IMPLEMENTAREA ALGORITMILOR DIN INTELIGENȚA ARTIFICIALĂ PENTRU DEPĂȘIREA DECALAJULUI DE COMUNICARE ÎNTRE PERSOANELE CU DIZABILITĂȚI (SURDO-MUȚI) ȘI OAMENII FĂRĂ DIZABILITĂȚI.

Dragoș COJOCARI

Departamentul Ingineria Software și Automatică, TI-214, Facultatea calculatoare, informatică și microelectronică, Universitatea Tehnică a Moldovei, Chișinău, Moldova

Îndrumător/coordonator științific: Veronica ANDRIEVSCHI-BAGRIN, asist. univ., UTM

Rezumat. *În articolul dat se va analiza posibilitatea de a integra algoritmii din Inteligența Artificială pentru a ajuta la depășirea decalajului de comunicare între persoanele cu dizabilități (surdo-muți) și oamenii fără dizabilități. Ca exemplu a fost ales limbajul de semne americane (ASL), el fiind cel mai des folosit în lume de oamenii surdo-muți. Pentru realizarea scopului dat se va folosi algoritmul "Support Vector Machine" (SVM) din Machine Learning (ML). Toate ecuațiile prezente sunt deja bine cunoscute, demonstrațiile vor fi ilustrate grafic și reprezentate prin exemple. În articol de asemenea este prezentată exactitatea prezicerii algoritmului și explicarea cauzei erorilor și inexactități algoritmului în unele momente. Realizarea algoritmului se va face în limbajul de programare Python cu ajutorul bibliotecilor "Mediapipe", "scikit-learn", "OpenCV" si "Turtle".*

Cuvinte cheie*: Support Vector Machine, Inteligenta Artificiala, Python, limbajul de semne americane*

Introducere

Potrivit Organizației Mondiale a Sanatatii (OMS), peste 5% din populația lumii suferă de dizabilități ale auzului și surditate. Decalajul de comunicare dintre persoanele fără dizabilități și surdo-mute poate prezenta provocări pentru o comunicare eficientă, surdo-muții pentru comunicare se bazează pe limbajul semnelor, cum ar fi ASL, iar oamenii fără dizabilități se bazează pe convorbirea verbală. Persoanele fără dizabilități pot să nu fie familiarizate cu forma de comunicare prin gesturi, ceea ce poate duce la ne înțelegere și dificultăți în comunicare, pentru a depăși acest decalaj se poate folosi algoritmul SVM pentru a transforma gesturile în text în timp real.

În cele ce urmează ne interesează:

- să minimizăm timpul de răspuns al aplicației;
- exactitatea algoritmului să fie cât mai mare.

Descrierea algoritmului

În ML SVM face parte din modelele cu "Învățarea supravegheată" (SL), ceea ce înseamnă că pentru a antrena astfel de model trebuie să-i transmitem un set de date cu N atribute, în care valorile au rezultatul predefinit, iar algoritmul deja la rândul său va modifica parametrii săi astfel încât să ajungă la rezultatul dorit de noi [1].

Scopul principal al algoritmului SVM este de a găsi un hiperplan optimal într-un domeniu N-dimensional. Hiperplanul pentru domeniul N-dimensional se află în N-1 dimensiune, ca spre exemplu pentru planul bidimensional hiperplanul va fi o linie prezentată în figura 1, iar pentru planul tridimensional hiperplanul va fi un plan bidimensional prezentat în figura 2.

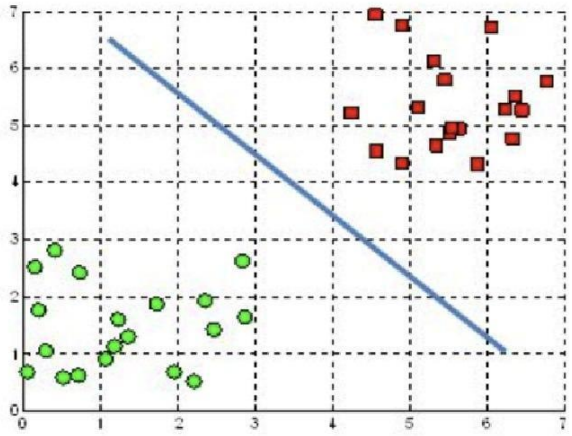

Figura 1. Hiperplanul optim pentru modelul cu două atribute (plan bidimensional)

