

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAŞI
FACULTATEA DE INFORMATICĂ



Lucrare de licență

Coordonator științific: Dr. Bogdan Pătruț

Coordonator științific: Dr. Anca Ignat

Absolvent: Luca Gheorghe-Vlăduț

Iași, 2025

Machine Learning Assistant

Cuprins

Introducere

- Contextul și motivația proiectului
- Scopul aplicației
- Structura lucrării

Notiuni teoretice

- Definirea învățării automate
- Tipuri de învățare
- Prezentare generală a algoritmilor selectați

Descrierea generală a aplicației

- Scopul și funcționalitatea principală
- Fluxul logic pentru utilizator
- Exemplificare scenarii de utilizare

Tehnologii utilizate

- Python, Flask
- Matplotlib
- HTML, CSS
- Alte unelte relevante

Arhitectura aplicației

- Structura folderelor și fișierelor
- Rutarea Flask
- Explicația modului MVC sau similar
- Gestionarea template-urilor și interacțiunea cu backend-ul

Prezentarea algoritmilor integrați

- Pentru fiecare algoritm:
 - Prezentare teoretică
 - Implementare
 - Cum este expus în aplicație

Interfața cu utilizatorul

- Prezentarea UI
- Explicarea funcționalităților interactive

Testare și validare

- Modul de testare a aplicației
- Testarea rezultatelor algoritmilor

Concluzii și direcții viitoare

- Ce ai învățat din proiect

Ce poate fi îmbunătățit
Propuneri pentru extindere

Bibliografie

Cărți, cursuri, articole online, tutoriale

1. Introducere

• Contextul și motivația proiectului

Acest proiect a fost realizat din dorința de a explica modul de funcționare al algoritmilor de Machine Learning persoanelor (în special studentilor) care doresc să învețe despre acest domeniu. Totul a început în prima săptămână de facultate din anul 3 când am avut primul contact cu acest domeniu, neștiind nimic de el până atunci. Pe tot parcursul semestrului cât am studiat această materie am întâlnit tot felul de dificultăți în înțelegerea anumitor algoritmi și a modului lor de rulare, doar că nu doar eu, ci și mai mulți dintre colegii mei. Din acest motiv, m-am hotărât să fac o listă cu lucrurile care ne-au pus pe mine și colegii mei în dificultate, anumite lucruri care nu se deduc ușor din notări și formule și să le explic prin intermediul unei aplicații web care pe baza unui set de date oferit de utilizator, trece prin toți pașii algoritmului oferind informații și explicații. În afară de explicații, mai sunt prezentate tabele și grafice care au rolul să facă utilizatorul să înțeleagă mai ușor evoluția de la o iteratăie la alta.

• Scopul aplicatiei

Scopul acestei aplicații este de a ajuta studenții să înțeleagă mai ușor cum funcționează algoritmi de Machine Learning prin explicarea fiecărui pas din algoritmi, explicarea termenilor cheie într-un mod informal și mai ușor de înțeles și înțelegerea mai ușoară prin utilizarea de grafice. Pentru cei cărora le place să facă asocieri în scopul de a reține sau învăța mai ușor, au fost adăugate în secțiunea de noțiuni teoretice a fiecărui algoritm câteva versete biblice care au strânsă legătură cu modul în care algoritmi funcționează. Pentru a face mai plăcută interacțiunea utilizator-aplicatie, paginile își pot schimba tema și culoarea, atât pentru a oferi un confort ochilor (tema luminoasă și culori deschise pe timp de zi, respectiv temă întunecată și culori închise pe timp de noapte) cât și pentru a atrage studenții pe această aplicație având posibilitatea să încerce ceva nou de fiecare dată.

• Structura lucrării

În această lucrare vor fi prezentate aspecte legate de aplicația web precum limbajele de programare utilizate, modul în care a fost gândită această aplicație, modul în care fiecare algoritm a fost explicat și cum au fost prelucrate datele, noțiuni teoretice despre fiecare algoritm, imagini cu interfață grafică și bucăți de cod importante. Vor fi prezentate explicații cât mai multe despre cod și modul de prelucrare al datelor, acestea fiind însotite și de imagini.

2. Noțiuni teoretice

• Definirea învățării automate

Învățarea automată este un subdomeniu al informaticii care se ocupă cu prelucrarea datelor în scopul de a ”învăța” din ele și pentru a lua decizii în situații asemănătoare sau diferite prin analiza unor seturi mari de date și găsirea unui model în acestea. Odată ce modelele de învățare automată au acces la cât mai multe date și sunt folosite din ce în ce mai des, acestea devin din ce în ce mai precise, având o acuratețe foarte mare în răspunsul oferit.

• Tipuri de invatare

Invatare supervizată: în acest tip de invatare datele sunt etichetate, acestea fiind reprezentate de perechi input-output unde cunoaștem valoarea output-ului, adică eticheta. Algoritmii care folosesc acest tip de invatare sunt regresia liniară, regresia logistică, K-NN, SVM, retelele neuronale. În viața de zi cu zi, invatarea supervizată este folosită în clasificarea e-mailurilor ca fiind spam sau non-spam, recunoasterea vocală sau detectarea de boli.

Invatare nesupervizată: în acest tip de invatare datele nu sunt etichetate, acestea neavând o valoare la output. Scopul acestui tip de invatare este de a se lega de toate datele oferite și de a încerca să creeze modele. Algoritmii care folosesc invatarea nesupervizată sunt K-Means și cei de clusterizare ierarhica. În viața de zi cu zi, invatarea nesupervizată este utilizată în securitatea cibernetică, recunoasterea facială sau analiza social media.

Invatare semi-supervizată: în acest tip de invatare sunt combinate cantități mici de date etichetate cu o cantitate mare de date neetichetate, scopul fiind de a da un punct de start în găsirea unui model sau în găsirea mai usoara a acestuia. Printre algoritmii care folosesc acest tip de invatare enumerăm S3VM și GMMs. În practică, acest tip de invatare este folosită în analiza medicală, recunoasterea scrisului de mâna sau clasificarea de imagini.

Invatare prin reîntârziere(Reinforcement Learning): în acest tip de invatare este vorba de un agent care interacționează cu o situație și primește o recompensă sau o penalizare în funcție de alegerile pe care le face, ducând la formarea unei strategii cat mai buna. Algoritmii care folosesc acest tip de invatare sunt Q-Learning și Deep Q Networks. În practică, invatarea prin reîntârziere este folosită în găsirea celor mai bune strategii pentru jocuri precum sah, Go sau table.

- Prezentare generala a algoritmilor selectati

Algoritmul AdaBoost

AdaBoost(Adaptive Boosting) este un algoritm de invatare supervizat care are rolul de a clasiifica datele combinand mai multe modele slabe(ex: compasi de decizie) in unul puternic. Acest algoritm a fost propus de Yoav Freund si Robert Schapire in anul 1995.

Modul de functionare este unul nici prea simplu dar nici prea complicat. Se incepe cu initializarea ponderilor. Fiecare punct primeste o pondere(greutate) care este echivalenta cu 1 supra numarul total de puncte. Aceasta asignare este valida doar in prima iteratie a algoritmului. Urmatorul pas este cel de stabilire a granitelor. Rolul granitelor este de a arata unde sunt puncte diferite pe grafic(daca un punct etichetat pozitiv este urmat de un punct etichetat negativ sau invers, atunci aici este un loc unde este necesara trasarea unei granite).

Aceste granite se traseaza la jumatatea distantei dintre doua cele mai apropiate puncte cu etichete diferite. Un lucru important la acest algoritm este trasarea unei granite sau a unor granite(depinde de spatiul numerelor), fara de care algoritmul ar functia gresit si ar oferi rezultate eronate. Aceasta granita se pune ori dupa ultimul punct de pe grafic, ori inainte de primul punct, fiind o mica distanta intre cele doua. Urmatorul pas este cel de calculare al erorilor pentru fiecare punct de pe grafic in raport cu granitele de decizie si determinarea erorii minime. Granitele de decizie sunt compasi de decizie care ajuta modelul sa clasifice datele.

La fiecare iteratie a algoritmului este ales cate un nou compas. Gasirea erorii minime determina si compasul de decizie la iteratia respectiva. Odata ce compasul de decizie a fost selectat, se pregatesc ponderile pentru iteratia urmatoare. Astfel, punctele clasificate corect primesc o pondere mare iar punctele clasificate gresit primesc o pondere mai mica. Deci, la o iteratie urmatoare, punctele clasificate gresit vor fi luate in evidenta avand eroarea cea mai mica. Se observa faptul ca doar in pasul initial punctele primesc pondere echivalenta cu 1 supra numarul total de puncte iar in iteratiile urmatoare ponderile au legatura cu punctele clasificate gresit si corect.

Acesti pasi se repeta pana cand toate punctele sunt clasificate corect sau pana cand s-a atins numarul de iteratii stabilite la inceput, acestea doua reprezentand doua criterii importante in oprirea rularii algoritmului.

Printre avantajele acestui algoritm se enumera gestionarea datelor zgomotoase si a outlier-elor si oferirea de rezultate cu o acuratete foarte mare. Ca si dezavantaje, algoritmul poate conduce la overfitting daca clasificatorii slabii sunt prea complexi, timp costisitor in cazul in care setul de date este unul destul de mare si sensibilitate la date zgomotoase si outlieri care nu sunt preprocesate.

In practica, acest algoritm este utilizat in recunoasterea faciala, determinarea daca un e-mail poate fi incadrat ca fiind spam sau non-spam sau clasificarea de date si imagini.

Acest algoritm a fost dezvoltat si imbunatatit in cursul anilor, astfel aparand mai multe versiuni si variante precum Discrete AdaBoost, Real AdaBoost, LogitBoost sau Gentle AdaBoost.

Algoritmul ID3

Algoritmul ID3(Iterative Dichotomiser 3) este a treia versiune a algoritmilor creati de J. Ross Quinlan in anul 1980. Acesta este un algoritm de clasificare a datelor bazandu-se pe cosntructia de arbori de decizie.

Modul de functionare este unul nici prea simplu dar nici prea complicat. Este important ca datele sa fie cat mai bine structurate intr-un tabel, tabel care detine informatii despre un anumit obiect sau lucru indiferent de mediul acestuia de provenienta, informatii numite caracteristici si valori. Unul dintre cei mai importanti termeni care are legatura cu acest algoritm este cel de entropie care semnifica nivelul mediu de incertitudine sau a impuritatii intr-un set de date. O valoare mare a entropiei inseamna un set de date mai amestecat.

Primul pas in algoritm este cel de calculare a entropiei mari, adica entropia variabilei tinta. Dupa urmeaza calcularea entropiei pentru fiecare atribut din tabel, iar pentru fiecare atribut se calculeaza entropiile tinand cont si de atribute. Odata ce toate entropiile sunt calculate, pentru fiecare atribut in parte se calculeaza castigul de informatie ca fiind diferența dintre entropia mare si entropiile atributelor in functie de variabile inmultite cu un raport dintre numarul de variabile corespunzatoare atributului si cardinalul atributului. Dupa ce toate castigurile de informatie au fost calculate, se compara pentru a se determina care atribut are castigul de informatie cel mai mare. Atributul respectiv este ales ca radacina a arborelui de decizie.

La pasul urmator, atributul ales la pasul precedent nu mai este luat in considerare si se aplica acelasi procedeu(calculare entropii, calculare castig de informatie, determinarea atributului cu castig de informatie maxim) pana cand tabelul ramane fara atribute si arborele este construit complet. Nodurile arborelui contin in interior numele atributelor din tabel si sunt legate prin arce pe care sunt notate atributele nodului parinte. In nodurile frunza sunt trecute valorile atributelor, valori care reprezinta raspunsul algoritmului in clasificarea altor date. Odata ce arborele este creat, se pot clasifica si alte date. Este necesar sa se inceapa de la radacina arborelui si sa se parcurga arborele tinand cont de ce atribute si valori are noua instanta de clasificat. Parcursarea are loc pana cand se ajunge intr-un nod frunza, moment in care algoritmul se opreste si se ia decizia in privinta raspunsului oferit.

Acest algoritm are avantaje precum construirea unui arbore simplu, scurt si rapid si verificarea intreg setului de date pentru a construi arborele de decizie. Printre dezavantaje se enumera gestionarea doar datelor categoriale, prioritizarea atributelor cu mai multe valori sau negostionarea seturilor de date nebalansate si a valorilor lipsa. In practica, acest algoritm este utilizat in luarea de decizii in domeniul medical, clasificarea e-mailurilor ca fiind spam sau non-spam sau a sistemelor de recomandare.

Algoritmul ID3 are mai multe versiuni precum C4.5, C5.0, CART, ID4, ID5, Oblique Decision Trees, Random Forest, Chi-Squared Automatic Interaction Detector si Cost-Sensitive ID3.

Algoritmul k-NN

Algoritmul k-NN este un algoritm de invatare supervizat, non-parametric, care face clasificare pe baza celor mai apropiate instante. A fost dezvoltat de Evelyn Fix si Joseph Hodges in anul 1951, urmand ca mai tarziu sa fie explicat de Thomas Cover. Acest algoritm poate fi generalizat si pentru regresie.

Modul de functionare al acestui algoritm este unul foarte simplu. Datele sunt reprezentate sub forma de puncte intr-un grafic impreuna cu etichetele acestora. De obicei, punctele sunt reprezentate de o bulina neagra daca au eticheta negativa si bulina alba daca au eticheta pozitiva. Apoi se alege o metrica de distanta.

Alegerea acesteia este un pas important deoarece folosirea de metriki de distanta diferita schimba si acuratetea modelului. Cea mai folosita metrica este distanta Euclidiana care se calculeaza ca radical din suma patratelor diferenței dintre coordonatele corespunzatoare axei Ox , respectiv Oy . Alte doua metriki foarte folosite sunt distanta Manhattan si distanta Chebyshev. Alte metriki mai putin intalnite sunt distantele Minkowski, Cosinus Distance, Hamming, Mahalanobis, Jaccard, Levenshtein.

Odata ce metrica de distanta a fost aleasa, trebuie aleasa o valoare pentru variabila k . Aceasta variabila reprezinta cati vecini vor fi luati in calcul in determinarea modelului. Daca k este ales ca fiind 5, atunci eticheta punctului pe care vrem sa il clasificam va fi determinata de etichetele celor mai apropiati 5 vecini. In determinarea clasificarii punctului dorit, se vor face doua multimi, prima reprezentand multimea vecinilor din cei k cei mai apropiati vecini care au etichete pozitive si a doua multime formata din vecinii din cei k cei mai apropiati vecini care au etichete negative. Multimea cu cardinalul mai mare va da eticheta punctului care se doreste a fi clasificat. Spre exemplu, daca din cei 5 cei mai apropiati vecini ai unui punct pe care dorim sa il clasificam 3 vecini au eticheta pozitiva iar 2 au eticheta negativa, atunci punctul nostru va avea si el tot eticheta pozitiva deoarece multimea vecinilor cu eticheta pozitiva domina.

Este important ca atunci cand se alege valoarea pentru variabila k , aceasta sa reprezinte un numar impar. Daca variabila k ar fi ales un numar par, atunci este foarte posibil sa se intample cazul in care exact jumata din vecini sunt etichetati pozitiv si jumata sunt etichetati negativ, caz in care nu se poate lua o decizie asupra etichetei care i se potriveste punctului ce trebuie clasificat. Aceasta este o ambiguitate care trebuie evitata deoarece algoritmul este in imposibilitatea de a oferi un raspuns.

Printre avantajele de folosire ale acestui algoritm se enumera usurinta de inteleghere a acestuia, implementarea simpla, adaptabilitatea la noi date, eficienta pe seturi mici de date. Ca si dezavantaje, acest algoritm necesita multa memorie si sensibilitatea la caracteristici irelevante. In viata de zi cu zi, acest algoritm este folosit in diagnosticul medical, recunoasterea scrisului de mana, recunoastere faciala si imagini. Ca si versiuni ale acestui algoritm avem Weighted k-NN, Distance-weighted k-NN, Condensed k-NN, Edited k-NN sau Fuzzy k-NN.

Algoritmul K-Means

Algoritmul K-Means este un algoritm de invatare nesupervizata dezvoltat de Stuart Lloyd in anul 1955, fiind mai apoi dezvoltat si imbunatatit pe parcursul anilor. Acest algoritm are ca scop principal gruparea datelor(clustering).

Modul sau de functionare este moderat ca dificultate. Primul pas este cel de a se stabili cu ce valoare este initializata variabila k . Alegerea acestei valori este foarte importanta deoarece o valoare diferita poate influenta precizia si acuratetea modelului. Un k prea mic ar grupa multe date diferite impreuna iar un k prea mare ar grupa date care sunt foarte asemanatoare la un loc. Una dintre metodele de alegere a valorii pentru variabila k este metoda cotului(Elbow). Aceasta metoda presupune rularea algoritmului pe mai multe valori posibile ale lui k . Rezultatele sunt reprezentate intr-un grafic pe care se cauta un "cot", adica un punct unde scaderea erorii devine cat mai mica posibil. O alta metoda utilizata este Silhouette care functioneaza pe baza unui scor acordat punctelor.

Odata ce k este ales, trebuie alesi cei k centroizi. In acest caz, centroizii pot fi chiar si punctele din grafic sau pot fi puncte care nu se regasesc in grafic. Urmeaza alegerea metricii de distanta care este un pas important, folosirea unei metriki diferite poate schimba si acuratetea modelului. Cea mai folosita metrica este distanta Euclidiana care se calculeaza ca radical din suma patratelor diferenței dintre coordonatele corespunzatoare axei Ox , respectiv Oy . Alte doua metriki foarte folosite sunt distanta Manhattan si distanta Chebyshev. Alte metriki mai putin intalnite sunt distantele Minkowski, Cosinus Distance, Hamming, Mahalanobis, Jaccard, Levenshtein. In cele ce urmeaza trebuie calculata distanta dintre fiecare punct si cei k centroizi. Se ia cate un punct pe rand, se calculeaza distanta dintre el si cei k centroizi, de aici rezultand k lungimi de distanta. Apoi se compara cele k lungimi urmand ca punctul sa fie asignat centroidului de care este cel mai apropiat. Dupa ce toate punctele au fost asignate unui centroid, urmeaza sa fie recalculate pozitiile centroizilor. Modul de a determina noua pozitie a fiecarui centroid este una destul de simpla. Se aduna toate coordonatele punctelor corespunzatoare clusterului in care au fost alocate si se imparte la numarul total de puncte din multime, astfel obtinandu-se noile coordonate ale centroidului. Cu alte cuvinte, centroidul este un centru de greutate. In acest punct se termina o iteratie a algoritmului.

Urmeaza ca algoritmul sa se repete, dar de data asta cu alte coordonate ale centroizilor. In functie de marimea setului de date, centroizii se pot schimba foarte frecvent sau aproape deloc. Daca vorbim de un dataset mic, undeva la 10-15 puncte, e posibil ca centroizii sa se schimbe doar intre una sau doua iteratii sau sa nu se schimbe deloc intre nicio iteratie. In cazul in care datasetul este foarte mare, centroizii isi vor schimba pozitiile intre multe iteratii, sansa ca centroizii sa fie fixati in pozitia potrivita din prima fiind foarte mica.

Algoritmul se opreste in doua situatii. Prima e atunci cand se atinge numarul de iteratii fixat la inceputul algoritmului iar a doua e atunci cand centroizii nu isi mai schimba pozitia intre doua iteratii succesive sau isi schimba pozitia dar foarte putin.

Acest algoritm are ca avantaje simplitatea implementarii lui, scalarea la seturi mari de date sau generalizarea la clustere de diferite forme si marimi precum clusterele eliptice. Ca si dezavantaje, k trebuie ales manual, dificultatea clusterizarii datelor de diferite marimi si densitati fara generalizare sau dificultatea scalarii cu numar de dimensiuni. Ca si versiuni ale acestui algoritm putem enumera K-Medoids, Lloyd's K-Means, MacQueen's K-Means, Hartigan-Wong K-Means, K-Means++ sau Kernel K-Means.

