destinatarul e-mailului și Content_Length indică lungimea conținutului.

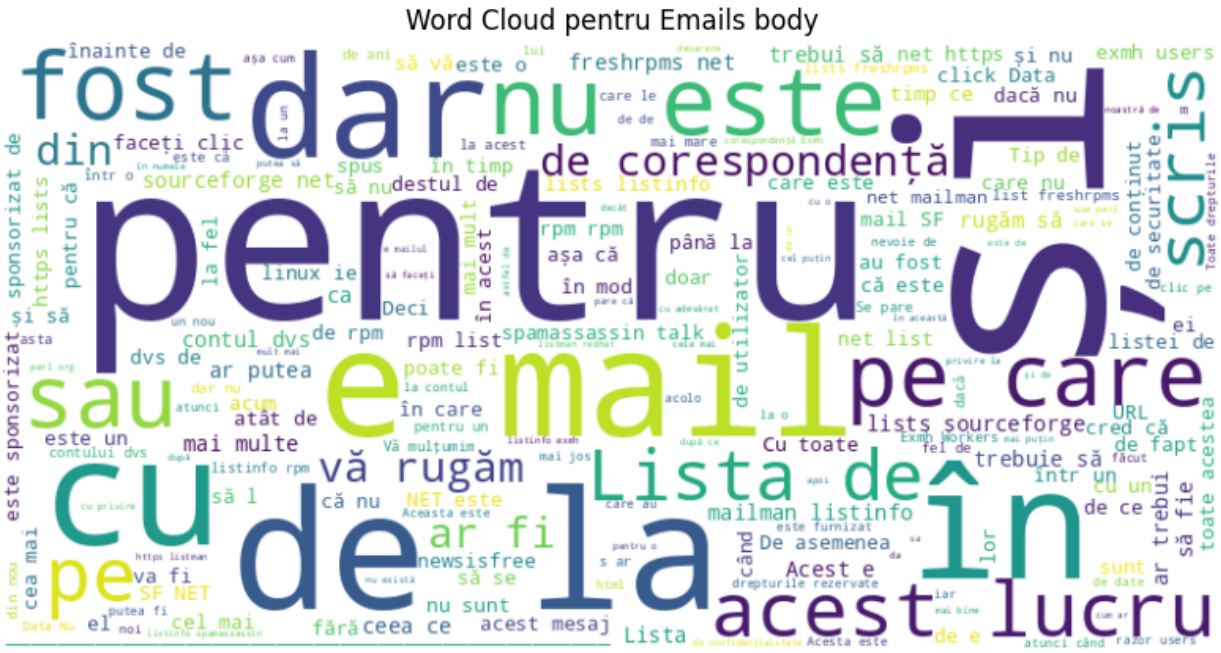


Figura 3.3. Word Cloud pentru setul de date email combinat

Word cloud-ul reprezintă o interpretare vizuală a cuvintelor cheie care au o pondere mai mare în cadrul setului de date. Prin examinarea word cloud-ului din Figura 3.3. Putem determina că ponderea cea mai mare a cuvintelor coincide cu cele mai uzuale cuvinte din limba română. Conform Lingvistul Adina Dragomirescu de la Institutul de lingvistică „Iorgu Iordan – Al. Rosetti" al Academiei Române cele mai des folosite cuvinte sunt prepoziţiile, conjuncţiile coordonatoare/ subordonatoare şi verbele auxiliare.[25]

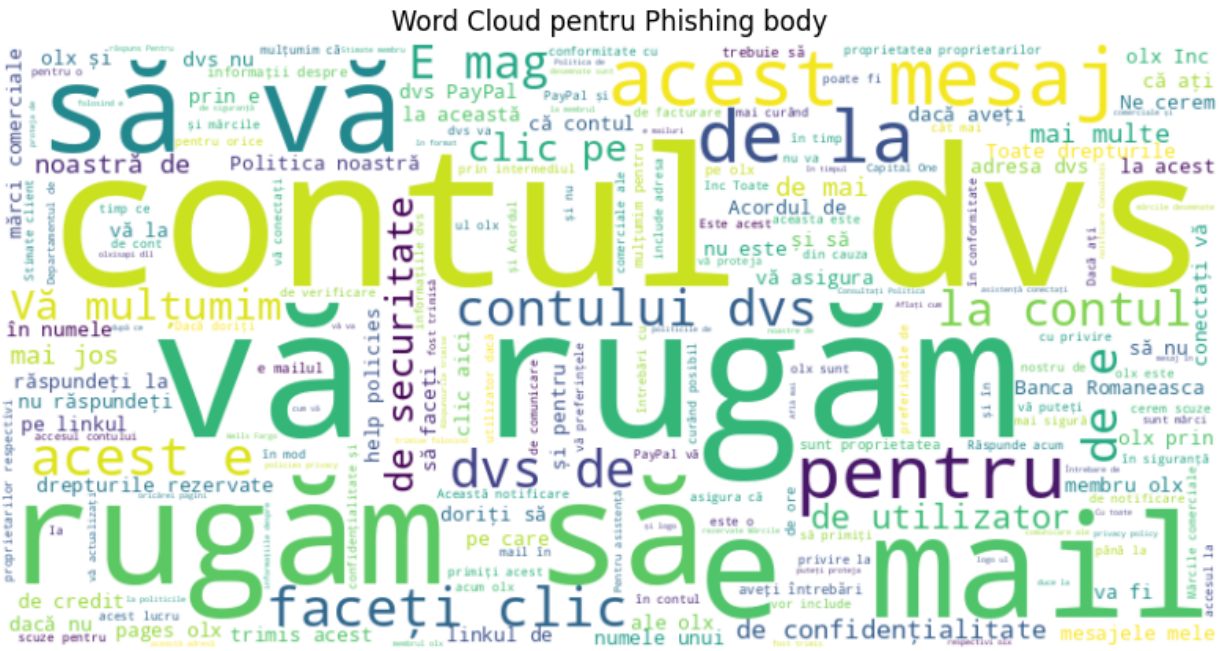


Figura 3.4. Word Cloud pentru setul de date email phishing

Spre deosebire de setul combinat, email-urile de phishing descriu cel mai des termeni de natura autoritară care solicită sau impun presiune emoțională asupra destinatarilor prin utilizarea unor declarații urgente sau amenințătoare, pentru a-i determina să acționeze rapid și să dezvăluie aceste informații.