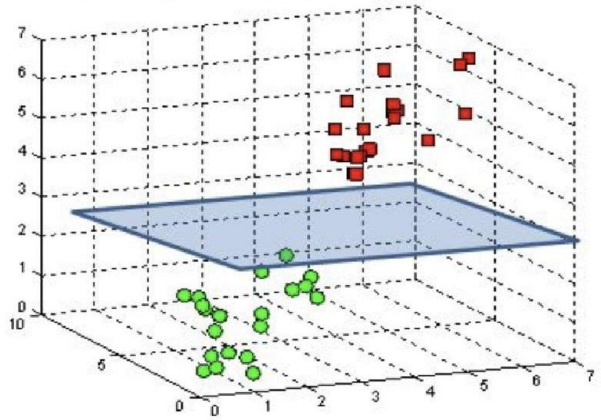

Figura 2. Hiperplanul optim pentru modelul cu trei atribute (plan tridimensional)

Hiperplanul se defineste prin formula:

$$x \cdot \underline{w} + b = 0 \quad (1)$$

unde $\underline{w}$ este n-vector, iar b este un număr real. Pentru reprezentarea grafică vom lua $\underline{w} = 1$, iar pentru calcularea distanței dintre hiperplan și punctul x vom egala formula (1) cu y.

$$x \cdot \underline{w} + b = y \quad (2)$$

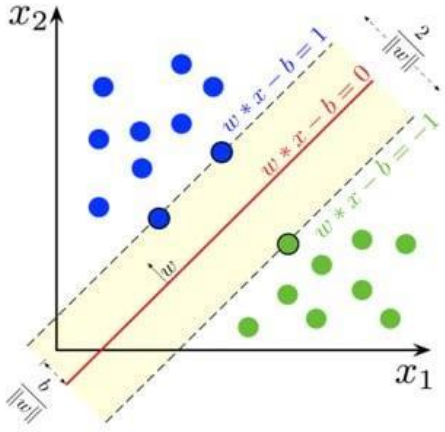

Figura 3. Distanța dintre hiperplan și punctul x

Pentru comoditate în figura 3 *y* a fost inlocuit cu ± 1 și vom primi că condiția de împărțire în clase pentru modelul nostru va fi:

$$y_n \begin{Bmatrix} +1 & x \cdot \underline{w} + b \geq 1 \\ -1 & x \cdot \underline{w} + b \leq -1 \end{Bmatrix} \quad (3)$$

Respectiv formula (3) poate fi rescrisa ca:

$$y_n(x \cdot \underline{w} + b) \geq 0 \quad (4)$$

sau

$$min \frac{1}{2} ||w||^2 \quad (5)$$

Pentru a găsi hiperplanul optim pentru modelul nostru trebuie să alegem planul astfel încât distanța dintre vectorii suport și hiperplan sa fie maximă, vectorii suport fiind punctele care se află cel mai aproape de hiperplan și influențează poziția și orientarea sa.

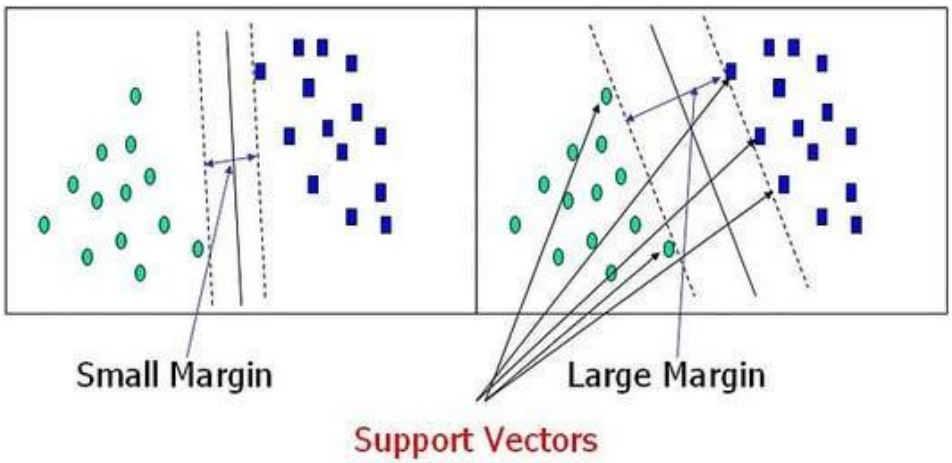

Figura 4. Vectorii suport și distanța lor de la hiperplan

În exemplele anterioare setul de date era unul ideal, fără erori sau valori aberante ce se vizualizează în figura 5, însă în realitate așa ceva nu se întâlnește și trebuie o modalitate de a aborda valorile date ca efectul lor asupra modelului să fie minimizat.