Algoritmul de Regresie Logistica

Algoritmul de Regresie Logistica este un algoritm de invatare supervizata dezvoltat in anul 1880 si integrat in machine learning dupa anii 1950, algoritm la care au contribuit Francis Galton, Karl Pearson si Ronald Fisher. Acest algoritm are ca scop prezicerea probabilitatii ca un element, obiect, sa apartina unei clase.

Modul de functionare al acestui algoritm poate parca putin mai dificil, acest lucru datorandu-se si de prezenta notiunilor si formulelor matematice. Acest algoritm foloseste o functie speciala, numita functia sigmoid. Aceasta este o functie non-liniara care are ca scop transformarea output-ului modelului de regresie logistica intr-o probabilitate. Aceasta functie este aleasa in detrimentul altor functii deoarece este stabila, mai usor de interpretat si mult mai comună.

Alta functie utilizata este functia de log-verosimilitate. Aceasta are rolul de a maximiza scorul. Aceasta functie trateaza fiecare instant din setul de date, comportandu-se diferit doar atunci cand output-ul difera. In aceasta formula intervine termenul de bias care este o valoare constanta(de obicei 1) care se aduna. Vectorul gradient, alta componenta importanta in aplicarea algoritmului, arata directia in care creste functia de log-verosimilitate. Acest vector are cardinalul egal cu numarul de instante din setul de date. Poate fi calculat ca fiind derivata partiala a functiei de log-verosimilitate in raport cu w. Matricea Hessiana este o matrice care contine derivatele de ordin doi si pe care se pot aplica alte metode de optimizare.

Odata ce functia de log-verosimilitate, vectorul gradient si matricea Hessiana au fost determinate, se pot face predictii pentru alte seturi noi de date. Pentru acest lucru, se initializeaza n ponderi cu 0, unde n reprezinta numarul de instante din setul de date, apoi folosindu-ne de valorile din vectorul gradient si una dintre metoda gradientului ascendent sau descendent, se poate aplica formula pentru a face predictia. Formula se aplica pana la convergenta. Valorile din noul vector obtinut dupa aplicarea formulei sunt inmultite cu valorile instantei ce se doreste a fi clasificata, apoi toate produse sunt adunate. Daca suma reprezinta un numar negativ, atunci algoritmul va produce output-ul 0, iar daca suma este reprezentata de un numar pozitiv, output-ul produs de regresia logistica va fi 1.

In practica mai este folosit si un alt tip de regresie, fiind vorba de cea liniara. Diferenta dintre cele doua este data de rezultatul pe care il ofera si contextul in care sunt folosite. Regresia liniara este folosita pentru pentru a prezice valori precum temperatura, pretul, inaltime, dimensiune, pe cand regresia logistica face o predictie ca un anumit obiect, eveniment sa se incadreze intr-o categorie.

Printre avantajele acestui algoritm avem usurinta de a fi implementat, intelese si interpretat, eficient in antrenare, poate fi usor extins la alte clase. Printre dezavantaje, problemele non-liniare nu pot fi rezolvate cu acest algoritm, este greu de a obtine legaturi complexe si poate fi folosit doar pentru a prezice functii discrete.

Algoritmul de regresie logistica are mai multe variante precum cea binara, multiclassa, penalizata sau cea cu interactiuni.

Algoritmii Bayes Naiv si Bayes Optimal

Algoritmii Bayes Naiv si Bayes Optimal sunt doi algoritmi de invatare automata dezvoltati dupa anul 1990, fiind algoritmi de clasificare care se bazeaza pe probabilitati in oferirea unui raspuns.

Modul de functionare al acestor algoritmi este unul destul de simplu deoarece se bazeaza pe calcularea unor probabilitati. In cazul algoritmului Bayes Naiv se foloseste Teorema lui Bayes si se face presupunerea ca toate atributele sunt independente intre ele, avand rolul de a simplifica cat mai mult si de a face calculul unul cat mai simplu. In schimb, in cazul algoritmului Bayes Optimal, acesta ia in calcul absolut toate probabilitatile pentru a oferi un rezultat cat mai precis. Din punct de vedere al corectitudinii si acuratetii, Bayes Optimal este cel mai bun. Dar din punct de vedere al complexitatii, Bayes Naiv este preferat in detrimentul acestuia.

In cazul celor doi algoritmi, odata ce probabilitatile au fost calculate, se poate incepe determinarea predictiei pentru o instanta ce se doreste a fi clasificata. Se calculeaza doua probabilitati, una cand noua instanta ar apartine clasei 0 si inca o probabilitate atunci cand noua instanta ar apartine clasei 1. Daca una dintre clase are o probabilitate mai mare decat cealalta, atunci aceasta clasa va oferi output-ul algoritmului, adica in ce clasa poate fi incadrata noua instanta. Cateodata, din cauza setului de date, se intampla ca ambele probabilitati sa fie egale. In acest caz algoritmul nu poate decide daca noua instanta apartine clasei 0 sau 1. Pentru a rezolva aceasta problema se foloseste o metoda numita Laplace. Aceasta metoda adauga constanta 1 la numarator si mai aduna o constanta la numitorul fractiei, constanta care reprezinta numarul total de valori posibile pentru acea variabila. Cu ajutorul acestei metode este evitat cazul in care avem doua probabilitati egale si algoritmul nu poate decide carei clase apartine instanta ce se vrea a fi clasificata. Tot cu aceasta metoda este evitat si cazul in care in calculul probabilitatilor se poate ajunge ca una dintre ele sa fie 0.

In cazul celor doi algoritmi difera doar modul in care sunt calculate probabilitatile. Dezavantajul major al algoritmului Bayes Naiv este ca de cele mai multe ori presupunerea de independenta conditionata este falsa, asta ducand la rezultate imprecise si o acuratete mai slaba decat a altor algoritmi de clasificare utilizati in machine learning. In schimb, acest algoritm compenseaza cu rapiditatea si usurinta calculelor. Algoritmul Bayes Optimal are ca dezavantaj spatiul de memorie pe care il foloseste deoarece ia in calcul toate probabilitatile, el neluand in calcul presupunerea independentei conditionate intre variabile. Cu toate ca ofera rezultate mult mai precise, in practica nu este preferat din cauza resurselor utilizate.

In viata de zi cu zi, doar algoritmul Bayes Naiv este folosit. Aceste este utilizat in clasificarea e-mailurilor ca fiind spam sau non-spam, diagnostic medical sau filtrarea comentariilor si textelor din social media. Algoritmul Bayes Optimal este dorit a fi utilizat doar in scop teoretic si educational.

Bayes Naiv a dezvoltat pe parcursul timpului mai multe variante/versiuni precum cel Gaussian, multinomial, Bernoulli sau cel categorial. Bayes Optimal are versiuni precum modelul averaging sau maximum a posteriori(clasificatorul MAP).

3. Descrierea generala a aplicatiei

• Scopul si functionalitatea principala

Scopul acestei aplicatii web este de a familiariza pe cei noi in domeniu cu conceptele de invatare automata si despre modul in care algoritmii utilizati in acest domeniu functioneaza. Explicarea algoritmilor este facuta pas cu pas, explicatiile fiind insotite de imagini, grafice, tabele sau asocieri biblice. Partea grafica are scopul de a atrage utilizatorul si pentru a-i ajuta pe cei care au o memorie vizuala sa retina unele lucruri mai usor.

Functionalitatea principala a acestei aplicatii este cea de prelucrare a datelor. Datele pot fi introduse intr-un tabel de catre utilizator in interfata grafica, acestea urmand sa fie trimise pe backend, loc in care sunt prelucrate si trimise inapoi catre interfata grafica unde sunt afisate intr-un mod cat mai placut.

• Fluxul logic pentru utilizator

Utilizatorul trebuie sa acceseze pagina principală a aplicatiei. In cest loc(Figura 1) ii este prezentat pe scurt ce inseamna termenul de invatare automata si ce va descoperi in acest site.

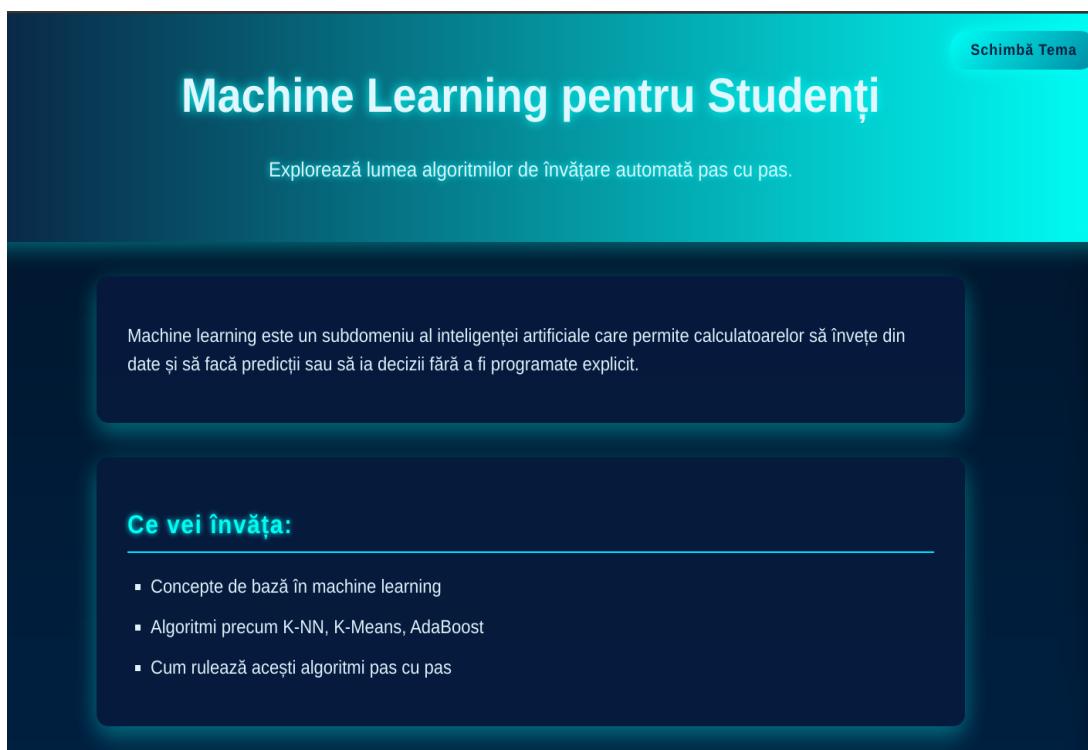


Figura 1: Pagina de start- informatii

In ultima sectiune a paginii(Figura 2) va gasi link-uri catre site-uri utile care il pot ajuta sa invete si mai multe lucruri despre acest domeniu, link-uri catre carti scrise de persoane importante care se occupa de acest domeniu sau link-uri catre site-uri cu seturi de date in caz ca utilizatorul doreste sa isi faca o idee despre ce este acela un set de date, cat de mare poate fie sau daca vrea sa aplice unul dintre algoritmii de machine learning pe un dataset.



Figura 2: Pagina de start- resurse utile

Inainte de aceasta sectiune se mai afla o sectiune cu algoritmii(Figura 3) pe care pagina web ii explica. Este nevoie doar ca utilizatorul sa selecteze ce algoritm doreste sa aprofundeze sau despre care sa invete cum ruleaza.



Figura 3: Pagina de start- algoritmi

Odata ce utilizatorul isi alege un algoritm, acesta va fi redirectionat catre pagina dedicata algoritmului(Figura 4). In partea de inceput a paginii, utilizatorul va gasi informatii despre modul in care algoritmul ruleaza, pasii algoritmului fiind explicati cat mai simplu pastrandu-se doar ideea principala fara a incarca pagina cu informatii care nu sunt necesare.

Algoritmul K-Means

K-Means este un algoritm de învățare nesupraveghetă folosit pentru gruparea (clustering) datelor. Scopul lui este să împartă datele în K grupuri (clustere), în funcție de cât de apropiate sunt între ele.

• Cum funcționează K-Means:

1. Alegem aleatoriu K centre (centroizi).
2. Fiecare punct este alocat celui mai apropiat centru.
3. Se recalculează centrul fiecărui grup ca media punctelor din acel grup.
4. Se repetă pașii 2–3 până când centrele nu se mai schimbă (convergență).

Figura 4: Pagina dedicată algoritmilor- modul de funcționare

Dupa aceasta secțiune, utilizatorul va gasi o nouă secțiune cu versete biblice(Figura 5). Scopul acestor versete biblice este de a face o legătură între sensul lor și modul de funcționare al algoritmilor. Aceasta metoda vine în ajutorul celor care vor să învețe despre ce fac algoritmii, ajutându-i pe cei care au o metodă vizuală sau cărora le plac asociările ca să retină mai ușor. Fiecare verset are o strânsă legătură cu ceea ce face algoritmul, acestea fiind apoi explicitate și prezentata legătura pe care o au.

• Versete biblice care reflectă principiile K-Means:

1. Matei 25:32

„Toate neamurile vor fi adunate înaintea Lui. El îi va despărți pe unii de alții, cum desparte păstorul oile de capre.”

Legătura cu K-Means:

Algoritmul K-Means face exact asta: separă datele în grupuri distincte în funcție de trăsăturile lor. Nu există confuzie — fiecare punct este separat clar, ca oile de capre.

2. Romani 12:4-5

„Căci, după cum într-un singur trup avem multe mădule, iar mădulele nu au toate aceeași funcție, tot așa și noi, deși suntem mulți, suntem un singur trup în Hristos și, fiecare în parte, mădule unui altora.”

Legătura cu K-Means:

Acest verset reflectă ideea de specializare și colaborare în cadrul unui grup, similar cu modul în care punctele dintr-un cluster contribuie la definirea centrului comun în K-Means.

3. Psalmul 133:1

„Iată, ce bine și ce plăcut este ca frații să locuiască împreună în unitate!”

Legătura cu K-Means:

Versetul subliniază frumusețea unității, analog cu modul în care K-Means grupează puncte similare, creând clustere armonioase.

4. Coloseni 3:14

„Si mai presus de toate acestea, îmbrăcați-vă cu dragostea, care leagă totul într-o armonie perfectă.”

Legătura cu K-Means:

Dragostea ca lant reflectă modul în care centrul unui cluster în K-Means acționează ca punct de coeziune pentru toate punctele din grup.

5. Petru 3:8

„În sfârșit, fiți toți cu aceleași gânduri, simțind cu alții, iubind ca frații, miloși, smeriți.”

Legătura cu K-Means:

Acest verset subliniază empatia și armonia, reflectând modul în care K-Means grupează puncte similare pentru a forma clustere omogene.

Figura 5: Pagina dedicată algoritmilor- versete biblice

In penultima secțiune a paginii(Figura 6) sunt prezentati termeni si cuvinte cheie care au legătura cu algoritmul respectiv. Acești termeni sunt explicitați într-un mod informal, fără cuvinte complicate pentru a facilita înțelegerea mai usoara a utilizatorului. Pe parcursul rularii algoritmului respectiv acești termeni vor fi utilizati destul de des, motiv pentru care studierea acestei pagini este esențială.

Termeni utilizati:

- 1.**k**- Numărul de clustere dorite. Trebuie specificat înainte de a rula algoritmul k-means.
- 2.**Centroid**- Punctul central al unui cluster – este media tuturor punctelor din acel cluster.
- 3.**Cluster**- Grup de puncte similare, apropiate în spațiul caracteristicilor, care împărtășesc un centroid comun.
- 4.**Initializare aleatoare**- Centroidele sunt alese aleatoriu la început. Poate afecta negativ rezultatul dacă inițializarea este slabă.
- 5.**K-Means++**- Metodă inteligentă de inițializare a centroidelor, care reduce riscul de a ajunge la un minim local slab.
- 6.**Convergența**- Algoritmul k-means oprește execuția când centroizii nu se mai schimbă sau modificările sunt sub un prag dat.
- 7.**Invatare nesupervizata**- Algoritmul k-means oprește execuția când centroidele nu se mai schimbă sau modificările sunt sub un prag dat.
- 8.**K-partiție**- Împărțirea setului de date în k clustere disjuncte.
- 9.**K-configuratie**- Setul centroids-urilor care definesc o anumită partiție.
- 10.**Criteriul J**-Funcția obiectiv a algoritmului k-means: suma pătratelor distanțelor dintre fiecare punct și centroidul său.
- 11.**Criteriul Elbow**- Ajută la alegerea optimă a lui k analizând grafic scăderea erorii (inertia) în funcție de k. Punctul în care această scădere încetinește brusc (cotul) indică valoarea potrivită pentru k.

Figura 6: Pagina dedicata algoritmilor- termeni si cuvinte cheie

Odata ce aceste sectiuni au fost parcurse, utilizatorul ajunge la ultima sectiunii a paginii(Figura 7), cea care ii permite sa navigheze intre pagini. Aici are posibilitatea de a se reinvoarce la pagina principala, avand butonul ”Go Back Home”. Celalalt buton, ”exemplu practic”, conduce utilizatorul catre o pagina noua, pagina unde poate introduce ce date doreste. Scopul algoritmilor este de a primi un set de date si o instantă, iar pe baza modelului format după acel set de date sa faca o clasificare sau o predictie pentru noua instantă.

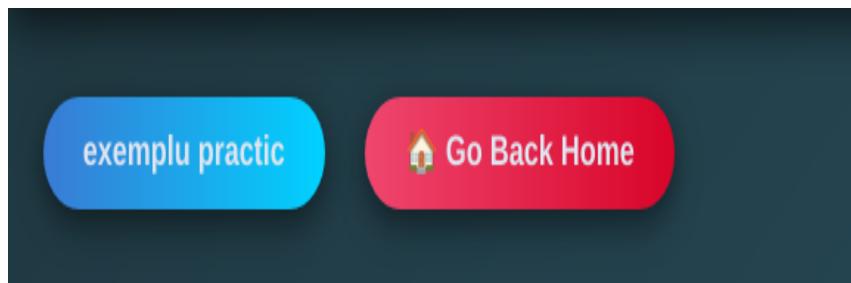


Figura 7: Pagina dedicata algoritmilor- navigarea intre pagini

Din moment ce utilizatorul apasă butonul ”exemplu practic” va fi redirectionat către o pagina (Figura 8) unde va trebui să completeze informații despre cat de multe instante(linii) dorește să aibă tabelul cu date sau cu cate atribute(coloane). De preferat ar fi să se completeze cu un set de date luat de pe un site care oferă aceste tipuri de dataseturi. Dacă este completat cu un dataset random cu valori total la întâmplare atunci și modelul creat și predictia vor avea un comportament total neașteptat.

Figura 8: Pagina completare informatii- dimensiuni tabel, numar de puncte

Dupa ce utilizatorul a completat informatiile despre ce dimensiune vrea sa aiba tabelul sau (numar de instante, numar de coloane, numar de centroizi etc.) va fi redirectionat catre o noua pagina (Figura 9), pagina unde va fi necesara introducerea datelor. Aici se vor introduce datele, coordonata cu coordonata, apoi daca este cazul se vor introduce si datele instantei ce se doreste a fi clasificata. Abia dupa ce datele au fost introduse, utilizatorul poate vizualiza modul de functionare al algoritmului.

Figura 9: Pagina completare informatii- coordonate puncte, coordonate instanta ce urmeaza sa fie clasificata

Abia dupa ce toate datele au fost introduse, utilizatorul va fi redirectionat catre o pagina unde algoritmul respectiv este aplicat pe toate liniile din tabel, oferind explicatii, grafice, tabele si rezultatul final, clasificarea instantei dorite.