3.2.2 Setul de date pentru url

Setul de date folosit pentru antrenarea și evaluarea modelului pentru analiza URL-urilor este este preluat din baza de date de pe site-ul kaggle.com, **Malicious URLs dataset [23]**. Acest set contine 651,191 URL-uri, din care 428103 benigne sau sigure, 96457 defacement, 94111 phishing și 32520 malware. Vom modifica acest set de date astfel încat vom obține doar 2 etichete agregând etichetele defacement, malware în categoria generală de phishing reprezentând e-mailuri periculoase.

url	type
br-icloud.com.br	phishing
mp3raid.com/music/krizz_kaliko.html	benign
bopsecrets.org/rexroth/cr/1.htm	benign

Tabelul 2. Exemplu date url

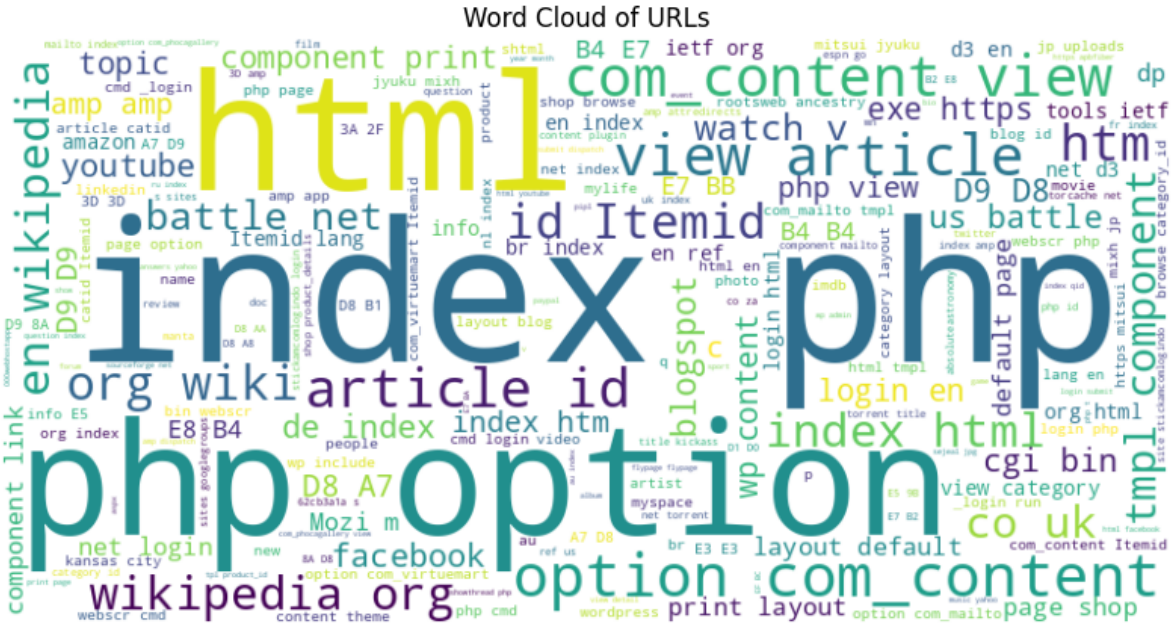


Figura 3.5. Word cloud pentru setul de date pentru url-uri

Din setul de date pentru URL-uri, s-a generat un word cloud reprezentat în figura 3.5. care are ca și cuvinte semnificative html, index, php acest fapt este în concordanță cu studiul celor de la w3techs [24] care arată că peste 75.5% dintre toate website-urile folosesc php ca limbaj de programare. Este interesant să observăm cum rezultatele obținute în studiul w3techs.com sunt confirmate parțial de analiza setului de date utilizat în cadrul prezentei lucrări. Acest lucru sugerează că setul de date este reprezentativ și corespunde cu realitatea.

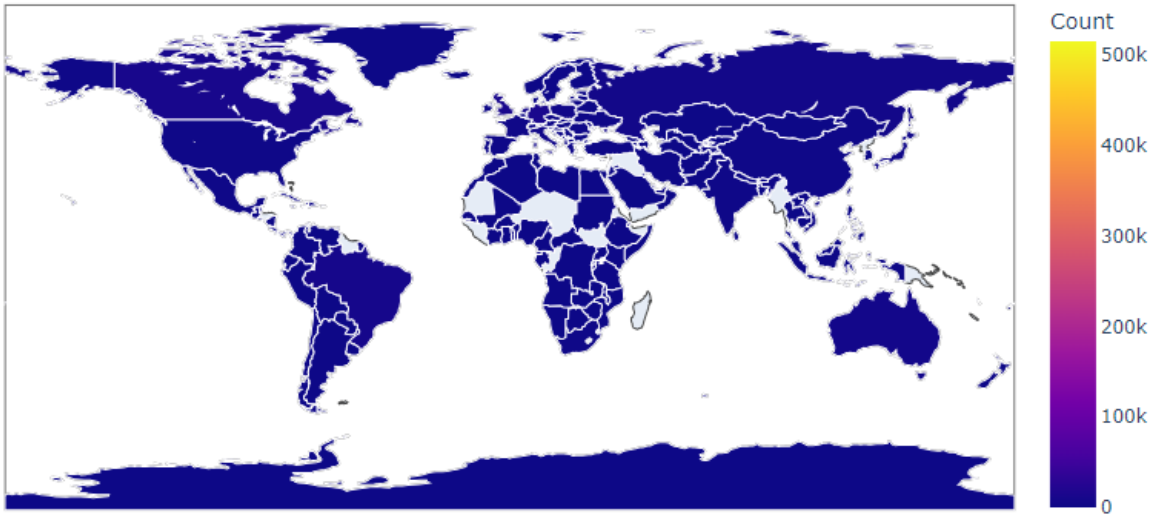


Figura 3.6. Distribuția URL-urilor pe regiuni ale globului

Un alt indiciu al faptului ca setul este reprezentativ este distribuția acestuia pe regiunile globului. Din figura 3.6. observăm faptul că acestea sunt distribuite geografic uniform pe toată suprafața terrei și că nu există o concentrare sau un dezechilibru semnificativ într-o anumită zonă.

3.3. Implementarea modelelor

Acest capitol se referă la etapa în care modelele sunt implementate și pregătite pentru a fi utilizate în analiza și clasificarea datelor.

3.3.1. Selectarea modelelor

Pentru a determina ce model de învățare automată s-ar potrivi am antrenat următoarele modele:

- LogisticRegression: Acest model aplică regresia logistică pentru a realiza clasificarea, potrivit pentru probleme de clasificare binară, poate furniza probabilități de apartenență la fiecare clasă.

- KNeighborsClassifier: Modelul utilizează algoritmul vecinilor k (KNN) pentru a realiza clasificarea. Se bazează pe similaritatea între punctele de date și vecinii lor în spațiul caracteristicilor.

- DecisionTree: Acest model construiește un arbore de decizie în care fiecare nod reprezintă o caracteristică și fiecare ramură reprezintând o decizie.

- RandomForest: Acest model este o combinație de mai mulți arbori de decizie (un ansamblu de arbori aleatori). Folosește tehnicile de "bagging" și "randomizare" pentru a obține o clasificare mai precisă și pentru a reduce overfitting-ul.

- SVC (Support Vector Classifier): Acest model utilizează vectori suport pentru a separa exemplele din diferitele clase. Poate trata atât probleme de clasificare binară, cât și probleme de clasificare multiplă.

Pentru determinarea algoritmului optim vom folosi o serie de metrici precum:

- Acuratețea, aceasta reprezintă proporția corectă de clasificari în raport cu numărul total înregistrări.
- Precizia, măsoară proporția corectă de instanțe pozitive identificate în raport cu numărul total de instanțe clasificate ca fiind pozitive.
- Recall, acest metric măsoară numărul de instanțe pozitive identificate de algoritm în raport cu numărul total de instanțe reale pozitive din întregul set de date.