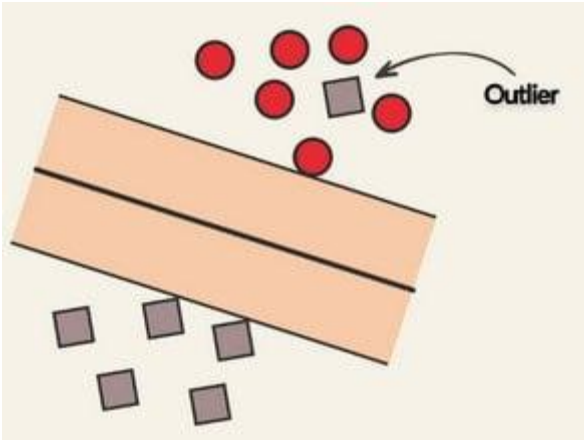

Figura 5. Valoare aberantă

Pentru a afla eroarea produsă de astfel de valoare vom folosi formula (6), graficul căruia poate fi vizualizat în figura 6.

$$\xi = max\{0, 1 - y_n(\underline{w} \cdot x + b)\} \quad (6)$$

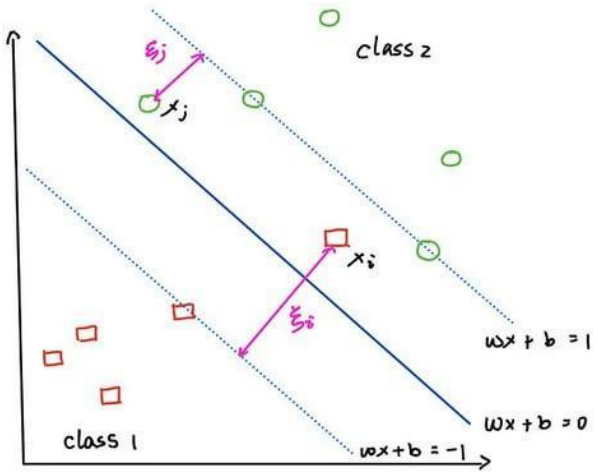

Figura 6. Eroarea valorilor aberante

Pentru a găsi eroarea produsă de toate valorile aberante vom lua suma erorilor care se calculează după formula (6):

$$\sum_{i=1}^{n} \xi_i \quad (7)$$

Unele valori aberante însă nu aduc o eroare mare în model și pot fi ignorate, pentru aceasta introducem o variabila C care va reprezenta nivelul de ignoranță față de valorile aberante și introducandu-se in formula vom primi:

$$C \cdot \sum_{i=1}^{n} \xi_i \quad (8)$$

După cate observăm din figura 7 cu cât mai mica e valoarea variabilei C cu atât mai tare se ignoră valorile aberante, iar cu cat e mai mare valoarea variabilei C cu atît mai mult se iau valorile date în considerație, aceasta fiind ilustrat în figura 7. Pentru a crea un model cât mai aproape de ideal trebuie de găsit valoarea optimă pentru C.

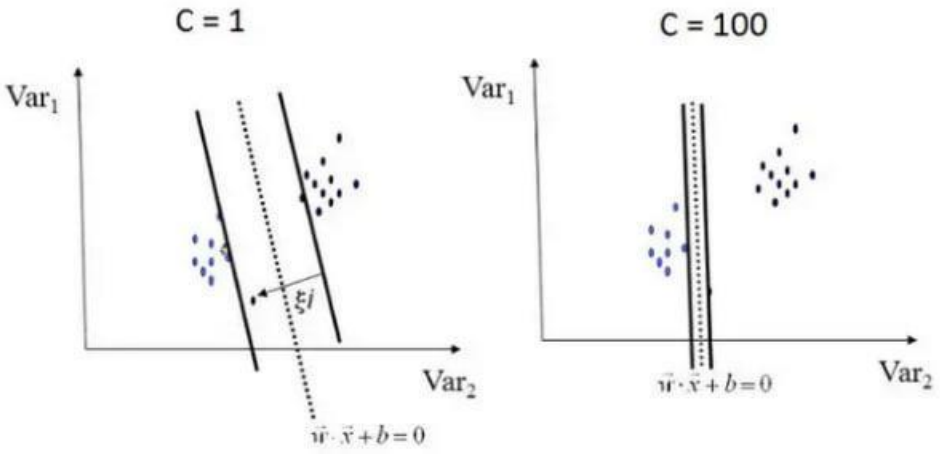

Figura 7. Impactul introducerii nivelului de ignoranță față de valorile aberante

Într-un final folosind formulele (5) și (8) vom primi formula cu ajutorul careia vom găsi hiperplanul optim:

$$min\frac{1}{2}||w||^2 + C \cdot \sum_{i=1}^{n} \quad \xi_i \qquad (9)$$

Pentru a găsi valorile optime pentru *w* și *b* se folosesc diferite modalități, cele mai populare fiind dualitatea și descendența gradientului [2-3].