- **Exemplificare scenarii de utilizare**

Aplicatia este destinata tuturor persoanelor care vor sa invete cum ruleaza si functioneaza algoritmi de Machine Learning dar nu stiu de unde sa inceapa si ce material sa foloseasca. Ca si scenarii de utilizare, aplicatia poate fi folosita de studentii de anul 2 care vor sa ia materia din timp ca pe parcursul anului 3 sa aiba mai mult timp liber, studentii de anul 3 carora li se par prea complicate notatiile si explicatiile din cartile de specialitate, studentii de master care doresc sa isi reaminteasca notiunile folosite, profesori care vor sa imbine predatul la tabla cu lumea digitala sau persoanele noi in domeniul informaticii care doresc sa dobandeasca informatii si cunostinte in acest subdomeniu.

4. Tehnologii utilizate

• Python, Flask

Aceasta aplicatie web a putut fi posibila cu ajutorul framework-ului Flask, un micro-framework destinat realizarii de aplicatii web. Este caracterizat prin usurinta sa de a crea pagini, oferind toate lucrurile esentiale in dezvoltarea unei pagini. Spre deosebire de alte framework-uri, acesta ne ofera strict doar lucrurile de care avem nevoie, pastrand simplitatea. Acest framework permite definirea de rute cu decoratori care mapeaza URL-urile in functii. Vine impreuna cu libraria Jinja2, o librerie ce permite combinarea codului HTML cu sechete de cod asemanatoare cu o combinatie intre limbajele Python si Bash, transformand pagina intr-o cat mai dinamica. Flask vine impreuna si cu alte librarii, "render_template" si "request" (Figura 10), prima librerie avand rol de a punce in lasare o pagina HTML sau de a trimite anumiti parametri catre acea pagina care mai apoi sa fie prelucratie, iar a doua librarie are rolul de a prelua datele din formulele HTML de pe frontend. Acest limbaj a fost ales deoarece ofera librarii precum cele de creare a graficelor sau pentru usurinta cu care datele sunt memorate si prelucrate pe backend.

```
62
63     @app.route("/kNN_Algorithm")
64     def kNN_Algorithm():
65         return render_template("kNN_Algorithm.html")
66
67     @app.route(rule: "/kNN_AlgorithmExample", methods=["GET", "POST"])
68     def kNN_AlgorithmExample():
69         elements = []
70         n = None
71         m = 3
72         x1=None
73         y1=None
74         if request.method == "POST":
75             n = int(request.form["n"]) if "n" in request.form else None
76             x1=int(request.form["x1"]) if "x1" in request.form else None
77             y1=int(request.form["y1"]) if "y1" in request.form else None
78
```

Figura 10: Cod Python- Flask, render_template, request

• Matplotlib

Matplotlib este o librerie din limbajul de programare Python, librerie cu ajutorul careia se pot crea grafice(Figura 11). Pe aceste grafice se pot adauga puncte de diferite dimensiuni si culori, se pot trage linii atat drepte cat si punctate, se pot trage mediane, mediatoare si multe alte lucruri. In acest proiect s-a optat pentru folosirea acestei librarii in scopul de a crea grafice care sa faca faza rularii algoritmului pas cu pas mult mai usoara si pentru a vizualiza datele intr-un plan, situatie in care utilizatorului ii este creata o perspectiva. In cod, sunt create diferite

functii de creare a graficelor, functii care primesc parametrii necesari precum lista coordonatelor punctelor sau punctul unde este necesara trasarea unei linii orizontale sau verticale, urmand ca la finalul functiei imaginea creata sa fie returnata sub o forma comprimata. Aceasta imagine este apoi trimisa de pe backend pe frontend, acolo unde este afisata in pagina cu ajutorul HTML si Jinja2.

```

850     def grafic_adaboost_cu_praguri(puncte, etichete, pragurix, praguriy): 1 usage
851         from io import BytesIO
852         fig, ax = plt.subplots()
853         for (x, y), et in zip(puncte, etichete):
854             if et == 1:
855                 ax.plot(*args: x, y, 'ko') # 'k' = negru, 'o' = punct
856             else:
857                 ax.plot(*args: x, y, 'wo', markeredgecolor='black') # punct gol cu contur negru
858
859             for x in pragurix:
860                 ax.axvline(x=x, color='green', linestyle='--')
861             for y in praguriy:
862                 ax.axhline(y=y, color='red', linestyle='--')
863             ax.set_facecolor('white')
864             ax.grid(True, linestyle=':', alpha=0.5)
865
866             buffer = BytesIO()
867             plt.savefig(*args: buffer, format='png', bbox_inches='tight')
868             buffer.seek(0)
869             img_base64 = base64.b64encode(buffer.read()).decode('utf-8')
870             buffer.close()
871             plt.close(fig)
872
873             return img_base64

```

Figura 11: Cod Python- Matplotlib

• HTML, CSS

Fiind o aplicatie web, este evidenta folosirea limbajelor de programare HTML(HyperText Markup Language) si CSS(Cascading Style Sheets). HTML(Figura 12) este cel care afiseaza cuvintele si formulele pe pagina iar CSS(Figura 13) este cel care ofera culoare, dimensiune, pozitionare a tuturor elementelor, animatii si se ocupa ca un site sa fie responsive.



```

1  <!doctype html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1">
6          <title>Logistic Regression</title>
7          <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-mml-chtml.js"></script>
8          <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.7/MathJax.js?config=TeX-MML-AM_CHTF">
9              <link rel="stylesheet" href="{{ url_for('static', filename='AlgorithmExampleInterface.css') }}"/>
10
11      </head>
12      <body>
13          <h1>Introduceti doua numere n si m</h1>
14          <form method="POST">
15              <label for="n">n:</label>
16              <input type="number" name="n" id="n" required><br><br>
17
18              <label for="m">m:</label>
19              <input type="number" name="m" id="m" required><br><br>
20
21              <button type="submit">Generare elemente</button>
22          </form>
23
24      {% if n and m %}
25          <h2>Introduceti {{ n }} linii si {{ m }} coloane pentru x (x1, x2, ..., xm) si y</h2>
26          <form method="POST">
27              <input type="hidden" name="n" value="{{ n }}"/>
28              <input type="hidden" name="m" value="{{ m }}"/>
29
30              <table border="1">
31                  <thead>

```

Figura 12: Cod HTML- formular

Toate cele trei limbaje au o stransa legatura. CSS ”da viata” codului afisat de HTML. Python comunica foarte mult cu HTML prin primirea de date prin intermediul formularelor, memorarea lor in liste, prelucrarea lor si trimiterea lor inapoi intr-o forma ce poate fi afisata si inteleasa de utilizator.

```

1  body {
2      margin: 0;
3      padding: 0;
4      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
5      background: linear-gradient(-45deg, #89f7fe, #66a6ff, #f7797d, #a18cd1);
6      background-size: 600% 600%;
7      animation: gradientBackground 30s ease infinite;
8      color: #fff;
9      text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.3);
10     min-height: 100vh;
11     transition: all 0.3s ease-in-out;
12 }
13
14 @keyframes gradientBackground {
15     0% {
16         background-position: 0% 50%;
17     }
18     50% {
19         background-position: 100% 50%;
20     }
21     100% {
22         background-position: 0% 50%;
23     }
24 }
25
26 @keyframes gradientBackground {
27     0% {background-position: 0% 50%;}
28     50% {background-position: 100% 50%;}
29     100% {background-position: 0% 50%;}
30 }

```

Figura 13: Cod CSS- design pagina

• Alte unelte relevante

O unealta foarte importanta este sintaxa Jinja2(Figura 14). Aceasta pare la prima vedere o combinatie dintre Python(deoarece sintaxa este asemanatoare) si Bash(deoarece pentru orice instructiune trebuie specificat foarte clar unde incepe si unde se termina). Alt lucru important este ca aceasta sintaxa nu tine cont de indentare, permitandu-se ca mai multe operatii sa fie scrise pe acelasi rand, un lucru care este recomandat a fi evitat. Aceasta este responsabila de prelucrarea si afisarea datelor pe pagina intr-un mod dinamic. De pe backend se pot primi liste cu elemente, liste ale caror elemente pot fi parcurse si afisate in pagina. Pot fi primite si imagini comprimate care mai apoi sunt afisate pe frontend.

```

168    {% if rez_final_log_ver %}
169        <p>Adunand toti logaritmii de mai sus, vom obtine forma finala a functiei de log-verosimilitate:</p>
170    <p>
171        \(\prod(w_i)\) =
172        \(\prod_{i=1}^{n-1} w_i^{\log_{w_i}(\sum_{j=i+1}^n w_j)}\)
173    </p>
174    <math>\prod_{i=1}^{n-1} w_i^{\log_{w_i}(\sum_{j=i+1}^n w_j)}</math>
175    <math>= \prod_{i=1}^{n-1} w_i^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
176    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} w_i^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
177    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(1 - \frac{w_i}{w_n}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
178    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
179    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
180    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
181    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
182    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
183    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
184    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
185    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
186    <math>= w_n^{\log_{w_n}(w_n + \sum_{j=1}^{n-1} w_j)} \prod_{i=1}^{n-1} \left(\frac{w_n}{w_n + \sum_{j=1}^{n-1} w_j}\right)^{\log_{w_i}(w_n + \sum_{j=1}^{n-1} w_j)}</math>
187    </p>
188    {% endif %}
189

```

Figura 14: Cod HTML impreuna cu Jinja2- afisare dinamica in pagina

5. Arhitectura aplicatiei

• Structura folderelor si a fisierelor

Toate componente ale acestei aplicatii sunt impartite intr-un folder(Figura 15). Prima componenta este ”.venv” care reprezinta mediul virtual. Urmatoarele componente sunt folderul ”static” in care se afla fisierele cu cod CSS si folderul ”templates” in care e afla fisierele cu cod HTML. Framework-ul Flask impune folosirea de doua foldere separate cu aceasta denumire pentru a face diferenta intre fisiere. A patra componenta este fisierul ”app.py”, un fisier care contine cod Python si care se ocupa de logica programului fiind cel care preia datele, le memoreaza, le prelucreaza si apoi le trimit inapoi la frontend. Ultima componenta este ”External Libraries” care contine librarii externe, librarii care sunt aduse odata cu instalarea librariei Flask.

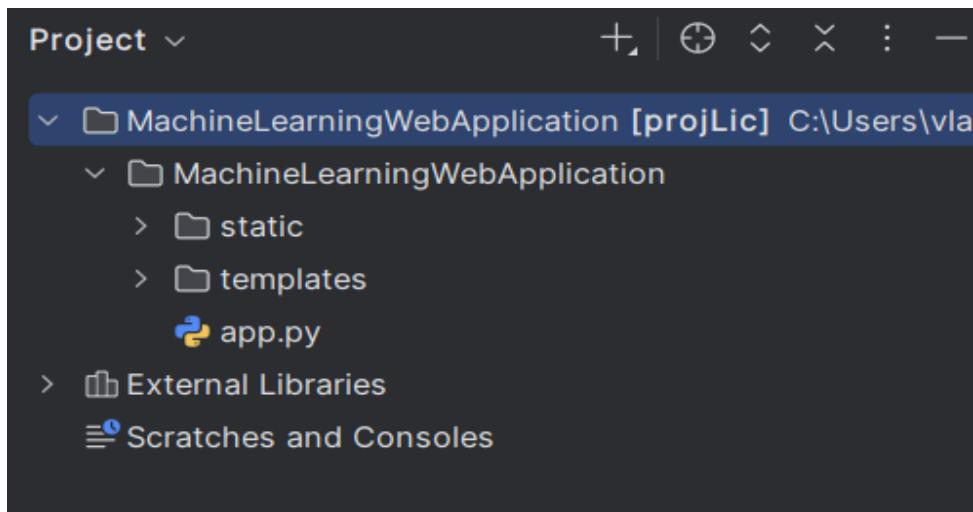


Figura 15: Structura folderelor

Folderul ”static”(Figura 16) este cel care contine fisierele CSS. Aceste fisiere cu cod sunt cele care ofera o culoare de fundal paginii, modifica dimensiune, culoarea si pozitionarea textului, creeaza animatii si se ocupa ca pagina sa fie responsive. Este important ca numele fisierului sa fie ”static”, asta fiind una dintre conditiile ca framework-ul Flask sa recunoasca paginile. S-a optat pentru folosirea unui fisier CSS mai complex si care ofera mai multa mobilitate, in interiorul acestuia fiind declarate mai multe teme astfel incat daca utilizatorul vrea sa schimbe tema paginii, o va putea face cu usurinta. La fel ca in cazul folderului ”static”, si folderul care contine fisierele cu cod HTML trebuie sa poarte o denumire specifica impusa de Flask, si anume ”templates”. Fara aceste denumiri si plasarea fisierelor in aceste foldere, framework-ul nu ar putea recunoaste fisierele si ar da eroare. In acest folder se afla paginile care au stransa legatura cu algoritmi. Fisierele care doar poarta nume de algoritm(de exemplu AdaBoost.html) sunt paginile care ofera informatii despre cum algoritmul functioneaza, pasii de rulare, versetele biblice care au o legatura cu modul lor de a se comporta si cuvintele cheie si importante folosite in contextul lor. Al doilea tip de fisier HTML este cel care poarta numele algoritmului urmat de cuvantul ”Example”(de exemplu AdaBoostExample.html), aceasta fiind pagina unde utilizatorul poate introduce informatii despre dimensiunea setului de date (numar de instante, linii, coloane, numar de centroizi) si setul de date(coordonate ale punctelor, etichete, instanta ce se doreste a fi clasificata). Tot pe acest tip de pagina sunt afisate explicatii despre cum algoritmul

funcioneaza, toti pasii algoritmului aplicati linie cu linie pe toate instancele din setul de date, grafice si tabele pentru o mai usoara vizualizare si inteleger.

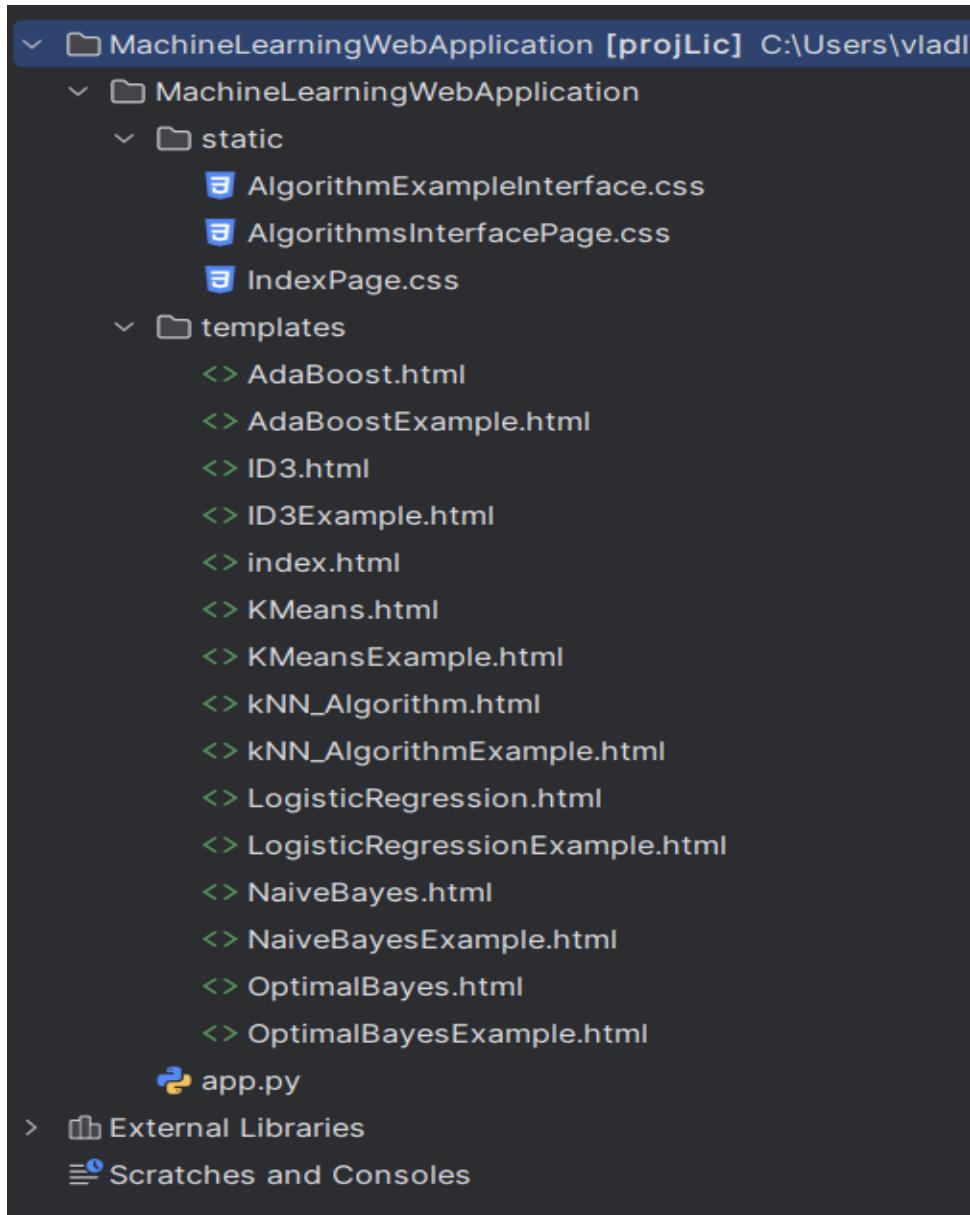


Figura 16: Structura folderelor "static" si "templates"

• Rutarea Flask

O functionalitate foarte importanta a acestui framework este rutarea. Aceasta permite ca o functie scrisa in Python sa fie legata de un URL(link)(Figura 17). Incepurul rutarii este marcata de simbolul @, urmat de numele aplicatiei asa cum este trecut in codul Python si de cuvantul cheie "route". Urmeaza ca apoi sa fie trecut intre o pereche de paranteze rotunde si o pereche de ghilimele linku-ul corespunzator paginii respective. Un exemplu de link este @app.route("/") care ne indica faptul ca aplicatia se numeste "app" iar simbolul "/" este corespunzator paginii principale, adica prima pagina care se afiseaza atunci cand aplicatia este pornita. Alt exemplu este @app.route("/LogisticRegression") care ne arata ca s-a navigat de la pagina principala catre

o alta pagina numita "LogisticRegression". Odata ce se face trecerea de la o pagina la alta, link-ul ori se va schimba complet ori va pastra linku-ul de la pagina anterioara la care se mai adauga niste cuvinte cheie separate de "/". La rutare este permisa si folosirea metodelor GET si POST ca in link-ul @app.route("/LogisticRegressionExample", methods=["GET", "POST"]) unde ni se indica faptul ca aplicatia va primi si trimit date si va utiliza una dintre aceste metode.

```

129     @app.route("/LogisticRegression")
130     def LogisticRegression():
131         return render_template("LogisticRegression.html")
132
133     @app.route(rule: "/LogisticRegressionExample", methods=["GET", "POST"])
134     def LogisticRegressionExample():

```

Figura 17: Rutarea Flask

• Gestionarea template-urilor si interacțiunea cu backend-ul

Template-urile sunt fisiere cu cod HTML la care se poate adauga si cod Python. Unul dintre scopurile principale este cel de a prelua datele si a le trimite catre backend unde urmeaza sa fie prelucrate. Acest lucru se intampla prin completarea cu date intr-un formular ce este creat in HTML. Odata ce datele sunt introduse, ele vor fi trimise catre backend unde vor fi preluate prin intermediul librariei "request"(Figura 18).

```

if request.method == "POST":
    n = int(request.form["n"]) if "n" in request.form else None
    m = int(request.form["m"]) if "m" in request.form else None

```

Figura 18: Libraria "request"