Cu ajutorul librariei sklearn am implementat, antrenat și evaluat cu setul de date descris mai sus, și am obtinut următoarele rezultate:

3.3.1.1. Rezultate Set Date Email

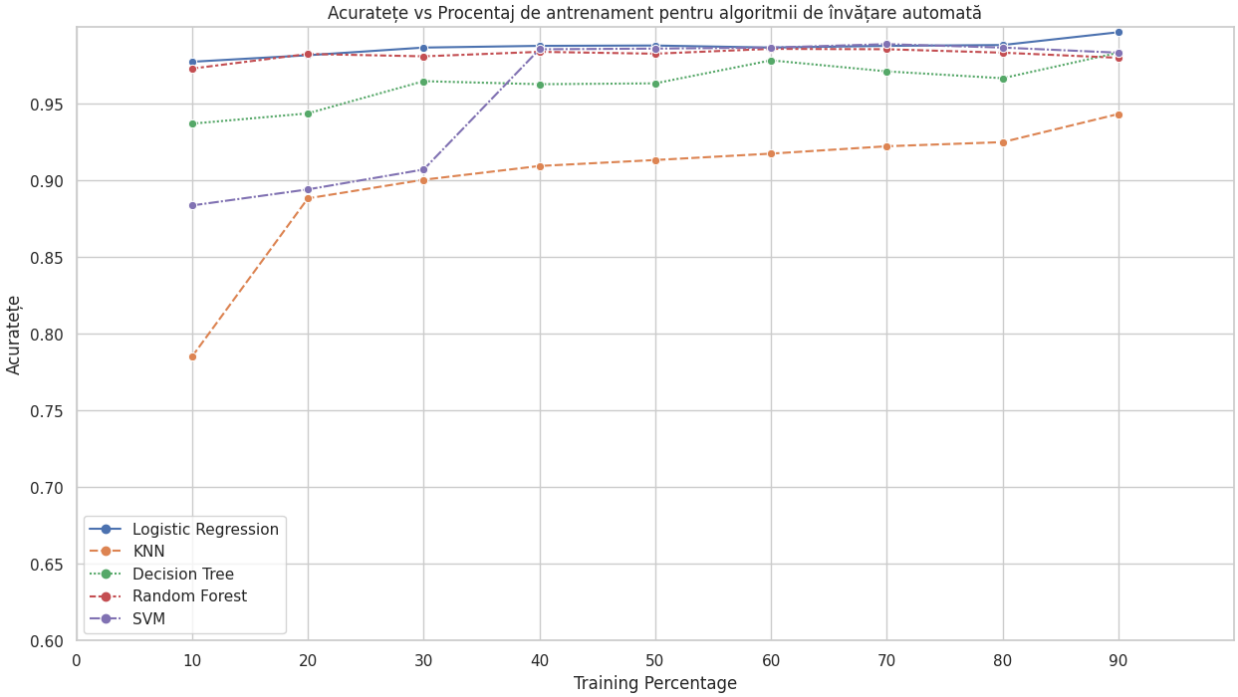


Figura 3.7. Comparatie între algoritmi pentru gradul de acuratețe în raport cu procentajul de antrenament

În urma preciziilor furnizate pentru fiecare algoritm, observăm câteva tendințe. Regresia logică are în mod constant o acuratețe ridicată, KNN începe cu o precizie scăzută dar se îmbunătățește odată cu creșterea procentului de antrenament, Decision Tree are inițial o precizie medie dar aceasta crește treptat odată cu procentul de antrenament, Random Forest obține rezultate bune constant pe tot intervalul de antrenament și SVM începe cu o precizie scăzută îmbunătățindu-se rapid concomitent cu creșterea procentului de antrenament.

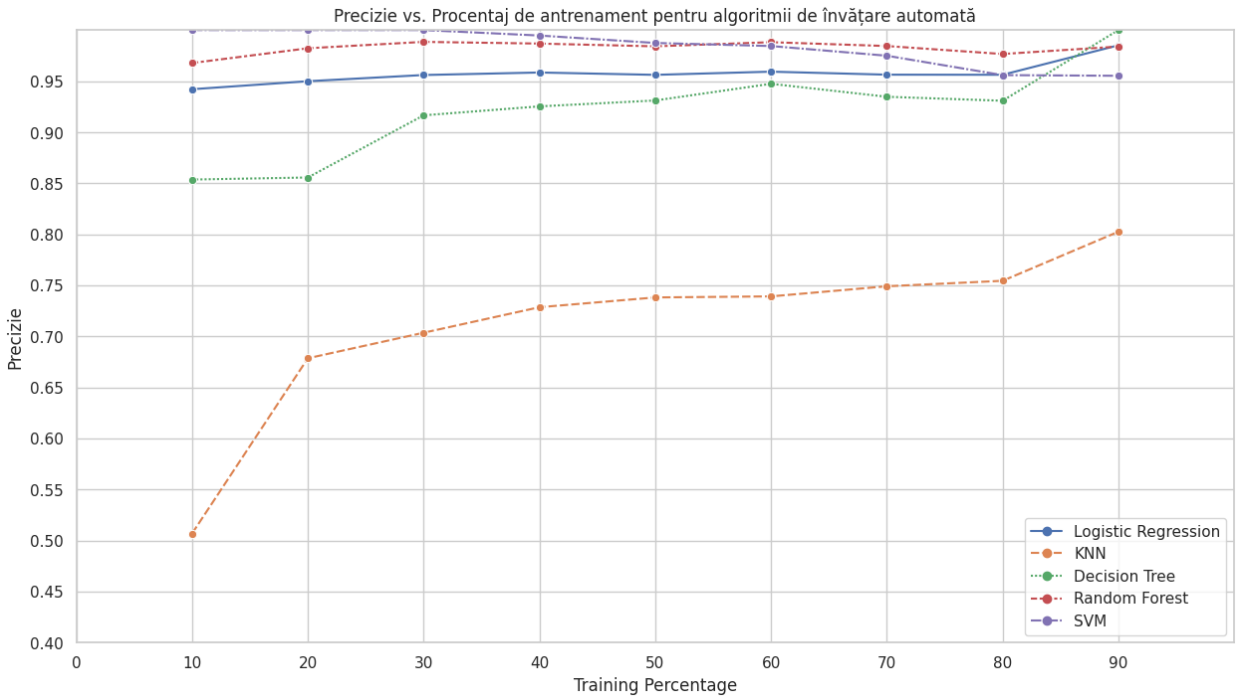


Figura 3.8. Comparatie între algoritmi pentru gradul de precizie în raport cu procentajul de antrenament

La fel ca și în cazul acurateții, Regresia logică prezintă valori în creștere însă se situează la mijlocul graficului între Random Forest și Decision Tree.