Implementarea algoritmului

Pentru implementarea algoritmului a fost ales limbajul de programare Python cu următoarele biblioteci:

- tkinter - crearea interfeței pentru aplicație;
- scikit-learn - crearea și antrenarea modelului [4];
- joblib - salvarea modelului;
- opencv - lucru cu camera web;
- mediapipe - căutarea coordonatelor mainii în spațiu.

Inițial pentru a primi coordonatele mâinii persoanei cu care vorbim vom folosi biblioteca mediapipe, care cu ajutorul la opencv capteaza imaginea și după ne transmite coordonatele la 21 de puncte de pe mână prezentată în figura 8 [5].

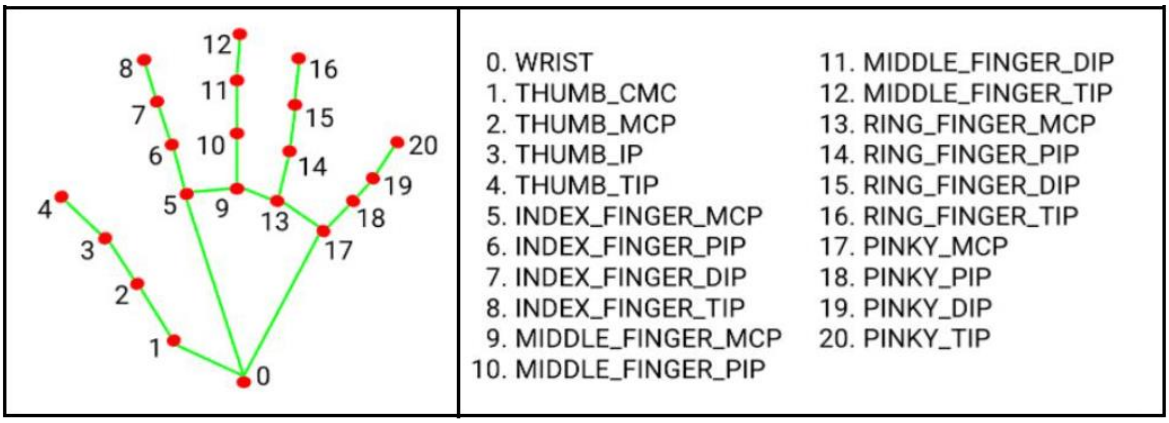

Figura 8. Punctele de pe mână luate de mediapipe.

După care informația dată se transmite modelului antrenat anticipat cu ajutorul scikit-learn, iar răspunsul dat de model va fi afișat în interfața noastră creată cu ajutorul bibliotecii tkinter ce se poate vizualiza în figura 9.

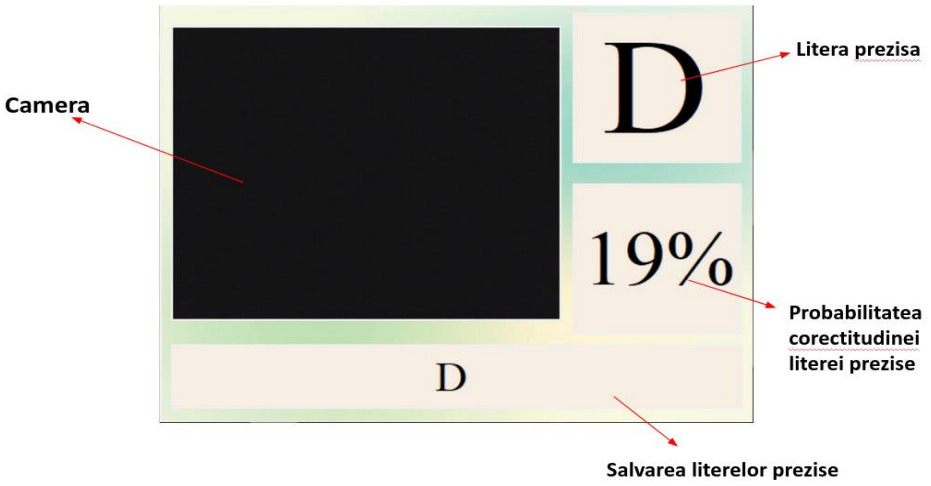

Figura 9. Interfața aplicației.