Din momentul in care datele sunt prelucrate, ele pot fi trimise inapoi catre frontend prin intermediul librariei "render_template"(Figura 19). Aceasta este o librerie care odata ce este accesata, va incarca o pagina HTML. Incarcarea paginii are loc in doua moduri, primul mod fiind fara parametri. In acest caz, pagina este incarcata fara a fi modificata. Al doilea caz este cel in care pagina este incarcata dar primeste si parametri. Parametrii reprezinta date ce au fost prelucrate cu ajutorul functiilor si a codului de Python. Pagina nu este imediat incarcata deoarece in codul HTML al paginii pot fi prelucrari de date, verificari de conditii daca anumite lucruri au fost prelucrate corect. Este posibil ca pagina sa nu se incarce datorita parametrilor primiti, un caz fiind cel in care datele nu sunt prelucrate corect pe backend si conditia de verificare a parametrilor trimisi prin intermediul librariei "render_template" este incalcata. Atunci cand se trimit un parametru cu ajutorul acestei librarii, trebuie mai intai sa specificam ce nume vrem sa poarte variabila noastra astfel incat sa fie recunoscuta pe frontend in momentul trimiterii. Pe langa asta, mai trebuie sa specificam ce valoare ii asignam variabilei, indiferent ca este o valoare

constanta, o valoare intreaga, o lista, un set, un dictionar sau o imagine comprimata. Odata ce variabilele au fost trimise catre a fi afisate in interfata grafica, trebuie facuta o verificare. Aceasta verificare se face in continuu pana cand variabila a fost primita. Din momentul in care s-a primit variabila, se poate intra in blocul de cod aferent prelucrarii variabilei respective. Variabila poate fi accesata prin utilizarea numelui folosit atunci cand a fost trimisa cu "render_template" intre doua perechi de acolade.

```
return render_template( template_name_or_list: "AdaBoostExample.html", n=n, elements=elements,
                      coordonate=coordonate, etichete=etichete, imagine1(imagine1,
                      praguriX(praguriX, praguriY(praguriY,
                      prag_exterior_x(prag_exterior_x, prag_exterior_y(prag_exterior_y,
                      nr_puncte(nr_puncte, imagine_cu_praguri(imagine_cu_praguri,
                      ponderi_it1(ponderi_it1, it1_axa_x_erori1(it1_axa_x_erori1, it1_axa_x_erori2(it1_axa_x_erori2,
                      it1_axa_y_erori1(it1_axa_y_erori1, it1_axa_y_erori2(it1_axa_y_erori2, it1_eroare_minima_tabel(prag_pentru_eroare_minima_it1_mesaj(prag_pentru_eroare_minima_it1_mesaj, prag_pentru_eroare_minima_it1_mesaj,
                      prag_pentru_eroare_minima_it1_mesaj(prag_pentru_eroare_minima_it1_mesaj, prag_pentru_eroare_minima_it1_mesaj,
                      gamma1(gamma1, alpha1(alpha1, valoare_X_it1_prag(valoare_X_it1_prag,
                      it1_puncte_clasificate_corect(it1_puncte_clasificate_corect,
                      it1_puncte_clasificate_gresit(it1_puncte_clasificate_gresit,
                      it1_index_puncte_clasificate_corect(it1_index_puncte_clasificate_corect,
                      it1_index_puncte_clasificate_gresit(it1_index_puncte_clasificate_gresit,
                      ponderi_it2(ponderi_it2, it2_eroare_minima_tabel(it2_eroare_minima_tabel,
                      it2_axa_x_erori1(it2_axa_x_erori1, it2_axa_x_erori2(it2_axa_x_erori2,
                      it2_axa_y_erori1(it2_axa_y_erori1, it2_axa_y_erori2(it2_axa_y_erori2,
                      alpha2(alpha2, gamma2(gamma2,
                      it2_prag_pentru_noi_ponderi(it2_prag_pentru_noi_ponderi,
                      prag_pentru_eroare_minima_it2_mesaj(prag_pentru_eroare_minima_it2_mesaj,
                      prag_pentru_eroare_minima_it2_valoare(prag_pentru_eroare_minima_it2_valoare,
                      valoare_X_it2_prag(valoare_X_it2_prag,
                      it2_puncte_clasificate_corect(it2_puncte_clasificate_corect,
                      it2_puncte_clasificate_gresit(it2_puncte_clasificate_gresit,
                      it2_index_puncte_clasificate_corect(it2_index_puncte_clasificate_corect,
                      it2_index_puncte_clasificate_gresit(it2_index_puncte_clasificate_gresit,
                      ponderi_it3(ponderi_it3, it3_eroare_minima_tabel(it3_eroare_minima_tabel,
                      it3_axa_x_erori1(it3_axa_x_erori1, it3_axa_x_erori2(it3_axa_x_erori2,
                      it3_axa_y_erori1(it3_axa_y_erori1, it3_axa_y_erori2(it3_axa_y_erori2,
                      alpha3(alpha3, gamma3(gamma3
```

Figura 19: Libraria "request"

6. Prezentarea algoritmilor integrati

• Algoritmul AdaBoost

AdaBoost(Adaptive Boosting) este un algoritm de invatare supervizata care are rolul de a clasiifica datele combinand mai multe modele slabe cum ar fi compasii de decizie.

Algoritmul, din punct de vedere teoretic, functioneaza astfel: fiecare punct primeste o pondere(greutate) care este echivalenta cu 1 supra numarul total de puncte, acesta fiind pasul initial. Aceasta asignare este valida doar in prima iteratie a algoritmului. Urmatorul pas este cel de stabilire a granitelor. Rolul granitelor este de a arata unde sunt puncte diferite pe grafic(daca un punct etichetat pozitiv este urmat de un punct etichetat negativ sau invers, atunci aici este un loc unde este necesara trasarea unei granite).

Aceste granite se traseaza la jumatarea distantei dintre doua cele mai apropiate puncte cu etichete diferite. Un lucru important la acest algoritm este trasarea unei granite sau a unor granite(depinde de spatiul numerelor), fara de care algoritmul ar functia gresit si ar oferi rezultate eronate. Aceasta granita se pune ori dupa ultimul punct de pe grafic, ori inainte de primul punct, fiind o mica distanta intre cele doua. Urmatorul pas este cel de calculare al erorilor pentru fiecare punct de pe grafic in raport cu granitele de decizie si determinarea erorii minime. Granitele de decizie sunt compasi de decizie care ajuta modelul sa clasifice datele.

La fiecare iteratie a algoritmului este ales cate un nou compas. Gasirea erorii minime determina si compasul de decizie la iteratia respectiva. Odata ce compasul de decizie a fost selectat, se pregatesc ponderile pentru iteratia urmatoare. Astfel, punctele clasificate corect primesc o pondere mare iar punctele clasificate gresit primesc o pondere mai mica. Deci, la o iteratie urmatoare, punctele clasificate gresit vor fi luate in evidenta avand eroarea cea mai mica. Se observa faptul ca doar in pasul initial punctele primesc pondere echivalenta cu 1 supra numarul total de puncte iar in iteratiile urmatoare ponderile au legatura cu punctele clasificate gresit si corect.

Acesti pasi se repeta pana cand toate punctele sunt clasificate corect sau pana cand s-a atins numarul de iteratii stabilite la inceput, acestea doua reprezentand doua criterii importante in oprirea rularii algoritmului.

In cod, mai exact in primul fisier HTML, prima parte ofera informatii despre modul in care algoritmul functioneaza(Figura 20). Aici sunt prezentati pasii de functionare a algoritmului, fiind prezentati intr-un mod informal, cat mai usor de intedes si fara cuvinte care ar putea ingreuna utilizatorul in a intelege modul de functionare. Pasii de rulare ai algoritmului au fost prezentati cat mai pe scurt deoarece cuvintele complicate si frazele lungi ar plictisi utilizatorul, aceasta pagina web dorind ca acest lucru sa fie evitat. In realizarea acestei aplicatii s-a dorit abordarea unei solutii noi si unice in a ajuta utilizatorii sa retina unele informatii mai usor, adica folosirea de versete biblice(Figura 21). In unele versete biblice regasim un comportament sau unele lucruri foarte specifice algoritmilor de Machine Learning. Din acest motiv, au fost adaugate cateva versete biblice care au o stransa legatura cu modul de functionarea al algoritmului, insotite de explicatii si de o interpretare.

```

2   <html lang="en">
3     <body>
4       <h1>Algoritmul AdaBoost</h1><br>
5
6       <p>
7         AdaBoost (Adaptive Boosting) este un algoritm de învățare automată utilizat pentru clasificare, care
8         funcționează prin combinarea mai multor modele slabe (de obicei arbori de decizie simpli sau "stumps")
9         într-un model puternic.
10        <br>
11        Principiul de bază al AdaBoost este următorul:<br>
12
13        Antrenarea unui model slab: Începe cu un model simplu (de obicei, un „stump”) care face predicții simple.<br>
14
15        Corectarea erorilor: După fiecare iteratie, AdaBoost ajustează greutatea exemplelor gresite, adică pune un
16        accent mai mare pe acele exemple care au fost corect clasificate gresit.
17        <br>
18        Combinarea modelelor: Modelul final este combinarea unui număr de clasificatori slabii, fiecare ponderat în
19        funcție de performanța sa. Modelele care fac mai multe greseli primesc o pondere mai mică.
20        <br>
21        AdaBoost repetă acest proces până când ajunge la un număr specificat de clasificatori sau până când
22        eroarea este suficient de mică. Ideea este că fiecare clasificator adaugă ceva valoros și corectează greselile
23        anterioare, iar modelul final este mult mai robust și precis.
24        <br><br>
25      </p>

```

Figura 20: Algoritmul AdaBoost- mod de functionare

```

†Versete biblice care reflectă principiile AdaBoost:<br>

1. Proverbe 24:16<br>
  „Căci cel neprihănit cade de sapte ori și se ridică, dar cei răi se prăbusesc în nenorocire.”<br>
  ✓ Legătura cu AdaBoost:<br>
  AdaBoost învăță din greselile anterioare, concentrându-se pe exemplele care au fost clasificate gresit.
  Modelul nu renunță după fiecare „cădere”, ci se ridică și devine mai puternic.
  <br><br>

2. Ecclesiastul 4:9<br>
  „Mai bine doi decât unul, căci au o răsplată bună pentru munca lor.”<br>
  ✓ Legătura cu AdaBoost:<br>
  AdaBoost folosește mai mulți clasificatori simpli (slabi) pentru a obține o performanță mai bună decât
  oricare dintre acestia individual. Colaborarea lor duce la rezultate mai bune.
  <br><br>

```

Figura 21: Algoritmul AdaBoost- versete biblice

In ultima sectiune a paginii se vor gasi termeni cheie explicati, termeni care sunt des folositi in contextul algoritmului AdaBoost(Figura 22). Acesti termeni au fost explicati intr-o maniera cat mai simpla, evitand sa fie date informatii si detalii care doar ar ingreuna procesul de intelegerere. Printre termenii explicati, regasim clasificatorul slab si cel puternic, greutatile si coeficientul de incredere.

```

72 Termeni utilizati:<br>
73 1.<b>Clasificator slab</b>- Un model simplu (de obicei un arbore de decizie mic) folosit ca baza in AdaBoost.<br>
74 2.<b>Clasificator puternic</b>- Un model obtinut prin combinarea mai multor clasificatori slabii pentru a
75 face predictii mai precise.<br>
76 3.<b>Greutati</b>- Fiecare exemplu din setul de date primeste o greutate care se ajusteaza pe
77 parcursul antrenarii, fiind mai mari pentru exemplele gresite clasificate.<br>
78 4.<b>Coeficientul de incredere</b>- Un factor care controleaza influenta fiecarui clasificator
79 slab asupra predictiei finale. Este calculat pe baza erorii fiecarui clasificator.<br>
80 <br>
```

Figura 22: Algoritmul AdaBoost- termeni importanti

In cel de-al doilea fisier HTML corespunzator algoritmului AdaBoost, mai exact cel in care se pot introduce date si se afiseaza pasii de executie ai algoritmului, utilizatorului i se cer date despre dataset. Aceste date pot fi completate si trimise cu ajutorul unui formular creat in HTML(Figura 23).

```

17  {% if n%}
18      <h2>Introduceti {{ n }} puncte (x, y) si eticheta (1 sau -1)</h2>
19      <form method="POST">
20          <input type="hidden" name="n" value="{{ n }}">
21
22          <table border="1">
23              <thead>
24                  <tr>
25                      <th>i</th>
26                      <th>x</th>
27                      <th>y</th>
28                      <th>Label</th>
29                  </tr>
30              </thead>
31              <tbody>
32                  {% for i in range(n) %}
33                      <tr>
34                          <td>{{ i + 1 }}</td>
35                          <td><input type="number" name="elements[]" required></td> <!-- x -->
36                          <td><input type="number" name="elements[]" required></td> <!-- y -->
37                          <td>
38                              <input type="number" name="elements[]" required min="-1" max="1" step="2">
39                          </td> <!-- label: 1 sau -1 -->
40                      </tr>
41                  {% endfor %}
42              </tbody>
43          </table>
> templates > <> AdaBoostExample.html                                         68:29  LF  UTF-8  4 spaces  Python 3.1
```

Figura 23: Algoritmul AdaBoost- formular informatii despre dataset

In acest formular se va introduce valoarea variabilei n, cea care ne spune cate instante(linii) va avea tabelul. Aceasta variabila este trimisa catre backend, unde va fi preluata cu ajutorul librariei "request" si memorata(Figura 24).

```
982     if request.method == "POST":  
983         n = int(request.form["n"]) if "n" in request.form else None
```

Figura 24: Algoritmul AdaBoost- preluarea numarului de instante

Aceasta variabila va fi trimisa inapoi prin intermediul librariei "render_template". In codul HTML, cu ajutorul sintaxei Jinja2, se va verifica daca variabila a fost primita. De abia dupa ce variabila a fost primita, se va crea un nou tabel cu n linii ce va trebui completat de utilizator cu setul de date(Figura 25).

```
49     {% if elements %}  
50         <h2 style="...">Punctele introduse</h2>  
51         <table border="1" cellpadding="10" cellspacing="0" style="...">  
52             <thead style="...">  
53                 <tr>  
54                     <th style="...">Id</th>  
55                     <th style="...">x</th>  
56                     <th style="...">y</th>  
57                     <th style="...">Etichetă</th>  
58                 </tr>  
59             </thead>  
60             <tbody>  
61                 {% for i in range(n) %}  
62                     <tr style="...">  
63                         <td>x<sub>{{ i + 1 }}</sub></td>  
64                         <td>{{ elements[i][0] }}</td>  
65                         <td>{{ elements[i][1] }}</td>  
66                         <td>{{ elements[i][2] }}</td>  
67                     </tr>  
68                 {% endfor %}  
69             </tbody>  
70         </table>  
71     {% endif %}
```

Figura 25: Algoritmul AdaBoost- formulat pentru completare date

In acest punct, utilizatorul va trebui sa completeze noul tabel(Figura 26) cu setul de date, urmand ca dupa sa apese pe butonul "Trimite date", urmand ca datele sa fie trimise catre backend iar utilizatorul sa fie redirectionat catre o noua pagina.

Introduceți 4 puncte (x, y) și eticheta (1 sau -1)			
i	x	y	Label
1			
2			
3			
4			

Trimite datele

Figura 26: Algoritmul AdaBoost- formulat pentru completare date

Dupa ce setul de date a fost completat iar utilizatorul a apasat butonul de trimitere a datelor, acestea vor fi trimise catre backend. Aici toate datele vor fi preluate si memorate in liste (Figura 27), urmand ca mai apoi sa fie prelucrate. Aici este si partea unde sunt declarate variabile care vor ajuta mai tarziu la memorarea si prelucrarea de date.

```

985     if n:
986         elements_form = request.form.getlist("elements[]")
987         try:
988             elements = [int(x) for x in elements_form]
989
990             if len(elements) == 3 * n:
991                 elements = [elements[i:i + 3] for i in range(0, len(elements), 3)]
992             else:
993                 elements = []
994             except ValueError:
995                 elements = []
996             etichete=[]
997             coordonate=[]
998             imagine1=[]
999             imagine_cu_praguri=[]
1000             praguriX=[]
1001             praguriY=[]
1002             prag_exterior_x=None
1003             prag_exterior_y=None
1004             nr_puncte=None
1005             ponderi_it1=[]
1006             it1_axa_x_erori1=[]
1007             it1_axa_x_erori2=[]
1008             it1_axa_y_erori1=[]

```

Figura 27: Algoritmul AdaBoost- preluate dataset

Ajuns in acest pas, primul lucru este cel de a crea un grafic. Pentru ca utilizatorul sa inteleaga mai usor, se va face un grafic cu reprezentarea in plan xOy a punctelor primite, impreuna cu etichetele lor. Pentru acest lucru, s-a creat o functie aparte (Figura 28) care primeste ca parametri doua liste, prima reprezentand coordonatele punctelor iar a doua reprezentand etichetele lor. Graficul este creat cu ajutorul librarii matplotlib, o librarie destinata vizualizarii si crearii de grafice. Graficul va avea punctele reprezentate intr-un grafic, punctele cu etichete negative fiind reprezentate de cerculete goale iar cele cu eticheta pozitiva de cerculete pline. Odata creat graficul, va fi returnat intr-o forma comprimata astfel incat sa poata fi trimis inapoi pe frontend. Dupa ce se termina de executat functia, imaginea va fi trimisa catre frontend unde va fi afisata in pagina (Figura 29). Pentru a fi afisata in pagina, este necesara verificarea unei conditii, aceea ca imaginea sa fie deja creata.

```

752     def adaBoost_plot1(list1, list2): 1 usage
753         if len(list1) != len(list2):
754             raise ValueError("Listele trebuie să aibă aceeași lungime")
755
756         fig, ax = plt.subplots()
757
758         for (x, y), label in zip(list1, list2):
759             if label == 1:
760                 ax.plot(*args: x, y, 'ko')
761             elif label == -1:
762                 ax.plot(*args: x, y, 'ko', markerfacecolor='white')
763             else:
764                 raise ValueError("Etichetele trebuie să fie 1 sau -1")
765
766         ax.set_xlabel('x')
767         ax.set_ylabel('y')
768         ax.set_title('Reprezentare puncte cu etichete')
769         ax.grid(True)
770         buf = io.BytesIO()
771         plt.savefig(*args: buf, format='png')
772         plt.close(fig)
773         buf.seek(0)
774
775         img_base64 = base64.b64encode(buf.getvalue()).decode('utf-8')
776
    return img_base64

```

Figura 28: Algoritmul AdaBoost- creare grafic folosind matplotlib

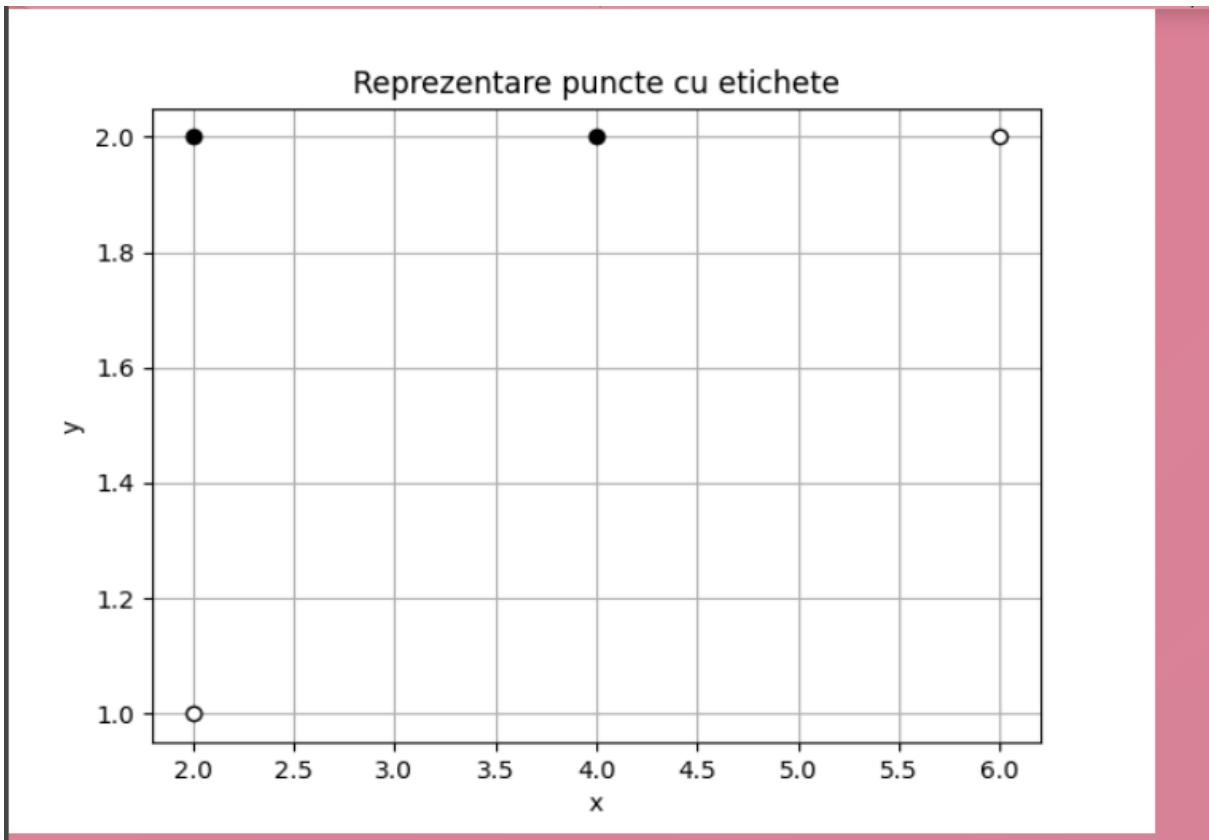


Figura 29: Algoritmul AdaBoost- grafic afisat in pagina

Urmeaza sa se determine granitele de decizie. Acestea au rolul de a separa doua puncte apropriate cu etichete diferite. Se cauta in grafic unde sunt doua puncte apropriate, indiferent daca e vorba de plan orizontal sau vertical, si se traseaza o dreapta punctata la jumatea distantei dintre ele. Acest lucru este facut cu ajutorul unei functii care primeste ca parametri coordonatele si etichetele, aranjeaza in ordine crescatoare atat in functie de axa Ox cat si de Oy punctele si returneaza doua liste. Prima lista contine split-urile(granitele) corespunzatoare axei orizontale iar a doua corespunzatoare axei verticale (Figura 20).