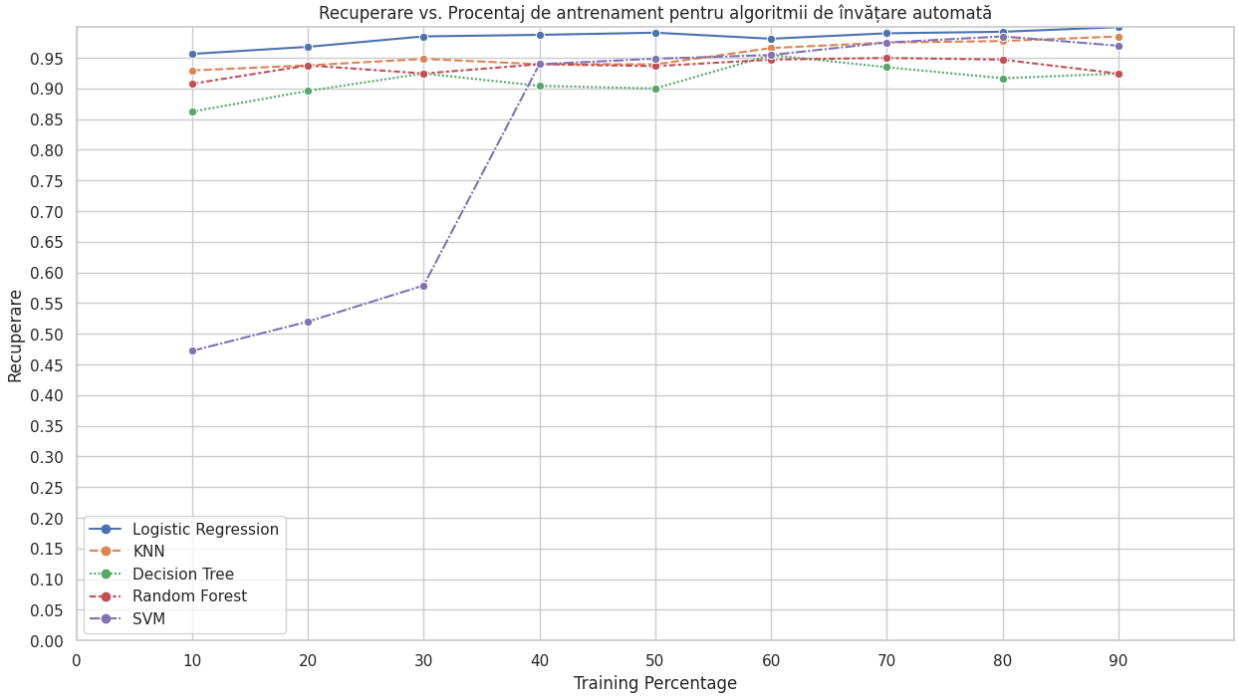


Figura 3.9. Comparatie între algoritmi pentru gradul de recuperare în raport cu procentajul de antrenament

Conform graficului din figura 3.9. Regresia logistică prezintă valoarea cea mai mare de recall variind între 0,956303 și 1,000000.

3.3.1.2. Rezultate Set Date Url

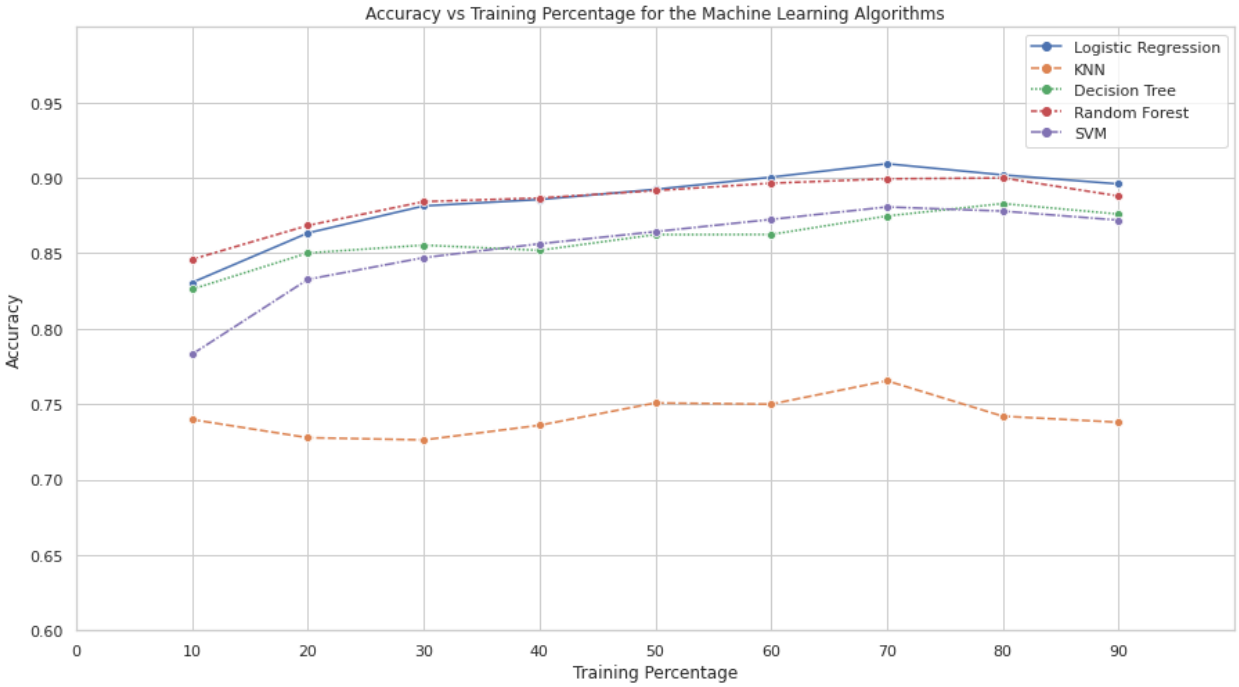


Figura 3.10. Comparație între algoritmi pentru gradul de acuratețe în raport cu procentajul de antrenament