Analiza rezultatelor

Folosind funcția incorporata a bibliotecii scikit-learn vom vedea că modelul nostru are o precizie de ~82.6% și analizând rezultatele cu ajutorul matricei de confuzie figura 10 observăm că modelul nostru întâmpină greutăți cu înțelegerea literelor ‘m’ și ‘n’, deseori se încurcă ‘a’ cu ‘s’, ‘q’ cu ‘p’, ‘r’ cu ‘u’ etc. Unele din cauza apariției erorilor date pot fi setul de date necalitativ și volumul mic de informație.

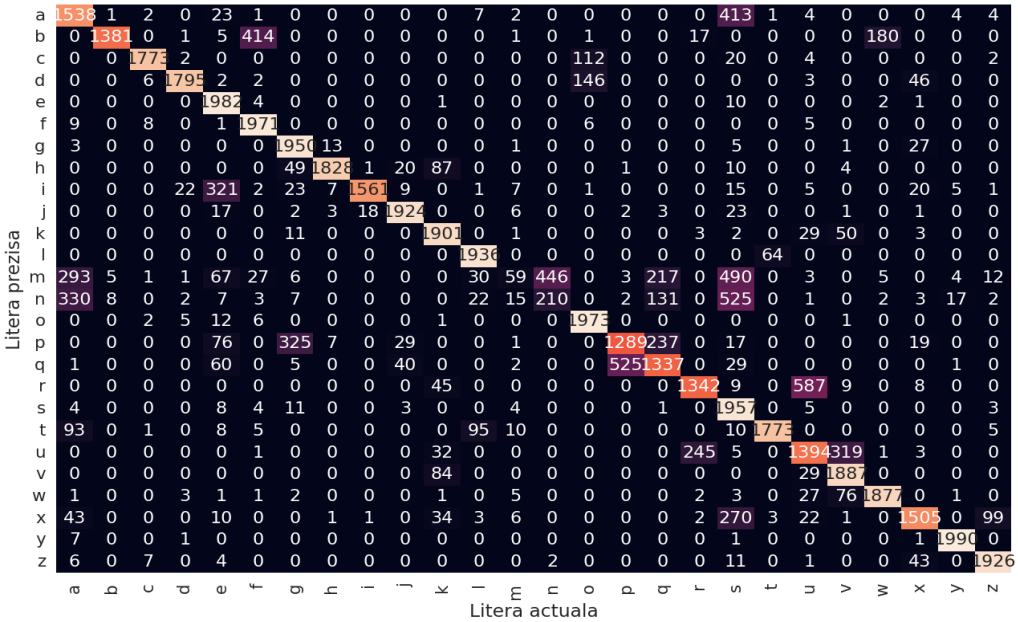

Figura 10. Matricea de confuzie al modelului

Concluzie

Ecuațiile deduse incă în anii 90 datorită hardware-ul din zilele noastre sa arătat ca fiind foarte eficiente. SVM este un algoritm care chiar și cu un set de date mic poate avea o precizie mare, în cazul nostru precizia este de aproximativ 83%, datorită diferitor posibilități care ne dă funcția, ea fiind efectivă indiferent de dimensiunea în care se folosește și are posibilitatea de a aborda valorile aberante folosind constanta *C*.

După cum observăm cu un set de date necalitativ chiar dacă ajungem la o precizie mare, oricum modelul are foarte multe erori, de unde înțelegem ca pentru a antrena bine modelul trebuie pentru început să lucrăm cu informația pe baza căreia îl vom antrena, pe lângă aceasta, valorile pentru constanta *C*, *W* și *B* deduse pentru modelul nostru nu vor fi bune pentru alte modele.

Deci pentru a crea un model cu o precizie mare cu ajutorul SVM se poate folosi un set de date mici, însă cu informație calitativă, și de asemenea ne trebuie să alegem valori optime pentru constante.

Referinţe

1. Explain Support Vector Machines in Mathematic. Details Hyperplane, maximal margin, hard-margin, soft-margin in math. [online]. [accesat 07.03.2023]. Disponibil: https://towardsdatascience.com/explain-support-vector-machines-in-mathematic-details-c7cc1be9f3b9
2. Duality (optimization) [online]. [accesat 07.03.2023]. Disponibil: https://en.wikipedia.org/wiki/Duality_(optimization)
3. Gradient Descent Algorithm — a deep dive [online]. [accesat 07.03.2023]. Disponibil: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21
4. Support Vector Machines [online]. [accesat 07.03.2023]. Disponibil: https://scikit-learn.org/stable/modules/svm.html
5. MediaPipe Hands [online]. [accesat 07.03.2023]. Disponibil: https://google.github.io/mediapipe/solutions/hands.html