Nr. prag	Prag
1	1.5
2	3.0
3	5.0

Figura 30: Algoritmul AdaBoost- granite de decizie

Dupa ce cele doua liste au fost returnate de functie, ele sunt trimise ca si parametru catre alta functie care are scopul de a crea alt grafic. Aceasta functie primeste ca parametri coordonatele punctelor, etichetele lor, granitele de decizie corespunzatoare axelor Ox si Oy, urmand ca apoi sa realizeze un desen cu punctele si granitele. Dupa ce graficul este creat, este returnat sub o forma comprimata si urmeaza ca sa fie exportat de pe backend catre frontend. Aici, daca este indeplinita conditia de verificare, adica daca imaginea exista si a fost creata, se va afisa pe pagina imaginea (Figura 31).



Figura 31: Algoritmul AdaBoost- granite de decizie

S-au utilizat culori, verde pentru granitele verticale si rosu pentru granitele orizontale pentru o mai buna vizualizare. In grafic s-au utilizat unitati mici de masura, totul pentru a fi cat mai bine vizualizat.

Unul dintre pasii importanti ai algoritmului este cel de determinare a erorilor. Pentru acest lucru, se ia fiecare punct in parte si se verifica daca este corect clasificat in functie de in ce parte a granitei este pozitionat. Aceste erori sunt memorate intr-o lista si apoi trimisa catre frontend unde cu ajutorul unei instructiuni repetitive (Figura 33), lista este parcursa iar erorile sunt afisate in pagina sub forma tabelara (Figura 32).

Prag	1.5	3.0	5.0
$err_{D1}(X_1 < Prag)$	0.5	0.5	0.25
$err_{D1}(X_1 \geq Prag)$	0.5	0.5	0.75

Figura 32: Algoritmul AdaBoost- erori

Erorile sunt parcurse cu ajutorul instructiunii repetitive "for". Initial este creat un tabel, iar pentru fiecare eroare este alocata cate o casuta in tabelul respectiv. Ca si pas initial, pe prima linie se afiseaza valorile din tabel unde au loc granitele, apoi pe prima coloana erorile.

```

160     <table border="1">
161     <tr>
162         <td><strong>Prag</strong></td>
163         {% for prag in praguriX %}
164             <td>{{ prag }}</td>
165         {% endfor %}
166     </tr>
167     <tr>
168         <td><strong><i>err<sub>D1</sub>(X<sub>1</sub>< Prag)</i></strong></td>
169         {% for eroare in it1_axa_x_erori1 %}
170             <td>{{ eroare }}</td>
171         {% endfor %}
172     </tr>
173     <tr>
174         <td><strong><i>err<sub>D1</sub>(X<sub>1</sub>>= Prag)</i></strong></td>
175         {% for eroare in it1_axa_x_erori2 %}
176             <td>{{ eroare }}</td>
177         {% endfor %}
178     </tr>
179 </table>
180     <br><br>
181 {%endif%}
182

```

Figura 33: Algoritmul AdaBoost- afisare tabelara a erorilor

Aceste informatii au fost alese sa fie reprezentate intr-un tabel pentru ca reprezinta niste informatii importante si pentru a fi vizualizate mai usor.

Dupa ce au fost oferite toate informatiile necesare, algoritmul va trebui sa se pregateasca pentru pasul urmator. Acesta este reprezentat de recalcularea ponderilor, oferind ponderi noi pentru pasul urmator. Acestea sunt calculate folosind ponderile de la pasul precedent si erorile din tabelul prezentat anterior. Acestea vor fi calculate iterativ, tinandu-se cont de faptul daca au fost clasificate corect sau gresit la pasul respectiv. In timp ce sunt calculate, acestea sunt memorate intr-o lista, urmand sa fie trimise catre frontend. In momentul primirii acestei liste, cu ajutorul instructiunii "for" din Jinja2, se parcurge lista iar ponderile sunt afisate in format tabelar pe ecran (Figura 34).

x1	x2	x3	x4
0.1666666666666666	0.5	0.1666666666666666	0.1666666666666666

Figura 34: Algoritmul AdaBoost- afisare tabelara a erorilor

Deoarece scopul acestei aplicatii este unul educational, s-a optat ca algoritmul sa se opreasca dupa trei iteratii. Daca i s-ar fi permis algoritmului sa execute mai multe iteratii, pagina ar fi intampinat probleme de performanta si timp deoarece prelucrarea datelor si crearea de imagini este costisitoare, iar utilizatorul si-ar fi pierdut interesul sa citeasca absolut toate explicatiile de la toate iteratiile algoritmului. Pe parcursul rularii celor trei iteratii, toate datele importante sunt memorate in variabile care au fost declarate la inceputul codului. S-a optat pentru folosirea cat mai multor variabile pentru a nu mai fi necesara parcurgerea de liste sau de liste care contin liste, asta complicand codul. Dupa ce toate datele au fost colectate, vor fi trimise ca parametri catre o functie bazata pe matplotlib ce va construi un grafic in care este reprezentat modul in care punctele au fost clasificate. Acest grafic va fi returnat sub o forma comprimata, urmand ca apoi sa fie trimis catre frontend si afisat in pagina. Cele trei ipoteze au fost colorate diferit iar semnele "+" si "-" reprezinta cum clasifica modelul punctele in zona respectiva.

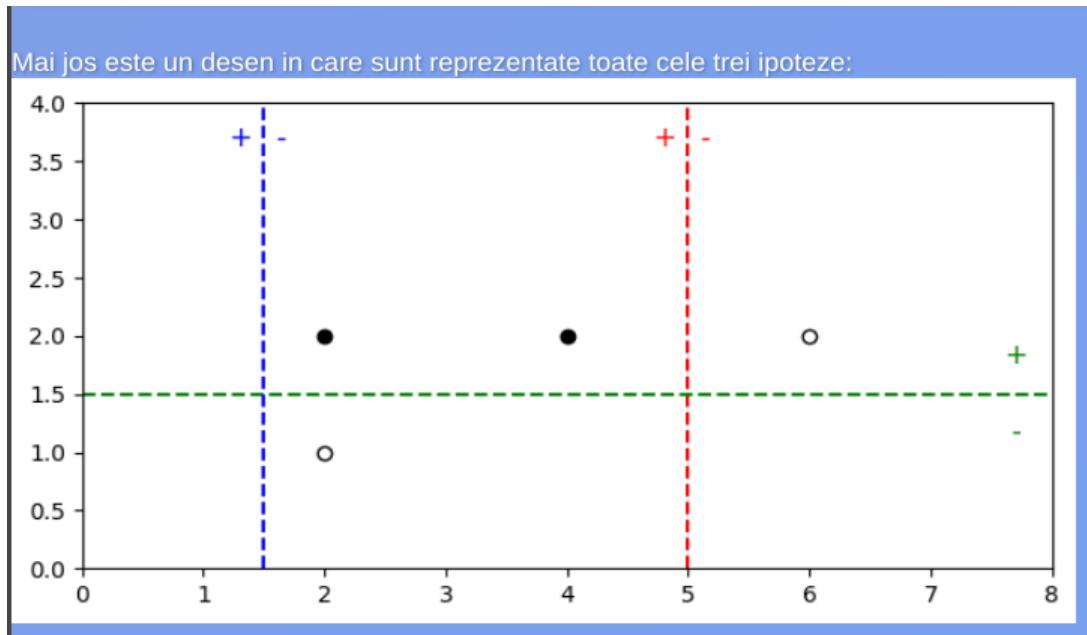


Figura 35: Algoritmul AdaBoost- clasificarea datelor

• Algoritmul k-NN

Algoritmul k-NN este un algoritm de invatare supervizata care face clasificare pe baza celor mai apropiate instante.

Din punct de vedere teoretic, algoritmul functioneaza astfel: datele sunt reprezentate sub forma de puncte intr-un grafic impreuna cu etichetele acestora. De obicei, punctele sunt reprezentate de o bulina neagra daca au eticheta negativa si bulina alba daca au eticheta pozitiva. Apoi se alege o metrica de distanta.

Alegerea acesteia este un pas important deoarece folosirea de metriki de distanta diferita schimba si acuratetea modelului. Cea mai folosita metrica este distanta Euclidiana care se calculeaza ca radical din suma patratelor diferenței dintre coordonatele corespunzatoare axei Ox, respectiv Oy. Alte doua metriki foarte folosite sunt distanta Manhattan si distanta Chebyshev. Alte metriki mai putin intalnite sunt distantele Minkowski, Cosinus Distance, Hamming, Mahalanobis, Jaccard, Levenshtein.

Odata ce metrica de distanta a fost aleasa, trebuie aleasa o valoare pentru variabila k. Aceasta variabila reprezinta cati vecini vor fi luati in calcul in determinarea modelului. Daca k este ales ca fiind 5, atunci eticheta punctului pe care vrem sa il clasificam va fi determinata de etichetele celor mai apropiati 5 vecini. In determinarea clasificarii punctului dorit, se vor face doua multimi, prima reprezentand multimea vecinilor din cei k cei mai apropiati vecini care au etichete pozitive si a doua multime formata din vecinii din cei k cei mai apropiati vecini care au etichete negative. Multimea cu cardinalul mai mare va da eticheta punctului care se doreste a fi clasificat. Spre exemplu, daca din cei 5 cei mai apropiati vecini ai unui punct pe care dorim sa il clasificam 3 vecini au eticheta pozitiva iar 2 au eticheta negativa, atunci punctul nostru va avea si el tot eticheta pozitiva deoarece multimea vecinilor cu eticheta pozitiva domina.

Este important ca atunci cand se alege valoarea pentru variabila k, aceasta sa reprezinte un numar impar. Daca variabila k ar fi ales un numar par, atunci este foarte posibil sa se intample cazul in care exact jumata din vecini sunt etichetati pozitiv si jumata sunt etichetati negativ, caz in care nu se poate lua o decizie asupra etichetei care i se potriveste punctului ce trebuie clasificat. Aceasta este o ambiguitate care trebuie evitata deoarece algoritmul este in imposibilitatea de a oferi un raspuns.

In primul fisier HTML corespunzator acestui algoritm se regasesc informatii despre modul lui de functionare (Figura 36). Sunt explicate pe rand etapele pe care algoritmul le parcurge pana la oferirea rezultatului final. Sunt explicati pasii in cuvinte cat mai simple, totul fiind enumerat. Ca si abordare a unei solutii unice, s-a ales introducerea de versete biblice (Figura 37) care au transa legatura cu modul de functionare a algoritmului. Aceste versete au fost alese cu atentie astfel incat sa reflecteze legatura intre cele doua. In pagina este prezentat versetul, informatii despre unde se gaseste in Biblie, legatura cu algoritmul si interpretarea acestuia.

Algoritmul k-NN

K-NN este un algoritm de clasificare. Când primește un exemplu nou (un punct cu caracteristici necunoscute), cauță în setul de date cei mai apropiati „vecini” și decide la ce categorie aparține exemplul nou, pe baza majorității vecinilor săi.

Cum funcționează pas cu pas:

- 1.Se dă un punct nou (necunoscut).
- 2.Se calculează distanțele față de toate celelalte puncte.
- 3.Se aleg cei mai apropiati K vecini.
- 4.Se „votează” pentru cea mai frecventă clasă dintre vecini.
- 5.Se atribuie aceea clasă punctului necunoscut.

Figura 36: Algoritmul k-NN- mod de functionare

Algoritmul k-NN clasifica punctele în funcție de etichetele vecinilor. Dacă majoritatea punctelor sunt pozitive, atunci și punctul care se dorește să fie clasificat tot pozitiv va fi, în caz contrar acesta va fi negativ. Acest lucru se regăseste și în versetele prezentate care ne spun că cine umbă cu intelectii va deveni și el unul. Prin aceasta analogie se dorește crearea unei legături și înțelegerea mai bună a modului de funcționare a algoritmului k-NN de către utilizator.

```
#Versete biblice care reflectă principiile K-NN:  

1. Proverbe 13:20<br>
„Cine umbă cu intelectii devine intelept, dar tovarăsul nebunilor suferă pagubă.”<br>
Legătura cu K-NN:<br>
Clasificarea se face în funcție de vecinii apropiati. Dacă aceștia sunt "intelepti", adică de o anumită clasă,
noul exemplu va fi asociat cu acea clasă.
<br>
2. 1 Corinteni 15:33<br>
„Nu vă înselați: tovarăsiile rele strică obiceiurile bune.”<br>
Legătura cu K-NN:<br>
Influența celor din jur determină categoria în care ești încadrat. Vecinii contează, chiar dacă tu ești
diferit - modelul te clasifică după ei.
```

Figura 37: Algoritmul k-NN- versete biblice

Ultima secțiune(Figura 38) din pagina este cea dedicată termenilor explicativi. Aici se regăsesc termeni importanți utilizati în contextul acestui algoritm. Termenii sunt explicati în maxim o fraza, într-un mod simplu și informal fără ca să complice procesul de învățare al utilizatorului. Tot în ultima secțiune a paginii se regăseste un dataset preluat din cartile de specialitate, fiind pus în cazul în care utilizatorul dorește să vada cum rulează programul dar întâmpina probleme în gasirea unui dataset.

Termini utilizati:

- 1.**k (numărul de vecini)** – numărul de vecini cei mai apropiati luati în considerare pentru a face predicția.
- 2.**Distanță Euclidiană** – cea mai comună metrică folosită pentru a calcula distanța între puncte; măsoară "lungimea" liniei drepte dintre două puncte.
- 3.**Spațiu de caracteristici (feature space)** – reprezentarea datelor ca puncte într-un spațiu n-dimensional, unde fiecare dimensiune corespunde unei caracteristici.
- 4.**Clasificare** – una dintre sarcinile pentru care se folosește k-NN; presupune atribuirea unei clase unui nou exemplu pe baza majorității dintre vecinii săi.
- 5.**Majoritate (majority voting)** – metoda prin care se stabilește clasa unui exemplu nou: cea mai frecventă clasă dintre vecini este aleasă.
- 6.**Curse of Dimensionality(Blestemul marilor dimensiuni)** – fenomen în care creșterea numărului de dimensiuni (features) poate reduce eficiența k-NN, deoarece distanțele devin mai puțin semnificative.
- 7.**Overfitting** – apare când k este prea mic (ex: k=1), iar modelul învăță prea specific, captând zgromotul din date.

Figura 38: Algoritmul k-NN- versete biblice

In josul paginii se regasesc doua butoane, unul care trimit utilizatorul de pe pagina respectiva catre pagina principala si inca un buton care trimit utilizatorul catre pagina unde va testa si primi informatii despre algoritm. In momentul apasarii pe butonul "exemplu practic", se va naviga de la pagina actuala la pagina unde utilizatorul va completa un chenar cu o valoare pentru variabila n (Figura 39), variabila care reprezinta numarul de linii pe care il va avea dataset-ul.

```
<h2>Introduceti {{ n }} linii si 3 coloane</h2>
<form method="POST">
    <input type="hidden" name="n" value="{{ n }}>

    <label for="x1">x1: </label>
    <input type="number" name="x1" id="x1" required><br><br>

    <label for="y1">y1: </label>
    <input type="number" name="y1" id="y1" required><br><br>

    <table border="1">
        <thead>
            <tr>
                <th>i</th>
                <th>x</th>
                <th>y</th>
                <th>Eticheta</th>
            </tr>
        </thead>
        <tbody>
            {% for i in range(n) %}
                <tr>
                    <td>{{ i + 1 }}</td>
                    <td><input type="number" name="elements[]" required></td>
                    <td><input type="number" name="elements[]" required></td>
                    <td><input type="number" name="elements[]" required></td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
```

Figura 39: Algoritmul k-NN- tabel informatii dataset

Variabila n este trimisa catre backend, memorata intr-o variabila, apoi urmand sa se creeze un tabel cu n linii(Figura 40) pe care utilizatorul sa il completeze cu date. Pe langa acel tabel ce trebuie completat cu coordonatele punctelor si etichetele lor, mai trebuie completat si instanta ce trebuie clasificata. In cazul acesteia, se vor trece doar coordonatele urmand ca eticheta sa fie determinata de algoritm.