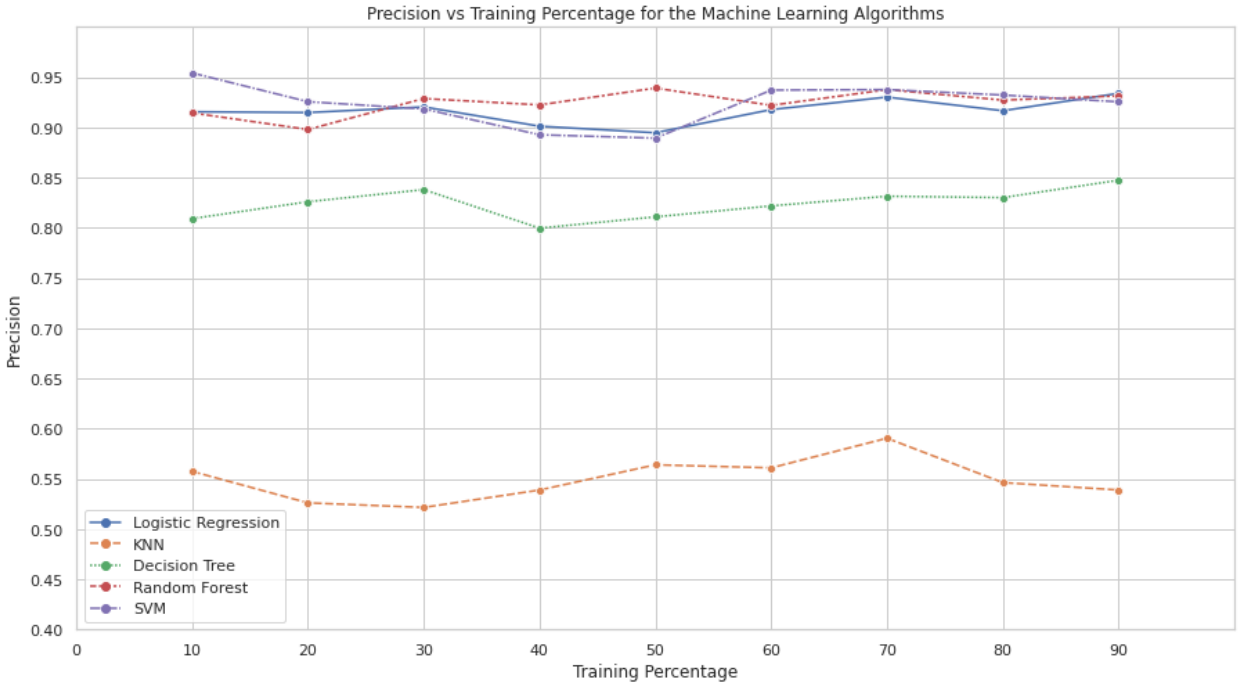


Figura 3.11. Comparație între algoritmi pentru gradul de precizie în raport cu procentajul de antrenament

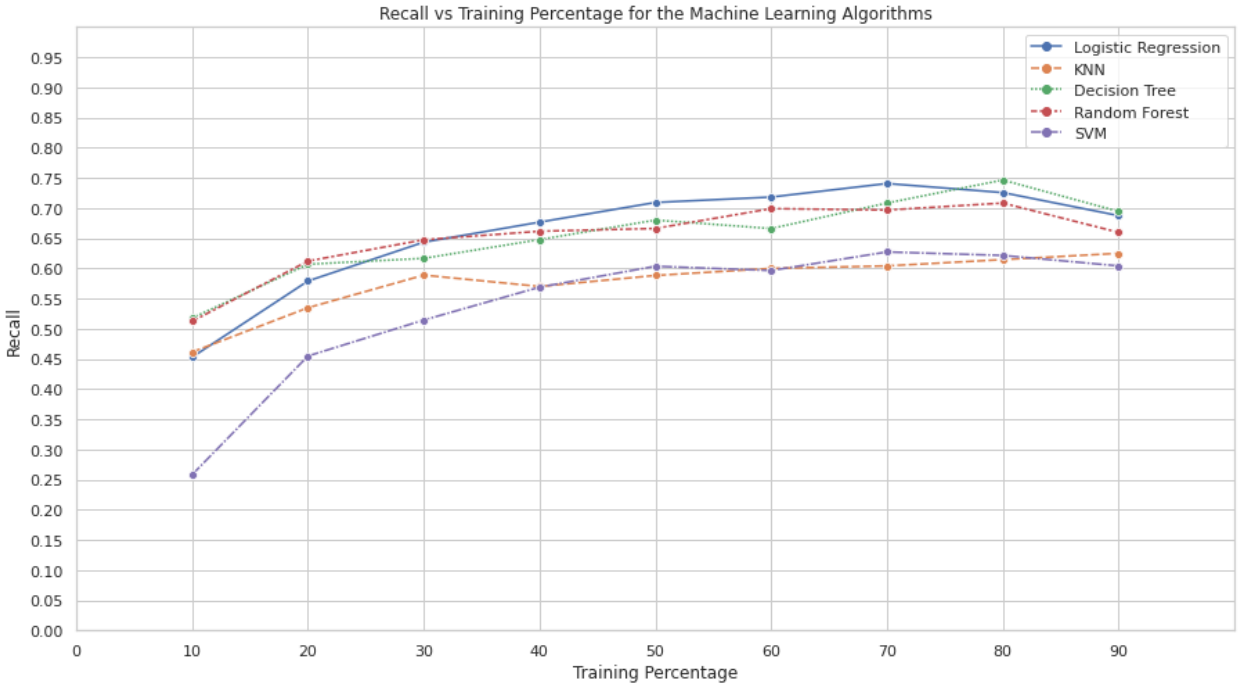


Figura 3.12. Comparație între algoritmi pentru gradul de recuperare în raport cu procentajul de antrenament

Pentru cel de-al doilea model observam ca obținem rezultate performanțe similare cu primul model.

3.3.1.3 Concluzia comparației

Pe baza acestor rezultate obținute în urma evaluării algoritmilor prezentați pentru clarificarea conținutului e-mailurilor, se poate observa că modelul de Regresie Logistică are performanțe consistente pe tot parcursul evaluării, obținând valori ridicate de acuratețe, precizie și recall în majoritatea scenariilor de antrenare.

3.3.3. Antrenarea modelelor

```
# Încarcă datele din fișierul CSV într-un DataFrame pandas
raw_mail_data = pd.read_csv('mail_data.csv')
raw_url_data = pd.read_csv('malicious_phish.csv')
# Înlocuiește valorile nule cu un șir nul
mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)), '')
url_data = raw_url_data.where((pd.notnull(raw_url_data)), '')

# Separă datele în coloane individuale
from_data = mail_data['From']
fullname_data = mail_data['Fullname']
to_data = mail_data['To']
subject_data = mail_data['Subject']
body_data = mail_data['Body']
return_path_data = mail_data['Return_Path']
category_data = mail_data['Category']
type_data = url_data['type']

# Asociază valorile 'Category' cu etichete numerice
category_data = category_data.map({'ham': 1, 'phishing': 0})
type_data = type_data.map({'benign': 1, 'defacement': 0, 'malware': 0, 'phishing': 0, 'malware': 0})
# Elimină rândurile cu valori infinite
mail_data = mail_data[np.isfinite(category_data)]
url_data = url_data[np.isfinite(type_data)]

# Combine valorile tuturor parametrilor într-un singur vector de caracteristici
features = pd.concat([from_data, fullname_data, to_data, subject_data, body_data, return_path_data],
axis=1)
features['Features'] = features.apply(lambda x: ' '.join(map(str, x)), axis=1)

url_data['url'] = url_data['url'].replace('www.', '', regex=True)
Xu = url_data['url']
Yu = type_data
url_data = pd.concat([Xu, Yu], axis=1).dropna()

# Separă datele în texte și etichete
```

```
X = features['Features']
Y = category_data

# Elimină rândurile cu valori NaN atât în X, cât și în Y
data = pd.concat([X, Y], axis=1).dropna()
X = data['Features']
Y = data['Category']

# Verifică dacă există cel puțin un eșantion în date
if X.shape[0] == 0:
raise ValueError("Nu s-au găsit eșantioane valide în date.")

# Transformă datele textuale în vectori de caracteristici
feature_extraction = TfidfVectorizer(min_df=1, stop_words='english', lowercase=True)
X_features = feature_extraction.fit_transform(X)
Xu_features = feature_extraction.transform(Xu)

# Antrenează modelul de clasificare
model = LogisticRegression(max_iter=1000, solver='sag', random_state=42)
modelUrl = LogisticRegression(max_iter=1000, solver='sag', random_state=42)
print("Se antrenează modelul...")
model.fit(X_features, Y)
modelUrl.fit(Xu_features, Yu)
print("Gata...")
```

Listing 2. Antrenarea celor 2 modele folosind Regresia Logistică

Antrenarea celor 2 modele este descrisă în listingul mai sus prezentat. Pentru primul set de date, modelul este antrenat folosind variabila model și metoda fit(X_features, Y), în timp ce pentru cel de-al doilea set de date, modelul este antrenat folosind variabila modelUrl și metoda fit(Xu_features, Yu).

3.4. Implementarea componentei de Backend

Așa cum am prezentat în capitolul proiectării, modulul de backend este implementat folosind o arhitectură bazată pe un API cu 2 endpoint-uri "urlpredict" și "predict".Aceste endpointuri reprezintă punctele de acces în serverul backend și sunt utilizate pentru a primi cereri specifice și a returna răspunsuri cu predicțiile făcute de cele 2 modele.

3.4.1. Implementare Endpoint API

- Endpoint-ul "urlpredict" poate fi utilizat pentru a primi cereri referitoare la predicția tipului de URL (benign, phishing) pe baza unei adrese sau a adreselor URL furnizate.

Acest endpoint primește adresa URL în corpul cererii HTTP și utilizează modelul antrenat pentru a realiza o predicție. Răspunsul poate fi în format JSON și poate conține rezultatul predicției “0”/”1”.

- Endpointul "predict" poate fi utilizat pentru a primi cereri referitoare la predicția categoriei unui email (ham, phishing) pe baza parametrilor specifici e-mailului. Acești parametri, precum "From", "To", "Subject", "Body" etc., sunt trimiși în corpul cererii HTTP și sunt utilizați pentru a realiza predicția cu ajutorul modelului antrenat.

```
@app.route('/urlpredict', methods=['POST'])
def urlpredict():
  input_urls = request.json
  print("Datele cererii:", input_urls)
  if not input_urls:
     return jsonify({'error': 'Nu s-au furnizat date URL.'}), 400
  predictions = []
  for url in input_urls:
# Transformă adresa URL în vectori de caracteristici
     url = url.replace('www.', '')
     url_features_transformed = feature_extraction.transform([url]).toarray()[0]
# Realizează predicția
     prediction = modelUrl.predict([url_features_transformed])[0]
     predictions.append(prediction)
# Calculează media predicțiilor
  mean_prediction = int(sum(predictions) / len(predictions))
# Pregătește răspunsul
  response = {'prediction': mean_prediction}
  return jsonify(response)


@app.route('/predict', methods=['POST'])
def predict():
  input_mail = request.json
  print("Datele cererii:", input_mail) # Printează datele cererii
  if not input_mail:
     return jsonify({'error': 'Nu s-au furnizat date de email.'}), 400
# Extrage caracteristicile e-mailului
  email_features = pd.DataFrame([input_mail], columns=features.columns)
  email_features['Features'] = email_features.apply(lambda x: ' '.join(map(str, x)), axis=1)
# Verifică dacă există cel puțin un eșantion în caracteristicile e-mailului
  if email_features.shape[0] == 0:
     return jsonify({'error': 'Datele de email nu conțin niciun eșantion.'}), 400
# Transformă caracteristicile e-mailului în vectori de caracteristici
  email_features_transformed =  feature_extraction.transform(email_features['Features']).toarray()
```

```
# Realizează predicții
  predictions = model.predict(email_features_transformed)

# Pregătește răspunsul
  response = {'prediction': int(predictions[0])}

  return jsonify(response)
```

Listing 3. Endpointuri API

3.4.2. Implementare Extensie

Implementarea extensiei prezentate se bazează pe utilizarea bibliotecii Gmail.js și Axios pentru a interacționa cu interfața Gmail și pentru a trimite cereri către endpointurile prezentate.

```
function parseEmail(emailData) {
  const from = emailData.from.address;
  const fullname = emailData.from.name;
  const to = emailData.to.address;
  const subject = emailData.subject;
  const body = emailData.content_html;
  const returnPath = emailData.from.address;
  // Sanitize the body
  const cleanBody = DOMPurify.sanitize(body, {
    USE_PROFILES: { html: false },
  });

  console.log("Clean body:", cleanBody);
  console.log("Original body:", body);
  // Create the request payload
  const payload = {
    From: from,
    Fullname: fullname,
    To: to,
    Subject: subject,
    Body: cleanBody
      .replace(/<[^>]+>/g, "")
      .replace(/ /g, " ")
      .replace(/\u200c/g, "")
      .replace(/&amp;/g, "&")
      .replace(/\s+/g, " ")
      .replace(/&lt;/g, "<")
```

```
    .replace(/&gt;/g, ">")
    .replace(/&quot;/g, "")
    .replace(/&#39;/g, "'")
    .replace(/u200c ;/g, "&")
    .trim(),
  Return_Path: returnPath,
 };
 const urlRegex = /(https?:\/\/[^\s]+)/g;
 const links = body.match(urlRegex) || [];

 console.log("Links:", links);
 return payload;
}
```

Listing 4. Funcția parseEmail

În cadrul funcției parseEmail descrise în Listing 4., se extrag datele relevante ale emailului precum expeditorul, destinatarul, subiectul și conținutul. De asemenea, se curăță conținutul HTML al emailului utilizând biblioteca DOMPurify pentru a asigura siguranța și a elimina orice elemente HTML nedorite sau potențial periculoase rămânând strict mesajul emailului.

Apelul către endpointuri ale serverului backend se realizează prin utilizarea bibliotecii Axios. Se trimite payload-ul cu datele emailului și se primește un răspuns care conține predicția referitoare la caracterul periculos al emailului.

```
{
"manifest_version": 3,
"name": "phishing-detection-ml",
"short_name": "gmailjsnode",
"version": "1.1",
"author": "Bogdan Iacob",

"description": "Phising detection using Machine Learning, Disertation project",

"content_scripts": [
{
"matches": [ "*://mail.google.com/*" ],
"js": [
"src/extensionInjector.js"
```

```
],
"run_at": "document_start"
}
],

"web_accessible_resources": [{
"resources": [
"dist/gmailJsLoader.js",
"dist/extension.js",
"dist/gmailJsLoader.js.map",
"dist/extension.js.map"
],
"matches": ["<all_urls>"]
}],
"host_permissions": [
"https://*/*"
]
}
```

Listing 5. Conținutul fișierului manifest

Manifestul descris în Listing 5. este un fișier de configurare care descrie extensia și specifică cum trebuie să funcționeze în mediul Chrome.

3.5. Provocări întâlnite

O dificultate a fost reprezentată de disponibilitatea limitată a datelor în limba română. Acest aspect a avut un impact mare asupra antrenarii modelului. Deoarece modelul se bazează pe algoritmi de învățare automată, are nevoie de date relevante și de înaltă calitate pentru a fi antrenat și pentru a produce rezultate precise. O altă provocare a fost în timpul implementării extensiei, am întâmpinat dificultăți pe parcursul procesului în manipularea conținutului e-mailului, extracția și prelucrarea corectă a datelor din email fiind greu de realizat fără biblioteca Gmail.js.

IV. Testarea aplicației

În cadrul acestui capitol, se va prezenta maniera de testare a modulelor aplicației și pașii necesari instalării aplicației. De asemenea, vor fi prezentate o parte a rezultatelor experimentale obținute până în momentul de față.