Introduceti un numar n

Generare tabel

Introduceti 4 linii si 3 coloane

i	x	y	Eticheta
1			
2			
3			
4			

Trimite elementele

Figura 40: Algoritmul k-NN- interfata tabel informatii dataset

Dupa ce punctele impreuna cu coordonatele si etichetele au fost trimise catre backend, aici vor fi memorate in liste separate. La inceput se va crea un grafic cu toate aceste puncte pentru o vizualizare mai usoara. Acest lucru este posibil cu ajutorul unei functii(Figura 43) ce foloseste libraria matplotlib si primeste ca parametri mai multe liste ce reprezinta coordonatele punctelor si etichetelor. Dupa ce imaginea este creata, se va returna si apoi va fi trimisa catre frontend unde ii va fi afisata utilizatorului (Figura 42). Pentru ca imaginea sa fie afisata utilizatorului, mai intai trebuie sa indeplineasca o conditie(Figura 41), cea de a fi creata si de a fi primita. De abia dupa ce aceste lucruri se intampla, utilizatorul va putea vizualiza imaginea.

```

81     {%if elements%}
82         
83     {%endif%}

```

Figura 41: Algoritmul k-NN- conditie de verificare

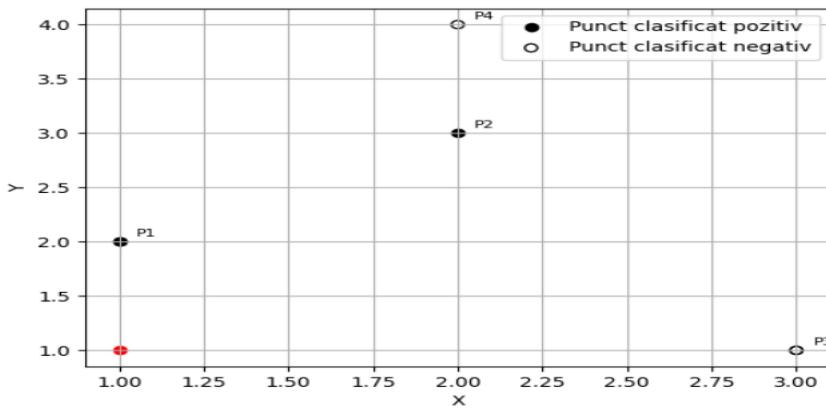


Figura 42: Algoritmul k-NN- grafic cu setul de date

Pentru a fi vizualizate mai usor si pentru o intelegerare mai simpla, s-a ales ca punctele pozitive sa fie reprezentate de un cerc negru, cele negative de un cerc gol iar punctul ce se doreste a fi clasificat este reprezentat de un cerculet de culoare rosie.

```
def plot_points_with_red(points, x1, y1, elements): 2 usages
    ax.scatter(x1, y1, color='red', marker='o')

    for x, y, filled in points:
        index_in_elements = elements.index([x, y, filled])
        label = f'P{index_in_elements + 1}'

        if filled == 1:
            ax.scatter(x, y, color='black', marker='o')
            ax.text(x + 0.05, y + 0.05, label, color='black', fontsize=8)
            if not positive_label_shown:
                ax.scatter(x, y, color='black', marker='o', label='Punct clasificat pozitiv')
                positive_label_shown = True
        elif filled == -1:
            ax.scatter(x, y, color='black', marker='o', facecolors='none')
            ax.text(x + 0.05, y + 0.05, label, color='black', fontsize=8)
            if not negative_label_shown:
                ax.scatter(x, y, color='black', marker='o', facecolors='none', label='Punct clasificat negativ')
                negative_label_shown = True

    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.legend()
    ax.grid(True)

    img = io.BytesIO()
    plt.savefig(*args: img, format='png')
    img.seek(0)
    return base64.b64encode(img.getvalue()).decode()
```

Figura 43: Algoritmul k-NN- codul aferent functiei de creare a graficului

Imaginea este returnata de functie in format comprimat pentru a putea fi trimisa cu ajutorul librariei "render_template" catre backend.

Algoritmul va face mai multe iteratii in functie de dimensiunea setului de date primit. Cu cat setul de date este mai mare, cu atat si numarul de iteratii creste. Daca de exemplu setul de date are 10 instante, atunci vom avea 5 iteratii. In schimb daca setul de date are o dimensiune mai mare, spre exemplu 100, numarul de iteratii creste la 50. La fiecare iteratie a algoritmului se va luta un k impar. Acest lucru este foarte important in gasirea rezultatului final. Daca k este impar, atunci cand multimea se va imparti in doua submultimi de vecini, mereu va exista o submultime care are macar un vecin in plus, de unde va rezulta si clasificarea punctului. Daca k ar fi par, atunci se poate ajunge la cazul in care impartirea multimii un doua ar duce la doua submultimi cu cardinal egal, caz in care algoritmul nu poate luta o decizie in ceea ce priveste clasificarea punctului. Pentru fiecare iteratie in parte, se vor determina cei mai apropiati k vecini ai punctului respectiv si se vor memora intr-o lista. Apoi in liste separate se vor memora distantele intre puncte, coordonatele, etichetele, urmand ca aceste elemente sa fie trimise pe frontend. Odata primite aceste elemente, cu ajutorul unei instructiuni repetitive se va itera printre aceste liste si se vor afisa intr-un mod placut in pagina.

Punctele cele mai apropiate de punctul nostru sunt:

P1 de coordonate (1, 2) **clasificat pozitiv** cu distanta de 1.0 unitati dintre punctul nostru de coordonate (1, 1) si P1 .

Formula distantei dintre două puncte este $d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Punctul nostru are coordonatele (1, 1) deci $x_1=1$ si $y_1=1$, iar punctul P1 are coordonatele (1, 2) deci $x_2=1$ si $y_2=2$.

Inlocuind in formula, obtinem $d(P, P1) = d((1, 1), (1, 2)) = \sqrt{(1 - 1)^2 + (2 - 1)^2} = \sqrt{(0)^2 + (1)^2} = \sqrt{0 + 1} = \sqrt{1} \approx 1.0$

P3 de coordonate (3, 1) **clasificat negativ** cu distanta de 2.0 unitati dintre punctul nostru de coordonate (1, 1) si P3 .

Formula distantei dintre două puncte este $d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Punctul nostru are coordonatele (1, 1) deci $x_1=1$ si $y_1=1$, iar punctul P3 are coordonatele (3, 1) deci $x_2=3$ si $y_2=1$.

Inlocuind in formula, obtinem $d(P, P3) = d((1, 1), (3, 1)) = \sqrt{(3 - 1)^2 + (1 - 1)^2} = \sqrt{(2)^2 + (0)^2} = \sqrt{4 + 0} = \sqrt{4} \approx 2.0$

P2 de coordonate (2, 3) **clasificat pozitiv** cu distanta de 2.24 unitati dintre punctul nostru de coordonate (1, 1) si P2 .

Formula distantei dintre două puncte este $d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Punctul nostru are coordonatele (1, 1) deci $x_1=1$ si $y_1=1$, iar punctul P2 are coordonatele (2, 3) deci $x_2=2$ si $y_2=3$.

Inlocuind in formula, obtinem $d(P, P2) = d((1, 1), (2, 3)) = \sqrt{(2 - 1)^2 + (3 - 1)^2} = \sqrt{(1)^2 + (2)^2} = \sqrt{1 + 4} = \sqrt{5} \approx 2.24$

Deoarece avem mai multe puncte clasificate pozitiv decat puncte clasificate negativ, atunci si punctul nostru tot pozitiv va fi clasificat.

Figura 44: Algoritmul k-NN- afisarea in pagina a modului de calculare a distanteelor

Fiecare pas este foarte detaliat explicit si evidentiat. Punctele clasificate pozitiv sunt evidențiate prin folosirea culorii verde iar cele negative prin culoarea albastru. Pe baza coordonatelor punctelor, formula de distanta Euclidiana este aplicata pas cu pas, nefacandu-se simplificari pana la rezultatul final. Acest lucru a fost posibil prin parcurgerea listelor primite de pe backend, liste in care s-au memorat calcule deja facute, singurul lucru ramas fiind doar afisarea intr-un mod placut si aranjat in pagina. Facerea calculelor din timp si memorarea lor in liste a fost necesara deoarece Jinja2 nu permite atribuirea de valori variabilelor. La final este oferita o mica explicatie din care reiese de ce punctul a fost clasificat intr-un anume fel.

• Algoritmul K-Means

Algoritmul K-Means este un algoritm de invatare nesupervizata, avand ca scop principal gruparea datelor.

Modul de functionare este urmatorul: primul pas este cel de a se stabili cu ce valoare este initializata variabila k . Alegerea acestei valori este foarte importanta deoarece o valoare diferita poate influenta precizia si acuratetea modelului. Un k prea mic ar grupa multe date diferite impreuna iar un k prea mare ar grupa date care sunt foarte asemanatoare la un loc. Una dintre metodele de alegere a valorii pentru variabila k este metoda cotului(Elbow). Aceasta metoda presupune rularea algoritmului pe mai multe valori posibile ale lui k . Rezultatele sunt reprezentate intr-un grafic pe care se cauta un "cot", adica un punct unde scaderea erorii devine cat mai mica posibil. O alta metoda utilizata este Silhouette care functioneaza pe baza unui scor acordat punctelor.

Odata ce k este ales, trebuie alesi cei k centroizi. In acest caz, centroizii pot fi chiar si punctele din grafic sau pot fi puncte care nu se regasesc in grafic. Urmeaza alegerea metricii de distanta care este un pas important, folosirea unei metrici diferite poate schimba si acuratetea modelului. Cea mai folosita metrica este distanta Euclidiana care se calculeaza ca radical din suma patratelor diferenței dintre coordonatele corespunzatoare axei Ox , respectiv Oy . Alte doua metrici foarte folosite sunt distanta Manhattan si distanta Chebyshev. Alte metrici mai putin intalnite sunt distantele Minkowski, Cosinus Distance, Hamming, Mahalanobis, Jaccard, Levenshtein.

In cele ce urmeaza trebuie calculata distanta dintre fiecare punct si cei k centroizi. Se ia cate un punct pe rand, se calculeaza distanta dintre el si cei k centroizi, de aici rezultand k lungimi de distanta. Apoi se compara cele k lungimi urmand ca punctul sa fie asignat centroidului de care este cel mai apropiat. Dupa ce toate punctele au fost asignate unui centroid, urmeaza sa fie recalculate pozitiile centroizilor. Modul de a determina noua pozitie a fiecarui centroid este una destul de simpla. Se aduna toate coordonatele punctelor corespunzatoare clusterului in care au fost alocate si se imparte la numarul total de puncte din multime, astfel obtinandu-se noile coordonate ale centroidului. Cu alte cuvinte, centroidul este un centru de greutate. In acest punct se termina o iteratie a algoritmului.

Urmeaza ca algoritmul sa se repete, dar de data asta cu alte coordonate ale centroizilor. In functie de marimea setului de date, centroizii se pot schimba foarte frecvent sau aproape deloc. Daca vorbim de un dataset mic, undeva la 10-15 puncte, e posibil ca centroizii sa se schimbe doar intre una sau doua iteratii sau sa nu se schimbe deloc intre nicio iteratie. In cazul in care datasetul este foarte mare, centroizii isi vor schimba pozitiile intre multe iteratii, sansa ca centroizii sa fie fixati in pozitia potrivita din prima fiind foarte mica.

Algoritmul se opreste in doua situatii. Prima e atunci cand se atinge numarul de iteratii fixat la inceputul algoritmului iar a doua e atunci cand centroizii nu isi mai schimba pozitia intre doua iteratii succesive sau isi schimba pozitia dar foarte putin.

Prima parte a fisierului HTML ce reprezinta acest algoritm este cea in care se regasesc informatii despre modul in care acest algoritm functioneaza (Figura 45). Pasii de executie au fost explicati pe scurt, fiind enuntati in patru idei. Motivul pentru care aplicatia ofera informatii intr-un mod informat si cat mai scurte este sa faca utilizatorul sa inteleaga mai bine si mai usor cum functioneaza un algoritm. Scopul este cel de a intelege, nu de a memora definitii si informatii pe de rost.

<p>

K-Means este un algoritm de învățare nesupravegheată folosit pentru gruparea (clustering) datelor.
 Scopul lui este să împartă datele în K grupuri (clustere), în funcție de cât de apropiate sunt între ele.

⌚ Cum funcționează K-Means:

1. Alegem aleatoriu K centre (centroizi).

2. Fiecare punct este alocat celui mai apropiat centru.

3. Se recalculează centrul fiecărui grup ca media punctelor din acel grup.

4. Se repetă pașii 2-3 până când centrele nu se mai schimbă (convergență).

</p>

Figura 45: Algoritmul K-Means- modul de functionare

Multe dintre versete(Figura 46) ilustreaza situatii si comportamente asemanatoare modului de functionare al algoritmilor. Oferind aceste versete impreuna cu o interpretare, utilizatorul poate alege daca sa fie de accord cu interpretarea oferita sau sa vina cu interpretarea personala. In cele mai multe cazuri, aceasta metoda functioneaza foarte bine, ajutand pentru o vizuire putin mai diferita. Dupa cum am mentionat anterior, scopul este de a intelege si invata, nu de a memora robotic niste definitii.

+Versete biblice care reflectă principiile K-Means:

1. Matei 25:32

„Toate neamurile vor fi adunate înaintea Lui. El îi va desparti pe unii de altii, cum desparte păstorul oile de capre.”

Legătura cu K-Means:

Algoritmul K-Means face exact asta: separă datele în grupuri distincte în funcție de trăsăturile lor. Nu există confuzie – fiecare punct este separat clar, ca oile de capre.

2. Romani 12:4-5

„Căci, după cum intr-un singur trup avem multe mădule, iar mădulele nu au toate aceeași funcție, tot așa și noi, deși suntem mulți, suntem un singur trup în Hristos și, fiecare în parte, mădule unui altora.”

Legătura cu K-Means:

Acest verset reflectă ideea de specializare și colaborare în cadrul unui grup, similar cu modul în care punctele dintr-un cluster contribuie la definirea centrului comun în K-Means.

Figura 46: Algoritmul K-Means- modul de functionare

In pasii de executie ai algoritmului se vor intalni termeni specifici, motiv pentru care in ultima sectiune a paginii s-au pus explicatii pentru cuvintele mai importante(Figura 47). Printre termenii explicati regasim centroidul, clusterul, convergenta, invatarea nesupervizata, criteriul J sau criteriul Elbow. Termenii sunt explicati simplu, intr-o fraza de cateva cuvinte si cu cuvinte cat mai simple.

Termeni utilizati:
1. k - Numărul de clustere dorite. Trebuie specificat înainte de a rula algoritmul k-means.
2. Centroid - Punctul central al unui cluster – este media tuturor punctelor din acel cluster.
3. Cluster - Grup de puncte similare, apropiate în spațiul caracteristicilor, care împărătesc un centroid comun.
4. Initializare aleatoare - Centroidele sunt alese aleatoriu la început. Poate afecta negativ rezultatul dacă initializarea este slabă.
5. K-Means++ - Metodă intelligentă de initializare a centroidelor, care reduce riscul de a ajunge la un minim local slab.
6. Convergenta - Algoritmul k-means oprește execuția când centroizi nu se mai schimbă sau modificările sunt sub un prag dat.
7. Invatare nesupervizata - Algoritmul k-means oprește execuția când centroidele nu se mai schimbă sau modificările sunt sub un prag dat.
8. K-partitie - Împărțirea setului de date în k clustere disjuncte.
9. K-configuratie - Setul centroidelor care definesc o anumită partitie.
10. Criteriul J Funcția obiectivă a algoritmului k-means: suma pătratelor distanțelor dintre fiecare punct și centroidul său.
11. Criteriul Elbow - Ajută la alegerea optimă a lui k analizând grafic scăderea erorii (inertia) în funcție de k. Punctul în care această scădere încetinește brusc (cotul) indică valoarea potrivită pentru k.

Figura 47: Algoritmul K-Means- termeni importanti

Dupa ce utilizatorul a studiat notiunile teoretice, poate trece la partea practica. Acest lucru se intampla prin apasarea butonului ”exemplu practic” care il va conduce catre o noua pagina unde va trebui sa introduca niste date informative (Figura 48) despre setul de date ce urmeaza o fi introdus. Pe noua pagina i se va cere sa introduca informatii despre numarul de instante ce va urma sa fie introdus si numarul de centroizi pe care algoritmul ii va folosi in realizarea clasificarii.

```
% if not n and not m %}
<h1>Introduceti cele doua numere</h1>
<form method="POST">
    <label for="n">Numarul de puncte: </label>
    <input type="number" name="n" id="n" required><br><br>

    <label for="m">Numarul de centroizi: </label>
    <input type="number" name="m" id="m" required><br><br>

    <button type="submit">Continuă</button>
</form>
```

Figura 48: Algoritmul K-Means- informatii despre dimensiunea setului de date

Dupa ce va introduce datele respective, acestea vor fi primite de backend si memorate in variabile separate. Pe pagina va aparea un nou tabel (Figura 49) ce va trebui completat cu setul de date, mai specific coordonatele punctelor si coordonatele centroizilor. Dupa ce utilizatorul a completat toate casutele tabelelor cu date, acestea vor fi trimise catre backend urmand sa fie modelate.

Introduceti coordonatele celor 4 puncte

i	x	y
1	1	2
2	1	4
3	3	5
4	3	2

Introduceti coordonatele celor 2 centroizi

i	x	y
1	1	1
2	2	2

Figura 49: Algoritmul K-Means- informatii despre date si centroizi

In momentul in care pe backend datele sunt primite, acestea sunt memorate in liste si variabile separate pentru o mai usoara prelucrare. Prima data se doreste crearea unui grafic pentru o mai buna vizualizare. Cu ajutorul unei functii ce foloseste libraria matplotlib, se creeaza un grafic corespunzator. Datele, adica coordonatele punctelor si ale centroizilor sunt trimise ca si parametri in liste separate, apoi functia creeaza graficul pe baza informatiilor primite. Punctele sunt reprezentate intr-un plan xOy, avand forma unui cerculet de culoare neagra. Apoi centroizii sunt reprezentati in acelasi plan dar cu o forma diferita, sub forma unui X de culoare rosie. S-au ales aceste culori pentru a se evidenta contrastul dintre cele doua tipuri de date diferite.

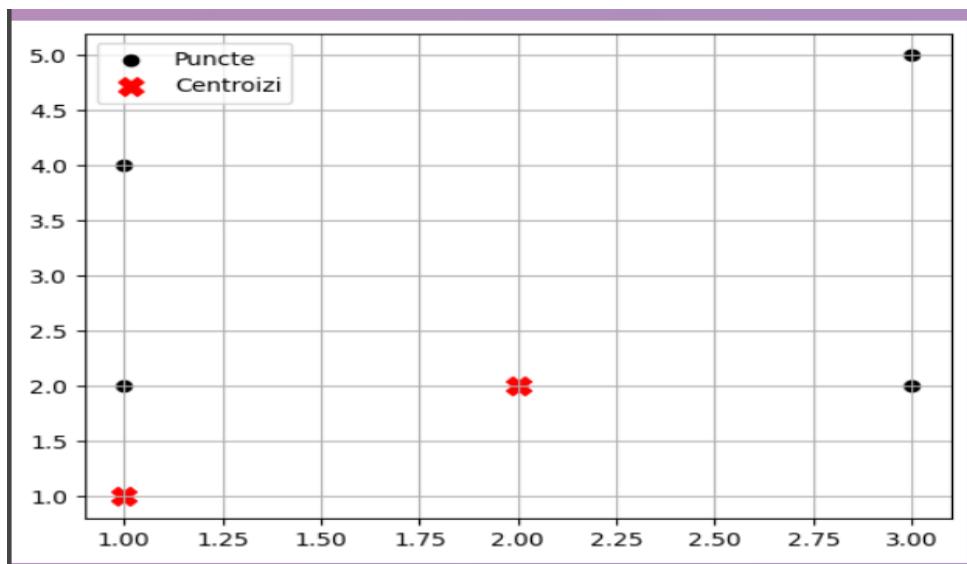


Figura 50: Algoritmul K-Means- reprezentarea grafica a datelor si centroizilor

Dupa ce imaginea este creata (Figura 50), este returnata intr-o forma comprimata si trimisa catre frontend unde va fi afisata in pagina.

Dupa ce graficul a fost creat, urmeaza modelarea datelor. Cu ajutorul unei functii separate de calculare a distantei dintre doua puncte ce primeste ca parametri coordonatele punctelor sub forma de liste, se va itera prin toate punctele pentru a se determina distantele dintre acestea si centroizi. Scopul este de a gasi centroidul cel mai apropiat de un punct si incadrarea punctului intr-un anumit cluster. Toate informatiile sunt memorate in liste care mai apoi vor fi trimise catre alta functie. Noua functie primeste coordonatele punctelor, ale centroizilor si componenta clusterelor. Functia va crea un nou grafic in care plaseaza punctele, acestea neschimbandu-si pozitia pe tot parcursul rularii algoritmului. In grafic sunt desenati si centroizii, acestia avand forma de X dar colorati diferit pentru a se deosebi. Pe langa asta, se mai adauga si o mediatoare intre centroizi pentru a se indica modul in care se grupeaza punctele.