4.1. Instalarea aplicației

Înainte de parcurgerea pașilor, trebuie instalate librăriile/programele:

- Python:
  - ➢ Flask
  - ➢ Pandas
  - ➢ Sklearn
  - ➢ Flask_cors
  - ➢ Numpy
  - ➢ pip

- Node.js:
  - ➢ Axios
  - ➢ dompurify
  - ➢ Gmail-js
  - ➢ jquery
  - ➢ Npm

Plecând de la operația de dezarhivare asupra fișierului cu codul sursa efectuăm următorii pași:

1. Navigăm în directorul AntiPhisingExtension
2. Npm i
3. Npm run build
4. Deschidem Chrome -> Extensii ->Activăm Developer mode -> Load unpacked -> Încărcăm Extensia

Vom putea vedea extensia încărcată

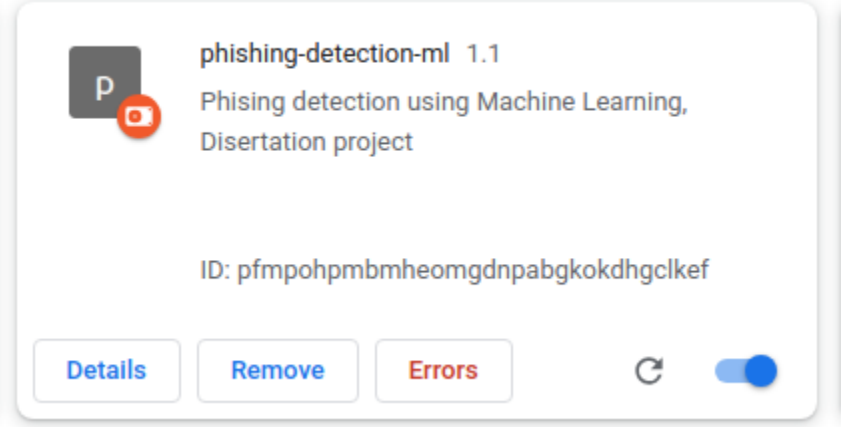


Figura 4.1. Extensia încărcată în browser-ul Chrome

5. Navigăm în directorul PhisingApiEndpoint
6. Pip install requrements.txt
7. Python app.py

Odată executați acești pași putem naviga în clientul de email și putem observa cum fiecare email selectat este analizat.

4.2. Testarea backend

Testarea modulelor s-a făcut inițial în terminal cu comanda curl, aceasta permite trimiterea de cereri HTTP către servere web și primirea răspunsurilor acestora. Prin intermediul comenzii curl, am putut testa și valida funcționalitatea endpoint-urilor noastre înainte de a le integra în extensie.

```
ubuntu@ubuntu1804:~/Public/PhisingApiEndpoint$ curl -X POST -H "Content-Type: application/json" -d '["example.com", "google.com"]'
http://localhost:5000/urlpredict
{
  "prediction": 1
}
```

Figura 4.2. Comanda curl

4.3. Testarea frontend

Testarea frontend-ului extensiei s-a realizat pentru a verifica și valida funcționalitatea și integrarea corectă a componentelor vizuale în mediul clientului de email. Aceasta a implicat testarea interacțiunii cu extensia în cadrul browserului, asigurându-ne că toate funcțiile și caracteristicile frontend funcționează conform așteptărilor.

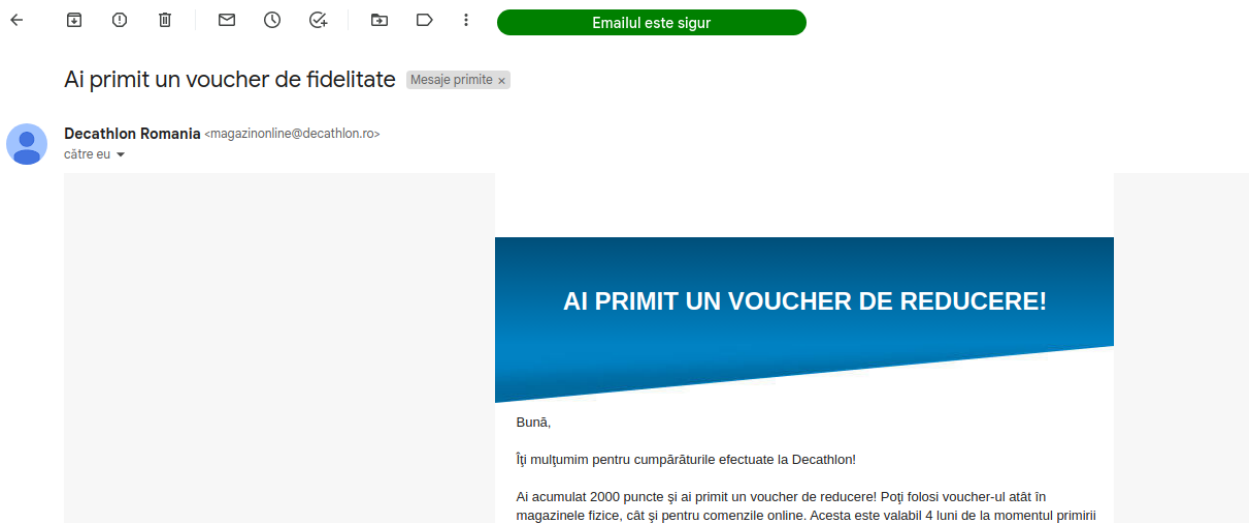


Figura 4.3. Afișarea label-ului valid în pagină

Figura 4.4. Afișarea label-ului periculos în pagină

Concluzii

În aceasta lucrare am prezentat Analiza automată a emailurilor pentru identificarea phishing-ului prin intermediul Machine Learning prin crearea unei aplicații de tip extensie browser.

Aplicația antiphishing s-a dovedit utilă în majoritatea cazurilor testate. A reușit să detecteze și să clasifice în mod eficient emailuri și URL-uri periculoase, contribuind la protecția utilizatorilor împotriva phishing-ului.

Una dintre principalele limitări întâmpinate a fost disponibilitatea unui set limitat de date în limba română. Acest lucru a putut afecta performanța extensiei în detectarea phishing-ului specific limbii române. Extinderea setului de date cu mai multe exemple relevante în limba română ar putea îmbunătăți capacitatea extensiei de a detecta și clasifica conținuturile periculoase specifice. Am identificat zone de optimizare și funcționalități suplimentare care ar putea fi adăugate pentru a spori eficacitatea și utilizabilitatea extensiei.

Tehnicile de învățare automată și-au arătat cu siguranță potențialul de transformare într-o varietate de domenii și aplicații. Cu toate acestea, este important să se ia în considerare și limitările tehnologiilor de învățare automată. Acestea pot include nevoia de seturi de date suficient de mari și diverse pentru a obține rezultate precise, riscul de erori și fals-pozitive, și dependența de calitatea și reprezentativitatea datelor de antrenament.

Bibliografie

[1] M. Khonji, Y. Iraqi, and A. Jones, “Phishing Detection: A Literature Survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013, doi: 10.1109/surv.2013.032213.00009.