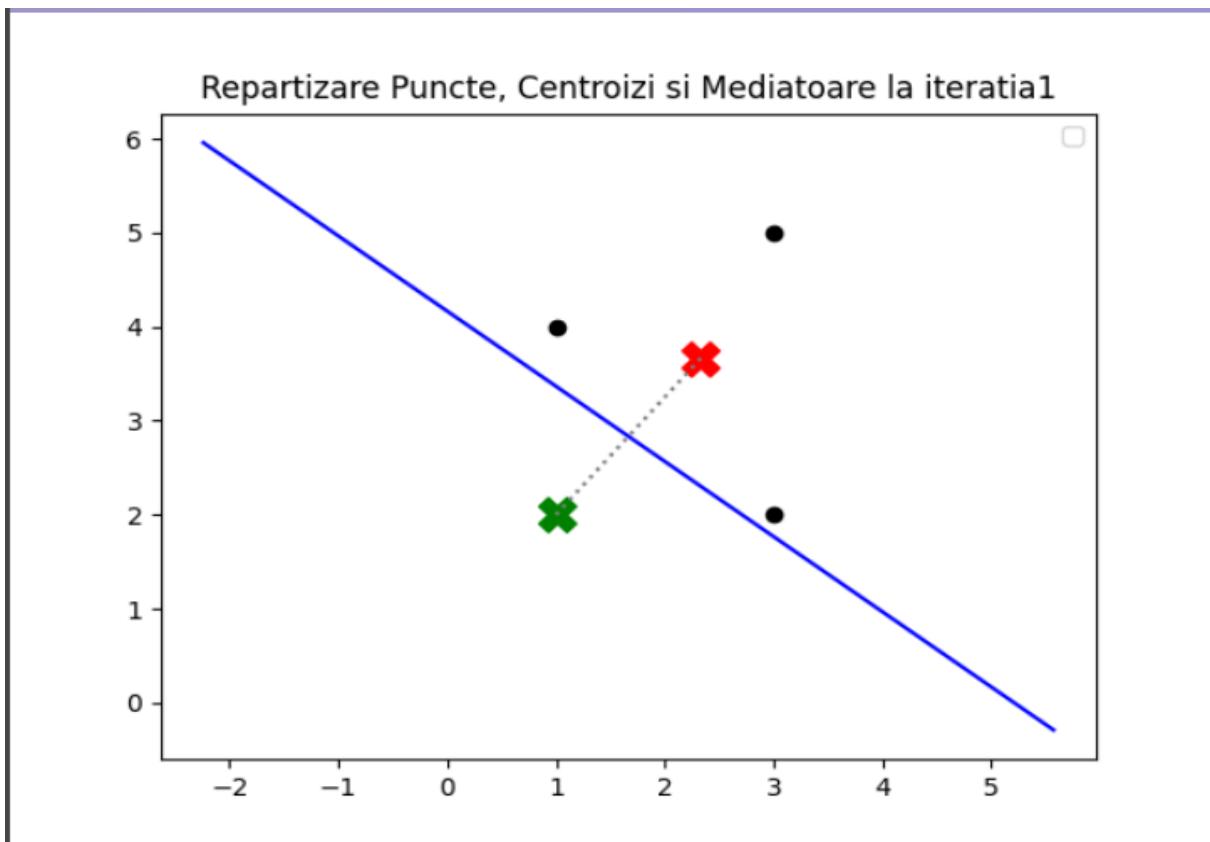


Figura 51: Algoritmul K-Means- reprezentarea grafica a centroizilor dupa o iteratie

Functia de desenare a graficului isi termina executia, urmand ca aceasta sa fie returnata intr-o forma comprimata. De pe backend este trimisa cu ajutorul librarii "render_template" catre frontend. Aici, cu ajutorul unei instructiuni de decizie se verifica daca imaginea a fost creata, exista si daca a fost primita. Odata ce conditia este indeplinita, imaginea poate si afisata in pagina web (Figura 51).

Pentru a se trasa aceste "delimitatoare" care indica grupurile si modul in care datele sunt grupate, se trage o linie dreapta intre doi centroizi iar pe linia respectiva se trage o mediatoare, adica o linie perpendiculara cu dreapta trasata care trece exact prin mijlocul acesteia. In functie de cati centroizi exista, numarul de mediatoare poate creste sau scadea.

Din moment ce toate datele sunt prelucrate, acestea sunt trimise pe frontend. Astfel, informatii precum componenta clusterelor insotita de coordonatele punctelor, pozitia noilor centroizi sau cardinalul multimilor sunt afisate in pagina.

• Algoritmul de Regresie Logistica

Algoritmul de Regresie Logistica este un algoritm de invatare supervizata care are ca scop precizarea probabilitatii ca un element sa apartina unei clase.

Modul de functionare este urmatorul: acest algoritm foloseste o functie speciala, numita functia sigmoid. Aceasta este o functie non-liniara care are ca scop transformarea output-ului modelului de regresie logistica intr-o probabilitate. Aceasta functie este aleasa in detrimentul altor functii deoarece este stabila, mai usor de interpretat si mult mai comună.

Alta functie utilizata este functia de log-verosimilitate. Aceasta are rolul de a maximiza scorul. Aceasta functie trateaza fiecare instanta din setul de date, comportandu-se diferit doar atunci cand output-ul difera. In aceasta formula intervine termenul de bias care este o valoare constanta(de obicei 1) care se aduna. Vectorul gradient, alta componenta importanta in aplicarea algoritmului, arata directia in care creste functia de log-verosimilitate. Acest vector are cardinalul egal cu numarul de instante din setul de date. Poate fi calculat ca fiind derivata partiala a functiei de log-verosimilitate in raport cu w. Matricea Hessiana este o matrice care contine derivatele de ordin doi si pe care se pot aplica alte metode de optimizare.

Odata ce functia de log-verosimilitate, vectorul gradient si matricea Hessiana au fost determinate, se pot face predictii pentru alte seturi noi de date. Pentru acest lucru, se initializeaza n ponderi cu 0, unde n reprezinta numarul de instante din setul de date, apoi folosindu-ne de valorile din vectorul gradient si una dintre metoda gradientului ascendent sau descendent, se poate aplica formula pentru a face predictia. Formula se aplica pana la convergenta. Valorile din noul vector obtinut dupa aplicarea formulei sunt inmultite cu valorile instantei ce se doreste a fi clasificata, apoi toate aceste produse sunt adunate. Daca suma reprezinta un numar negativ, atunci algoritmul va produce output-ul 0, iar daca suma este reprezentata de un numar pozitiv, output-ul produs de regresia logistica va fi 1.

In prima parte a fisierului HTML corespunzator acestui algoritm sunt prezentati pasii de executie ai algoritmului (Figura 52).

```
<p>
    Algoritmul de regresie logistică este folosit pentru clasificare binară (ex: da/nu, adevărat/fals, 0/1).
    El estimează probabilitatea ca o instantă să aparțină unei clase, folosind o funcție sigmoidă care transformă
    orice valoare într-o probabilitate între 0 și 1.<br><br>
    <br>Pasi cheie:<br>
    1.Se calculează o combinatie liniară a caracteristicilor (ex:  $z = w_0 + w_1 \cdot x_1 + \dots + w_n \cdot x_n$ ).<br>
    2.Se aplică funcția sigmoid:  $\sigma(z) = 1 / (1 + e^{-z})$ , care returnează o probabilitate între 0 și 1.<br>
    3.Dacă probabilitatea este  $\geq 0.5$ , se clasifică ca 1; altfel, ca 0.<br>
    4.Se ajustează coeficientii folosind funcția de cost (log-loss) și algoritmul de gradient descent.<br>
</p>
```

Figura 52: Regresia Logistica- modul de executie

Acestia sunt pasii elementari pe care algoritmul de regresie logistica ii aplica in oferirea raspunsului final. A fost prezentata ideea de baza a algoritmului insotita de pasii de rulare enumerati in patru idei simple.

Ca si la algoritmii precedenti, si la acest algoritm s-au pus in evidenta cateva versete care au o legatura cu modul algoritmului de a functiona (Figura 53). Acestea au rolul de a oferi o asemnanare intre cele doua oferind si o interpretare a versetului respectiv dar lasand si utilizatorul sa vina cu propria interpretare. Acestea au fost adaugate cu scopul de a adauga aplicatiei un strop de originalitate si pentru a se diferenția de alte aplicatii care ofera informatii despre algoritmii de Machine Learning.

```
+Versete biblice care reflecta principiile Regresiei Logistice:<br>
1. Luca 16:13 <br>
„Niciun slujitor nu poate sluji la doi stăpâni. Căci ori îl va urî pe unul și-l va iubi pe celălalt, ori se va atașa de unul și-l va disprețui pe celălalt. Nu puteti sluji și lui Dumnezeu, și banilor.” <br>
Legătura cu Regresia Logistică:<br>
Regresia logistică nu permite ambiguități: nu poti fi în două clase în același timp. Odată calculată probabilitatea, punctul este atribuit clar unei clase.<br>
<br><br>

2. Proverbe 4:26<br>
„Netezeste cărarea picioarelor tale și toate căile tale să fie bine hotărâte!”<br>
Legătura cu Regresia Logistică:<br>
Modelul logistic „netezeste” decizia folosind o funcție matematică (sigmoidă), astfel încât fiecare pas spre o clasă este clar și bine determinat.
<br><br>
```

Figura 53: Regresia Logistică- modul de executie

In penultima sectiune a paginii (Figura 54) se afla termeni importanti folositi atunci cand se aplica acest algoritm. Acesti termeni au fost explicati pe cat de simplu posibil si intr-un mod putin mai informal. Printre termenii explicati se regasesc functia sigmoid, bias-ul, gradientul descendente sau overfitting-ul.

```
Termeni utilizati<br>
1.<b>Functia sigmoid</b>- Transformă orice valoare reală într-o probabilitate între 0 și 1.<br>
2.<b>Greutati (weights)</b>- Coeficientii  $w_{1}, w_{2}, \dots, w_{n}$  asociati fiecarei caracteristici, sunt invatati in timpul antrenarii.<br>
3.<b>Bias(termen liber)</b>- Constanta  $w_0$  adaugata la combinatia liniara care deplaseaza functia de decizie.<br>
4.<b>Functia de cost</b>- Măsoară cât de departe sunt predictiile de valorile reale. Este convexă și favorizează probabilități apropiate de eticheta reală.<br>
5.<b>Gradient descendente</b>- Algoritm de optimizare folosit pentru a minimiza functia de cost prin actualizarea treptată a greutătilor.<br>
6.<b>Overfitting</b>- Situatie în care modelul se potriveste prea bine pe datele de antrenare și generalizează prost pe date noi.<br>
```

Figura 54: Regresia Logistică- termeni importanti

Acesti termeni au un rol foarte important, necunoasterea lor duce la confuzii si neintelegeri in explicatiile oferite mai tarziu de aceasta aplicatie web.

In ultima sectiune a paginii (Figura 55), este oferit un set de date ca exemplu. Setul de date este preluat din cartea "Exercitii de invatare automata" scrisa de Dr. Liviu Ciortuz. Acesta se poate folosi de utilizator pentru a vedea cum ruleaza algoritmul.

i	x ₁	x ₂	x ₃	x ₄	y
1	1	0	0	0	1
2	1	0	1	0	1
3	0	1	0	1	1
4	0	0	0	1	0
5	1	1	1	0	0
6	1	0	1	1	0
7	1	0	0	1	0
8	0	1	0	0	0

Dataset preluat din cartea „Exerciții de învățare automată” de Dr. Liviu Ciortuz.

Figura 55: Regresia Logistica- dataset oferit ca exemplu

Dupa ce utilizatorul a parcurs pagina cu informatii teoretice, poate trece la pagina urmatoare. Acest lucru se face prin apasarea butonului "exemplu practic" care il va conduce catre o noua pagina unde va trebui sa ofere anumite informatii. Aici va trebui sa completeze doua chenare ce reprezinta numarul de linii si de coloane din tabelul ce il va da mai tarziu ca input. In momentul in care completeaza aceste date, ele vor fi trimise catre backend si memorate in doua variabile separate. Din acest punct, utilizatorului ii va fi dat un nou tabel de completat, anume cel cu setul de date (Figura 56). Aici va avea la dispozitie n linii si m+1 coloane, cele m coloane fiind acordate variabilelor x iar ultima coloana variabilei y care are valoarea 0 sau 1. Dupa ce completeaza acest tabel, datele vor fi preluate de backend care le va memora in liste separate.

i	x1	x2	y
1	1	1	1
2	1	0	1
3	0	0	1

Figura 56: Regresia Logistica- dataset oferit ca input

Din acest punct, datele incep sa fie prelucrate. Tot in acelasi timp pe frontend incep sa apara informatii precum formule, notatii si explicatii.

```
%if elements%
| <p>Primul pas este cel de calculare a functiei de log-verosimilitate. Formula este urmatoarea:</p>
<p>
\[
\ell(w) = \sum_{i=1}^n y^{(i)} \ln \sigma(w \cdot x^{(i)}) + (1 - y^{(i)}) \ln(1 - \sigma(w \cdot x^{(i)}))
\]
</p>

<p>Deoarece in tabelul nostru avem {{n}} instante, formula va fi urmatoarea:</p>
<p>
\[
\ell(w) = \sum_{i=1}^{{n}} y^{(i)} \ln \sigma(w \cdot x^{(i)}) + (1 - y^{(i)}) \ln(1 - \sigma(w \cdot x^{(i)}))
\]
</p>
```

Figura 57: Regresia Logistica- informatii scrise in HTML

Formulele sunt scrise si prelucrate in HTML cu ajutorul librarii mathjax, o librarie care permite afisarea in pagina de simboluri si notatii matematice (Figura 57). Aceste informatii incep sa apara in pagina (Figura 58) odata cu prelucrarea datelor pe backend. Pana cand nu se incepe prelucrarea datelor si primele date modelate nu sunt trimise de backend, in pagina nu se va afisa absolut nimic, nici formula matematica care nu depinde la nivel teoretic de natura datelor.

Prin primul pas este cel de calculare a functiei de log-verosimilitate. Formula este urmatoarea:

$$\ell(w) = \sum_{i=1}^n y^{(i)} \ln \sigma(w \cdot x^{(i)}) + (1 - y^{(i)}) \ln(1 - \sigma(w \cdot x^{(i)}))$$

Deoarece in tabelul nostru avem 3 instante, formula va fi urmatoarea:

$$\ell(w) = \sum_{i=1}^3 y^{(i)} \ln \sigma(w \cdot x^{(i)}) + (1 - y^{(i)}) \ln(1 - \sigma(w \cdot x^{(i)}))$$

Variabila y ia valorile 0 sau 1. Asta inseamna ca pentru fiecare instant din tabel, formula va fi $y^{(i)} \ln \sigma(w \cdot x^{(i)})$ atunci cand y este egal cu 1 si $(1 - y^{(i)}) \ln(1 - \sigma(w \cdot x^{(i)}))$ atunci cand y este 0. De mentionat faptul ca $y^{(i)}$ reprezinta valoarea lui y de pe linia i. A nu se confunda $y^{(i)}$ cu y^i (y la puterea i)!

Pentru a determina functia de log-verosimilitate, parcurgem fiecare linie(instanta) din tabel si aplicam formula.

Linia 1 din tabel:

Deoarece y=1, vom folosi prima parte a formulei

Figura 58: Regresia Logistica- informatii afisate in pagina

Pe pagina vor aparea informatii precum formule corespunzatoare functiei de log-verosimilitate, a vectorului gradient, a functiei de cost sau pentru matricea Hessiana. Pe langa formule, se va explica cum aceste formule se aplica pe fiecare linie din setul de date si care este rezultatul final.

• Algoritmii Bayes Naiv si Bayes Optimal

Algoritmii Bayes Naiv si Bayes Optimal sunt doi algoritmi de invatare automata de clasificare care se bazeaza pe probabilitati in oferirea unui raspuns.

In cazul algoritmului Bayes Naiv se foloseste Teorema lui Bayes si se face presupunerea ca toate atributele sunt independente intre ele, avand rolul de a simplifica cat mai mult si de a face calculul unul cat mai simplu. In schimb, in cazul algoritmului Bayes Optimal, acesta ia in calcul absolut toate probabilitatile pentru a oferi un rezultat cat mai precis. Din punct de vedere al corectitudinii si acuratetii, Bayes Optimal este cel mai bun. Dar din punct de vedere al complexitatii, Bayes Naiv este preferat in detrimentul acestuia.

In cazul celor doi algoritmi, odata ce probabilitatile au fost calculate, se poate incepe determinarea predictiei pentru o instanta ce se doreste a fi clasificata. Se calculeaza doua probabilitati, una cand noua instanta ar apartine clasei 0 si inca o probabilitate atunci cand noua instanta ar apartine clasei 1. Daca una dintre clase are o probabilitate mai mare decat cealalta, atunci aceasta clasa va oferi output-ul algoritmului, adica in ce clasa poate fi incadrata noua instanta. Cateodata, din cauza setului de date, se intampla ca ambele probabilitati sa fie egale. In acest caz algoritmul nu poate decide daca noua instanta apartine clasei 0 sau 1. Pentru a rezolva aceasta problema se foloseste o metoda numita Laplace. Aceasta metoda adauga constanta 1 la numarator si mai aduna o constanta la numitorul fractiei, constanta care reprezinta numarul total de valori posibile pentru acea variabila. Cu ajutorul acestei metode este evitat cazul in care avem doua probabilitati egale si algoritmul nu poate decide carei clase apartine instanta ce se vrea a fi clasificata. Tot cu aceasta metoda este evitat si cazul in care in calculul probabilitatilor se poate ajunge ca una dintre ele sa fie 0.

In cazul celor doi algoritmi difera doar modul in care sunt calculate probabilitatile. Dezavantajul major al algoritmului Bayes Naiv este ca de cele mai multe ori presupunerea de independenta conditionata este falsa, asta ducand la rezultate imprecise si o acuratete mai slaba decat a altor algoritmi de clasificare utilizati in machine learning. In schimb, acest algoritm compenseaza cu rapiditatea si usurinta calculelor. Algoritmul Bayes Optimal are ca dezavantaj spatiul de memorie pe care il foloseste deoarece ia in calcul toate probabilitatile, el neluand in calcul presupunerea independentei conditionate intre variabile. Cu toate ca ofera rezultate mult mai precise, in practica nu este preferat din cauza resurselor utilizate.

Ca si in cazul algoritmilor precedenti, paginile HTML ale celor doi algoritmi au ca prima sectiune o parte unde algoritmul este explicitat pe scurt. Este prezentata ideea principala a algoritmului insotita de pasii de rulare enumerati pe scurt.

Ca si la ceilalti algoritmi, si la acestei doi algoritmi au fost adaugate versete biblice insotite de interpretare dar care lasa loc si interpretarilor utilizatorului. Acestea au rolul de a ajuta persoana care doreste sa invete din aceasta aplicatie sa retina mai usor, de a oferi un prilej ca utilizatorul sa isi readuca aminte mai bine si mai repede anumite informatii. Acestea au scop educational si faciliteaza invatarea si memorarea mai simpla, dar care vrea sa ocoleasca memorarea fortata(robotica). Ambele pagini dedicate acestor algoritmi ofera explicatii pentru cuvinte cheie, termeni importanti, fara de care explicarea iteratiilor algoritmilor ar fi una destul de dificila.

In momentul in care utilizatorul a parcurs aparatul teoretic si il stapaneste, poate trece la pagina urmatoare, cea dedicata prelucrarii de date, aplicarea algoritmului si oferirea de informatii si explicatii.