[2] J. Camenisch, J. Lopez, F. Massacci, M. Ciscato, and T. Skordas, “JCS special issue on EU-funded ICT research on Trust and Security,” *Journal of Computer Security*, vol. 18, no. 1, pp. 1–5, Jan. 2010, doi: 10.3233/jcs-2010-0375.

[3] M. N. Banu and S. M. Banu, “A comprehensive study of phishing attacks,” *International Journal of Computer Science and Information Technologies*, vol. 4, no. 6, pp. 783–786, 2013.

[4] V. Bhavsar, A. Kadlak, and S. Sharma, “Study on phishing attacks,” *Int. J. Comput. Appl*, vol. 182, pp. 27–29, 2018.

[5] R. G. Brody, E. Mulig, and V. Kimball, “PHISHING, PHARMING AND IDENTITY THEFT.,” *Academy of Accounting & Financial Studies Journal*, vol. 11, no. 3, 2007.

[6] R. E. Schapire and Y. Freund, “Foundations of Machine Learning,” in *Boosting: Foundations and Algorithms*, 2012, pp. 23–52.

[7] P. Langley, *Elements of Machine Learning*. Morgan Kaufmann, 1996.

[8] T. Oladipupo Ayodele, “Types of Machine Learning Algorithms,” in *New Advances in Machine Learning*, Y. Zhang, Ed. Rijeka: IntechOpen, 2010. [Online]. Available: https://doi.org/10.5772/9385

[9] G. Shmueli, P. C. Bruce, P. Gedeck, and N. R. Patel, *Data Mining for Business Analytics: Concepts, Techniques and Applications in Python*. John Wiley & Sons, 2019.

[10] F. Leon, *Inteligenta artificiala: rationament probabilistic, tehnici de clasificare*. 2012.

[11] J. Gou, L. Du, Y. Zhang, and T. Xiong, “A New Distance-weighted k -nearest Neighbor Classifier,” *J. Inf. Comput. Sci.*, vol. 9, Nov. 2011.

[12] D. Vecliuc, C. Artene, M.-N. Tibeică, and F. Leon, “An Experimental Study of Machine Learning for Phishing Detection.pdf,” May 2023.

[13] “HTML.” *Wikipedia*. N.p., 29 June 2023. Web. 17 June 2023.

[14] “CSS.” *Wikipedia*. N.p., 6 July 2023. Web. 17 June 2023.

[15] “JavaScript.” *Wikipedia*. N.p., n.d. Web. 17 June 2023.

[16] Node.js. “Node.Js.” *Node.Js*, https://nodejs.org/en/. Accessed 17 June 2023.

[17] “pandas,” *Python Data Analysis Library*. https://pandas.pydata.org/ (accessed Jun. 17, 2023).

[18] “Getting Started,” *Axios Docs*. https://axios-http.com/docs/intro (accessed Jun. 17, 2023).
[19] “npm Docs.” https://docs.npmjs.com/ (accessed Jun. 17, 2023).
[20] “NumPy documentation — NumPy v1.24 Manual.” https://numpy.org/doc/stable/ (accessed Jun. 17, 2023).
[21] J. F.- js.foundation, “jQuery.” https://jquery.com/ (accessed Jun. 17, 2023).
[22] “scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation.” https://scikit-learn.org/stable/ (accessed Jun. 17, 2023).
[23] M. Siddhartha, “Malicious URLs dataset,” *Kaggle*. https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset (accessed Jun. 17, 2023).

[24] “Usage Statistics and Market Share of PHP for Websites, June 2023.” https://w3techs.com/technologies/details/pl-php (accessed Jun. 17, 2023).
[25] I. Nicolescu, “Care sunt cele mai folosite cuvinte în limba română,” *Adevărul*, Feb. 12, 2015. https://adevarul.ro/stiri-interne/educatie/care-sunt-cele-mai-folosite-cuvinte-in-limba-1600077.html (accessed Jun. 17, 2023).
[26] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, “PhishNet: Predictive Blacklisting to Detect Phishing Attacks,” in *2010 Proceedings IEEE INFOCOM*, Mar. 2010. Accessed: Jun. 17, 2023. [Online]. Available: http://dx.doi.org/10.1109/infcom.2010.5462216

Anexe

extensionInjector.js

```
"use strict";

function addScript(src) {
const script = document.createElement("script");
script.type = "text/javascript";
script.src = chrome.runtime.getURL(src);
(document.body || document.head ||
document.documentElement).appendChild(script);
}

addScript("dist/gmailJsLoader.js");
addScript("dist/extension.js");
```

gmailJsLoader.js

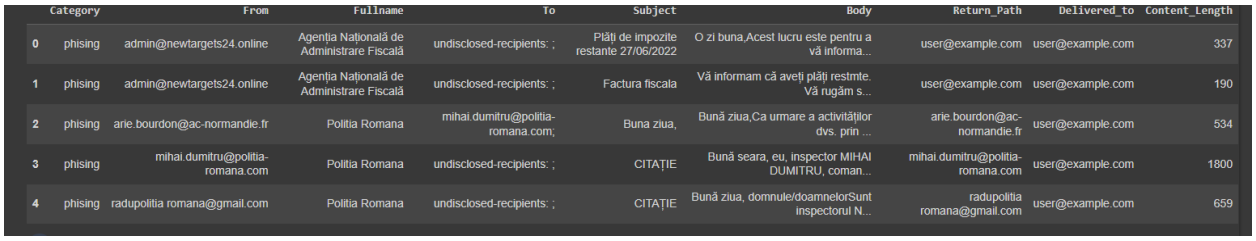
```
"use strict";

// gmail.js needs to be loaded as early as possible to be able to

const GmailFactory = require("gmail-js");
const jQuery = require("jquery");


window._gmailjs = window._gmailjs || new GmailFactory.Gmail(jQuery);
```

Set date email:



	Category	From	Fullname	To	Subject	Body	Return_Path	Delivered_to	Content_Length
0	phising	admin@newtargets24.online	Agenția Națională de Administrare Fiscală	undisclosed-recipients: ;	Plăți de impozite restante 27/06/2022	O zi buna,Acest lucru este pentru a vă informa...	user@example.com	user@example.com	337
1	phising	admin@newtargets24.online	Agenția Națională de Administrare Fiscală	undisclosed-recipients: ;	Factura fiscala	Vă informam că aveți plăți restmte. Vă rugăm s...	user@example.com	user@example.com	190
2	phising	arie.bourdon@ac-normandie.fr	Politia Romana	mihai.dumitru@politia-romana.com;	Buna ziua,	Bună ziua,Ca urmare a activităților dvs. prin ...	arie.bourdon@ac-normandie.fr	user@example.com	534
3	phising	mihai.dumitru@politia-romana.com	Politia Romana	undisclosed-recipients: ;	CITAȚIE	Bună seara, eu, inspector MIHAI DUMITRU, coman...	mihai.dumitru@politia-romana.com	user@example.com	1800
4	phising	radupolitia romana@gmail.com	Politia Romana	undisclosed-recipients: ;	CITAȚIE	Bună ziua, domnule/doamnelorSunt inspectorul N...	radupolitia romana@gmail.com	user@example.com	659