Pe noua pagina (Figura 59) unde va fi redirectionat va trebui sa asigneze variabilelor n si m doua valori. Variabila n reprezinta cate linii va avea tabelul iar variabila m ne spune cate coloane. Dupa ce aceste date au fost introduse, ele sunt trimise catre backend unde sunt memorate in doua variabile diferite. Urmeaza ca utilizatorul sa completeze un nou tabel, de dimensiune n*m. Pe langa aceste date, mai trebuie completate si informatii despre instanta ce se doreste a fi clasificata.

Completați tabelul cu 2 rânduri și 2 coloane + 1 coloană K		
x1	x2	K
1	1	1
1	0	1

Introduceți un exemplu de predicție (fără K):

0	0
---	---

Figura 59: Bayes Naiv si Bayes Optimal- set de date oferit ca input

Din moment ce toate datele au fost introduse, ele vor fi trimise catre backend. Aici sunt memorate si prelucrate in variabile diferite. Se folosesc cat mai multe liste in care se memoreaza diferite etape ale calculelor care mai apoi sunt trimise pe frontend. Aici sunt parcurse cu ajutorul instructiunilor repetitive si afisate in pagina.

Formula pentru predicție

Deoarece clasificatorul *Bayes Optimal* nu lucreaza cu presupunerea de independenta conditionala a atributelor de intrare in raport cu atributul de iesire, el va face predictia folosind formula de mai jos:

$$\hat{y}_{JB} = \operatorname{argmax}_{k \in \{0,1\}} P(K=k) P(x_1=1, x_2=0 | K=k)$$

Deoarece $k \in \{0,1\}$, formula de mai sus va avea doua forme:

$$p_0 = P(x_1=1, x_2=0 | K=0) \cdot P(K=0)$$

$$p_1 = P(x_1=1, x_2=0 | K=1) \cdot P(K=1)$$

In calcularea p_0 ne vom folosi de valorile din tabel:

$$p_0 = P(x_1=1, x_2=0 | K=0) \cdot P(K=0) =$$

$$= 0 \cdot 0 = 0.0$$

In calcularea p_1 ne vom folosi de valorile din tabel:

$$p_1 = P(x_1=1, x_2=0 | K=1) \cdot P(K=1) =$$

$$= 1 \cdot 1 = 1.0$$

Deoarece $p_0 < p_1$ ($0.0 < 1.0$), atunci clasificatorul *Bayes Optimal* va prezice $K=1$ pentru instanta data cu probabilitatea:

$$P(K=1 | x_1=1, x_2=0) =$$

$$= \frac{p_1}{p_1 + p_0} = \frac{1.0}{1.0 + 0.0} = 1.0$$

[Go Back Home](#)

Figura 60: Bayes Naiv si Bayes Optimal- explicatii oferite in pagina

7. Interfata cu utilizatorul

• Prezentarea UI

Interfata grafica este una dintre cele mai importante lucruri atunci cand vorbim de o pagina web. Chiar daca pagina are o functionalitate foarte buna, o acuratete si performanta ridicata, ofera intr-un timp foarte scurt informatiile necesare, dar daca informatiile sunt doar simplu afisate, aranjate intr-un mod dezordonat si doar scris alb pe negru nu va atrage prea multa lume.

In contextul acestei aplicatii, tot ce tine de interfata grafica a putut fi creat cu ajutorul limbajului CSS (Cascade Style Sheets). Aceste este cel care a oferit culoare, dimensiune, aranjare in pagina a obiectelor, oferirea unor forme frumoase ale tabelelor sau crearea de animatii pe butoane.

Ca si tema de fundal s-a optat folosirea unui gradient (Figura 61). Gradientul este o combinatie de culori specificate de programator care sunt imbinante intr-un mod placut.



Figura 61: Interfata grafica- gradient

Gradientul are mai multe functionalitati, precum cea de a inclina culorile intr-un anumit unghi sau cea care da senzatia ca gradientul se misca. Senzatia de miscare este creata in felul urmator: se creeaza un gradient care este mult mai mare decat dimensiunea ecranului, de regula de trei sau patru ori mai mare, dupa care elementul pe care se doreste sa se aplice gradientul este fixat iar in spatele acestuia gradientul executa o miscare in functie de unghiul specificat. In paginile unde trebuie introduse date despre setul de date ce urmeaza sa fie introdus se poate vedea mai bine cum apare efectul de gradient miscator.

Toate butoanele din aplicatie, cele care duc catre pagina cu notiuni teoretice, cel de schimbare a temei, cel care trimit utilizatorul catre pagina cu exemplul practic sau pagina Home, au animatii. Butoanele (Figura 62) sunt rotunjite la colturi si au efectul de hover, adica atunci cand cursorul mouse-ului este dus pe butonul respectiv, acesta ori isi schimba culoarea ori dimensiunea.



Figura 62: Interfata grafica- butoane

• Explicarea functionalitatilor interactive

Pentru functionalitatile interactive ale acestei pagini se regasesc efectul de gradient miscator, efectul de hover la butoane si butonul de schimbare a temei. Primul a fost explicat la pasul anterior. Efectul de hover se realizeaza prin adaugarea unui tag atributului pe care dorim sa il stilizam. Apoi, in pagina CSS de care este legata pagina HTML, se adauga efectul de hover (Figura 63) atributului prin specificarea tag-ului, procesul fiind unul foarte simplu.

```
81  tr:hover {  
82      background-color: rgba(255, 255, 255, 0.2);  
83      transition: background-color 0.3s ease;  
84  }  
85  ⚡
```

Figura 63: Interfata grafica- efectul hover

Una dintre functionalitatile care au adus o imbunatatire destul de placuta pe partea grafica este adaugarea unui buton cu ajutorul caruia utilizatorul poate schimba tema aplicatiei. In fisierul CSS asociat paginii respective este creat un template care doar ofera paginii o aranjare placuta a cuvintelor, tabelelor si butoanelor. Pe langa acest template mai sunt adaugate cateva cazuri care cuprind atribute ce determina o anumita culoare. Utilizatorul are la alegere intre mai multe teme, teme de fundal ce sunt memorate intr-o lista. In momentul intrarii pe pagina, tema de fundal va fi mereu prima din lista. In momentul in care butonul de schimbare a temei este apasat, indexul este crescut cu o unitate, trecand la urmatoarea tema. Acest lucru este facut cu ajutorul unui script scris in JavaScript si integrat in codul HTML. Astfel, utilizatorului ii sunt puse la dispozitie mai multe teme la alegere. In mod implicit, tema aplicatiei este una mai intunecata, scopul fiind de a nu oferi un disconfort ochilor din cauza luminii prea puternice. Daca se prefera o tema mai luminoasa, se poate schimba foarte usor intr-o tema cu imagini deschise si cu mult alb (Figura 64).

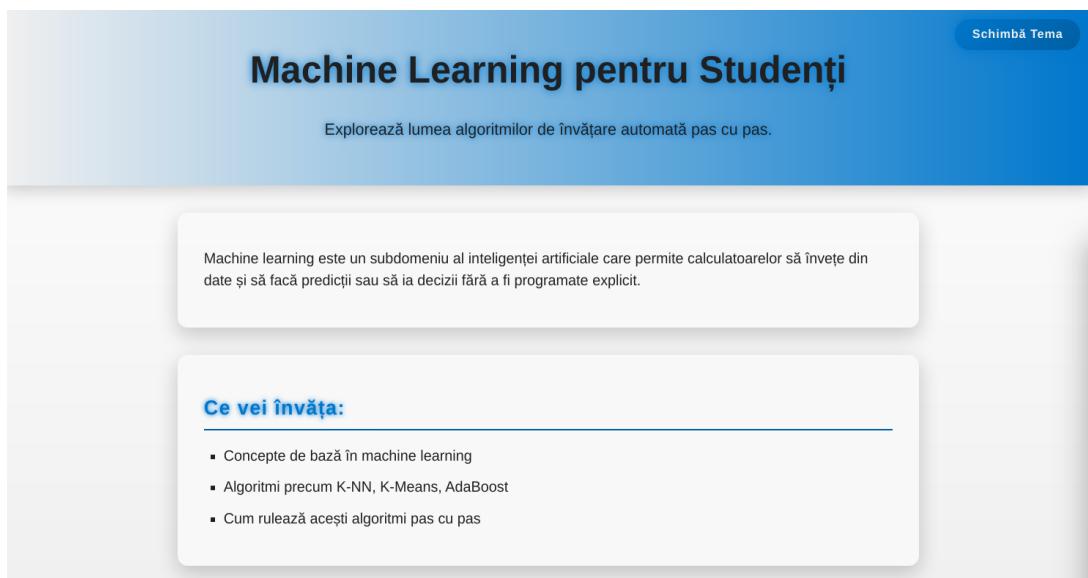


Figura 64: Interfata grafica- efectul hover

8. Testare si validare

• Modul de testare a aplicatiei

Pe parcursul dezvoltarii acestei aplicatii a fost necesara si faza de testare si validare a codului. Aceasta faza s-a desfasurat in mai multe etape:

-Testarea functionala: Functiile care au ajutat in procesul de prelucrare al datelor au fost testate separat. Unul dintre cazuri este testarea la algoritm k-NN a functiei de calculare a distantei. Aceasta functie primea patru parametri, primii doi reprezentant coordonatele primului punct iar ultimele doua corespunzatoare celui de-al doilea punct. Separat, pe o foaie, s-au luat mai multe cazuri, adica mai multe combinatii de coordonate de diferite dimensiuni si semn, si s-a aplicat formula. In cele ce a urmat, s-a verificat daca rezultatele de pe foaie coincid cu cele oferite de functia scrisa in Python.

-Testare de compatibilitate: Aplicatia a fost testata daca e compatibila pe mai multe sisteme de operare. Mai intai, codul a fost incarcat pe Github iar link-ul aferent acestui repository a fost adaugat pe Render. Pe aceasta platforma s-a putut hosta aplicatia pe web, astfel incat sa aiba cat mai multa lume acces la ea. Din momentul in care s-a terminat procesul de deployment, aplicatia a putut fi testata de pe telefon cu sistemul de operare Android si de pe laptopuri cu sistemele de operare Windows si Linux.

-Testare de interfata: Etapa de creare a interfetei grafice a fost lasata la urma. Dupa ce a fost scris codul CSS corespunzator fisierelor HTML, s-a verificat daca aplicatia ofera interfata dorita. Au fost necesare unele modificari din anumite cauze precum culoarea anumitor zone sau obiecte care nu se potriveau cu fundalul, asezarea in pagina a anumitor elemente, unele pasaje de text erau prea ingramadite sau aplicatia nu oferea efectul de responsiveness.

In procesul de testare si intalnire a anumitor bug-uri si erori, au fost necesare cateva actiuni precum modificarea integrala a unei bucati mari de cod, renuntarea la anumite obiective din realizarea aplicatiei sau memorarea de cod din diferite stari ale evolutiei sale.

• Testarea rezultatelor algoritmilor

De fiecare data ce o functionalitate noua era introdusa sau un algoritm era gata de implementat, a fost necesara testarea corectitudinii codului ce a fost implementat. Prima data am testat algoritmi oferind seturi mici de date pe care am aplicat algoritmul pe hartie. Au aparut si situatii in care rezultatele nu coincideau si a fost necesara revizuirea codului si a logicii implementarii. Dupa ce algoritmul a oferit rezultatul asteptat pe seturi mici de date, am trecut si la seturi de date mai mari. Aceste seturi au fost preluate din carti de specialitate precum cartea "Exercitii de invatare automata" scrisa de Dr. Liviu Ciortuz sau din cartea "Machine Learning" scrisa de Tom Mitchell.

x_i	X_1	X_2	y_i
x_1	1	2	+1
x_2	2	3	+1
x_3	3	4	-1
x_4	3	2	-1
x_5	3	1	-1
x_6	4	4	-1
x_7	5	4	-1
x_8	5	2	+1
x_9	5	1	+1

Figura 65: Set de date pentru testare

In figura 65 este un set de date preluat din cartea "Exercitii de invatare automata" de Dr. Liviu Ciortuz de la sectiunea "Algoritmul AdaBoost", iar al doilea dataset (Figura 66) este din cartea "Machine Learning" de Tom Mitchell de la sectiunea "Retele neuronale".

Instance	Classification	a_1	a_2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

Figura 66: Set de date pentru testare

Primul set de date a fost folosit de multe ori in testarea corectitudinii algoritmului implementat de mine, verificand rezultatele din carte cu cele oferite de programul meu. Pe prima pagina, pagina Home, la sectiunea "Resusrse utile" este oferit un link catre o pagina numita Kaggle, pagina pe care se pot gasi seturi de date mult mai mari si consistente. Pe acest site se pot gasi date reale precum date despre boli medicale, informatii despre automobile sau despre social-media.

9. Concluzii si directii viitoare

- Ce am invatat din proiect, ce poate fi imbunatatit si propuneri pentru extindere**

Lucrul cel mai important din acest proiect pe care l-am invatat a fost cum sa imi gestionez timpul. Este important ca in dezvoltarea unui proiect de dimensiuni mai mari cum este acesta, sa se imparta task-urile in subtask-uri care sa fie rezolvate zilnic. Cu cat de mult se amana mai mult finalizarea unui task, cu atat apar si unele probleme si erori in cod datorate presiunii de a termina ceva inceput cat mai repede. Am invatat ca scrierea de cod insotita de mici pauze ajuta mult iar lucrarea zilnica la proiect ajuta deoarece esti "tinut in priza" cu proiectul. Amanarea unei bucati de cod sa fie terminata in alta zi este o problema mare din cauza faptului ca se pot uita anumite detalii ce nesesita sa fie implementate sau se pierde timp in a intelege ce a fost scris pana in un moment dat.

De asemenea, am reusit sa invat sa lucrez cu un framework nou si sa fac o pagina web sa raspunda in mod dinamic in functie de anumiti parametri. Am mai invatat cum se creeaza grafice cu ajutorul unor librarii, iar la nivel teoretic mi-am imbunatatit cunoostintele despre algoritmii de Machine Learning si mi-am fixat anumiti termeni importanti si cuvinte cheie. Una dintre marile provocari a fost host-area aplicatiei pe web, din care am invatat cum o aplicatie ajunge sa fie publicat pe internet.

Aplicatia nu este perfecta, ea putand fi imbunatatita in multe moduri. Eu as imbunatati modul in care unele iteratii ale algoritmilor sunt explicate, adaugarea de informatii suplimentare acolo unde nu sunt de ajuns sau modul de afisare in pagina. Pe partea de cod, o imbunatatire majora ar fi adusa de fragmentarea codului de pe backend, fiecarui algoritm sa ii fie dedicata o pagina speciala unde poate fi scris cod Python si chiar intretinut, imbunatatit si scalat.

Ca si propuneri pentru extindere, eu consider ca adaugarea si altor algoritmi decat cei prezentati ar fi o imbunatatire majora. Adaugarea unor pagini interactive unde utilizatorii pot completa rebusuri cu cuvinte cheie utilizate in acest domeniu al informaticii sau pagini cu lucruri interesante din viata de zi cu zi in care acesti algoritmi si-au pus amprenta. O alta imbunatatire ar fi aceea de a memora seturile de date introduse de utilizatori intr-o baza de date ca pe urma cand alti utilizatori vor sa completeze cu un set de date pe care sa se aplice unul dintre algoritmi, el sa primeasca o sugestie de dataset bazat pe cel al celorlalți utilizatori. S-ar mai putea adauga si imagini cu animatii din care sa se vada cum se misca punctele de la o iteratie la alta sau diferenta intre iteratii. Pentru cei care doresc sa faca mai multe lucruri in acelasi timp, s-ar putea adauga o functie care citeste textul de pe paginile cu notiuni teoretice. Nu este recomandata aceasta functie deoarece unii utilizatori nu ar fi atenti, dar este o functie care ar aduce o noutate paginii si poate un moment de amuzament.

Din punctul meu de vedere, dezvoltarea acestei aplicatii m-a ajutat mult deoarece am invatat lucruri utile, notiuni teoretice si modul de lucru cu diferite unelte din domeniul programarii. Cu toate ca nu este perfecta si i se mai pot aduce imbunatatiri, este o aplicatie de care sunt multumit deoarece lipsa intr-o companie la o aplicatie de acest nivel s-a simtit. Este o aplicatie de care imi voi aduce aminte cu drag si pe care o voi adauga in CV-ul de angajare.

10. Bibliografie

• Carti, cursuri, articole online, tutoriale

- <https://www.bibliaortodoxa.ro/>
- https://ro.wikipedia.org/wiki/nvare_automat
- <https://www.sap.com/romania/products/artificial-intelligence/what-is-machine-learning.html>
- https://en.wikipedia.org/wiki/Reinforcement_learning
- https://ro.wikipedia.org/wiki/%C3%8Env%C4%83%C8%9Bare_automat%C4%83
- <https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algo>
- <https://lazyprogrammer.me/mlcompendium/ensemble/adaboost.html>
- [https://en.wikipedia.org/wiki/Entropy_\(information_theory\)](https://en.wikipedia.org/wiki/Entropy_(information_theory))
- https://www.researchgate.net/figure/Advantages-and-disadvantages-of-tbl1_374795587
- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- <https://keylabs.ai/blog/k-nearest-neighbors-knn-real-world-applications>
- [https://en.wikipedia.org/wiki/Elbow_method_\(clustering\)](https://en.wikipedia.org/wiki/Elbow_method_(clustering))
- <https://developers.google.com/machine-learning/clustering/kmeans/advantages-disadvantages>
- <https://towardsdatascience.com/three-versions-of-k-means-cf939b65f4ea>
- <https://medium.com/@karan.kamat1406/how-logistic-regression-works-theory-practical-examples-10c3a2a2a2d>
- <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/>
- <https://www.geeksforgeeks.org/python/flask-tutorial/>
- <https://doxologia.ro/biblia-ortodoxa>
- https://biblia.resursecrestine.ro/?gad_source=1&gad_campaignid=895007671&gbraids=0AAAAADJNkXJbvJX054ZX02jGmq3EFWcOQ&gclid=Cj0KCQjwgvn-OFpq5UTokaAoKpEALw_wcBhttps://www.crestinortodox.ro/dogmatica/dogma/interpretarea-sfintei-scripturi-68921.html
- <https://www.crestinortodox.ro/dogmatica/dogma/interpretarea-sfintei-scripturi-68921.html>
- <http://www.softwaretesting.ro/Romana/Files/TestMethods/Software%20Testing%20Methods.html>

- <https://blogdeit.ro/7-idei-pentru-un-cod-de-testare-automata-eficienta/>
- <https://uncoded.ro/testarea-automata-a-codului-in-aplicatii-python/>
- <https://www.zaptest.com/ro/ce-este-testarea-functională-tipuri-exemple/>
- <https://www.geeksforgeeks.org/python/how-to-use-css-in-python-flask/>
- <https://stackoverflow.com/questions/22259847/application-not-picking-up-css-style-sheets>
- <https://flask.palletsprojects.com/en/stable/tutorial/static/>
- <https://pythonhow.com/python-tutorial/flask/Adding-CSS-styling-to-your-flask-app/>
- <https://medium.com/an-idea/beautify-flask-web-app-using-css-html-d574a6a2a2d>
- <https://blogdeit.ro/7-idei-pentru-un-cod\protect\penalty\z@-de-testare\protect\penalty\z@-automata-eficienta/>
- <https://pythonhow.com/python-tutorial/flask/HTML-templates-in-flask/>
- <https://stackoverflow.com/questions/74962606/how-to-use-flask-for-rendering-templates>
- <https://www.geeksforgeeks.org/python/flask-rendering-templates/>
- <https://matplotlib.org/>
- https://www.w3schools.com/python/matplotlib_intro.asp
- https://matplotlib.org/stable/gallery/user_interfaces/web_application_server_sgskip.html
- <https://www.geeksforgeeks.org/python/create-scatter-charts-in-matplotlib/>

